

Volume 1

# Academic Press Library in Signal Processing

Signal Processing Theory  
and Machine Learning

Rama Chellappa  
Sergios Theodoridis



Academic Press is an imprint of Elsevier  
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK  
225 Wyman Street, Waltham, MA 02451, USA

First edition 2014

Copyright © 2014 Elsevier Ltd. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recording or otherwise without the prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; email: [permissions@elsevier.com](mailto:permissions@elsevier.com). Alternatively you can submit your request online by visiting the Elsevier web site at <http://elsevier.com/locate/permissions>, and selecting Obtaining permission to use Elsevier material.

#### **Notice**

No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

#### **Library of Congress Cataloging in Publication Data**

A catalog record for this book is available from the Library of Congress

#### **British Library Cataloguing in Publication Data**

A catalogue record for this book is available from the British Library

ISBN: 978-0-12-396502-8

For information on all Elsevier publications  
visit our website at [www.store.elsevier.com](http://www.store.elsevier.com)

Printed and bound in Poland.

14 15 16 17 10 9 8 7 6 5 4 3 2 1



Working together  
to grow libraries in  
developing countries

[www.elsevier.com](http://www.elsevier.com) • [www.bookaid.org](http://www.bookaid.org)

# Introduction

## Signal Processing at Your Fingertips!

Let us flash back to the 1970s when the editors-in-chief of this e-reference were graduate students. One of the time-honored traditions then was to visit the libraries several times a week to keep track of the latest research findings. After your advisor and teachers, the librarians were your best friends. We visited the engineering and mathematics libraries of our Universities every Friday afternoon and poured over the IEEE Transactions, Annals of Statistics, the Journal of Royal Statistical Society, Biometrika, and other journals so that we could keep track of the recent results published in these journals. Another ritual that was part of these outings was to take sufficient number of coins so that papers of interest could be xeroxed. As there was no Internet, one would often request copies of reprints from authors by mailing postcards and most authors would oblige. Our generation maintained thick folders of hard-copies of papers. Prof. Azriel Rosenfeld (one of RC's mentors) maintained a library of over 30,000 papers going back to the early 1950s!

Another fact to recall is that in the absence of Internet, research results were not so widely disseminated then and even if they were, there was a delay between when the results were published in technologically advanced western countries and when these results were known to scientists in third world countries. For example, till the late 1990s, scientists in US and most countries in Europe had a lead time of at least a year to 18 months since it took that much time for papers to appear in journals after submission. Add to this the time it took for the Transactions to go by surface mails to various libraries in the world. Scientists who lived and worked in the more prosperous countries were aware of the progress in their fields by visiting each other or attending conferences.

Let us race back to 21st century! We live and experience a world which is fast changing with rates unseen before in the human history. The era of Information and Knowledge societies had an impact on all aspects of our social as well as personal lives. In many ways, it has changed the way we experience and understand the world around us; that is, the way we learn. Such a change is much more obvious to the younger generation, which carries much less momentum from the past, compared to us, the older generation. A generation which has grew up in the Internet age, the age of Images and Video games, the age of IPAD and Kindle, the age of the fast exchange of information. These new technologies comprise a part of their "real" world, and Education and Learning can no more ignore this reality. Although many questions are still open for discussions among sociologists, one thing is certain. Electronic publishing and dissemination, embodying new technologies, is here to stay. This is the only way that effective pedagogic tools can be developed and used to assist the learning process from now on. Many kids in the early school or even preschool years have their own IPADs to access information in the Internet. When they grow up to study engineering, science, or medicine or law, we doubt if they ever will visit a library as they would by then expect all information to be available at their fingertips, literally!

Another consequence of this development is the leveling of the playing field. Many institutions in lesser developed countries could not afford to buy the IEEE Transactions and other journals of repute. Even if they did, given the time between submission and publication of papers in journals and the time it took for the Transactions to be sent over surface mails, scientists and engineers in lesser developed countries were behind by two years or so. Also, most libraries did not acquire the proceedings of conferences and so there was a huge gap in the awareness of what was going on in technologically advanced

countries. The lucky few who could visit US and some countries in Europe were able to keep up with the progress in these countries. This has changed. Anyone with an Internet connection can request or download papers from the sites of scientists. Thus there is a leveling of the playing field which will lead to more scientist and engineers being groomed all over the world.

The aim of Online Reference for Signal Processing project is to implement such a vision. We all know that asking any of our students to search for information, the first step for him/her will be to click on the web and possibly in the Wikipedia. This was the inspiration for our project. To develop a site, related to the Signal Processing, where a selected set of reviewed articles will become available at a first “click.” However, these articles are fully refereed and written by experts in the respected topic. Moreover, the authors will have the “luxury” to update their articles regularly, so that to keep up with the advances that take place as time evolves. This will have a double benefit. Such articles, besides the more classical material, will also convey the most recent results providing the students/researchers with up-to-date information. In addition, the authors will have the chance of making their article a more “permanent” source of reference, that keeps up its freshness in spite of the passing time.

The other major advantage is that authors have the chance to provide, alongside their chapters, any multimedia tool in order to clarify concepts as well as to demonstrate more vividly the performance of various methods, in addition to the static figures and tables. Such tools can be updated at the author’s will, building upon previous experience and comments. We do hope that, in future editions, this aspect of this project will be further enriched and strengthened.

In the previously stated context, the Online Reference in Signal Processing provides a revolutionary way of accessing, updating and interacting with online content. In particular, the Online Reference will be a living, highly structured, and searchable peer-reviewed electronic reference in signal/image/video Processing and related applications, using existing books and newly commissioned content, which gives tutorial overviews of the latest technologies and research, key equations, algorithms, applications, standards, code, core principles, and links to key Elsevier journal articles and abstracts of non-Elsevier journals.

The audience of the Online Reference in Signal Processing is intended to include practicing engineers in signal/image processing and applications, researchers, PhD students, post Docs, consultants, and policy makers in governments. In particular, the readers can be benefited in the following needs:

- To learn about new areas outside their own expertise.
- To understand how their area of research is connected to other areas outside their expertise.
- To learn how different areas are interconnected and impact on each other: the need for a “helicopter” perspective that shows the “wood for the trees.”
- To keep up-to-date with new technologies as they develop: what they are about, what is their potential, what are the research issues that need to be resolved, and how can they be used.
- To find the best and most appropriate journal papers and keeping up-to-date with the newest, best papers as they are written.
- To link principles to the new technologies.

The Signal Processing topics have been divided into a number of subtopics, which have also dictated the way the different articles have been compiled together. Each one of the subtopics has been coordinated by an AE (Associate Editor). In particular:

1. Signal Processing Theory (Prof. P. Diniz)
2. Machine Learning (Prof. J. Suykens)
3. DSP for Communications (Prof. N. Sidiropoulos)
4. Radar Signal Processing (Prof. F. Gini)
5. Statistical SP (Prof. A. Zoubir)
6. Array Signal Processing (Prof. M. Viberg)
7. Image Enhancement and Restoration (Prof. H. J. Trussell)
8. Image Analysis and Recognition (Prof. Anuj Srivastava)
9. Video Processing (other than compression), Tracking, Super Resolution, Motion Estimation, etc. (Prof. A. R. Chowdhury)
10. Hardware and Software for Signal Processing Applications (Prof. Ankur Srivastava)
11. Speech Processing/Audio Processing (Prof. P. Naylor)
12. Still Image Compression
13. Video Compression

We would like to thank all the Associate Editors for all the time and effort in inviting authors as well as coordinating the reviewing process. The Associate Editors have also provided succinct summaries of their areas.

The articles included in the current editions comprise the first phase of the project. In the second phase, besides the updates of the current articles, more articles will be included to further enrich the existing number of topics. Also, we envisage that, in the future editions, besides the scientific articles we are going to be able to include articles of historical value. Signal Processing has now reached an age that its history has to be traced back and written.

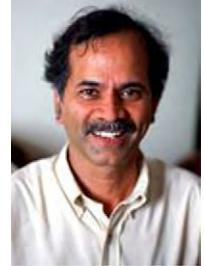
Last but not least, we would like to thank all the authors for their effort to contribute in this new and exciting project. We earnestly hope that in the area of Signal Processing, this reference will help level the playing field by highlighting the research progress made in a timely and accessible manner to anyone who has access to the Internet. With this effort the next breakthrough advances may be coming from all around the world.

The companion site for this work: <http://booksite.elsevier.com/9780124166165> includes multimedia files (Video/Audio) and MATLAB codes for selected chapters.

Rama Chellappa  
Sergios Theodoridis

# About the Editors

**Rama Chellappa** received the B.E. (Hons.) degree in Electronics and Communication Engineering from the University of Madras, India in 1975 and the M.E. (with Distinction) degree from the Indian Institute of Science, Bangalore, India in 1977. He received the M.S.E.E. and Ph.D. Degrees in Electrical Engineering from Purdue University, West Lafayette, IN, in 1978 and 1981, respectively. During 1981–1991, he was a faculty member in the department of EE-Systems at University of Southern California (USC). Since 1991, he has been a Professor of Electrical and Computer Engineering (ECE) and an affiliate Professor of Computer Science at University of Maryland (UMD), College Park. He is also affiliated with the Center for Automation Research, the Institute for Advanced Computer Studies (Permanent Member) and is serving as the Chair of the ECE department. In 2005, he was named a Minta Martin Professor of Engineering. His current research interests are face recognition, clustering and video summarization, 3D modeling from video, image and video-based recognition of objects, events and activities, dictionary-based inference, compressive sensing, domain adaptation and hyper spectral processing.



Prof. Chellappa received an NSF Presidential Young Investigator Award, four IBM Faculty Development Awards, an Excellence in Teaching Award from the School of Engineering at USC, and two paper awards from the International Association of Pattern Recognition (IAPR). He is a recipient of the K.S. Fu Prize from IAPR. He received the Society, Technical Achievement, and Meritorious Service Awards from the IEEE Signal Processing Society. He also received the Technical Achievement and Meritorious Service Awards from the IEEE Computer Society. At UMD, he was elected as a Distinguished Faculty Research Fellow, as a Distinguished Scholar-Teacher, received an Outstanding Innovator Award from the Office of Technology Commercialization, and an Outstanding GEMSTONE Mentor Award from the Honors College. He received the Outstanding Faculty Research Award and the Poole and Kent Teaching Award for Senior Faculty from the College of Engineering. In 2010, he was recognized as an Outstanding ECE by Purdue University. He is a Fellow of IEEE, IAPR, OSA, and AAAS. He holds four patents.

Prof. Chellappa served as the Editor-in-Chief of IEEE Transactions on Pattern Analysis and Machine Intelligence. He has served as a General and Technical Program Chair for several IEEE international and national conferences and workshops. He is a Golden Core Member of the IEEE Computer Society and served as a Distinguished Lecturer of the IEEE Signal Processing Society. Recently, he completed a two-year term as the President of the IEEE Biometrics Council.

**Sergios Theodoridis** is currently Professor of Signal Processing and Communications in the Department of Informatics and Telecommunications of the University of Athens. His research interests lie in the areas of Adaptive Algorithms and Communications, Machine Learning and Pattern Recognition, Signal Processing for Audio Processing and Retrieval. He is the co-editor of the book “Efficient Algorithms for Signal Processing and System Identification,” Prentice Hall 1993, the co-author of the best selling book “Pattern Recognition,” Academic Press, 4th ed. 2008, the co-author of the book “Introduction to Pattern Recognition: A MATLAB Approach,” Academic Press, 2009, and the co-author of three books in Greek, two of them for the Greek Open University. He is Editor-in-Chief for the Signal Processing Book Series, Academic Press and for the E-Reference Signal Processing, Elsevier.



He is the co-author of six papers that have received best paper awards including the 2009 IEEE Computational Intelligence Society Transactions on Neural Networks Outstanding paper Award. He has served as an IEEE Signal Processing Society Distinguished Lecturer. He was *Otto Monstead Guest Professor*, Technical University of Denmark, 2012, and holder of the *Excellence Chair*, Department of Signal Processing and Communications, University Carlos III, Madrid, Spain, 2011.

He was the General Chairman of EUSIPCO-98, the Technical Program co-Chair for ISCAS-2006 and ISCAS-2013, and co-Chairman and co-Founder of CIP-2008 and co-Chairman of CIP-2010. He has served as President of the European Association for Signal Processing (EURASIP) and as member of the Board of Governors for the IEEE CAS Society. He currently serves as member of the Board of Governors (Member-at-Large) of the IEEE SP Society.

He has served as a member of the Greek National Council for Research and Technology and he was Chairman of the SP advisory committee for the Edinburgh Research Partnership (ERP). He has served as Vice Chairman of the Greek Pedagogical Institute and he was for 4 years member of the Board of Directors of COSMOTE (the Greek mobile phone operating company). He is Fellow of IET, a Corresponding Fellow of the Royal Society of Edinburgh (RSE), a Fellow of EURASIP, and a Fellow of IEEE.

# Section Editors

## Section 1

**Paulo S. R. Diniz** was born in Niterói, Brazil. He received the Electronics Engineering degree (Cum Laude) from the Federal University of Rio de Janeiro (UFRJ) in 1978, the M.Sc. degree from COPPE/UFRJ in 1981, and the Ph.D. from Concordia University, Montreal, P.Q., Canada, in 1984, all in electrical engineering.

Since 1979 he has been with the Department of Electronic Engineering (the undergraduate department) UFRJ. He has also been with the Program of Electrical Engineering (the graduate studies department), COPPE/UFRJ, since 1984, where he is presently a Professor. He served as Undergraduate Course Coordinator and as Chairman of the Graduate Department. He has also received the Rio de Janeiro State Scientist award, from the Governor of Rio de Janeiro state.



From January 1991 to July 1992, he was a visiting Research Associate in the Department of Electrical and Computer Engineering of University of Victoria, Victoria, B.C., Canada. He also holds a Docent position at Helsinki University of Technology (now Aalto University). From January 2002 to June 2002, he was a Melchor Chair Professor in the Department of Electrical Engineering of University of Notre Dame, Notre Dame, IN, USA. His teaching and research interests are in analog and digital signal processing, adaptive signal processing, digital communications, wireless communications, multirate systems, stochastic processes, and electronic circuits. He has published several refereed papers in some of these areas and wrote the books *ADAPTIVE FILTERING: Algorithms and Practical Implementation*, Springer, NY, Fourth Edition 2013, *DIGITAL SIGNAL PROCESSING: System Analysis and Design*, Cambridge University Press, Cambridge, UK, Second Edition 2010 (with E. A. B. da Silva and S. L. Netto), and *Block Transceivers: OFDM and Beyond*, Morgan & Claypool, Fort Collins, CO, 2012, (with W. A. Martins and M. V. S. Lim). He has served as General co-Chair of ISCAS2011 and Technical Program Chair of the 1995 MWSCAS both held in Rio de Janeiro, Brazil. He was also the Technical Program co-Chair of SPAWC2008. He has been on the technical committee of several international conferences including ICASSP, ISCAS, ICECS, EUSIPCO, and MWSCAS. He has served as the Vice President for region 9 of the IEEE Circuits and Systems Society and as Chairman of the DSP technical committee of the same Society. He is also a Fellow of IEEE (for fundamental contributions to the design and implementation of fixed and adaptive filters and Electrical Engineering Education). He has served as Associate Editor for the following Journals: IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing from 1996 to 1999, IEEE Transactions on Signal Processing from 1999 to 2002, and the Circuits, Systems and Signal Processing Journal from 1998 to 2002. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society for the year 2000–2001. In 2004 he served as Distinguished Lecturer of the IEEE Signal Processing Society and received the 2004 Education Award of the IEEE Circuits and Systems Society.

## Section 2

**Johan A.K. Suykens** received the master degree in ElectroMechanical Engineering and the Ph.D. degree in Applied Sciences from the Katholieke Universiteit Leuven, in 1989 and 1995, respectively. In 1996 he has been a Visiting Postdoctoral Researcher at the University of California, Berkeley. He has been a Postdoctoral Researcher with the Fund for Scientific Research FWO Flanders and is currently a Professor (Hoogleraar) with KU Leuven. He is author of the books “Artificial Neural Networks for Modeling and Control of Non-linear Systems” (Kluwer Academic Publishers) and “Least Squares Support Vector Machines” (World Scientific), co-author of the book “Cellular Neural Networks, Multi-Scroll Chaos and Synchronization” (World Scientific), and editor of the books “Nonlinear Modeling: Advanced Black-Box Techniques” (Kluwer Academic Publishers) and “Advances in Learning Theory: Methods, Models and Applications” (IOS Press). He is a Senior IEEE member and has served as associate editor for the IEEE Transactions on Circuits and Systems (1997–1999 and 2004–2007) and for the IEEE Transactions on Neural Networks (1998–2009). He received an IEEE Signal Processing Society 1999 Best Paper (Senior) Award and several Best Paper Awards at International Conferences. He is a recipient of the International Neural Networks Society INNS 2000 Young Investigator Award for significant contributions in the field of neural networks. He has served as a Director and Organizer of the NATO Advanced Study Institute on Learning Theory and Practice (Leuven 2002), as a program co-chair for the International Joint Conference on Neural Networks 2004 and the International Symposium on Non-linear Theory and its Applications 2005, as an organizer of the International Symposium on Synchronization in Complex Networks 2007 and a co-organizer of the NIPS 2010 workshop on Tensors, Kernels and Machine Learning. He has been awarded an ERC Advanced Grant 2011.



# Authors Biography

## CHAPTER 1

**Isabela Ferrão Apolinário** was born in Brasília, Brazil. She is currently studying Electronics Engineering in the Federal University of Rio de Janeiro (UFRJ) and will soon join an M.Sc. course in the same university.

Since 2010 she has been a member of the Audio Processing Group (GPA) from the Signal Processing Laboratory (LPS) and has been studying audio processing. She has worked as an undergraduate teaching assistance in Calculus II and Digital Signal Processing.

She is currently a student member of the IEEE Circuits and Systems Society. Her main interest is in digital audio signal processing, more specifically in Psychoacoustics and Audio 3D.



## CHAPTER 2

**José Antonio Apolinário** Junior graduated from the Military Academy of Agulhas Negras (AMAN), Brazil, in 1981 and received the B.Sc. degree from the Military Institute of Engineering (IME), Brazil, in 1988, the M.Sc. degree from the University of Brasília (UnB), Brazil, in 1993, and the D.Sc. degree from the Federal University of Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, Brazil, in 1998, all in electrical engineering. He is currently an Adjoint Professor with the Department of Electrical Engineering, IME, where he has already served as the Head of Department and as the Vice-Rector for Study and Research. He was a Visiting Professor at the em Escuela Politécnica del Ejército (ESPE), Quito, Ecuador, from 1999 to 2000 and a Visiting Researcher and twice a Visiting Professor at Helsinki University of Technology (HUT), Finland, in 1997, 2004, and 2006, respectively. His research interests comprise many aspects of linear and nonlinear digital signal processing, including adaptive filtering, speech, and array processing. He has organized and has been the first Chair of the Rio de Janeiro Chapter of the IEEE Communications Society. He has recently edited the book "QRDRRLS Adaptive Filtering" (Springer, 2009) and served as the Finance Chair of IEEE ISCAS 2011 (Rio de Janeiro, Brazil, May 2011). He is a senior member of the IEEE.



**Carla Liberal Pagliari** received the Ph.D. degree in electronic systems engineering from the University of Essex, UK, in 2000. Since 1993 she has been with the Department of Electrical Engineering at the Military Institute of Engineering (IME), Rio de Janeiro, Brazil. She took part in the team that worked toward the development of the Brazilian Digital Television System. Her research interests include image processing, digital television, image and video coding, stereoscopic and multiview systems, and computer vision. She is a senior member of the IEEE and served as the Local Arrangements Chair



of IEEE ISCAS 2011 (Rio de Janeiro, Brazil, May 2011). She is currently an Associate Editor of the journal Multidimensional Systems and Signal Processing.

## CHAPTER 3

**Leonardo Gomes Baltar** received his B.Sc. and M.Sc. degrees in Electrical Engineering from the Federal University of Rio de Janeiro (UFRJ), Brazil, in 2004 and 2006, respectively. Since 2006, he is with the Chair of Circuit Theory and Signal Processing at the Technical University of Munich (TUM), Germany, as a research and teaching assistant pursuing a Ph.D. degree. His activities are mainly in the area of digital signal processing techniques applied to wired and wireless communications systems including multirate systems, filter banks, block transforms, digital filters design, and efficient processing structures. He has been involved in the European FP7 project PHYDYAS and in projects with industrial partners.



**Josef A. Nossek** received his Dipl.-Ing. and Dr. techn. degrees from the Technical University of Vienna, Austria, in 1974 and 1980, respectively. He was for 15 years with Siemens AG, Munich, in the field of communications, including the position of Head of Radio Systems Design for digital communications. In 1989 he joined the Technical University of Munich (TUM), Germany, where he is full Professor and Head of the Chair for Circuit Theory and Signal Processing. He is a Fellow of IEEE since 1993 for his contributions to the design of discrete-time networks and technical leadership in the development of radio communication systems. He was the Editor-in-Chief of the IEEE Transactions on Circuits and Systems (1993–1995). He was President-Elect, President, and Past President of the IEEE Circuits and Systems Society (2001/02/03). In 1998 he received the Innovations Award of the Vodafone Foundation for excellent research in mobile communications. In 2008 he received the Education Award of the IEEE Circuits and Systems Society. In 2011 he received the IEEE Guillemin-Cauer Best Paper Award. Since 2009 he is member of acatech (National Academy of Science and Engineering). He is author or co-author of numerous publications and has given a number of invited plenary lectures. He was the President of VDE (2007–2008), the German Association of Electrical, Electronics and Information Engineering, and Vice-President of VDE (2009–2010).



## CHAPTER 4

**Luiz W. P. Biscainho** was born in Rio de Janeiro, Brazil, in 1962. He received the Electronics Engineering degree (magna cum laude) from the EE (now Poli) at Universidade Federal do Rio de Janeiro (UFRJ), Brazil, in 1985, and the M.Sc. and D.Sc. degrees in Electrical Engineering from the COPPE at UFRJ in 1990 and 2000, respectively. Having worked in the telecommunication industry between 1985 and 1993, Dr. Biscainho is now Associate



Professor at the Department of Electronics and Computer Engineering (DEL) of Poli and the Electrical Engineering Program (PEE) of COPPE (serving as Academic Coordinator in 2010), at UFRJ. His research area is digital signal processing, particularly audio processing and adaptive systems. He is currently a member of the IEEE (Institute of Electrical and Electronics Engineers), the AES (Audio Engineering Society), the SBrT (Brazilian Telecommunications Society), and the SBC (Brazilian Computer Society).

## CHAPTER 5

**Håkan Johansson** received the Master of Science degree in computer science and the Licentiate, Doctoral, and Docent degrees in Electronics Systems from Linkoping University, Sweden, in 1995, 1997, 1998, and 2001, respectively. During 1998 and 1999 he held a postdoctoral position at Signal Processing Laboratory, Tampere University of Technology, Finland. He is currently Professor in Electronics Systems at the Department of Electrical Engineering of Linkoping University. His research encompasses design and implementation of efficient and flexible signal processing (SP) systems, mainly for communication applications. During the past decade, he has developed many different SP algorithms for various purposes, including filtering, sampling rate conversion, signal reconstruction, and parameter estimation. He has developed new estimation and compensation algorithms for errors in analog circuits such as compensation of mismatch errors in time-interleaved analog-to-digital converters and mixers. He is one of the founders of the company Signal Processing Devices Sweden AB that sells this type of advanced signal processing. He is the author or co-author of four books and some 160 international journal and conference papers. He is the co-author of three papers that have received best paper awards and he has authored one invited paper in IEEE Transactions and four invited chapters. He has served as Associate Editor for IEEE Trans. on Circuits and Systems I and II, IEEE Trans. Signal Processing, and IEEE Signal Processing Letters, and he is currently an Area Editor of the Elsevier Digital Signal Processing journal and a member of the IEEE Int. Symp. Circuits. Syst. DSP track committee.



## CHAPTER 6

**Lars Wanhammar** was born in Vansbro, Sweden, on August 19, 1944. He has received the following degrees: Teknisk magister (teknisk fysik) in 1970, civilingenjör in 1980, teknisk doktor in 1981, and docent in 1986, in electrical engineering from Linköping University, Sweden. During 1964–1970 he worked at Televerket (Royal Swedish Telephone Board), Division of Communication and during 1970–1971 as a Lecturer at the technical college in Norrköping. Since 1971 he has been working at Linköping University, Department of Electrical Engineering, Division of Applied Electronics, as Assistant, Research Assistant, and from 1982 as Associate Professor (universitetslektor) and from 1997 as full Professor and Head of the Division of Electronics Systems, at the Department of Electrical Engineering, Linköping University, Sweden. From 2011 he is currently Professor Emeritus.



## **xxxviii Authors Biography**

In addition, from 1995 to 2004 he worked as Adjunct Professor at the Norwegian Institute of Technology (NTNU) at the departments of Physical Electronics and Telecommunications.

His research interests are primary theory and design of digital signal processing and telecommunication systems, particularly analog and digital filters, and discrete transforms as well as computational properties of DSP algorithms, bit-serial, digit-serial and distributed arithmetic, CAD tools, and globally asynchronous locally synchronous techniques for ULSI.

He is the author or co-author of five books in analog and digital filters, one in parallel processing in industrial real-time applications, and one in DSP integrated circuits.

**Ya Jun Yu** received the B.Sc. and M.Eng. degrees in biomedical engineering from Zhejiang University, Hangzhou, China, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2004.

From 1997 to 1998, she was a Teaching Assistant with Zhejiang University. She joined the Department of Electrical and Computer Engineering, National University of Singapore as a Post Master Fellow in 1998 and remained in the same department as a Research Engineer until 2004. She joined the Temasek Laboratories at Nanyang Technological University as a Research Fellow in 2004. Since 2005, she has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where she is currently an Assistant Professor. Her research interests include digital signal processing and VLSI circuits and systems design.

He has served as an associate editor for Circuits Systems and Signal Processing and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II since 2009 and 2010, respectively.



## **CHAPTER 7**

**Fred Harris** holds the Signal Processing Chair of the Communication Systems and Signal Processing Institute at San Diego State University where since 1967 he has taught courses in Digital Signal Processing and Communication Systems. He holds 20 patents on digital receiver and DSP technology and lectures throughout the world on DSP applications. He consults for organizations requiring high-performance, cost-effective DSP solutions. He is an adjunct member of the IDA-Princeton Center for Communications Research.

Fred Harris has written over 200 journal and conference papers, the most well known being his 1978 paper “On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform”. He is the author of the book **Multirate Signal Processing for Communication Systems** and he has contributed to a number of other books on DSP applications including the “Source Coding” chapter in Bernard Sklar’s 1988 book, **Digital Communications** and the “Multirate FIR Filters for Interpolation and Resampling” and the “Time Domain Signal Processing with the DFT” chapters in Doug Elliot’s 1987 book **Handbook of Digital Signal Processing**, and “A most Efficient Digital Filter: The Two-Path Recursive All-Pass Filter” Chapter and the “Ultra Low Phase Noise DSP Oscillator” Chapter in Rick Lyon’s 2012 book **Streamlining Digital Signal Processing**. He is also co-author of the book **Software Radio Sampling Rate Selection, Design and Synchronization**.



In 1990 and 1991 he was the Technical and then the General Chair of the Asilomar Conference on Signals, Systems, and Computers and was Technical Chair of the 2003 Software Defined Radio Conference and of the 2006 Wireless Personal Multimedia Conference. He became a Fellow of the IEEE in 2003, cited for contributions of DSP to communications systems. In 2006 he received the Software Defined Radio Forum's "Industry Achievement Award". His paper at the 2006 SDR conference was selected for the best paper award as was his paper at the 2010 Autotestcon conference and again his paper at the 2011 Wireless Personal Mobile Communications Conference and once again the 2011 SDR conference. He is the former Editor-in-Chief of the Elsevier DSP Journal.

The spelling of my name with all lower case letters is a source of distress for typists and spell checkers. A child at heart, I collect toy trains and old slide-rules.

**Elettra Venosa** received the "Laurea" (BS/MS) degree, summa cum laude, in Electrical Engineering in January 2007 from Seconda Università degli Studi di Napoli, Italy. From January 2007 to November 2007 she was a researcher at the Italian National Inter-University Consortium for Telecommunications. In January 2011, she received the Ph.D. in Telecommunication/DSP from Seconda Università degli Studi di Napoli. From June 2008 to September 2008 she worked as a project manager for Kiranet –ICT Research Center – to develop an advanced radio identification system for avionics, in collaboration with the Italian Center for Aerospace Research (CIRA). From April 2009 to September 2009 she worked as associate researcher at Communications and Signal Processing Laboratory (CSPL) in the Department of Electrical and Computer Engineering at Drexel University, Philadelphia, where she worked on Sparse Sampling Techniques for Software Defined Radio Receivers. From January 2011 to July 2012 she worked as principal system engineer in IQ-Analog, CA, developing algorithms for digital correction in TI-ADCs. From August 2012 to December 2013 she was associate researcher in Qualcomm, CA. Her focus was to improve the current commercial modem architectures. Currently, she is working, as a postdoctoral researcher, on Multirate Signal Processing for Software Defined Radios in the Department of Electrical Engineering at San Diego State University, San Diego, CA, USA where she also teaches graduate and undergraduate courses. She is also working as a software defined radio engineer in Space Micro, CA. She is the author of more than 40 scientific publications on SDR and of the book "Software Radio: Sampling Rate Selection Design and Synchronization".



**Xiaofei** Chen received the Bachelor's degree in Electrical Engineering in June 2006 from Xi'an University of Posts & Telecommunications, China. In December 2008, he received the Master's degree at Electrical & Computer Engineering department, San Diego State University, USA. In March 2009, he joined the Joint Doctoral Program between San Diego State University and University of California, San Diego. His current research interests are in the area of multirate signal processing and software defined radio.



## CHAPTER 8

**Trac D. Tran** (S'94–M'98–SM'08) received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, in 1993 and 1994, respectively, and the Ph.D. degree from the University of Wisconsin, Madison, in 1998, all in electrical engineering.

In July 1998, he joined the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD, where he currently holds the rank of Professor. His research interests are in the field of signal processing, particularly in sparse representation, sparse recovery, sampling, multirate systems, filter banks, transforms, wavelets, and their applications in signal analysis, compression, processing, and communications. His pioneering research on integer-coefficient transforms and pre-/post-filtering operators has been adopted as critical components of Microsoft Windows Media Video 9 and JPEG XR—the latest international still-image compression standard ISO/IEC 29199–2. He is currently a regular consultant for the US Army Research Laboratory, Adelphi, MD.



He was the codirector (with Prof. J. L. Prince) of the 33rd Annual Conference on Information Sciences and Systems (CISS'99), Baltimore, in March 1999. In the summer of 2002, he was an ASEE/ONR Summer Faculty Research Fellow at the Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, CA. He has served as Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING as well as the IEEE TRANSACTIONS ON IMAGE PROCESSING. He was a former member of the IEEE Technical Committee on Signal Processing Theory and Methods (SPTM TC) and is a current member of the IEEE Image Video and Multidimensional Signal Processing (IVMSP) Technical Committee. He received the NSF CAREER award in 2001, the William H. Huggins Excellence in Teaching Award from The Johns Hopkins University in 2007, and the Capers and Marion McDonald Award for Excellence in Mentoring and Advising in 2009.

## CHAPTER 9

**Yufang Bao** has been an Assistant Professor in the Department of Mathematics and Computer Science at UNC Fayetteville State University (UNCFSU) since 2007. She is also a scholar of the Center of Defense and Homeland Security (CDHS) at UNCFSU. She received her first Ph.D. degree in probability/statistics from Beijing Normal University (BNU), Beijing, China, and her second Ph.D. degree in Electrical Engineering from North Carolina State University (NCSU), Raleigh, NC. Her research focus was in probability/statistics. She subsequently directed her research into the area of applying mathematics in signal/image processing and analysis. Her contributions in signal/image processing included algorithm development that bridged stochastic diffusion and multi-scale wavelet theory with scale space analysis methods for image denoising and segmentation. Between 2002 and 2007, she has worked at the VA Center, UCSF, CA and then at the University of Miami, School of Medicine, FL, both as a research scientist focusing on statistical image reconstruction in Frequency domain with MR spectroscopy imaging, and with parallel



MR image reconstruction using mathematical modeling. Currently, her research interests are in applying mathematics to statistical digital signal/image processing and analysis, mathematical modeling, and their applications.

**Hamid Krim** (ahk@ncsu.edu) received his degrees in ECE from University of Washington and Northeastern University. He was a Member of Technical Staff at *AT&T Bell Labs*, where he has conducted research and development in the areas of telephony and digital communication systems/subsystems. Following an NSF postdoctoral fellowship at Foreign Centers of Excellence, LSS/University of Orsay, Paris, France, he joined the *Laboratory for Information and Decision Systems*, Massachusetts Institute of Technology, Cambridge, MA as a Research Scientist and where he was performing and supervising research.



He is presently Professor of Electrical Engineering in the ECE Department, North Carolina State University, Raleigh, leading the *Vision, Information and Statistical Signal Theories and Applications group*. His research interests are in statistical signal and image analysis and mathematical modeling with a keen emphasis on applied problems in classification and recognition using geometric and topological tools.

## CHAPTER 10

**Lisandro Lovisolo** was born in Neuquen, Argentina, but considers himself brazilian. He received the Electronics Engineering degree from Universidade Federal do Rio de Janeiro, in 1999, the M.Sc. degree in Electrical Engineering in 2001, and the D.Sc. degree in Electrical Engineering both from Universidade Federal do Rio de Janeiro (COPPE/UFRJ). Since 2003 he has been with the Department of Electronics and Communications Engineering (the undergraduate department), UERJ. He has also been with the Postgraduate in Electronics Program, since 2008. His research interests lie in the fields of digital signal and image processing and communications.



**Eduardo A. B. da Silva** was born in Rio de Janeiro, Brazil. He received the Electronics Engineering degree from Instituto Militar de Engenharia (IME), Brazil, in 1984, the M.Sc. degree in Electrical Engineering from Universidade Federal do Rio de Janeiro (COPPE/UFRJ) in 1990, and the Ph.D. degree in Electronics from the University of Essex, England, in 1995. In 1987 and 1988 he was with the Department of Electrical Engineering at Instituto Militar de Engenharia, Rio de Janeiro, Brazil. Since 1989 he has been with the Department of Electronics Engineering (the undergraduate department), UFRJ. He has also been with the Department of Electrical Engineering (the graduate studies department), COPPE/UFRJ, since 1996. His research interests lie in the fields of digital signal and image processing, especially signal compression, digital television, wavelet transforms, mathematical morphology, and applications to telecommunications.



## CHAPTER 11

**Suleyman Serdar Kozat** received the B.S. degree with full scholarship and high honors from Bilkent University, Turkey. He received the M.S. and Ph.D. degrees in Electrical and Computer Engineering from University of Illinois at Urbana Champaign, Urbana, IL, in 2001 and 2004, respectively.

After graduation, he joined IBM Research, T.J. Watson Research Center, Yorktown, NY as a Research Staff Member in Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. While doing his Ph.D., he was also working as a Research Associate at Microsoft Research, Redmond, WA, in Cryptography and Anti-Piracy Group. He holds several patent applications for his works performed in IBM Research and Microsoft Research. Currently, he is an Assistant Professor at the electrical and electronics engineering department, Koc University, Turkey. He coauthored more than 50 papers in refereed high impact journals and conference proceedings and has several patent applications. Overall, his research interests include intelligent systems, adaptive filtering for smart data analytics, online learning, and machine learning algorithms for signal processing.



He has been serving as an Associate Editor for the IEEE Transactions on Signal Processing and he is a Senior Member of the IEEE. He has been awarded IBM Faculty Award by IBM Research in 2011, Outstanding Faculty Award by Koc University in 2011, Outstanding Young Researcher Award by the Turkish National Academy of Sciences in 2010, ODTU Prof. Dr. Mustafa N. Parlar Research Encouragement Award in 2011 and holds Career Award by the Scientific Research Council of Turkey, 2009. He has won several scholarships and medals in international and national science and math competitions.

**Andrew C. Singer** received the S.B., S.M., and Ph.D. degrees, all in Electrical Engineering and Computer Science, from the Massachusetts Institute of Technology. Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently a Professor in the ECE department and the Coordinated Science Laboratory. During the academic year 1996, he was a Postdoctoral Research Affiliate in the Research Laboratory of Electronics at MIT. From 1996 to 1998, he was a Research Scientist at Sanders, A Lockheed Martin Company in Manchester, New Hampshire, where he designed algorithms, architectures, and systems for a variety of DOD applications. His research interests include signal processing and communication systems. He was a Hughes Aircraft Masters Fellow and was the recipient of the Harold L. Hazen Memorial Award for excellence in teaching in 1991. In 2000, he received the National Science Foundation CAREER Award, in 2001 he received the Xerox Faculty Research Award, and in 2002 he was named a Willett Faculty Scholar. He has served as an Associate Editor for the IEEE Transactions on Signal Processing and is a member of the MIT Educational Council, and of Eta Kappa Nu and Tau Beta Pi. He is a Fellow of the IEEE.



In 2005, he was appointed as the Director of the Technology Entrepreneur Center (TEC) in the College of Engineering. He also co-founded Intersymbol Communications, Inc., a venture-funded

fabless semiconductor IC company, based in Champaign Illinois. A developer of signal processing enhanced chips for ultra-high speed optical communications systems, Intersymbol was acquired by Finisar Corporation (NASD:FNSR) in 2007. He serves on the board of directors of a number of technology companies and as an expert witness for electronics, communications, and circuit-related technologies.

## CHAPTER 12

**Vítor H. Nascimento** was born in São Paulo, Brazil. He obtained the B.S. and M.S. degrees in Electrical Engineering from the University of São Paulo in 1989 and 1992, respectively, and the Ph.D. degree from the University of California, Los Angeles, in 1999. From 1990 to 1994 he was a Lecturer at the University of São Paulo, and in 1999 he joined the faculty at the same school, where he is now an Associate Professor. One of his papers received the 2002 IEEE SPS Best Paper Award. He served as an Associate Editor for the IEEE Signal Processing Letters from 2003 to 2005, for the IEEE Transactions on Signal Processing from 2005 to 2008, and for the EURASIP Journal on Advances in Signal Processing from 2006 to 2009, and was a member of the IEEE-SPS Signal Processing Theory and Methods Technical Committee from 2007 to 2012. Since 2010 he is the chair of the São Paulo SPS Chapter. His research interests include signal processing theory and applications, robust and nonlinear estimation, and applied linear algebra.



**Magno T. M. Silva** was born in São Sebastião do Paraíso, Brazil. He received the B.S., M.S., and Ph.D. degrees, all in electrical engineering, from Escola Politécnica, University of São Paulo, São Paulo, Brazil, in 1999, 2001, and 2005, respectively. From February 2005 to July 2006, he was an Assistant Professor at Mackenzie Presbyterian University, São Paulo, Brazil. Since August 2006, he has been with the Department of Electronic Systems Engineering at Escola Politécnica, University of São Paulo, where he is currently an Assistant Professor. From January to July 2012, he worked as a Postdoctoral Researcher at Universidad Carlos III de Madrid, Leganés, Spain. His research interests include linear and non-linear adaptive filtering.



## CHAPTER 14

**Ambuj Tewari** is with the Department of Statistics, University of Michigan, Ann Arbor. He has served on senior program committees of the conferences on Algorithmic Learning Theory (ALT), Conference on Learning Theory (COLT), and Neural Information Processing Systems (NIPS). His work has received both the student paper award (2005) and the best paper award (2011) at COLT. He received his M.A. in Statistics (2005) and Ph.D. in Computer Science (2007) from the University of California at Berkeley where his advisor was Peter Bartlett. He was a research Assistant Professor in Toyota Technological Institute at Chicago (2008–2010), an Assistant Professor



(part-time) in the Department of Computer Science, University of Chicago (2008–2010), and a postdoctoral fellow in the Institute for Computational Engineering and Sciences, University of Texas at Austin (2010–2012). He has also been a Visiting Researcher at Microsoft Research, Redmond.

**Peter L. Bartlett** is with the Division of Computer Science and Department of Statistics, University of California at Berkeley, and with the School of Mathematical Sciences, Queensland University of Technology. He is the co-author of the book “Learning in Neural Networks: Theoretical Foundations.” He has served as Associate Editor of the journals Machine Learning, Mathematics of Control Signals and Systems, the Journal of Machine Learning Research, the Journal of Artificial Intelligence Research, and the IEEE Transactions on Information Theory. He was awarded the Malcolm McIntosh Prize for Physical Scientist of the Year in Australia for his work in statistical learning theory. He was a Miller Institute Visiting Research Professor in Statistics and Computer Science at U.C. Berkeley in Fall 2001, and a Fellow, Senior Fellow, and Professor in the Research School of Information Sciences and Engineering at the Australian National University’s Institute for Advanced Studies (1993–2003).



## CHAPTER 15

**Barbara Hammer** received her Ph.D. in Computer Science in 1995 and her venia legendi in Computer Science in 2003, both from the University of Osnabrueck, Germany. From 2000 to 2004, she was a leader of the junior research group “Learning with Neural Methods on Structured Data” at University of Osnabrueck before accepting an offer as professor for Theoretical Computer Science at Clausthal University of Technology, Germany, in 2004. Since 2010, she is holding a professorship for Theoretical Computer Science for Cognitive Systems at the CITEC cluster of excellence at Bielefeld University, Germany. Several research stays have taken her to Italy, UK, India, France, the Netherlands, and the USA. Her areas of expertise include hybrid systems, self-organizing maps, clustering, and recurrent networks as well as applications in bioinformatics, industrial process monitoring, or cognitive science. She is leading the task force “Data Visualization and Data Analysis” of the IEEE CIS Technical Committee on Data Mining, and the Fachgruppe Neural Networks of the GI.



## CHAPTER 16

**John Shawe-Taylor** is a Professor at the Department of Computer Science, University College London (UK). His main research area is Statistical Learning Theory, but his contributions range from Neural Networks, to Machine Learning, to Graph Theory. He has published

over 150 research papers. He obtained a Ph.D. in Mathematics at Royal Holloway, University of London in 1986. He subsequently completed an M.Sc. in the Foundations of Advanced Information Technology at Imperial College. He was promoted to Professor of Computing Science in 1996. He moved to the University of Southampton in 2003 to lead the ISIS research group. He was appointed the Director of the Center for Computational Statistics and Machine Learning at University College, London in July 2006. He has coordinated a number of Europeanwide projects investigating the theory and practice of Machine Learning, including the NeuroCOLT projects. He is currently the scientific coordinator of a Framework VI Network of Excellence in Pattern Analysis, Statistical Modeling and Computational Learning (PASCAL) involving 57 partners.



**Shiliang Sun** received the B.E. degree in automatic control from the Department of Automatic Control, Beijing University of Aeronautics and Astronautics in 2002, and the Ph.D. degree in pattern recognition and intelligent systems from the State Key Laboratory of Intelligent Technology and Systems, Department of Automation, Tsinghua University, Beijing, China, in 2007. In 2004, he was entitled Microsoft Fellow. Currently, he is a Professor at the Department of Computer Science and Technology and the Founding Director of the Pattern Recognition and Machine Learning Research Group, East China Normal University. From 2009 to 2010, he was a Visiting Researcher at the Department of Computer Science, University College London, working within the Center for Computational Statistics and Machine Learning. He is a member of the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) network of excellence, and on the editorial boards of multiple international journals. His research interests include machine learning, pattern recognition, computer vision, natural language processing, and intelligent transportation systems.



## CHAPTER 17

**Konstantinos Slavakis** received the M.E. and Ph.D. degrees in electrical and electronic engineering from Tokyo Institute of Technology (TokyoTech), Tokyo, Japan, in 1999 and 2002, respectively. For the period from 2004 to 2006, he was with TokyoTech as a JSPS PostDoc, and from 2006 to 2007, he was a Postdoc in the Department of Informatics and Telecommunications, University of Athens, Greece. From 2007 to 2012, he served as an Assistant Professor at the Department of Informatics and Telecommunications, University of Peloponnese, Tripolis, Greece. Currently, he is a Research Associate at the University of Minnesota, Digital Technology Center.



He serves as an Associate and Area Editor of the IEEE Transactions on Signal Processing. His research interests include applications of convex analysis and computational algebraic geometry to signal processing, machine learning, array, and multidimensional systems problems.

**Pantelis Bouboulis** received the M.Sc. and Ph.D. degrees in informatics and telecommunications from the National and Kapodistrian University of Athens, Greece, in 2002 and 2006, respectively. From 2007 till 2008, he served as an Assistant Professor in the Department of Informatics and Telecommunications, University of Athens. Since 2008, he teaches mathematics in Greek High Schools. His current research interests lie in the areas of machine learning, fractals, wavelets, and image processing.



## CHAPTER 18

**Franz Pernkopf** received his M.Sc. (Dipl. Ing.) degree in Electrical Engineering at Graz University of Technology, Austria, in summer 1999. He earned a Ph.D. degree from the University of Leoben, Austria, in 2002. In 2002 he was awarded the Erwin Schrödinger Fellowship. He was a Research Associate in the Department of Electrical Engineering at the University of Washington, Seattle, from 2004 to 2006. Currently, he is Associate Professor at the Laboratory of Signal Processing and Speech Communication, Graz University of Technology, Austria. His research interests include machine learning, discriminative learning, graphical models, feature selection, finite mixture models, and image- and speech processing applications.



**Robert Peharz** received his M.Sc. degree in Telematics at Graz University of Technology (TUG) in 2010. He currently pursues his Ph.D. studies at the SPSC Lab, TUG. His research interests include probabilistic graphical models, sparse coding, nonnegative matrix factorization, and machine learning in general, with applications to signal processing, audio engineering, and computer vision.



**Sebastian Tschiatschek** received the B.Sc. degree and M.Sc. degree in Electrical Engineering at Graz University of Technology (TUG) in 2007 and 2010, respectively. He conducted his Master thesis during a one-year stay at ETH Zürich, Switzerland. Currently, he is with the Signal Processing and Speech Communication Laboratory at TUG where he is pursuing the Ph.D. degree. His research interests include Bayesian networks, information theory in conjunction with graphical models and statistical pattern recognition.



## CHAPTER 19

**A. Taylan Cemgil** (M'04) received his Ph.D. (2004) from SNN, Radboud University Nijmegen, the Netherlands. Between 2004 and 2008 he worked as a postdoctoral researcher at Amsterdam University and the Signal Processing and Communications Laboratory, University of Cambridge, UK. He is currently an Associate Professor of Computer Engineering at Bogazici University, Istanbul, Turkey. He is a member of the IEEE MLSP Technical Committee and an Associate Editor of IEEE Signal Processing Letters and Digital Signal Processing. His research interests are in Bayesian statistical methods and inference, machine learning, and audio signal processing.



## CHAPTER 20

**Dao Lam** is a Ph.D. Candidate at Missouri University of Science and Technology, Rolla, MO. He received the B.S. degree from Post and Telecommunications Institute of Technology, Ho Chi Minh, Vietnam in 2003. He got his M.S. from Waseda University, Japan in 2008. His research interests are image processing, robotics, supervised and unsupervised learning, and computational intelligence.



**Donald Wunsch** is the Mary K. Finley Missouri Distinguished Professor at Missouri University of Science & Technology (Missouri S&T). Earlier employers were: Texas Tech University, Boeing, Rockwell International, and International Laser Systems. His education includes: Executive MBA—Washington University in St. Louis, Ph.D., Electrical Engineering—University of Washington (Seattle), M.S., Applied Mathematics (same institution), B.S., Applied Mathematics—University of New Mexico, and Jesuit Core Honors Program, Seattle University. Key research contributions are: Clustering; Adaptive Resonance and Reinforcement Learning architectures, hardware and applications; Neurofuzzy regression; Traveling Salesman Problem heuristics; Robotic Swarms; and Bioinformatics. He is an IEEE Fellow and previous INNS President, INNS Fellow and Senior Fellow 07—present, and served as IJCNN General Chair, and on several Boards, including the St. Patrick's School Board, IEEE Neural Networks Council, International Neural Networks Society, and the University of Missouri Bioinformatics Consortium. He has produced 16 Ph.D. recipients in Computer Engineering, Electrical Engineering, and Computer Science; has attracted over \$8 million in sponsored research; and has over 300 publications including nine books. His research has been cited over 6000 times.



## CHAPTER 21

**Andrzej Cichocki** received the M.Sc. (with honors), Ph.D. and Dr.Sc. (Habilitation) degrees, all in electrical engineering, from *Warsaw University of Technology* in Poland.

Since 1972, he has been with the *Institute of Theory of Electrical Engineering, Measurement and Information Systems, Faculty of Electrical Engineering* at the *Warsaw University of Technology*, where he obtained a title of a full Professor in 1995.

He spent several years at *University Erlangen-Nuerenberg* in Germany, at the Chair of Applied and Theoretical Electrical Engineering directed by Professor Rolf Unbehauen, as an Alexander-von-Humboldt Research Fellow and Guest Professor.

In 1995–1997 he was a team leader of the laboratory for Artificial Brain Systems, at Frontier Research Program RIKEN (Japan), in the Brain Information Processing Group.

He is currently senior team leader and the head of the *Cichocki Laboratory for Advanced Brain Signal Processing*, at *RIKEN Brain Science Institute* in Japan.



## CHAPTER 22

**Xueyuan Zhou** received his Ph.D. degree in computer science from the University of Chicago in 2011. His research interests include statistical machine learning theory and application in non-linear high dimensional data.

**Mikhail Belkin** is an Associate Professor in the Department of Computer Science and Engineering at the Ohio State University. He received his Ph.D. degree in mathematics from the University of Chicago in 2003. His research interests include a range of theoretical questions concerning the computational and statistical limits of learning and mathematical foundations of learning structure from data.

**Yannis Kopsinis** received the B.Sc. degree from the Department of Informatics and Telecommunications, University of Athens, Greece, in 1998 and his Ph.D. degree in 2003 from the same department. From January 2004 to December 2005 he has been a research fellow with the Institute for Digital Communications, School of Engineering and Electronics, the University of Edinburgh, UK. From January 2006 to September 2009 he was a senior researcher in the same department. From January 2012 to April 2013 he was a Ramon Y Cajal Fellow in the School of Applied Physics, University of Granada, Spain. He currently holds a Marie Curie Intra-European fellowship



on sparse online learning. His current research interests include adaptive signal processing, time-frequency analysis, and compressed sensing.

**Konstantinos Slavakis** received the M.E. and Ph.D. degrees in electrical and electronic engineering from Tokyo Institute of Technology (TokyoTech), Tokyo, Japan, in 1999 and 2002, respectively. For the period of 2004–2006, he was with TokyoTech as a JSPS PostDoc, and from 2006 to 2007, he was a Postdoc in the Department of Informatics and Telecommunications, University of Athens, Greece. From 2007 to 2012, he served as an Assistant Professor at the Department of Informatics and Telecommunications, University of Peloponnese, Tripolis, Greece. Currently, he is a Research Associate at the University of Minnesota, Digital Technology Center.



He serves as an Associate and Area Editor of the IEEE Transactions on Signal Processing. His research interests include applications of convex analysis and computational algebraic geometry to signal processing, machine learning, array, and multidimensional systems problems.

## CHAPTER 24

**José C. Principe** is currently a Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, Gainesville, USA. He is BellSouth Professor and Founder and Director of the University of Florida Computational Neuro-Engineering Laboratory (CNEL). He is an IEEE fellow and AIMBE fellow, and is the past Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, past President of the International Neural Network Society, former Secretary of the Technical Committee on Neural Networks of the IEEE Signal Processing Society, and a former member of the Scientific Board of the Food and Drug Administration. He is involved in biomedical signal processing, in particular, the electroencephalogram (EEG) and the modeling and applications of adaptive systems.



**Badong Chen** received his Ph.D. degree in Computer Science and Technology from Tsinghua University, Beijing, China, in 2008. He was a Postdoctoral Associate at the University of Florida Computational NeuroEngineering Laboratory (CNEL) during the period October, 2010 to September, 2012. He is currently a Professor at the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China. His research interests are in statistical signal processing, information theoretic learning, online kernel learning, and their applications in cognition and neuroscience.



## I Authors Biography

**Luis G. Sanchez Giraldo** was born in 1983 in Manizales, Colombia. He received the B.S. in electronics engineering and M.Eng. in industrial automation from Universidad Nacional de Colombia in 2005 and 2008, respectively, and his Ph.D. in electrical and computer engineering from University of Florida in 2012. Between 2004 and 2008, he was appointed as a research assistant at the Control and Digital Signal Processing Group (GCPDS) at Universidad Nacional de Colombia. During his Ph.D. studies he worked as a research assistant at the Computational Neuro-Engineering Laboratory (CNEL) at University of Florida. His main research interests are in machine learning and signal processing.



## CHAPTER 25

**Enes Makalic** was born in 1980. He received the Bachelor of Computer Science (Honors) degree in 2002 and the Ph.D. degree in 2007, both from Monash University, Australia. His research interests include information theoretic model selection using Minimum Message Length (MML) and Minimum Description Length (MDL) theories of statistical inference. He currently holds a Postdoctoral position with The University of Melbourne, Australia.

**Daniel F. Schmidt** was born in 1980. He received the Bachelor of Digital Systems (Honors) degree in 2002 and the Ph.D. degree in 2008, both from Monash University, Australia. His research interests are primarily information theoretic approaches to statistical inference, model selection, and estimation. He currently holds a Postdoctoral position with The University of Melbourne, Australia.

**Abd-Krim Seghouane** received his Ph.D. degree from University of Paris sud (Paris XI) in 2003. His research interests are within statistical signal and image processing. He is currently a Senior Research Fellow within the department of Electrical and Electronic Engineering, The University of Melbourne.

## CHAPTER 26

**George Tzanetakis** is an Associate Professor and Canada Research Chair in Computer Analysis of Audio and Music at the Department of Computer Science with cross-listed appointments in ECE and Music at the University of Victoria, Canada. In 2011 he was a visiting scientist at Google Research. He received his Ph.D. in Computer Science at Princeton University in 2002 and was a Post-Doctoral fellow at Carnegie Mellon University in 2002–2003. His research spans all stages of audio content analysis such as feature extraction, segmentation, classification with specific emphasis on music information retrieval. He is also the primary designer and developer of Marsyas an open



source framework for audio processing with specific emphasis on music information retrieval applications. His pioneering work on musical genre classification received a IEEE signal processing society young author award and is frequently cited. More recently he has been exploring new interfaces for musical expression, music robotics, computational ethnomusicology, and computer-assisted music instrument tutoring. These interdisciplinary activities combine ideas from signal processing, perception, machine learning, sensors, actuators, and human-computer interaction with the connecting theme of making computers better understand music to create more effective interactions with musicians and listeners.

# Introduction to Signal Processing Theory

# 1

**Isabela F. Apolinário and Paulo S.R. Diniz**

*Program of Electrical Engineering and the Department of Electronics and Computer Engineering, COPPE/Poli,  
Universidade Federal do Rio de Janeiro, Brazil*

---

## 1.01.1 Introduction

Signal processing is a key area of knowledge that finds applications in virtually all aspects of modern life. Indeed the human beings are employing signal processing tools for centuries without realizing it [1]. In present days the younger generation might not be able to understand how one can live without carrying a mobile phone, traveling long distances without an almost self piloted airplane, exploring other planets without human presence, and utilizing a medical facility without a wide range of diagnostic and intervention equipments.

Signal processing consists of mapping or transforming information bearing signals into another form of signals at the output, aiming at some application benefits. This mapping defines a continuous or analog system if it involves functions representing the input and output signals. On the other hand, the system is discrete or digital if its input and output signals are represented by sequences of numbers.

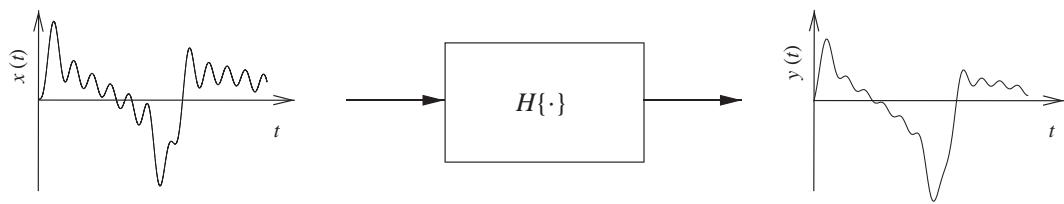
Signal processing theory is very rich and as a reward it has been finding applications in uncountable areas, among which we can mention bioengineering, communications, control, surveillance, environment monitoring, oceanography, and astronomy, just to mention a few. In particular, the enormous advance in digital integrated circuit technology, responsible for the computing and information revolutions that we are so used today, enabled the widespread use of Digital Signal Processing Systems. These systems allowed advances in fields such as: speech, audio, image, video, multirate processing, besides in several applications of wireless communications and digital control.

This chapter is an overview of the classical signal processing tools whose details are encountered in numerous textbooks such as: [1–15]. The topics treated in the following chapters entail the description of many signal processing tools, usually not covered in the classical textbooks available in the market. As a result, we believe that the readers can benefit from reading many of these chapters, if not all, in order to deepen and widen their current knowledge as well as to start exploiting new ideas.

---

## 1.01.2 Continuous-time signals and systems

The processing of signals starts by accessing the type of information we want to deal with. Many signals originating from nature are continuous in time and as such, the processing involved must include analog signal acquisition systems followed by the implementation of a continuous-time system if the processing

**FIGURE 1.1**

Continuous-time system.

remains analog all the way. Alternatively, assuming the original continuous-time signal contains limited spectral information, we can sample it in order to generate a sequence representing the continuous-time signal unambiguously<sup>1</sup>. In this latter form one can benefit from the advanced digital integrated circuit technology to process the signal. However, many real life applications still includes continuous-time signal processing at least at the acquisition and actuator phases of the processing.

A continuous-time system maps an analog input signal represented by  $x(t)$  into an output signal represented by  $y(t)$ , as depicted in Figure 1.1. A general representation of this mapping is

$$y(t) = \mathcal{H}\{x(t)\},$$

where  $\mathcal{H}\{\cdot\}$  denotes the operation performed by the continuous-time system. If the system is linear and time-invariant (LTI), as will be described in Chapter 2, there are several tools to describe the mapping features of the system. Typically, a general description of the systems consists of a differential equation which in turn can be solved and analyzed by employing the Laplace transform. A key feature of the LTI systems is their full representation through their impulse responses. The behavior of a nonlinear system is more complicated to analyze since it can not be fully characterized by its impulse response.

Another important tools to deal with the continuous-time signals which are periodic and non-periodic are the Fourier series and Fourier transform, respectively. As will be discussed in Chapter 2, the Fourier and Laplace transforms are closely related, being both essential tools to describe the behavior of continuous-time signals when applied to LTI systems.

The Laplace transform is suitable to represent a time-domain non-periodic function of a continuous and real (time) variable, resulting in a frequency-domain non-periodic function of a continuous and complex frequency variable. That is,

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st} dt \iff x(t) = \frac{1}{2\pi} e^{\sigma t} \int_{-\infty}^{\infty} X(\sigma + j\omega)e^{j\omega t} d\omega.$$

The Fourier transform is employed to represent a time-domain non-periodic function of a continuous and real (time) variable with a frequency-domain non-periodic function of a continuous and imaginary

<sup>1</sup>As will be seen in Chapter 5, we can completely reconstruct a continuous-time band-limited signal  $x(t)$  from its sampled version  $x(n)$  if the sampling frequency is chosen correctly.

frequency variable. The Fourier transform is described by

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt \iff x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} d\Omega.$$

The representation of a time-domain periodic function of a continuous and real (time) variable is performed by the Fourier series. In the frequency-domain, the representation consists of a non-periodic function of a discrete and integer frequency variable, as following described:

$$X(k) = \frac{1}{T} \int_0^T x(t) e^{-j(2\pi/T)kt} dt \iff x(t) = \sum_{k=-\infty}^{\infty} X(k) e^{j(2\pi/T)kt}.$$

### 1.01.3 Discrete-time signals and systems

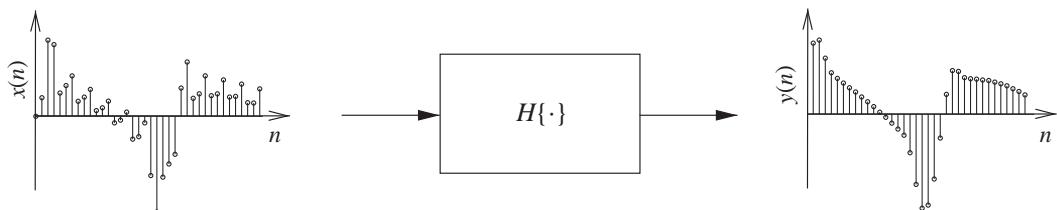
A discrete-time signal is represented by a sequence of numbers and can be denoted as  $x(n)$  with  $n \in Z$ , where  $Z$  is the set of integer numbers. The samples  $x(n)$  might represent numerically the amplitude of a continuous-time signal sample at every  $T$  seconds or can be originated from a discrete information. In many applications, the sampling period  $T$  represents the time interval between two acquired samples of the continuous-time signal. However, in other situations it might represent the distance between two sensors of two pixels of an image, or the separation between two antennas, just to mention a few examples.

Discrete-time systems map input sequences into output sequences. A general representation of this mapping is

$$y(n) = \mathcal{H}\{x(n)\},$$

where  $\mathcal{H}\{\cdot\}$  denotes the operation performed by the discrete-time system. Figure 1.2 depicts the input-to-output mapping of sequences. According to the properties of  $\mathcal{H}\{\cdot\}$ , the discrete-time system might be LTI. This way, it benefits from a number of analysis and design tools such as frequency-domain representations. However, there are many applications employing nonlinear, time-varying, and even non-causal systems [2,5,7,8,11,12,16].

If the system is linear and time-invariant, as will be described in Chapter 2, there are several tools to describe the mapping features of the system. Typically, a general description of discrete-time systems through a difference equation can be solved and analyzed by employing the  $z$  transform. Also a



**FIGURE 1.2**

Discrete-time signal representation.

discrete-time LTI system is fully described by its impulse response, whereas a nonlinear system can not be fully characterized by its impulse response.

The  $z$  transform is the key tool to represent a time-domain non-periodic function of a discrete and integer variable through a frequency-domain non-periodic function of a continuous and complex frequency variable. That is,

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \iff x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1} dz.$$

The discrete-time Fourier transform (DTFT) represents a time-domain non-periodic function of a discrete and integer variable by a periodic function of a continuous frequency variable in the frequency domain as follows:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \iff x(n) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(e^{j\omega})e^{j\omega n} d\omega.$$

Finally, the discrete Fourier transform (DFT) is the right tool to represent a time-domain periodic function of a discrete and integer variable through a periodic function of a discrete and integer frequency variable in the frequency domain. The DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn} \iff x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j(2\pi/N)kn}.$$

The DFT plays a key role in digital signal processing since it represents a finite-length sequence in the time domain through a finite-length sequence in the frequency domain. Since both domains utilize sequences, this feature makes the DFT a natural choice for time-frequency representation of information in a digital computer. A little mental exercise allows us to infer that the DFT is the perfect representation in the frequency domain of a finite-length sequence in the time domain, if we interpret the latter as a period of a periodic infinite-length sequence. In addition, by employing appropriate zero-padding in two finite-length sequences, the product of their DFTs represents the DFT of their linear convolution (see [Chapter 3](#)), turning the DFT a valuable tool for LTI system implementation. There is a plethora of applications for the DFT in signal processing and communications and some of them can be accessed in the references [2–13].

Often, whenever a DSP system is LTI, it can be described through a difference equation as follows:

$$\sum_{i=0}^N a_i y(n-i) + \sum_{l=0}^M b_l x(n-l) = 0.$$

The above description is suitable for implementation in digital computers. The difference equation represents a causal LTI system if its auxiliary conditions correspond to its initial conditions and those are zeros [2]. Assuming  $a_0 = 1$ , the above equation can be rewritten as

$$y(n) = - \sum_{i=1}^N a_i y(n-i) + \sum_{l=0}^M b_l x(n-l).$$

This class of system has infinite impulse response (i.e.,  $y(n) \neq 0$  when  $n \rightarrow \infty$ ) and, as such, it is called IIR system or filter.

The nonrecursive system generates the output signal from past input samples, that is,

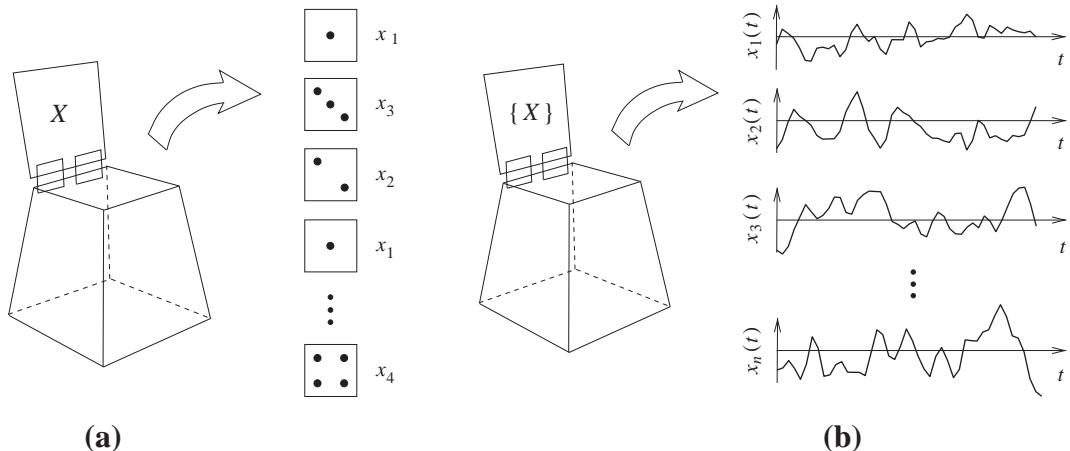
$$y(n) = \sum_{l=0}^M b_l x(n-l).$$

In this case, the resulting system has finite impulse response and is known as FIR systems or filters.

### 1.01.4 Random signals and stochastic processes

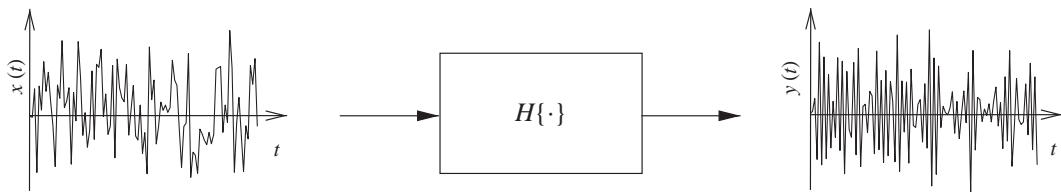
In most practical applications, we have to deal with signals that can not be described by a function of time, since their waveforms follow a random pattern [17,18]. Take, for example, the thermal noise inherent to any material. Despite the lack of exact knowledge of the signal values, it is possible to analyze and extract information the signals contain by employing the mathematical tools available to deal with random signals. Chapter 4 will present a compact and yet clarifying review of random signals and stochastic processes which are crucial to understand several concepts that will be discussed in the more advanced chapters.

The theory starts by defining a random variable as a mapping of the result of a random process experiment onto the set of numbers. A random variable  $X$  is a function that assigns a number  $x$  to every outcome of a random process experiment. An example is given in Figure 1.3(a), in which each outcome of a throw of a dice, numbers 1–6, corresponds to a determined value,  $x_1$ – $x_6$ , respectively.



**FIGURE 1.3**

Examples of random variable and stochastic process. (a) Random variable. (b) Stochastic process.

**FIGURE 1.4**

Filtering of a random signal.

The stochastic process is a rule to describe the time evolution of the random variable depending on the random process experiment, whereas the set of all experimental outcomes is its domain known as the ensemble. An example is given in Figure 1.3(b), in which, at any time instant  $t_0$ , the set formed by the output samples of the stochastic process  $\{X\}$ ,  $\{x_1(t_0), x_2(t_0), x_3(t_0), \dots, x_n(t_0)\}$ , represents a random variable. A single outcome  $x_i(t)$  of  $\{X\}$  is a random signal. Since random signals do not have a precise description of their waveforms, we have to characterize them either via measured statistics or through a probabilistic model. In general, the first- and second-order statistics (mean and variance, respectively) are sufficient for characterization of the stochastic process, particularly due to the fact that these statistics are suitable for measurements.

As an illustration, Figure 1.4 shows a random signal as an input of a highpass filter. We can observe that the output signal is still random, but clearly shows a faster changing behavior given that the low frequency contents of the input signal were attenuated by the highpass filter.

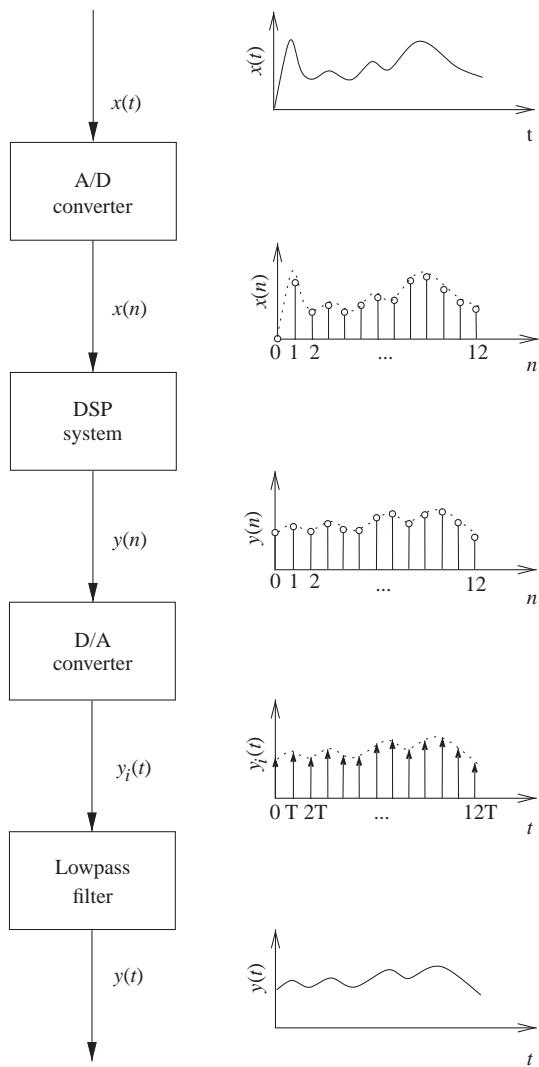
### 1.01.5 Sampling and quantization

A digital signal processing system whose signals originated from continuous-time sources includes several building blocks, namely: an analog-to-digital (A/D) converter; a digital signal processing (DSP) system; a digital-to-analog (D/A) converter; and a lowpass filter. Figure 1.5 illustrates a typical digital signal processing setup where:

- The A/D converter produces a set of samples in equally spaced time intervals, which may retain the information of the continuous-time signal in the case the latter is band limited. These samples are converted into a numeric representation, in order to be applied to the DSP system.
- The DSP performs the desired mapping between the input and output sequences.
- The D/A converter produces a set of equally spaced-in-time analog samples representing the DSP system output.
- The lowpass filter interpolates the analog samples to produce a smooth continuous-time signal.

[Chapter 5](#) will discuss in detail the conditions which a continuous-time signal must satisfy so that its sampled version retains the information of the original signal, dictated by the sampling theorem. This theorem determines that a band limited continuous-time signal can be theoretically recovered from its sampled version by filtering the sequence with an analog filter with prescribed frequency response.

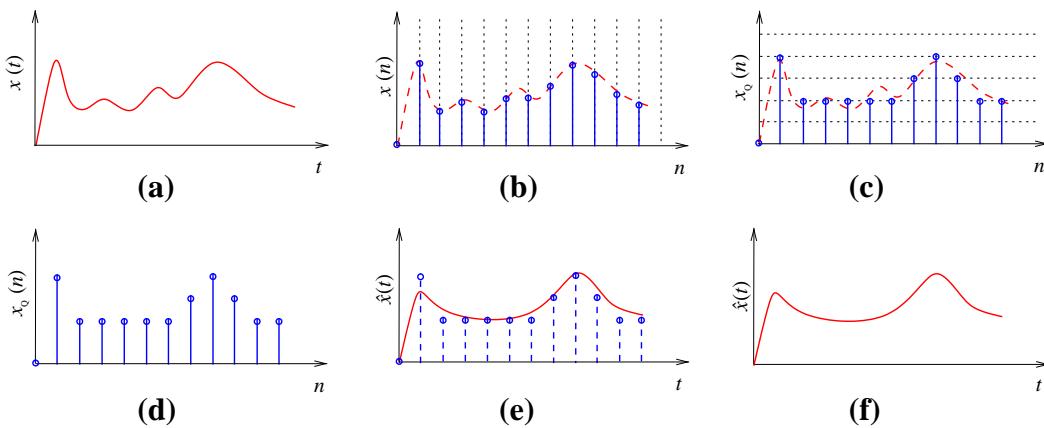
It is also important to mention that, while processing the signals in the digital domain, these are subject to quantization errors, such as: roundoff errors originated from internal arithmetic operations

**FIGURE 1.5**

DSP system.

performed in the signals; deviations in the filter response due to finite wordlength representation of the multiplier coefficients inherent to the signal processing operation; and errors due to representation of the acquired continuous-time signals with a set of discrete levels.

The actual implementation of a DSP system might rely on general purpose digital machines, where the user writes a computer software to implement the DPS tasks. This strategy allows fast prototyping and testing. Other mean is to use special-purpose commercially available CPUs, known as Digital Signal

**FIGURE 1.6**

A digital signal generation. (a) Original continuous-time signal. (b) A/D converter: sampling. (c) A/D converter: quantization. (d) Digital signal. (e) D/A converter.(f) Recovered continuous-time signal.

Processors, which are capable of implementing sum of product operations in a very efficient manner. Yet another approach is to employ special purpose hardware tailored for the given application.

Figure 1.6 depicts a continuous-time signal, its digitized version, and its recovered continuous-time representation. We can notice, from Figures 1.6(a) to 1.6(f), the effects of a low sampling frequency and a small number of quantization steps on the A/D and D/A processes.

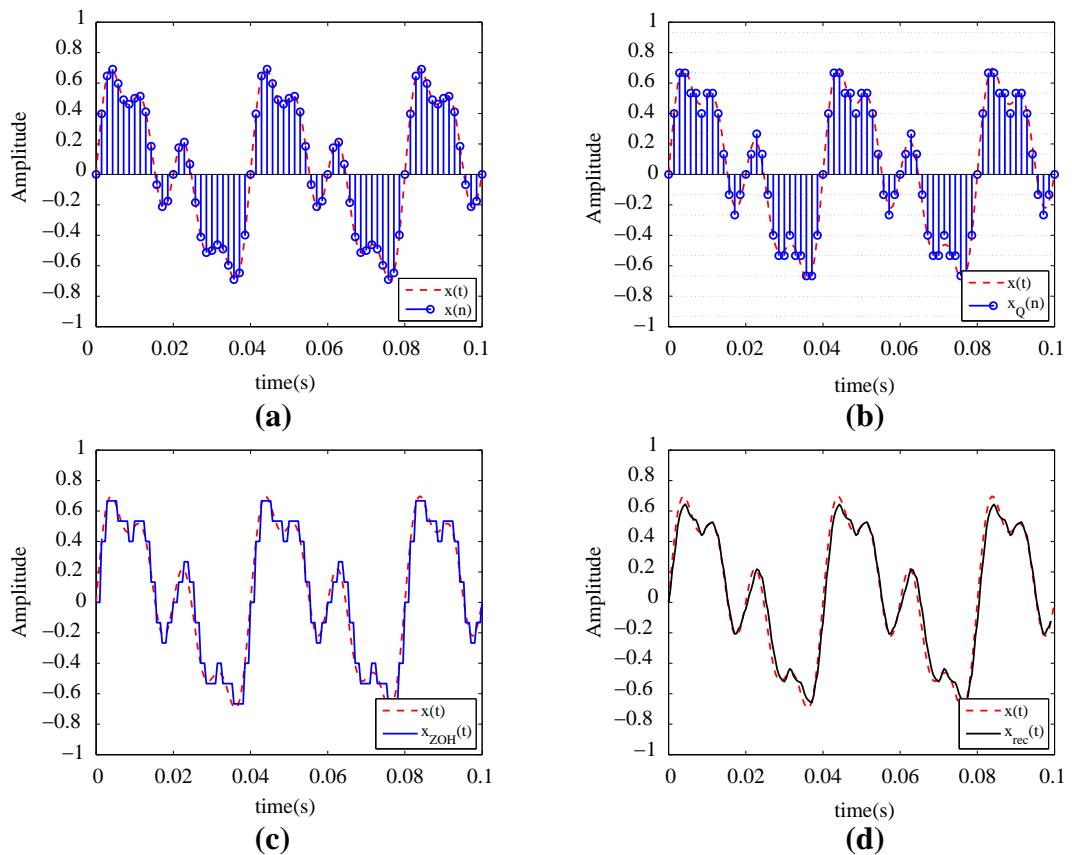
Figure 1.7 is a detailed example of the steps entailing an A/D and a D/A conversion, where in Figure 1.7(a) a continuous-time signal is depicted along with its equally spaced samples. These samples are then quantized as illustrated Figure 1.7(b). Assuming the quantized samples are converted into an analog signal through a zero-order hold, the output of the D/A converter becomes as illustrated in Figure 1.7(c). The sampled-and-held continuous-time signal is lowpass filtered in order to recover the original continuous-time signal. As can be observed in Figure 1.7(d), the recovered signal resembles the original where the difference originates from the quantization effects and the nonideal lowpass filter at the D/A converter output.

### 1.01.6 FIR and IIR filter design

The general form of an FIR transfer function is given by

$$H(z) = \sum_{l=0}^M b_l z^{-l} = H_0 z^{-M} \prod_{l=0}^M (z - z_l).$$

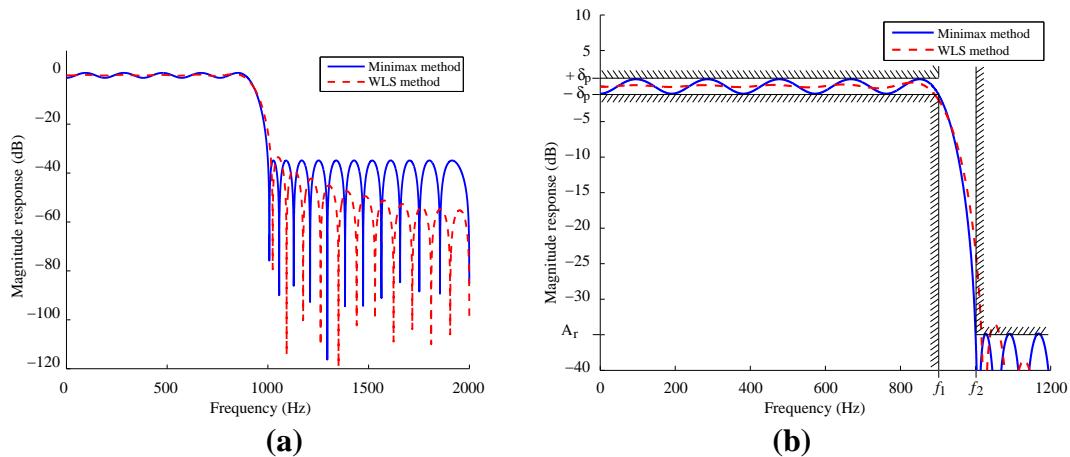
Since all the poles of the FIR transfer function are placed at  $z = 0$ , it is always stable. As a result, the transfer function above will always represent a stable filter. The main feature of the FIR filters is

**FIGURE 1.7**

A real digital signal generation. (a) A/D converter: sampling. (b) A/D converter: quantization. (c) D/A converter: zero-order hold. (d) D/A converter: lowpass filter.

the possibility of designing structurally induced linear-phase filters. On the other hand, an FIR filter requires a higher number of multiplications, additions and storage elements when compared to their IIR counterparts in order to satisfy a prescribed specification for the magnitude response.

There are many methods to design an FIR filter [19], ranging from the simple ones, such as the window and frequency sampling methods, to more efficient and sophisticated, that rely on some kind of optimization method [20–22]. The simple methods lead to solutions which are far from optimum in most practical applications, in the sense that they either require higher order or cannot satisfy prescribed specifications. A popular optimization solution is the minimax method based on the Remez exchange algorithm, which leads to transfer functions with minimum order to satisfy prescribed specifications.

**FIGURE 1.8**

Typical magnitude responses of FIR filters: Minimax and WLS. (a) Magnitude response. (b) Specifications for a lowpass filter using minimax design.

Nowadays, with the growing computational power, it is possible to design quite flexible FIR filters by utilizing weighted least squares (WLS) solutions. These filters are particularly suitable for multirate systems, since the resulting transfer functions can have decreasing energy in their stopband [2]. The WLS method enables the design of filters satisfying prescribed specifications, such as the maximum deviation in the passband and in part of the stopband, while minimizing the energy in a range of frequencies in the stopband. As can be observed in Figure 1.8, for a given filter order, the minimax design leads to lower maximum stopband attenuation, whereas the WLS solution leads to lower stopband energy. Note there are solutions in between where some maximum values of stopband ripples can be minimized while the energy of the remaining ones is also minimized. It is possible to observe that the WLS solution does not satisfy the prescribed specifications given that it has the same filter order as the minimax solution.

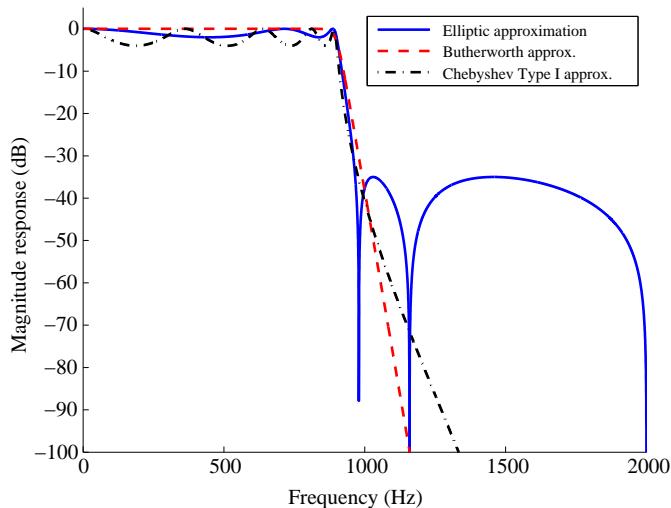
The typical transfer function of an IIR filter is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{l=0}^M b_l z^{-l}}{1 + \sum_{i=1}^N a_i z^{-i}},$$

which can be rewritten in a form explicitly showing the poles positions as

$$H(z) = H_0 \frac{\prod_{l=0}^M (1 - z^{-1} z_l)}{\prod_{i=0}^N (1 - z^{-1} p_i)} = H_0 z^{N-M} \frac{\prod_{l=0}^M (z - z_l)}{\prod_{i=0}^N (z - p_i)}.$$

IIR filters are usually designed by using well-established analog filter approximations, such as Butterworth, Chebyshev, and elliptic methods (Figure 1.9). The prototype analog transfer function is then

**FIGURE 1.9**

Typical magnitude response of an IIR filter.

transformed into a digital transfer function by using an adequate transformation method. The most widely used transformation methods are bilinear and the impulse invariance methods [2].

### 1.01.7 Digital filter structures and implementations

From a closer examination of the FIR and IIR transfer functions, one can infer that they can be realized with three basic operators: adder, multiplier and delay (represented by  $z^{-1}$ ).

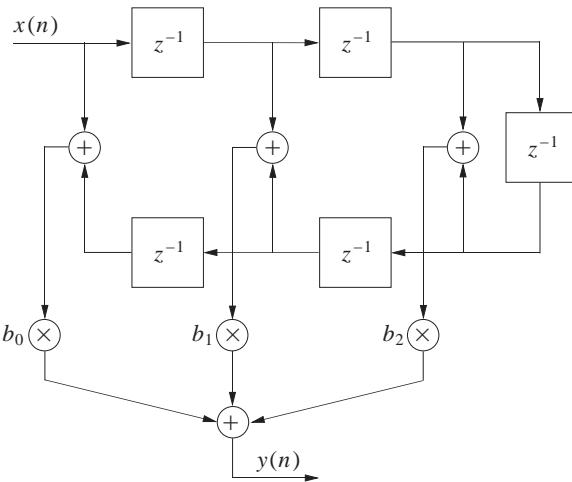
For example, Figure 1.10 shows a linear-phase FIR filter realization of odd order. Linear-phase filters have symmetric or anti-symmetric impulse response and the symmetry should be exploited in order to minimize the number of multipliers in the filter implementation, as illustrates Figure 1.10.

Figure 1.11 depicts a possible direct-form realization of an IIR transfer function. This realization requires the minimum number of multiplications, adders and delays to implement the desired IIR transfer function.

There are many alternative structures to implement the FIR and IIR filters. For example, IIR transfer functions can be implemented as a product or as a summation of lower order IIR structures by starting with their description as:

$$\begin{aligned} H(z) &= \prod_{k=1}^m \frac{\gamma_{0k} + \gamma_{1k}z^{-1} + \gamma_{2k}z^{-2}}{1 + m_{1k}z^{-1} + m_{2k}z^{-2}} \\ &= h_0^p + \sum_{k=1}^m \frac{\gamma_{1k}^p z + \gamma_{2k}^p}{z^2 + m_{1k}z + m_{2k}}, \end{aligned}$$

respectively. The right choice for a filter realization must take into consideration some issues, such as: modularity, quantization effects, design simplicity, power consumption, etc. [2]. Chapter 6 will address

**FIGURE 1.10**

Odd-order linear-phase FIR filter structure with  $M = 5$ .

advanced methods for FIR and IIR filter realization and implementation, where elegant and creative solutions are introduced in a clear manner. This chapter will also discuss strategies to implement digital filters efficiently.

### 1.01.8 Multirate signal processing

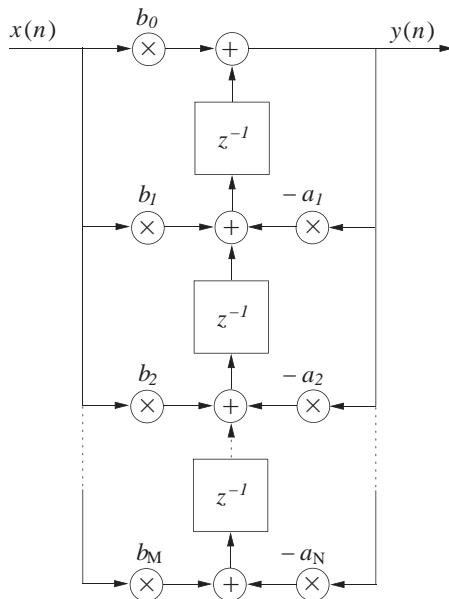
In digital signal processing it is easy to change the sampling rate of the underlying sequences by a ratio of integer values. This feature is highly desirable whenever we want to merge information of systems employing distinct sampling rates. In these cases, those rates should be made compatible [6,9,10].

Systems that internally employ multiple sampling rates are collectively referred to as multirate systems. In most cases, these systems are time-varying or, at most, periodically time-invariant.

The basic operations of the multirate systems are the decimation and interpolation. The right composition of these operations allows arbitrary rational sampling rate changes. If this sampling rate change is arbitrary and non-rational, the only solution is to recover the bandlimited continuous-time signal  $x(t)$  from its samples  $x(m)$ , and then re-sample it with a different sampling rate, thus generating a distinct discrete-time signal  $\bar{x}(n)$ .

If we assume that a discrete-time signal  $x(m)$  was generated from a continuous-time signal  $y(t)$  with sampling period  $T_1$ , so that  $x(m) = y(mT_1)$ , with  $m = \dots, 0, 1, 2, \dots$ , the sampling theorem requires that  $y(t)$  should be bandlimited to the range  $[-\frac{\pi}{T_1}, \frac{\pi}{T_1}]$ . The sampled continuous-time signal is given by:

$$y'(t) = \sum_{m=-\infty}^{\infty} x(m)\delta(t - mT_1).$$

**FIGURE 1.11**

Direct-Form IIR filter structure, for  $M = N$ .

The spectrum of the sampled signal is periodic with period  $\frac{2\pi}{T_1}$ . The original continuous-time signal  $y(t)$  can be recovered from  $y'(t)$  through an ideal lowpass filter. This interpolation filter, whose impulse response is denoted as  $h(t)$ , has ideal frequency response  $H(e^{j\omega})$  as follows:

$$H(e^{j\omega}) = \begin{cases} 1, & \omega \in [-\frac{\pi}{T_1}, \frac{\pi}{T_1}] \\ 0, & \text{otherwise,} \end{cases}$$

so that

$$y(t) = y'(t) * h(t) = \frac{1}{T_1} \sum_{m=-\infty}^{\infty} x(m) \operatorname{sinc} \frac{\pi}{T_1} (t - mT_1).$$

By re-sampling  $y(t)$  with period  $T_2$ , we obtain the discrete-time signal in the new desired sampling rate as follows:

$$\bar{x}(n) = \frac{1}{T_1} \sum_{m=-\infty}^{\infty} x(m) \operatorname{sinc} \frac{\pi}{T_1} (nT_2 - mT_1),$$

where  $\bar{x}(n) = y(nT_2)$ , with  $n = \dots, 0, 1, 2 \dots$

In this general equation governing sampling rate changes, there is no restriction on the values of  $T_1$  and  $T_2$ . However, if  $T_2 > T_1$  and aliasing is to be avoided, the interpolation filter must have a zero gain for  $\omega \notin [-\frac{\pi}{T_2}, \frac{\pi}{T_2}]$ .

In the case the sampling rate change corresponds to a ratio of integer numbers, all we need is simple decimation and interpolation operations. The decimation operation, or down-sampling, consists of retaining every  $M$ th sample of a given sequence  $x(m)$ . Since we are disposing samples from the original sequence, either the signal is sufficiently limited in band or the signal has to be filtered by a lowpass filter so that the decimated signal retains the useful information of the original signal. Decimation is a time-varying operation, since, if  $x(m)$  is shifted by  $m_0$ , the output signal will, in general, differ from the unshifted output shifted by  $m_0$ . Indeed, decimation is a periodically time-invariant operation, which consist of an extra degree of freedom with respect to the traditional time-invariant systems.

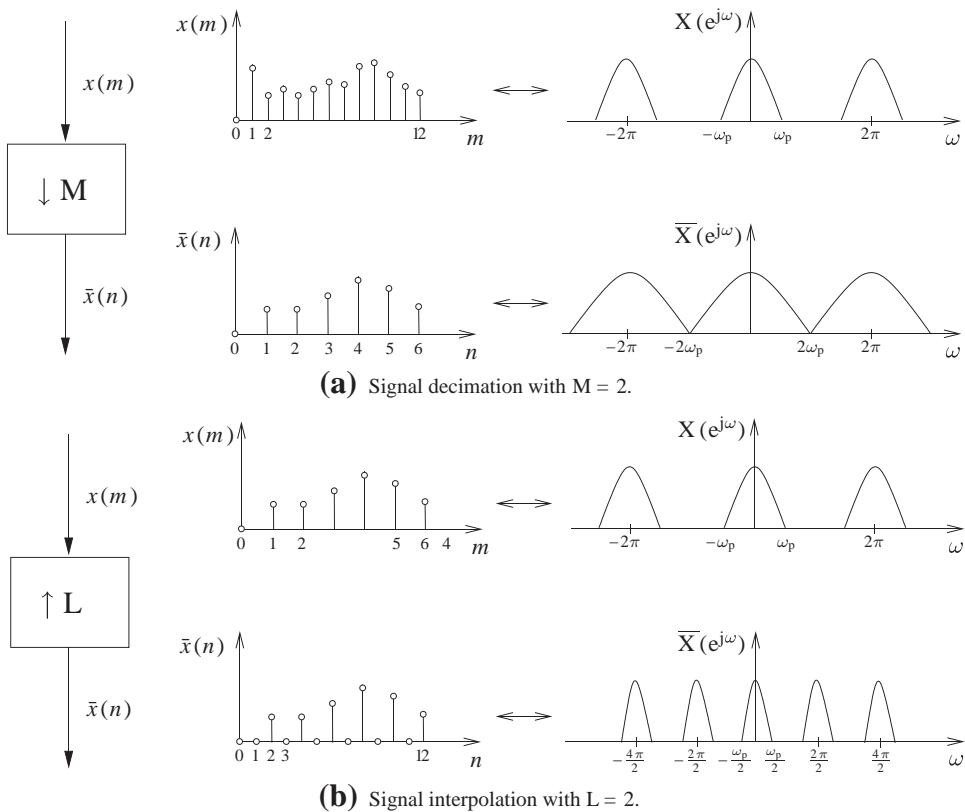
The interpolation, or up-sampling, of a discrete-time signal  $x(m)$  by a factor of  $L$  consists of inserting  $L - 1$  zeros between each of its samples. In the frequency domain, the spectrum of the up-sampled signal is periodically repeated. Given that the spectrum of the original discrete-time signal is periodic with period  $2\pi$ , the interpolated signal will have period  $\frac{2\pi}{L}$ . Therefore, in order to obtain a smooth interpolated version of  $x(m)$ , the spectrum of the interpolated signal must have the same shape of the spectrum of  $x(m)$ . This can be obtained by filtering out the repetitions of the spectra beyond  $[-\frac{\pi}{L}, \frac{\pi}{L}]$ . Thus, the up-sampling operation is generally followed by a lowpass filter. The interpolation is only periodically time-invariant and does not entail any loss of information.

Figure 1.12 illustrates discrete-time signal decimation and interpolation operations. As can be observed in Figure 1.12(a), the decimation factor of two widens the spectrum of the sampled signal in comparison to the new sampling rate. Figure 1.12(b) depicts the effect of increasing the sampling rate of a given sequence by two, where it can be seen that the frequency contents of the original signal repeats twice as often and appears narrower in the new sampling rate.

Chapter 7 will further discuss the theory and practice of multirate signal processing. This chapter will show how sophisticated real-life signal processing problems can be addressed starting from the basic theory. In particular, the authors address the design of flexible communications systems incorporating several advanced signal processing tools [4, 23–25].

### 1.01.9 Filter banks and wavelets

In many applications it is desirable to decompose a wideband discrete-time signal into several non-overlapping narrowband subsignals in order to be transmitted, quantized, or stored. In this case, each narrow-band channel can have its sampling rate reduced, since the subband signals have lower bandwidth than their originator signal. The signal processing tool that performs these tasks is the analysis filter bank. The analysis filters, represented by the transfer functions  $H_i(z)$ , for  $i = 0, 1, \dots, M - 1$ , consist of a lowpass filter  $H_0(z)$ , bandpass filters  $H_i(z)$ , for  $i = 1, 2, \dots, M - 2$ , and a highpass filter  $H_{M-1}(z)$ . Ideally, these filters have non-overlapping passbands, whereas their spectrum combination should cover the overall spectrum of the input signal. The outputs of the analysis filters, denoted as  $x_i(n)$ , for  $i = 0, 1, \dots, M - 1$ , have together  $M$  times the number of samples of the original signal  $x(n)$ . However, since each subband signal occupies a spectrum band  $M$  times narrower than the input signal, they can be decimated so the number of samples in the subbands does not increase. Indeed, if the input signal is uniformly split into subbands, we can decimate each  $x_i(n)$  by a factor of  $L$  smaller or equal to  $M$  without generating unrecoverable aliasing effects. Figure 1.13 shows the general configuration of a maximally (or critically) decimated analysis filter, where the decimation factor is equal to the number of subbands, i.e.,  $L = M$ .

**FIGURE 1.12**

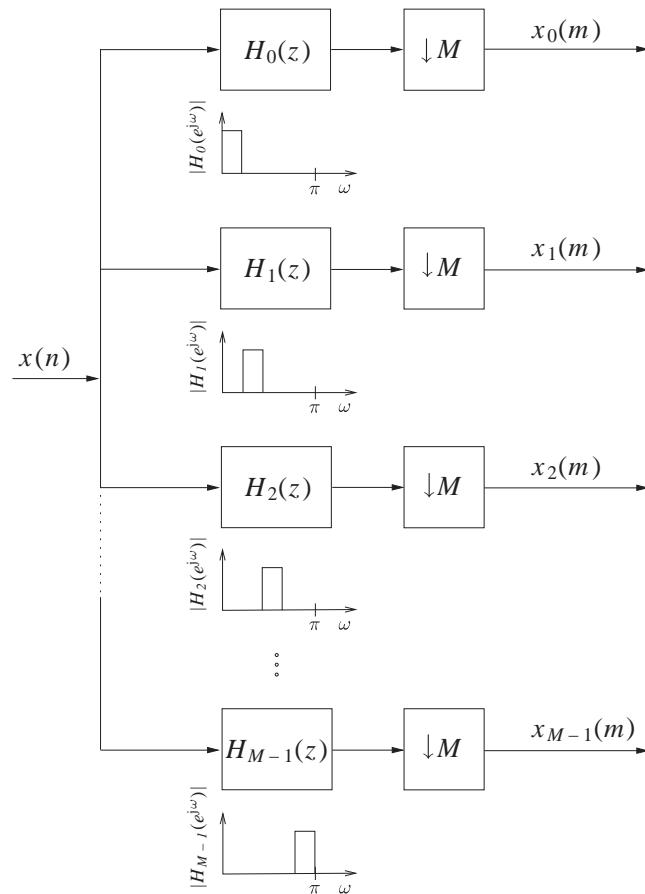
Signal decimation and interpolation. (a) Signal decimation with  $M = 2$ . (b) Signal interpolation with  $L = 2$ .

Whenever the input signal is split in subbands and decimated, if  $L \leq M$ , it is always possible to recover the input signal by properly designing the analysis filters in conjunction with the synthesis filters  $G_i(z)$ , for  $i = 0, 1, \dots, M - 1$ . The synthesis filters are placed after interpolators and they have the task of smoothing the up-sampled signals. Figure 1.14 illustrates the general configuration of the synthesis filter bank. A key feature of the synthesis filter bank is to cancel out or reduce the aliasing effects.

If the signals in the subbands are not modified, the filter bank output  $y(n)$  can be a delayed copy signal of  $x(n)$ . The cascade connection of the analysis and synthesis filter banks satisfying this condition is called a perfect reconstruction filter bank.

The cascade of an  $M$ -channel synthesis filter bank with an  $M$ -channel analysis filter bank gives rise to a transmultiplex system. The transmultiplex model is widely used in communications to model multiuser, multicarrier, and multiple access systems [4,23,24]. Chapter 7 will utilize multirate signal processing and transmultiplexers to discuss a design procedure to be applied in cognitive radio.

Chapter 8 will present several methods for designing the analysis filters  $H_i(z)$  and the synthesis filters  $G_i(z)$ , such that perfect reconstruction, as well as other features, are met. This chapter will also discuss

**FIGURE 1.13**

Analysis Filter Bank.

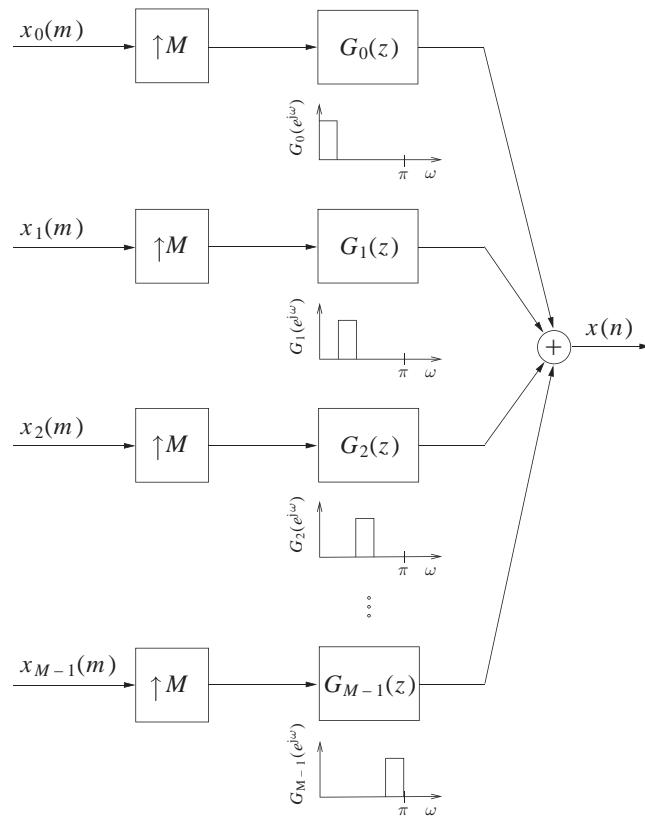
the design of filter banks with the subbands divided in octave bands, giving rise to the discrete-time wavelet series.

Figure 1.15 shows a four-band filter bank design and its effect in a composed signal.

Figure 1.16 illustrates how a four-band synthesis filter bank performs the recomposition of the input signal starting from the decimated subband signals.

### 1.01.10 Discrete multiscale and transforms

Discrete-time signals and systems can be characterized in the frequency domain by their Fourier transforms. One of the main advantages of discrete-time signals is that they can be processed and represented

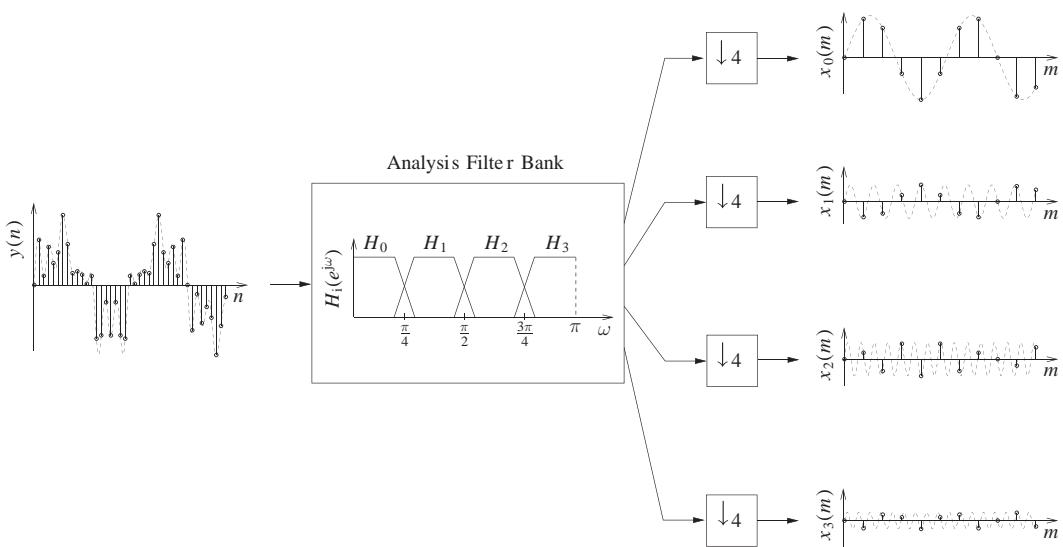
**FIGURE 1.14**

Synthesis Filter Bank.

in digital computers. However, when we examine the definition of the discrete-time Fourier transform,

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n},$$

we notice that such a characterization in the frequency domain depends on the continuous variable  $\omega$ . This implies that the Fourier transform, as it is, is not suitable for the processing of discrete-time signals in digital computers. We need a transform depending on a discrete-frequency variable that, if possible, preserves the handy interpretations of the Fourier-based tools, which retains important information. This can be obtained from the Fourier transform itself in a very simple way, by sampling uniformly the continuous-frequency variable  $\omega$  as long as the input sequence has finite length, otherwise the time-domain signal will be affected by aliasing. Using this strategy it is possible to obtain a mapping of a signal depending on a discrete-time variable  $n$  to a transform depending on a discrete-frequency

**FIGURE 1.15**

Analysis filter bank design.

variable  $k$ , leading to another interpretation for the DFT. Unfortunately, the DFT has its limitations in representing more general class of signals as will be following discussed.

The wavelet transform of a function belonging to  $\mathcal{L}^2(\mathbb{R})$ , the space of the square integrable functions, is the function decomposition in a basis composed by expansions, compressions, and translations of a single mother function  $\psi(t)$ , called wavelet.

The wavelet transform can be defined as

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n} \overline{\psi}_{m,n}(t)$$

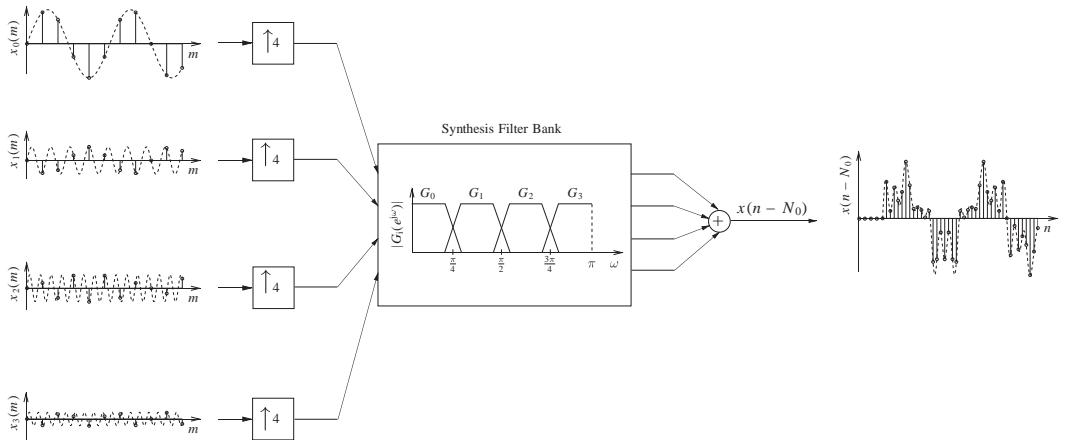
$$c_{m,n} = \int_{-\infty}^{\infty} x(t) \psi_{m,n}^*(t) dt,$$

where

$$\psi_{m,n}(t) = 2^{-m/2} \psi(2^{-m}t - n)$$

$$\overline{\psi}_{m,n}(t) = 2^{-m/2} \overline{\psi}(2^{-m}t - n).$$

This pair of equations defines a biorthogonal wavelet transform which are characterized by two wavelets: the analysis wavelet,  $\psi(t)$ , and the synthesis wavelet,  $\overline{\psi}(t)$ . Any function  $x(t) \in \mathcal{L}^2(\mathbb{R})$  can be

**FIGURE 1.16**

Synthesis filter bank design.

decomposed as a linear combination of contractions, expansions, and translations of the synthesis wavelet  $\bar{\psi}(t)$ . The weights of the expansion can be computed via the inner product of  $x(t)$  with expansions, contractions, and translations of the analysis wavelet  $\psi(t)$ .

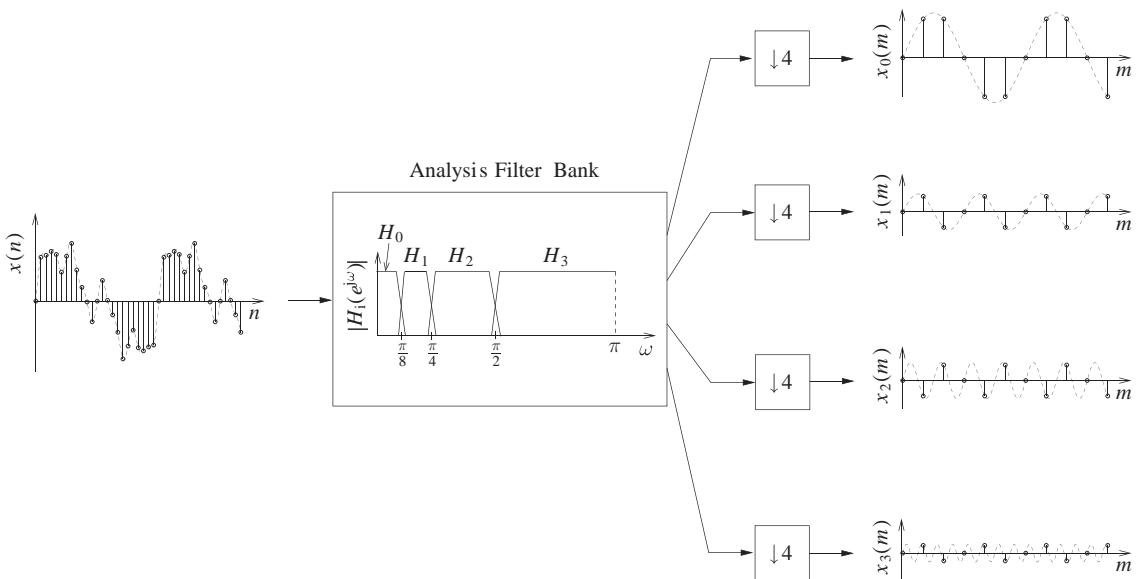
Functions  $\psi_{m,n}(t)$  do not comprise an orthogonal set, so neither do the functions  $\bar{\psi}_{m,n}(t)$ . However, functions  $\psi_{m,n}(t)$  are orthogonal to  $\bar{\psi}_{m,n}(t)$  so that

$$\langle \psi_{m,n}(t), \bar{\psi}_{k,l}(t) \rangle = \delta(m - k)\delta(n - l).$$

A byproduct of the development of the wavelet transforms is the so called wavelet series, allowing the representation of signals in the time and frequency domains through sequences. The cascade of two-band filter banks can produce many alternative maximally decimated decompositions. Of particular interest is the hierarchical decomposition achieving an octave-band decomposition, in which only the lowpass band is further decomposed. This configuration gives rise to the widely used wavelets series.

As will be discussed in Chapter 10, there are many transform-based tools in signal processing theory available to meet the requirement of different applications. The classical Fourier based transforms have been used for centuries, but their efficiency in dealing with nonstationary signals is limited. With wavelet series, for example, it is possible to obtain good resolution in time and frequency domains, unlike the Fourier-based analysis. Chapter 9 will include a myriad of transform solutions tailored for distinct practical situations.

Figure 1.17 illustrates a wavelet series representation of a composed signal, where the input signal is decomposed by an octave-band filter bank representing a wavelet series.

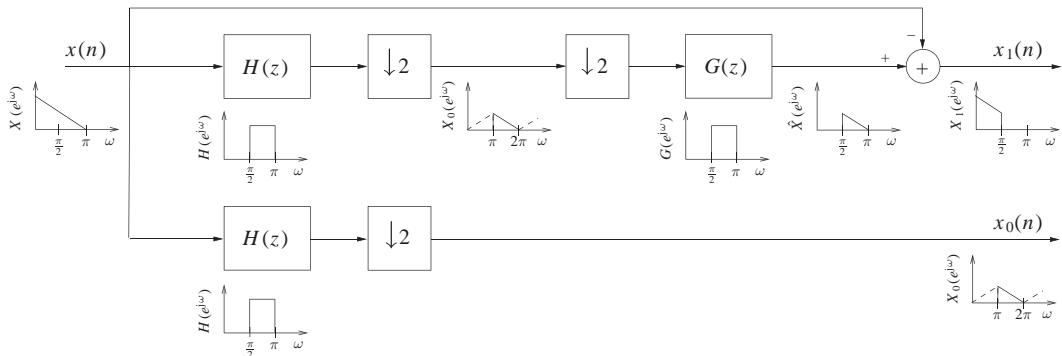
**FIGURE 1.17**

Wavelet series of a composed sinusoidal signal.

### 1.01.11 Frames

Frames consist of a set of vectors that enables a stable representation of a signal starting from an analysis operation whereby a set of coefficients is generated from the signal projection with the frame vectors, and a synthesis operation where the synthesis vectors are linearly combined to approximate the original signal. Chapter 10 will introduce the idea of frames by discussing the concepts of overcomplete representations, frames and their duals, frame operators, inverse frames and frame bounds. In particular, it will show how to implement signal analysis and synthesis employing frames generated from a fixed prototype signal by using translations, modulations and dilations; and analyzes frames of translates, Gabor frames and wavelet frames. The frames representation is usually redundant by requiring more coefficients than the minimum, where redundancy is measured by frame bounds [13]. The end result of employing the framework of frames is a set of signal processing tools such as: the Gabogram; time-frequency analysis; the matching pursuit algorithm, etc.

Figure 1.18 shows the representation of a composed signal into frames. As can be observed the representation of  $x(n)$  by  $x_1(n)$  and  $x_0(n)$  is redundant, since  $x_1(n)$  has the same rate as the input signal  $x(n)$ , whereas  $x_0(n)$  has half rate.

**FIGURE 1.18**

Frame representation.

## 1.01.12 Parameter estimation

In many signal processing applications it is crucial to estimate the power spectral density (PSD) of a given discrete-time signal. Examples of the use of PSD are vocal-track modeling, radar systems, antenna array, sonar systems, synthesis of speech and music, just to name a few. Usually, the first step in PSD estimation consists of estimating the autocorrelation function associated with the given data, followed by a Fourier transform to obtain the desired spectral description of the process.

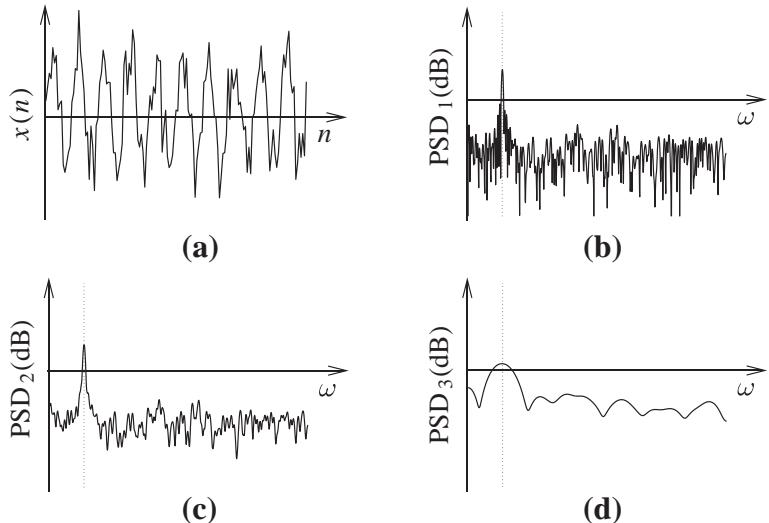
In the literature we can find a large number of sources dealing with algorithms for performing spectral estimation. The alternative solutions differ in their computational complexity, precision, frequency resolution, or other statistical aspects. In general, the spectral estimation methods can be classified as nonparametric or parametric methods. The nonparametric methods do not prescribe any particular structure for the given data, whereas parametric schemes assume that the provided process follows the pattern of a prescribed model characterized by a specific set of parameters [14].

Since the parametric approaches are simpler and more accurate, they will be emphasized in [Chapter 11](#). However, it should be mentioned that the parametric methods rely on some (a priori) information regarding the problem at hand.

[Figure 1.19](#) shows a noisy signal record and the estimated PSDs utilizing distinct methods. As can be observed, some of the resulting PSDs expose the presence of a sinusoid, not clearly observed in the time-domain signal.

## 1.01.13 Adaptive filtering

In many practical situations we can extract information from a signal by comparing it to another signal or a reference. For the cases where the specifications are neither known nor time-invariant, an adaptive filter is the natural solution. The coefficients of these filters are data driven and the resulting processing

**FIGURE 1.19**

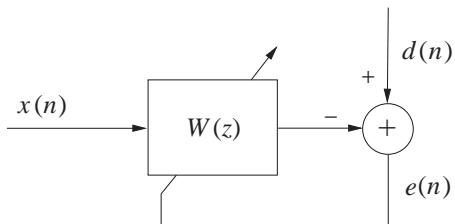
Spectral estimation. (a) Time-domain signal. (b) Estimated PSD1.(C) Estimated PSD2. (d) Estimated PSD3.

task does not meet the homogeneity and additive conditions of linear filtering. An adaptive filter is also time-varying since its parameters are continually changing in order to minimize a cost function.

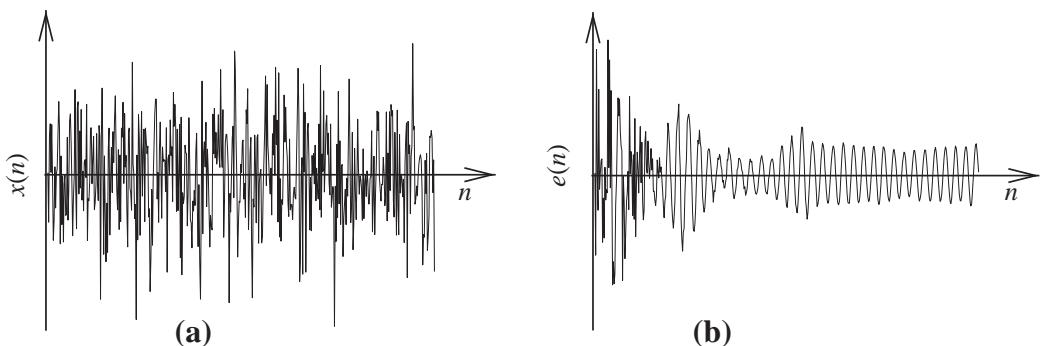
Adaptive filters are self-designing and perform the approximation step on-line. Starting from an initial condition, the adaptive filter searches an optimal solution in an iterative format. Since adaptive filters are nonlinear filters, the analysis of their performance behavior requires the solution of time-domain equations usually involving the second-order statistics of their internal and external quantities. Although the analysis of an adaptive filter is more complicated than for fixed filters, it is a simple solution to deal with unknown stationary and nonstationary environments. On the other hand, since the adaptive filters are self-designing filters, their design can be considered less involved than those of standard fixed digital filters [3]. An important topic related to adaptive filtering is neural networks and machine learning [26], not directly covered in this book.

Figure 1.20 depicts the general setup of an adaptive-filtering environment, where, for a given iteration  $n$ , an input signal  $x(n)$  excites the adaptive filter which will produce an output signal  $y(n)$  which is then compared with a reference signal  $d(n)$ . From this comparison, an error signal  $e(n)$  is formed according to  $d(n) - y(n)$ . The error signal is the argument of a cost (or objective) function whose minimization is achieved by properly adapting the adaptive filter coefficients. In Figure 1.21, it is shown a signal enhancement where it is recognized in the error signal that the input signal contained a sinusoidal component. In Figure 1.21 (a) it is difficult to observe the presence of a sinusoidal signal whereas the error signal clearly shows a nearly sinusoidal behavior.

Chapter 12 will present a comprehensive overview related to adaptive signal processing that unifies several concepts that go beyond the concepts covered in textbooks such as [3, 27, 28].

**FIGURE 1.20**

General adaptive filtering configuration.

**FIGURE 1.21**

An adaptive filtering application. (a) Input signal. (b) Error signal.

## 1.01.14 Closing comments

The theory of signal processing has been developing for over four decades at a fast pace. The following chapters will cover many of the classical tools widely employed in applications that we perceive in our daily life through the use of mobile phones, media players, medical equipments, transportation and so on. These chapters will also cover recent advances and describe many new tools which can potentially be utilized in new applications, as well as can be further investigated. The authors of the chapters are frequent and important contributors to the field of signal processing theory. The goal of each chapter is to provide an overview and a brief tutorial of important topic pertaining to the signal processing theory, including key references for further studies. The present chapter attempted to describe the context in which each family of tools is employed.

The rest of this book is organized as follows:

- [Chapter 2](#) covers the basic concepts of Continuous-Time Signals and Systems highlighting the main tools that can be employed to analyze and design such systems. Several examples are included to

illustrate the use of the tools in a full continuous-time environment. The content of this chapter is also essential to make the connection between the continuous-time and the discrete-time systems.

- [Chapter 3](#) addresses Discrete-Time Signals and Systems emphasizing the basic analysis tools that will be crucial to understand some more advanced techniques presented in the forthcoming chapters. This chapter shows how the powerful state-space representation of discrete-time systems can be employed as an analysis and design tool.
- [Chapter 4](#) on Random Signals and Stochastic Processes describes in a comprehensive way the basic concepts required to deal with random signals. It starts with the concept of probability useful to model chance experiments, whose outcomes give rise to the random variable. The time-domain signals representing non-static outcomes are known as stochastic processes where their full definition is provided. The chapter describes the tools to model the interactions between random signals and linear-time invariant systems.
- [Chapter 5](#) covers Sampling and Quantization where the basic concepts of properly sampling a continuous-time signal and its representation as a sequence of discrete-valued samples are addressed in detail. The chapter discusses a wide range of topics rarely found in a single textbook such as: uniform sampling and reconstruction of deterministic signals; the extension to sampling and reconstruction of stochastic processes; time-interleaved ADCs for high-speed A/D conversion. In addition, topics related to the correction to analog channel mismatches, principles of quantization, and over-sampled ADCs and DACs are briefly discussed. The chapter also includes the more advanced method for discrete-time modeling of mixed-signal systems employed in  $\Sigma\Delta$ -modulator-based ADCs.
- [Chapter 6](#) takes us to a tour on design of FIR and IIR transfer functions of fixed filters satisfying prescribed specifications, explaining their basic realizations and exploring more sophisticated realizations. For IIR filters, the chapter introduces the concept of wave digital filter in a concise and clear manner, providing the motivation for its conception from the analog filter structures. The resulting IIR filter realizations keep the low sensitivity properties of their analog originators. For FIR filter structures, the chapter covers the frequency masking approach that exploits redundancy in the impulse response of typical FIR filters with high selectivity, aiming at reducing the computational complexity. The chapter also explains in detail how to implement digital filters efficiently in specific hardware by properly scheduling the arithmetic operations.
- [Chapter 7](#) on Multirate Signal Processing for Software Radio Architecture applies several concepts presented in the previous chapters as tools to develop and conceive the implementation of radio defined by software, a subject of great interest in modern communications systems. Software defined radio entails implementing radio functionality into software, resulting in flexible radio systems whose main objective is to provide multi-service, multi-standard, multi-band features, all reconfigurable by software. From the signal processing perspective, several basic concepts of multirate systems such as, interpolation, decimation, polyphase decomposition, and transmultiplex play a central role. The chapter reviews the required concepts of signal processing theory and propose a software based radio architecture.
- [Chapter 8](#) on Modern Transform Design for Practical Audio/Image/Video Coding Applications addresses the design of several applications-oriented transform methods. Most signal processing textbooks present the classical transforms requiring large amount of multiplications, however the demand for low-power multimedia platforms requires the development of computationally efficient

transforms for coding applications. This chapter has the unique feature of presenting systematic procedures to design these transforms while illustrating their practical use in standard codecs.

- [Chapter 9](#) entitled Discrete Multi-Scale Transforms in Signal Processing presents a deeper high level exposition of the theoretical frameworks of the classical wavelets and the anisotropic wavelets. The aim is to enable the reader with enough knowledge of the wavelet-based tools in order to exploit their potential for applications in information sciences even further. Indeed the material covered in this chapter is unique in the sense that every discrete multi-scale transform is linked to a potential application.
- [Chapter 10](#) on Frames discusses the use of overcomplete signal representations employing frames and their duals, frame operators, inverse frames, and frame bounds. The signal analysis and synthesis using frame representation is also addressed in detail. In particular, the chapter emphasis is on frames generated from a fixed prototype signal by using translations, modulations and dilations, and analyzes frames of translates, Gabor frames and wavelet frames. The chapter also presents signal analysis techniques based on the Gaborgram and time-frequency analysis using frames and the matching pursuit algorithm.
- [Chapter 11](#) on Parametric Estimation exploits the key engineering concept related to modeling signals and systems, so that the model provides understanding of the underlying phenomena and possibly their control. This chapter emphasizes the parametric models leading to simple estimation of the parameters while exposing the main characteristics of the signals or systems under study. By employing both statistical and deterministic formulations, the chapter explains how to generate auto-regressive and moving average models which are then applied to solve problems related to spectrum estimation, prediction, and filtering.
- [Chapter 12](#) on Adaptive Filtering presents a comprehensive description of the current trends as well as some open problems in this area. The chapter discusses the basic building blocks utilized in adaptive filtering setup as well as its typical applications. Then, the chapter discusses what are the optimal solutions following by the derivation of the main algorithms. The authors confront for the first time two standard approaches to access the performance of the adaptive filtering algorithms. The chapter closes by discussing some current research topics and open problems for further investigation.

---

## References

- [1] A. Antoniou, On the Roots of Digital Signal Processing, *IEEE Circuits Syst. Mag.* 7 (1) (2007) 8–18; A. Antoniou, *IEEE Circuits Syst. Mag.* 7 (4) (2007) 8–19.
- [2] P.S.R. Diniz, E.A. Silva, S.L. Netto, *Digital Signal Processing: System Analysis and Design*, second ed., Cambridge University Press, Cambridge, UK, 2010.
- [3] P.S.R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, fourth ed., Springer, New York, NY, 2013.
- [4] P.S.R. Diniz, W.A. Martins, M.V.S. Lima, *Block Transceivers: OFDM and Beyond*, Morgan & Claypool Publishers, Ft, Collins, CO., 2012.
- [5] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*, McGraw-Hill, New York, NY, 2006.
- [6] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [7] A.V. Oppenheim, A.S. Willsky, A.H. Nawab, *Signals and Systems*, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1997.

- [8] A.V. Oppenheim, R.W. Schaffer, *Discrete-Time Signal Processing*, third ed., Prentice-Hall, Englewood Cliffs, NJ, 2010.
- [9] P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [10] M. Vetterli, J. Kovac̆ević, *Wavelets and Subband Coding*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [11] J.G. Proakis, D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, fourth ed., Prentice-Hall, Englewood Cliffs, NJ, 2007.
- [12] J.G. Proakis, D.G. Manolakis, *Applied Digital Signal Processing*, Cambridge University Press, Cambridge, UK, 2011.
- [13] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, third ed., Academic Press, Burlington, MA, 2009.
- [14] S.M. Kay, *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [15] J.H. McClellan, R.W. Schafer, M.A. Yoder, *DSP First: A Multimedia Approach*, third ed., Prentice-Hall, Englewood Cliffs, NJ, 1998.
- [16] S.K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, third ed., McGraw-Hill, New York, NY, 2006.
- [17] A. Papoulis, S. UnniKrishna Pillai, *Probability, Random Variables, and Stochastic Processes*, fourth ed., McGraw-Hill, New York, NY, 2002.
- [18] J.J. Shynk, *Probability, Random Variables, and Random Processes: Theory and Signal Processing Applications*, John Wiley & Sons, Hoboken, NJ, 2013.
- [19] T. Saramäki, Finite-impulse response filter design, in: S.K. Mitra, J.F. Kaiser (Eds.), *Handbook of Digital Signal Processing*, John Wiley & Sons, New York, NY, pp. 155–177.
- [20] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, Chichester, UK, 1980.
- [21] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, second ed., Addison-Wesley, Boston, MA, 1984.
- [22] A. Antoniou, W.-S. Lu, *Practical Optimization: Algorithms and Engineering Applications*, Springer, New York, NY, 2013.
- [23] D. Tse, P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, Cambridge, UK, 2005.
- [24] P.P. Vaidyanathan, S.-M. Phoong, Y.-P. Lin, *Signal Processing and Optimization for Transceiver Systems*, Cambridge University Press, Cambridge, UK, 2010.
- [25] H.L. Van Trees, *Optimum Array Processing*, John Wiley & Sons, New York, NY, 2002.
- [26] S. Haykin, *Neural Networks and Learning Machines*, Prentice-Hall, Englewood Cliffs, NJ, 2009.
- [27] S. Haykin, *Adaptive Filter Theory*, fourth ed., Prentice-Hall, Englewood Cliffs, NJ, 2002.
- [28] A.H. Sayed, *Adaptive Filters*, John Wiley & Sons, Hoboken, NJ, 2008.

# Continuous-Time Signals and Systems

# 2

José A. Apolinário Jr. and Carla L. Pagliari<sup>1</sup>

*Military Institute of Engineering (IME), Department of Electrical Engineering (SE/3), Rio de Janeiro, RJ, Brazil*

---

## Nomenclature

C	capacitance, in Faraday (F)
f	frequency, in cycles per second or Hertz (Hz)
L	inductance, in Henry (H)
R	resistance, in Ohms ( $\Omega$ )
t	time, in Second (s)
$x(t)$	input signal of a given continuous-time system, expressed in volt (V) when it corresponds to an input voltage; $y(t)$ is usually used as the output signal
$\Omega$	angular frequency, in radian per second (rad/s) (it is sometimes referred to as <i>frequency</i> although corresponding to $2\pi f$ )

---

### 1.02.1 Introduction

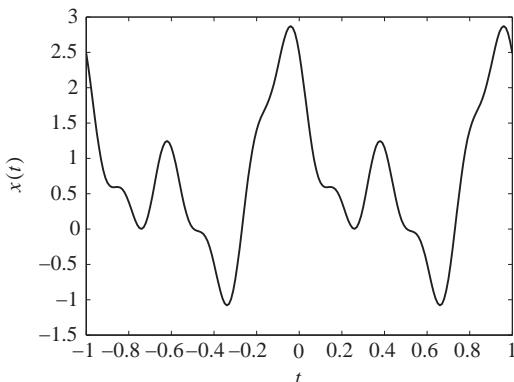
Most signals present in our daily life are continuous in time such as music and speech. A signal is a function of an independent variable, usually an observation measured from the real world such as position, depth, temperature, pressure, or time. Continuous-time signals have a continuous independent variable. The velocity of a car could be considered a continuous-time signal if, at any time  $t$ , the velocity  $v(t)$  could be defined. A continuous-time signal with a continuous amplitude is usually called an analog signal, speech signal being a typical example.

Signals convey information and are generated by electronic or natural means as when someone talks or plays a musical instrument. The goal of signal processing is to extract the useful information embedded in a signal.

Electronics for audio and last generation mobile phones must rely on universal concepts associated with the flow of information to make these devices fully functional. Therefore, a system designer, in order to have the necessary understanding of advanced topics, needs to master basic signals and systems theory. System theory could then be defined as the relation between signals, input and output signals.

---

<sup>1</sup> Authors thank Prof. Ney Bruno for his kind and competent review of this chapter.

**FIGURE 2.1**

Example of a continuous-time signal.

As characterizing the complete input/output properties of a system through measurement is, in general, impossible, the idea is to infer the system response to non-measured inputs based on a limited number of measurements. System design is a challenging task. However, several systems can be accurately modeled as linear systems. Hence, the designer has to create between the input and the output, an expected, or predictable, relationship that is always the same (time-invariant). In addition, for a range of possible input types, the system should generate a predictable output in accordance to the input/output relationship.

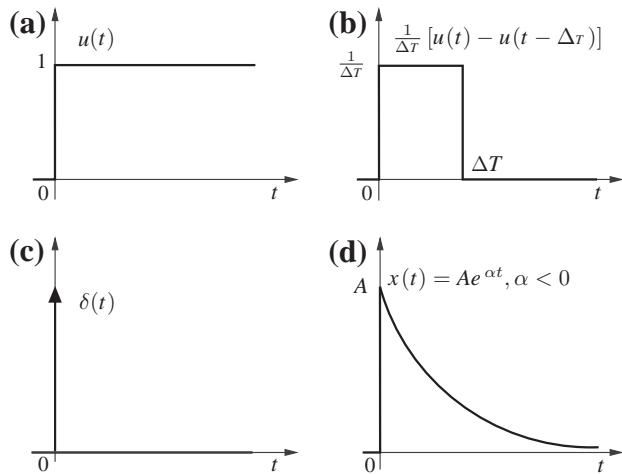
A continuous-time signal is defined on the continuum of time values as the one depicted in Figure 2.1,  $x(t)$  for  $t \in \mathbb{R}$ .

Although signals are real mathematical functions, some transformations can produce complex signals that have both real and imaginary parts. Therefore, throughout this book, complex domain equivalents of real signals will certainly appear.

Elementary continuous-time signals are the basis to build more intricate signals. The unit step signal,  $u(t)$ , displayed by Figure 2.2a is equal to 1 for all time greater than or equal to zero, and is equal to zero for all time less than zero. A step of height  $K$  can be made with  $Ku(t)$ .

A unit area pulse is pictured in Figure 2.2b; as  $\Delta\tau$  gets smaller, the pulse gets higher and narrower with a constant area of one (unit area). We can see the impulse signal, in Figure 2.2c, as a limiting case of the pulse signal when the pulse is infinitely narrow. The impulse response will help to estimate how the system will respond to other possible *stimuli*, and we will further explore this topic on the next section.

Exponential signals are important for signals and systems as eigensignals of linear time-invariant systems. In Figure 2.2d, for  $A$  and  $\alpha$  being real numbers,  $\alpha$  negative, the signal is a decaying exponential. When  $\alpha$  is a complex number, the signal is called a complex exponential signal. The periodic signals sine and cosine are used for modeling the interaction of signals and systems, and the usage of complex exponentials to manipulate sinusoidal functions turns trigonometry into elementary arithmetic and algebra.

**FIGURE 2.2**

Basic continuous-time signals: (a) unit-step signal  $u(t)$ , (b) unit-area pulse, (c) impulse signal (unit-area pulse when  $\Delta\tau \rightarrow 0$ ), and (d) exponential signal.

## 1.02.2 Continuous-time systems

We start this section by providing a simplified classification of continuous-time systems in order to focus on our main interest: a linear and time-invariant (LTI) system.

Why are linearity and time-invariance important? In general, real world systems can be successfully modeled by theoretical LTI systems; in an electrical circuit, for instance, a system can be designed using a well developed linear theory (valid for LTI systems) and be implemented using real components such as resistors, inductors, and capacitors. Even though the physical components, strictly speaking, do not comply with linearity and time-invariance for any input signals, the circuit will work quite well under reasonable conditions (input voltages constrained to the linear working ranges of the components).

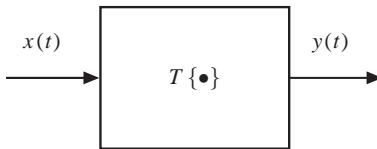
A system can be modeled as a function that transforms, or maps, the input signal into a new output signal. Let a continuous-time system be represented as in Figure 2.3 where  $x(t)$  is the input signal and  $y(t)$ , its output signal, corresponds to a transformation applied to the input signal:  $y(t) = \mathcal{T}\{x(t)\}$ .

The two following properties define a LTI system.

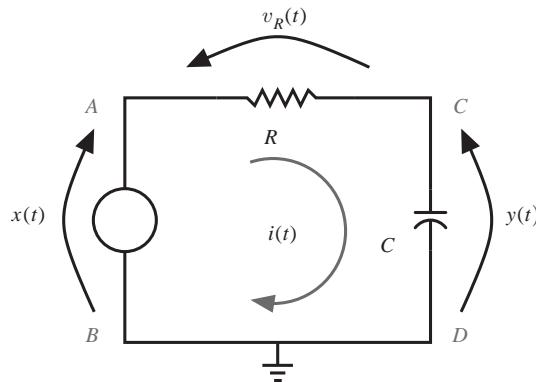
**Linearity:** A continuous-time system is said to be linear if a linear combination of input signals, when applied to this system, produces an output that corresponds to a linear combination of individual outputs, i.e.,

$$\begin{aligned} y(t) &= \mathcal{T}\{a_1x_1(t) + a_2x_2(t) + \dots\} \\ &= a_1\mathcal{T}\{x_1(t)\} + a_2\mathcal{T}\{x_2(t)\} + \dots \\ &= a_1y_1(t) + a_2y_2(t) + \dots, \end{aligned} \tag{2.1}$$

where  $y_i(t) = \mathcal{T}\{x_i(t)\}$ .

**FIGURE 2.3**

A generic continuous-time system representing the output signal as a transformation applied to the input signal:  $y(t) = T\{x(t)\}$ .

**FIGURE 2.4**

An example of an electric circuit representing a continuous-time system with input  $x(t)$  and output  $y(t)$ .

**Time-invariance:** A continuous-time system is said to be time-invariant when the output signal,  $y(t)$ , corresponding to an input signal  $x(t)$ , will be a delayed version of the original output,  $y(t - t_0)$ , whenever the input is delayed accordingly, i.e.,

$$\begin{aligned} &\text{if } y(t) = T\{x(t)\}; \\ &\text{then } y(t - t_0) = T\{x(t - t_0)\}. \end{aligned}$$

The transformation caused in the input signal by a linear time-invariant system may be represented in a number of ways: with a set of differential equations, with the help of a set of state-variables, with the aid of the concept of the impulse response, or in a transformed domain.

Let the circuit in Figure 2.4 be an example of a continuous-time system where the input  $x(t)$  corresponds to the voltage between terminals  $\mathcal{A}$  and  $\mathcal{B}$  while its output  $y(t)$  is described by the voltage between terminals  $\mathcal{C}$  and  $\mathcal{D}$ .

We know for this circuit that  $x(t) = y(t) + v_R(t)$  and that the current in the capacitor,  $i(t)$ , corresponds to  $C dy(t)/dt$ . Therefore,  $v_R(t) = Ri(t) = RC dy(t)/dt$ , and we can write

$$RC \frac{dy(t)}{dt} + y(t) = x(t), \quad (2.2)$$

which is an input-output (external) representation of a continuous-time system given as a differential equation.

Another representation of a linear system, widely used in control theory, is the state-space representation. This representation will be presented in the next section since it is better explored for systems having a higher order differential equation.

In order to find an explicit expression for  $y(t)$  as a function of  $x(t)$ , we need to solve the differential equation. The complete solution of this system, as shall be seen in the following section, is given by the sum of an homogeneous solution (zero-input solution or natural response) and a particular solution (zero-state solution or forced solution):

$$y(t) = \underbrace{y_h(t)}_{\text{homogeneous solution}} + \underbrace{y_p(t)}_{\text{particular solution}}. \quad (2.3)$$

The homogeneous solution, in our example, is obtained from (2.2) by setting  $RD dy_h(t)/dt + y_h(t) = 0$ . Since this solution and its derivatives must comply with the homogeneous differential equation, its usual choice is an exponential function such as  $Ke^{st}$ ,  $s$  being, in general, a complex number. Replacing this general expression in the homogeneous differential equation, we obtain  $RCs + 1 = 0$  (characteristic equation) and, therefore,  $s = -1/RC$ , such that the homogenous solution becomes

$$y_h(t) = Ke^{-\frac{t}{RC}}, \quad (2.4)$$

where  $K$  is a constant.

The particular solution is usually of the same form as the forcing function (input voltage  $x(t)$  in our example) and it must comply with the differential equation without an arbitrary constant. In our example, if  $x(t) = Me^{-mt}$ ,  $y_p(t)$  would be  $\frac{M}{1-mRC}e^{-mt}$ .

If we set  $x(t) = u(t)$ , the step function which can be seen as a DC voltage of 1 Volt switched at time instant  $t = 0$ , the forced solution is assumed to be equal to one when  $t \geq 0$ . For this particular input  $x(t) = u(t)$ , the output signal is given by  $y(t) = (Ke^{-\frac{t}{RC}} + 1)u(t)$ . Constant  $K$  is obtained with the knowledge of the initial charge of the capacitor (*initial conditions*); assuming it is not charged ( $v_c(t) = 0$ ,  $t \leq 0$ ), we have

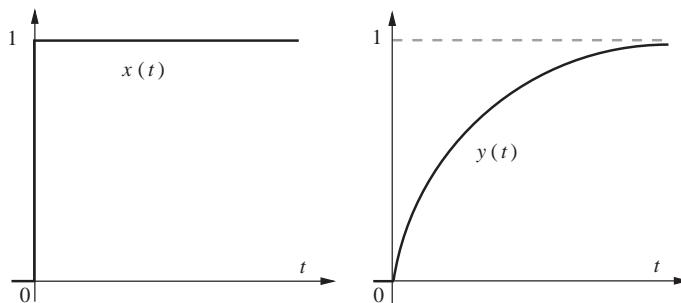
$$y(t) = (1 - e^{-\frac{t}{RC}})u(t), \quad (2.5)$$

which can be observed in Figure 2.5.

Note that, for this case where  $x(t) = u(t)$ , the output  $y(t)$  is known as the *step response*, i.e.,  $r(t) = y(t)|_{x(t)=u(t)}$ . Moreover, since the impulse  $\delta(t)$  corresponds to the derivative of  $u(t)$ ,  $\delta(t) = du(t)/dt$ , and the system is LTI (linear and time-invariant), we may write that  $dr(t)/dt = dT\{u(t)\}/dt$  which corresponds to  $T\{du(t)/dt\}$ , the transformation applied to the impulse signal leading to

$$h(t) = \frac{dr(t)}{dt}, \quad (2.6)$$

$h(t)$  known as the *impulse response* or  $h(t) = y(t)|_{x(t)=\delta(t)}$ .

**FIGURE 2.5**

Input and output signals of the system depicted in Figure 2.4. See [Video 1](#) to watch animation.

The impulse response is, particularly, an important characterization of a linear system. It will help to estimate how the system will respond to other *stimuli*. In order to show the relevance of this, let us define the unit impulse as

$$\delta(t) = \lim_{\Delta\tau \rightarrow 0} \frac{1}{\Delta\tau} [u(t) - u(t - \Delta\tau)], \quad (2.7)$$

such that we may see an input signal \$x(t)\$ as in Figure 2.6:

$$x(t) \approx \sum_{k=-\infty}^{\infty} x(k\Delta\tau) \underbrace{\left[ \frac{u(t - k\Delta\tau) - u(t - k\Delta\tau - \Delta\tau)}{\Delta\tau} \right]}_{\delta(t - k\Delta\tau) \text{ when } \Delta\tau \rightarrow 0} \Delta\tau. \quad (2.8)$$

The idea is to show that any signal, e.g., \$x(t)\$, can be expressed as a sum of scaled and shifted impulse functions.

In (2.8), making \$\Delta\tau \rightarrow 0\$ and representing the resulting continuous time by \$\tau\$ instead of \$k\Delta\tau\$, we can use the integral instead of the summation and find

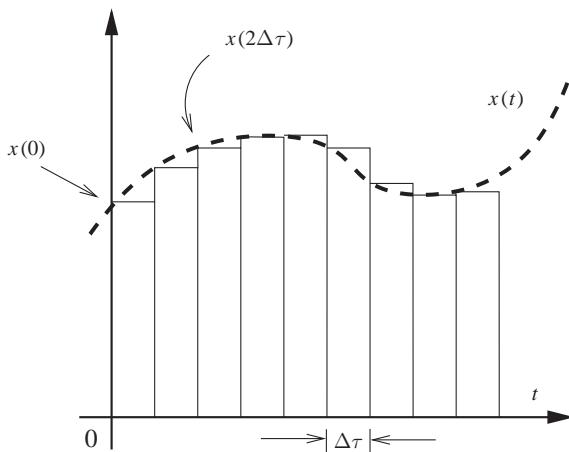
$$x(t) = \int_{-\infty}^{\infty} x(\tau) \delta(t - \tau) d\tau. \quad (2.9)$$

In a LTI system, using the previous expression as an input signal to compute the output \$y(t) = T\{x(t)\}\$, we obtain \$\int\_{-\infty}^{\infty} x(\tau) T\{\delta(t - \tau)\} d\tau\$ such that the output signal can be written as

$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau. \quad (2.10)$$

This expression is known as the *convolution* integral between the input signal and the system impulse response, and is represented as

$$y(t) = x(t) * h(t). \quad (2.11)$$

**FIGURE 2.6**

A signal  $x(t)$  can be obtained from the pulses if we make  $\Delta\tau \rightarrow 0$ .

Please note that the output signal provided by the convolution integral corresponds to the zero-state solution.

From Figure 2.6, another approximation for  $x(t)$  can be devised, leading to another expression relating input and output signals. At a certain instant  $t = \tau$ , let the angle of a tangent line to the curve of  $x(t)$  be  $\theta$  such that  $\tan \theta = \frac{dx(\tau)}{d\tau}$ . Assuming a very small  $\Delta\tau$ , this tangent can be approximated by  $\frac{\Delta x(\tau)}{\Delta\tau}$  such that

$$\Delta x(\tau) = \frac{dx(\tau)}{d\tau} \Delta\tau. \quad (2.12)$$

Knowing each increment at every instant  $t = k\Delta\tau$ , we can visualize an approximation for  $x(t)$  as a sum of shifted and weighted step functions  $u(t - k\Delta\tau)$ , i.e.,

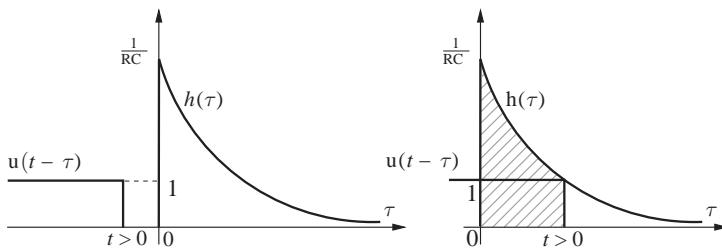
$$x(t) \approx \sum_{k=-\infty}^{\infty} \Delta x(k\Delta\tau) u(t - k\Delta\tau) = \sum_{k=-\infty}^{\infty} \frac{dx(k\Delta\tau)}{d\tau} u(t - k\Delta\tau) \Delta\tau. \quad (2.13)$$

From the previous expression, if we once more, as in (2.9), make  $\Delta\tau \rightarrow 0$  and represent the resulting continuous-time by  $\tau$  instead of  $k\Delta\tau$ , we can drop the summation and use the integral to obtain

$$x(t) = \int_{-\infty}^{\infty} \frac{dx(\tau)}{d\tau} u(t - \tau) d\tau. \quad (2.14)$$

Assuming, again, that the system is LTI, we use the last expression as an input signal to compute the output  $y(t) = \mathcal{T}\{x(t)\}$ , obtaining  $\int_{-\infty}^{\infty} \frac{dx(\tau)}{d\tau} \mathcal{T}\{u(t - \tau)\} d\tau$  which can be written as

$$y(t) = \int_{-\infty}^{\infty} \frac{dx(\tau)}{d\tau} r(t - \tau) d\tau \quad (2.15)$$

**FIGURE 2.7**

Graphical interpretation of the convolution integral. See [Video 2](#) to watch animation.

or,  $r(t)$  being the step-response, as

$$y(t) = \frac{dx(t)}{dt} * r(t). \quad (2.16)$$

In our example from Figure 2.4, taking the derivative of  $r(t) = (1 - e^{-t/RC})u(t)$ , we obtain the impulse response (the computation of this derivative is somehow tricky for we must bear in mind that the voltage in the terminals of a capacitor cannot present discontinuities; just as in the case of the current through an inductor) as:

$$h(t) = \frac{1}{RC}e^{-\frac{t}{RC}}. \quad (2.17)$$

In order to have a graphical interpretation of the convolution, we make  $x(t) = u(t)$  in (2.11) and use

$$r(t) = h(t) * u(t) = \int_{-\infty}^{\infty} h(\tau)u(t - \tau)d\tau; \quad (2.18)$$

we then compute this integral, as indicated in Figure 2.7, in two parts:

$$\begin{aligned} &\text{for } t < 0 : r(t) = 0; \\ &\text{for } t > 0 : r(t) = \int_0^t h(\tau)d\tau = \frac{1}{RC} \int_0^t e^{-\frac{\tau}{RC}}d\tau. \end{aligned} \quad (2.19)$$

Finally, from (2.19), we obtain  $r(t) = (1 - e^{-t/RC})u(t)$ , as previously known.

A few mathematical properties of the convolution are listed in the following:

- *Commutative*:  $x(t) * h(t) = h(t) * x(t)$ .
- *Associative*:  $[x(t) * h_1(t)] * h_2(t) = x(t) * [h_1(t) * h_2(t)]$ .
- *Distributive*:  $x(t) * [h_1(t) + h_2(t)] = x(t) * h_1(t) + x(t) * h_2(t)$ .
- *Identity element of convolution*:  $x(t) * \delta(t) = x(t)$ .

The properties of convolution can be used to analyze different system combinations. For example, if two systems with impulse responses  $h_1(t)$  and  $h_2(t)$  are cascaded, the whole cascade system will present the impulse response  $h_1(t) * h_2(t)$ . The commutative property allows the order of the systems of the cascade combination to be changed without affecting the whole system's response.

Two other important system properties are causability and stability.

*Causability:* A causal system, also known as non-anticipative system, is a system in which its output  $y(t)$  depends only in the current and past (not future) information about  $x(t)$ . A non-causal, or anticipative, system is actually not feasible to be implemented in real-life. For example,  $y(t) = x(t + 1)$  is non-causal, whereas  $y(t) = x(t - 1)$  is causal.

With respect to the impulse response, it is also worth mentioning that, for a causal system,  $h(t) = 0$  for  $t < 0$ .

*Stability:* A system is stable if and only if every bounded input produces a bounded output, i.e., if  $|x(t)| < B_x$  then  $|y(t)| < B_y$  for all values of  $t$ .

More about stability will be addressed in a forthcoming section, after the concept of poles and zeros is introduced.

The classic texts for the subjects discussed in this section include [1–5].

### 1.02.3 Differential equations

In many engineering applications, the behavior of a system is described by a differential equation. A differential equation is merely an equation with derivative of at least one of its variables. Two simple examples follow:

$$a \frac{d^2x}{dt^2} + b \frac{dx}{dt} + c = \sin(t), \text{ and} \quad (2.20)$$

$$\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} = x^2 + y^2 \quad (2.21)$$

where  $a$ ,  $b$ , and  $c$  are constants.

In (2.20), we observe an *ordinary differential equation*, i.e., there is only one independent variable and  $x = f(t)$ ,  $t$  being the independent variable. On the other hand, in (2.21), we have a *partial differential equation* where  $z = f(x, y)$ ,  $x$  and  $y$  being two independent variables. Both examples have *order 2* (highest derivative appearing in the equation) and *degree 1* (power of the highest derivative term).

This section deals with the solution of differential equations usually employed to represent the mathematical relationship between input and output signals of a linear system. We are therefore most interested in linear differential equations with constant coefficients having the following general form:

$$a_0 y + a_1 \frac{dy}{dt} + \cdots + a_N \frac{d^N y}{dt^N} = b_0 x + b_1 \frac{dx}{dt} + \cdots + b_M \frac{d^M x}{dt^M}. \quad (2.22)$$

The expression on the right side of (2.22) is known as *forcing function*,  $f(t)$ . When  $f(t) = 0$ , the differential equation is known as *homogeneous* while a non-zero forcing function corresponds to a nonhomogeneous differential equation such as in

$$a_0 y + a_1 \frac{dy}{dt} + \cdots + a_N \frac{d^N y}{dt^N} = f(t). \quad (2.23)$$

As mentioned in the previous section, the general solution of a differential equation as the one in (2.22) is given by the sum of two expressions:  $y_H(t)$ , the solution of the associated homogeneous differential equation

$$a_0 y + a_1 \frac{dy}{dt} + \cdots + a_N \frac{d^N y}{dt^N} = 0, \quad (2.24)$$

and a particular solution of the nonhomogeneous equation,  $y_P(t)$ , such that

$$y(t) = y_H(t) + y_P(t). \quad (2.25)$$

*Solution of the homogeneous equation:* The natural response of a linear system is given by  $y_H(t)$ , the solution of (2.24). Due to the fact that a linear combination of  $y_H(t)$  and its derivatives must be equal to zero in order to comply with (2.24), it may be postulated that it has the form

$$y_H(t) = e^{st}, \quad (2.26)$$

where  $s$  is a constant to be determined.

Replacing the assumed solution in (2.24), we obtain, after simplification, the characteristic (or auxiliary) equation

$$a_0 + a_1 s + \cdots + a_N s^N = 0. \quad (2.27)$$

Assuming  $N$  distinct roots of the polynomial in (2.27),  $s_1$  to  $s_N$ , the homogeneous solution is obtained as a linear combination of  $N$  exponentials as in

$$y_H(t) = k_1 e^{s_1 t} + k_2 e^{s_2 t} + \cdots + k_N e^{s_N t}. \quad (2.28)$$

Two special cases follow.

1. *Non-distinct roots:* if, for instance,  $s_1 = s_2$ , it is possible to show that  $e^{s_1 t}$  and  $t e^{s_2 t}$  are independent solutions leading to

$$y_H(t) = k_1 e^{s_1 t} + k_2 t e^{s_2 t} + \cdots.$$

2. *Characteristic equation with complex roots:* it is known that, for real coefficients ( $a_0$  to  $a_N$ ), all complex roots will occur in complex conjugate pairs such as  $s_1 = \alpha + j\beta$  and  $s_2 = \alpha - j\beta$ ,  $\alpha$  and  $\beta$  being real numbers. Hence, the solution would be

$$\begin{aligned} y_H(t) &= k_1 e^{(\alpha+j\beta)t} + k_2 e^{(\alpha-j\beta)t} + \cdots \\ &= [(k_1 + k_2) \cos \beta t + j(k_1 - k_2) \sin \beta t] e^{\alpha t} + \cdots \\ &= (k_3 \cos \beta t + k_4 \sin \beta t) e^{\alpha t} + \cdots, \end{aligned}$$

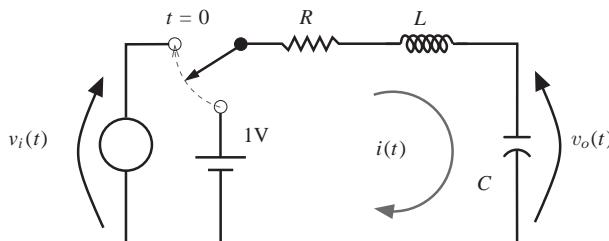
$k_3$  and  $k_4$  being real numbers if we make  $k_1 = k_2^*$ . In that case, we will have

$$y_H(t) = k_5 \cos(\beta t - \theta) e^{\alpha t} + \cdots,$$

with  $k_5 = \sqrt{k_3^2 + k_4^2}$  and  $\theta = \arctan \frac{k_4}{k_3}$ . Also note from the last expression that, in order to have a stable system (bounded output),  $\alpha$  should be negative (causing a decaying exponential).

**Table 2.1** Assumed Particular Solutions to Common Forcing Functions

Forcing function $f(t)$	Assumed $y_P(t)$
$K$	$A_0$
$Kt$	$A_0 t + A_1$
$Kt^n$	$A_0 t^n + A_1 t^{n-1} + \dots + A_{n-1} t + A_n$
$K e^{-\alpha t}$	$A_0 e^{-\alpha t}$
$\sin \Omega t$ or $\cos \Omega t$	$A_0 \sin \Omega t + A_1 \cos \Omega t$
$e^{-\alpha t} \sin \Omega t$ or $e^{-\alpha t} \cos \Omega t$	$e^{-\alpha t} (A_0 \sin \Omega t + A_1 \cos \Omega t)$

**FIGURE 2.8**

RLC series circuit with a voltage source as input and the voltage in the capacitor as output.

*Solution of the nonhomogeneous equation:* A (any) forced solution of the nonhomogeneous equation must satisfy (2.23) containing no arbitrary constant. A usual way to find the forced solution is employing the so called method of undetermined coefficients: it consists in estimating a general form for  $y_P(t)$  from  $f(t)$ , the forcing function. The coefficients ( $A_0, A_1, \dots$ ), as seen in Table 2.1 that shows the most common assumed solutions for each forced function, are to be determined in order to comply with the nonhomogeneous equation. A special case is treated slightly differently: when a term of  $f(t)$  corresponds to a term of  $y_H(t)$ , the corresponding term in  $y_P(t)$  must be multiplied by  $t$ . Also, when we find non-distinct roots of the characteristic equation (assume multiplicity  $m$  as an example), the corresponding term in  $y_P(t)$  shall be multiplied by  $t^m$ .

An *example* of a second order ordinary differential equation (ODE) with constant coefficients is considered as follows:

$$LC \frac{d^2 v_o(t)}{dt^2} + RC \frac{dv_o(t)}{dt} + v_o(t) = v_i(t), \quad (2.29)$$

where  $R = 2.0$  (in  $\Omega$ ),  $L = 2.0$  (in  $H$ ),  $C = 1/2$  (in  $F$ ), and  $v_i(t) = e^{-t}$  (in  $V$ ).

This equation describes the input  $\times$  output relationship of the electrical circuit shown in Figure 2.8 for  $t > 0$ .

Replacing the values of the components and the input voltage, the ODE becomes

$$\frac{d^2 v_o(t)}{dt^2} + \frac{dv_o(t)}{dt} + v_o(t) = \underbrace{e^{-t}}_{f(t)}. \quad (2.30)$$

The associated characteristic equation  $s^2 + s + 1 = 0$  has roots  $s = \frac{-1 \pm j\sqrt{3}}{2}$  leading to an homogeneous solution given by

$$\begin{aligned} v_{oH}(t) &= k_1 \cos\left(\frac{\sqrt{3}}{2}t\right) e^{-t/2} + k_2 \sin\left(\frac{\sqrt{3}}{2}t\right) e^{-t/2} \\ &= k_3 \cos\left(\frac{\sqrt{3}}{2}t + k_4\right) e^{-t/2}. \end{aligned} \quad (2.31)$$

Next, from the forcing function  $f(t) = e^{-t}$ , we assume  $v_{oP}(t) = k_5 e^{-t}$  and replace it in (2.30), resulting in  $k_5 = 1$  such that:

$$\begin{aligned} v_o(t) &= k_1 \cos\left(\frac{\sqrt{3}}{2}t\right) e^{-t/2} + k_2 \sin\left(\frac{\sqrt{3}}{2}t\right) e^{-t/2} + e^{-t} \\ &= k_3 \cos\left(\frac{\sqrt{3}}{2}t + k_4\right) e^{-t/2} + e^{-t}, \end{aligned} \quad (2.32)$$

where  $k_1$  and  $k_2$  (or  $k_3$  and  $k_4$ ) are constants to be obtained from previous knowledge of the physical system, the RLC circuit in this example. This knowledge comes as the initial conditions: an  $N$ -order differential equation having  $N$  constants and requiring  $N$  initial conditions.

*Initial conditions:* Usually, the differential equation describing the behavior of an electrical circuit is valid for any time  $t > 0$  (instant 0 assumed the initial reference in time); the initial conditions corresponds to the solution (and its  $N - 1$  derivatives) at  $t = 0^+$ . *In the absence of pulses, the voltage at the terminals of a capacitance and the current through an inductance cannot vary instantaneously* and must be the same value at  $t = 0^-$  and  $t = 0^+$ .

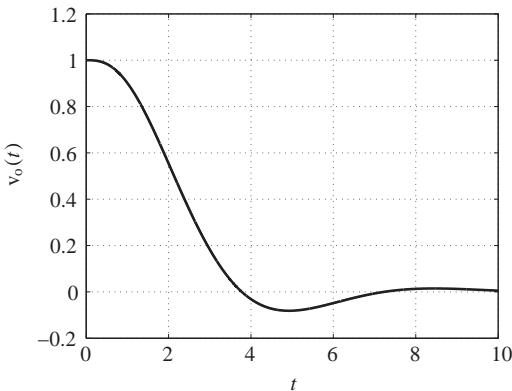
In the case of our example, based on the fact that there is a key switching  $v_i(t)$  to the RLC series at  $t = 0$ , we can say that the voltage across the inductor is  $v_o(0^-) = 1$  (the capacitor assumed charged with the DC voltage). Therefore, since the voltage across  $C$  and the current through  $L$  do not alter instantaneously, we know that  $v_o(0^+) = 1$  and  $i(0^+) = 0$ . Since we know that  $i(t) = C \frac{dv_o(t)}{dt}$ , we have  $\frac{dv_o(t)}{dt} \Big|_{t=0^+} = 0$ . With these initial conditions and from (2.32), we find  $k_1 = 0$  and  $k_2 = \frac{2\sqrt{3}}{3}$ .

Finally, the general solution is given as

$$v_o(t) = 1.1547 \sin(0.866t) e^{-t/2} + e^{-t}. \quad (2.33)$$

Figure 2.9 shows  $v_o(t)$  for  $0 \leq t \leq 10$ . An easy way to obtain this result with Matlab<sup>©</sup> is:

```
> y=dsolve ('D2y+Dy+y=exp(-t)', 'y(0)=1', 'Dy(0)=0');
> ezplot (y,[0 10])
```

**FIGURE 2.9**

Output voltage as a function of time,  $v_o(t)$ , for the circuit in Figure 2.8. See [Video 3](#) to watch animation.

To end this section, we represent this example using the state-space approach. We first rewrite (2.29) using the first and the second derivatives of  $v_o(t)$  as  $\dot{v}$  and  $\ddot{v}$ , respectively, and also  $u = v_i(t)$  as in

$$LC\ddot{v} + RC\dot{v} + v = u. \quad (2.34)$$

In this representation, we define a state vector  $\mathbf{x} = [v \ \dot{v}]^T$  and its derivative  $\dot{\mathbf{x}} = [\dot{v} \ \ddot{v}]^T$ , and from these definitions we write the *state equation* and the *output equation*:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} = \begin{bmatrix} 0 & 1 \\ \frac{-1}{LC} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} v \\ \dot{v} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{LC} \end{bmatrix} v_i(t), \\ y = \mathbf{Cx} + Du = [10] \begin{bmatrix} v \\ \dot{v} \end{bmatrix} + 0v_i(t), \end{cases} \quad (2.35)$$

where **A** is known as the *state matrix*, **B** as the *input matrix*, **C** as the *output matrix*, and **D** (equal to zero in this example) would be the *feedthrough* (or *feedforward*) matrix.

For further reading, we suggest [6–9].

## 1.02.4 Laplace transform: definition and properties

The Laplace transform [10] is named after the French mathematician Pierre-Simon Laplace. It is an important tool for solving differential equations, and it is very useful for designing and analyzing linear systems [11].

The Laplace transform is a mathematical operation that maps, or transforms, a variable (or function) from its original domain into the Laplace domain, or  $s$  domain. This transform produces a time-domain response to transitioning inputs, whenever time-domain behavior is more interesting than frequency-domain behavior. When solving engineering problems one has to model a physical phenomenon that

is dependent on the rates of change of a function (e.g., the velocity of a car as mentioned in Section 1.02.1). Hence, calculus associated with differential equations (Section 1.02.3), that model the phenomenon, are the natural candidates to be the mathematical tools. However, calculus solves (ordinary) differential equations provided the functions are continuous and with continuous derivatives. In addition, engineering problems have often to deal with impulsive, non-periodic or piecewise-defined input signals.

The Fourier transform, to be addressed in Section 1.02.7, is also an important tool for signal analysis, as well as for linear filter design. However, while the unit-step function (Figure 2.2a), discontinuous at time  $t = 0$ , has a Laplace transform, its forward Fourier integral does not converge. The Laplace transform is particularly useful for input terms that are impulsive, non-periodic or piecewise-defined [4].

The Laplace transform maps the time-domain into the  $s$ -domain, with  $s = \sigma + j\Omega$ , converting integral and differential equations into algebraic equations. The function is mapped (transformed) to the  $s$ -domain, eliminating all the derivatives. Hence, solving the equation becomes simple algebra in the  $s$ -domain and the result is transformed back to the time-domain. The Laplace transform converts a time-domain 1-D signal,  $x(t)$ , into a complex representation,  $X(s)$ , defined over a complex plane ( $s$ -plane). The complex plane is spanned by the variables  $\sigma$  (real axis) and  $\Omega$  (imaginary axis) [5].

The *two-sided* (or bilateral) *Laplace transform* of a signal  $x(t)$  is the function  $\mathcal{L}\{x(t)\}$  defined by:

$$X(s) = \mathcal{L}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-st} dt. \quad (2.36)$$

The notation  $X(s) = \mathcal{L}\{x(t)\}$  denotes that  $X(s)$  is the Laplace transform of  $x(t)$ . Conversely, the notation  $x(t) = \mathcal{L}^{-1}\{X(s)\}$  denotes that  $x(t)$  is the inverse Laplace transform of  $X(s)$ . This relationship is expressed with the notation  $x(t) \xrightarrow{\mathcal{L}} X(s)$ .

As  $e^{-st} = e^{-(\sigma+j\Omega)t} = e^{-\sigma t}(\cos \Omega t - j \sin \Omega t)$ , Eq. (2.36) can be rewritten as

$$\mathcal{L}\{x(t)\} = X(s)|_{s=\sigma+j\Omega} = \int_{-\infty}^{\infty} x(t)e^{-\sigma t} \cos \Omega t dt - j \int_{-\infty}^{\infty} x(t)e^{-\sigma t} \sin \Omega t dt. \quad (2.37)$$

This way, one can identify that the Laplace transform real part ( $\text{Re}(X(s))$ ) represents the contribution of the (combined) exponential and cosine terms to  $x(t)$ , while its imaginary part ( $\text{Im}(X(s))$ ) represents the contribution of the (combined) exponential and sine terms to  $x(t)$ . As the term  $e^{-st}$  is an eigenfunction, we can state that the Laplace transform represents time-domain signals in the  $s$ -domain as weighted combinations of eigensignals.

For those signals equal to zero for  $t < 0$ , the limits on the integral are changed for the *one-sided* (or unilateral) *Laplace transform*:

$$X(s) = \mathcal{L}\{x(t)\} = \int_{0^-}^{\infty} x(t)e^{-st} dt, \quad (2.38)$$

with  $x(t) = 0$  for  $t < 0^-$ , in order to deal with signals that present singularities at the origin, i.e., at  $t = 0$ .

As the interest will be in signals defined for  $t \geq 0$ , let us find the one-sided Laplace transform of  $x(t) = e^t u(t)$ :

$$\mathcal{L}\{x(t)\} = X(s) = \int_{0^-}^{\infty} e^t e^{-st} dt = \int_{0^-}^{\infty} e^{(1-s)t} dt = \frac{1}{1-s} e^{(1-s)t} \Big|_{0^-}^{\infty} = \frac{1}{s-1} \quad (2.39)$$

The integral, in (2.39), defining  $\mathcal{L}\{x(t)\}$  is true if  $e^{(1-s)t} \rightarrow 0$  as  $t \rightarrow \infty$ ,  $\forall s \in \mathcal{C}$  with  $\text{Re}(s) > 1$ , which is the region of convergence of  $X(s)$ .

As the analysis of convergence, as well as the conditions that guarantee the existence of the Laplace integral are beyond the scope of this chapter, please refer to [4,5].

For a given  $x(t)$ , the integral may converge for some values of  $\sigma$ ,  $\text{Re}(s = \sigma + j\Omega)$ , but not for others. So, we have to guarantee the existence of the integral, i.e.,  $x(t)e^{-\sigma t}$  has to be absolutely integrable. The region of convergence (ROC) of the integral in the complex  $s$ -plane should be specified for each transform  $\mathcal{L}\{x(t)\}$ , that exists if and only if the argument  $s$  is inside the ROC. The transformed signal,  $X(s)$  will be well defined for a range of values in the  $s$ -domain that is the ROC, which is always given in association with the transform itself.

When  $\sigma = 0$ , so that  $s = j\Omega$ , the Laplace transform reverts to the Fourier transform, i.e.,  $x(t)$  has a Fourier transform if the ROC of the Laplace transform in the  $s$ -plane includes the imaginary axis.

For finite duration signals that are absolutely integrable, the ROC contains the entire  $s$ -plane. As  $\mathcal{L}\{x(t)\} = X(s)$  cannot uniquely define  $x(t)$ , it is necessary  $X(s)$  and the ROC. If we wish to find the Laplace transform of a one-sided real exponential function  $x(t) = e^{at} u(t)$ ,  $a \in \mathcal{R}$ , given by:

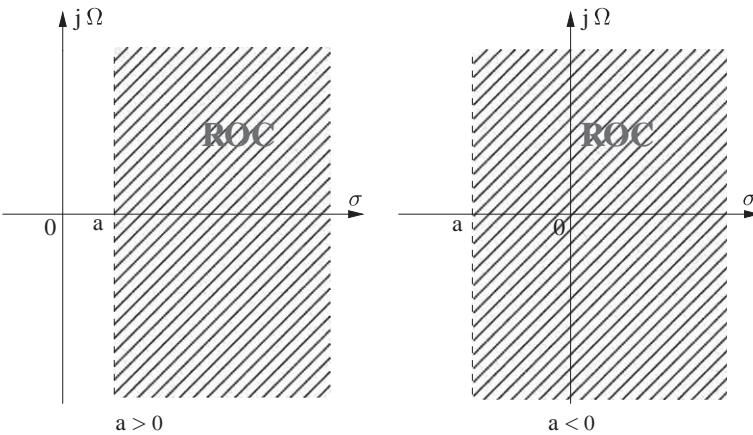
$$x(t) = \begin{cases} 0, & t \leq 0, \\ e^{at}, & t > 0, \end{cases} \quad (2.40)$$

we have

$$X(s) = \int_{0^-}^{\infty} e^{-(s-a)t} dt = -\frac{1}{s-a} e^{-(s-a)t} \Big|_{0^-}^{\infty} = \frac{1}{s-a}. \quad (2.41)$$

The integral of (2.41) converges if  $\sigma > a$ , and the ROC is the region of the  $s$ -plane to the right of  $\sigma = a$ , as pictured in Figure 2.10. If  $\sigma < a$  the integral does not converge as  $X(s) \rightarrow \infty$ , and if  $\sigma = a$  we cannot determine  $X(s)$ .

In other words, if a signal  $x(t)$  is nonzero only for  $t \geq 0$ , the ROC of its Laplace transform lies to the right hand side of its poles (please refer to Section 1.02.5). Additionally, the ROC does not contain

**FIGURE 2.10**

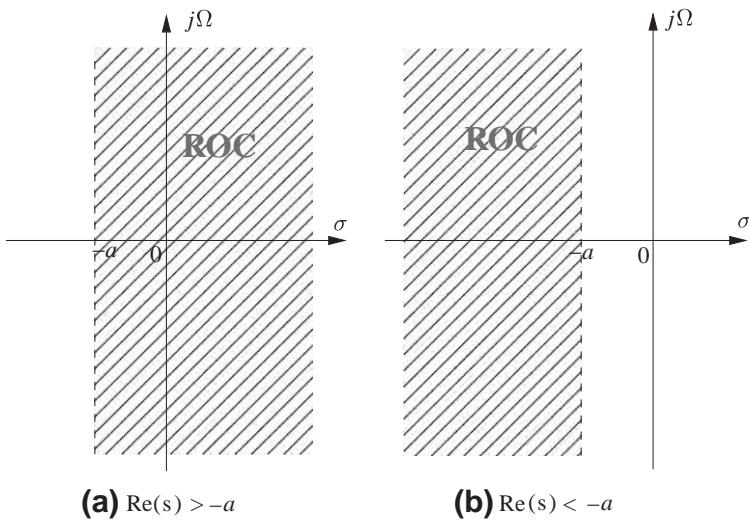
Region of Convergence (ROC) on the  $s$ -plane for the signal defined by Eq. (2.40). Note that, for a stable signal ( $a < 0$ ), the ROC contains the vertical axis  $s = j\Omega$ .

any pole. Poles are points where the Laplace transform reaches infinite value in the  $s$ -plane (e.g.,  $s = a$  in Figure 2.10).

An example of the two-sided Laplace transform of a right-sided exponential function  $x(t) = e^{-at} u(t)$ ,  $a \in \mathbb{R}$ , is given by

$$\begin{aligned}
 X(s) &= \int_{-\infty}^{\infty} e^{-at} u(t) e^{-st} dt \\
 &= \int_{0^-}^{\infty} e^{-(s+a)t} dt \\
 &= \int_{0^-}^{\infty} e^{-(\sigma+a)t} e^{-j\Omega t} dt \\
 &= -\frac{1}{s+a} e^{-(s+a)t} \Big|_{0^-}^{\infty} = \frac{1}{s+a}.
 \end{aligned} \tag{2.42}$$

As the term  $e^{-j\Omega t}$  is sinusoidal, only  $\sigma = \text{Re}(s)$  is important, i.e., the integral given by (2.42), when  $t$  tends to infinity, converges if  $e^{-(\sigma+a)t}$  is finite ( $\sigma > -a$ ) in order for the exponential function to decay. The integral of Eq. (2.42) converges if  $\text{Re}(s+1) > 0$ , i.e., if  $\text{Re}(s) > -a$ , and the ROC is the region of the  $s$ -plane to the right of  $\sigma = -a$ , as pictured in the left side of Figure 2.11. The ROC of the Laplace transform of a right-sided signal is to the right of its rightmost pole.

**FIGURE 2.11**

Regions of Convergence (ROCs) on the  $s$ -plane: (a) for right-sided signals,  $\text{Re}(s) > -a$ , and (b) for left-sided signals,  $\text{Re}(s) < -a$ .

If, we wish to find the two-sided Laplace transform of a left-sided signal,  $x(t) = -e^{-at}u(-t)$ ,  $a \in \mathcal{R}$ , we have

$$\begin{aligned} X(s) &= - \int_{-\infty}^{\infty} e^{-at} u(-t) e^{-st} dt \\ &= - \int_{-\infty}^{0^-} e^{-(s+a)t} dt \\ &= \frac{1}{s+a} e^{-(s+a)t} \Big|_{\infty}^{0^-} = \frac{1}{s+a}. \end{aligned} \quad (2.43)$$

The integral of Eq. (2.43) converges if  $\text{Re}(s) < -a$ , and the ROC is the region of the  $s$ -plane to the left of  $\sigma = -a$ , as pictured in the right side of Figure 2.11. The ROC of the Laplace transform of a left-sided signal is to the left of its leftmost pole.

For causal systems, where  $h(t) = 0$ ,  $t < 0$ , and right-sided signals, where  $x(t) = 0$ ,  $t < 0$ , the unilateral (one-sided) and bilateral (two-sided) transforms are equal. However, it is important to stress that some properties change, such as the differentiation property reduced to  $sX(s)$  for the bilateral case. Conversely, other properties, such as convolution, hold as is, provided the system is causal and the input starts at  $t = 0^-$ .

Common Laplace transform pairs [12] are summarized in Table 2.2.

Recalling Figure 2.3 from Section 1.02.1, where  $x(t)$  is the input to a linear system with impulse response  $h(t)$ , to obtain the output  $y(t)$  one could convolve  $x(t)$  with  $h(t)$  (2.11).

**Table 2.2** Laplace Transform Pairs

$x(t) = \mathcal{L}^{-1}\{X(s)\}$	$X(s) = \mathcal{L}\{x(t)\}$	ROC
$\delta(t)$	1	for all s
$\delta(t - \tau), \tau > 0$	$e^{-s\tau}$	for all s
$u(t)$	$\frac{1}{s}$	$\text{Re}(s) > 0$
$-u(-t)$	$\frac{1}{s}$	$\text{Re}(s) < 0$
$tu(t)$	$\frac{1}{s^2}$	$\text{Re}(s) > 0$
$t^n u(t)$	$\frac{n!}{s^{n+1}}, n = 1, 2, \dots$	$\text{Re}(s) > 0$
$e^{-at} u(t)$	$\frac{1}{s + a}$	$\text{Re}(s) > -\text{Re}(a)$
$-e^{-at} u(-t)$	$\frac{1}{s + a}$	$\text{Re}(s) < -\text{Re}(a)$
$t^n e^{-at} u(t)$	$\frac{n!}{(s + a)^{n+1}}, n = 1, 2, \dots$	$\text{Re}(s) > -\text{Re}(a)$
$\sin \Omega t u(t)$	$\frac{\Omega}{s^2 + \Omega^2}$	$\text{Re}(s) > 0$
$\cos \Omega t u(t)$	$\frac{s}{s^2 + \Omega^2}$	$\text{Re}(s) > 0$
$e^{-at} \sin \Omega t u(t)$	$\frac{\Omega}{(s + a)^2 + \Omega^2}$	$\text{Re}(s) > -\text{Re}(a)$
$e^{-at} \cos \Omega t u(t)$	$\frac{s + a}{(s + a)^2 + \Omega^2}$	$\text{Re}(s) > -\text{Re}(a)$
$\sin(\Omega t + \theta) u(t)$	$\frac{s \sin \theta + \Omega \cos \theta}{s^2 + \Omega^2}$	$\text{Re}(s) > 0$
$\cos(\Omega t + \theta) u(t)$	$\frac{s \cos \theta - \Omega \sin \theta}{s^2 + \Omega^2}$	$\text{Re}(s) > 0$
$e^{-at} (\Omega \cos \Omega t - a \sin \Omega t) u(t)$	$\frac{\Omega s}{(s + a)^2 + \Omega^2}$	$\text{Re}(s) > -\text{Re}(a)$

However, if  $X(s)$  and  $Y(s)$  are the associated Laplace transforms, the (computationally demanding) operation of convolution in the time-domain is mapped into a (simple) operation of multiplication in the  $s$ -domain:

$$y(t) = h(t) * x(t) \xleftarrow{\mathcal{L}} Y(s) = H(s)X(s). \quad (2.44)$$

Equation (2.44) shows that convolution in time-domain is equivalent to multiplication in Laplace domain. In the following, some properties of the Laplace transform disclose the symmetry between operations in the time- and  $s$ -domains.

*Linearity:* If  $x_1(t) \xrightarrow{\mathcal{L}} X_1(s)$ , ROC =  $R_1$ , and  $x_2(t) \xrightarrow{\mathcal{L}} X_2(s)$ , ROC =  $R_2$ , then

$$\mathcal{L}\{a_1x_1(t) + a_2x_2(t)\} = a_1\mathcal{L}\{x_1(t)\} + a_2\mathcal{L}\{x_2(t)\} = a_1X_1(s) + a_2X_2(s), \quad (2.45)$$

with  $a_1$  and  $a_2$  being constants and  $\text{ROC} \supseteq R_1 \cap R_2$ . The Laplace transform is a linear operation.

*Time shifting:* If  $x(t) \xrightarrow{\mathcal{L}} X(s)$ , ROC =  $R$ , then  $x(t - t_0) \xrightarrow{\mathcal{L}} e^{-st_0}X(s)$ ,  $\forall t_0 \in \mathbb{R}$ , ROC =  $R$ .

$$\begin{aligned} \int_{0^-}^{\infty} x(t - t_0)e^{-st} dt &= \int_{0^-}^{\infty} x(\tau)e^{-s(\tau+t_0)} d\tau \\ &= e^{-st_0} \int_{0^-}^{\infty} x(\tau)e^{-s(\tau)} d\tau \\ &= e^{-st_0} X(s), \end{aligned} \quad (2.46)$$

where  $t - t_0 = \tau$ . Time shifting (or time delay) in the time-domain is equivalent to modulation (alteration of the magnitude and phase) in the  $s$ -domain.

*Exponential scaling (frequency shifting):* If  $x(t) \xrightarrow{\mathcal{L}} X(s)$ , ROC =  $R$ , then  $e^{at}x(t) \xrightarrow{\mathcal{L}} X(s-a)$ , ROC =  $R + \text{Re}(a)$ ,  $\forall a \in \mathbb{R}$  or  $\mathcal{C}$ .

$$\mathcal{L}\{e^{at}x(t)\} = \int_{0^-}^{\infty} e^{at}x(t)e^{-st} dt = \int_{0^-}^{\infty} x(t)e^{-(s-a)t} dt = X(s-a). \quad (2.47)$$

Modulation in the time-domain is equivalent to shifting in the  $s$ -domain.

*Convolution:* If  $x_1(t) \xrightarrow{\mathcal{L}} X_1(s)$ , ROC =  $R_1$ , and  $x_2(t) \xrightarrow{\mathcal{L}} X_2(s)$ , ROC =  $R_2$ , then  $x_1(t) * x_2(t) \xrightarrow{\mathcal{L}} X_1(s)X_2(s)$ , ROC  $\supseteq R_1 \cap R_2$ :

$$\begin{aligned} \int_{0^-}^{\infty} [x_1(t) * x_2(t)]e^{-st} dt &= \int_{0^-}^{\infty} \left[ \int_{0^-}^{\infty} x_1(\tau)x_2(t-\tau) d\tau \right] e^{-st} dt \\ &= \int_{0^-}^{\infty} \int_{0^-}^{\infty} x_1(\tau)x_2(t-\tau)e^{-s(t-\tau+\tau)} d\tau dt \\ &= \int_{0^-}^{\infty} x_1(\tau)e^{-s\tau} d\tau \int_{0^-}^{\infty} x_2(v)e^{-sv} dv \\ &= X_1(s)X_2(s), \end{aligned} \quad (2.48)$$

where in the inner integral  $v = t - \tau$ . Convolution in time-domain is equivalent to multiplication in the  $s$ -domain.

*Differentiation in time:* If  $x(t) \xrightarrow{\mathcal{L}} X(s)$ , ROC =  $R$ , then  $\dot{x}(t) \xrightarrow{\mathcal{L}} sX(s) - x(0^-)$ , ROC  $\supseteq R$ . Integrating by parts, we have:

$$\begin{aligned} \int_{0^-}^{\infty} \frac{dx(t)}{dt} e^{-st} dt &= x(t)e^{-st} \Big|_{0^-}^{\infty} - \int_{0^-}^{\infty} x(t)(-s)e^{-st} dt \\ &= 0 - x(0^-) + s \int_{0^-}^{\infty} x(t)e^{-st} dt \\ &= sX(s) - x(0^-). \end{aligned} \quad (2.49)$$

where any jump from  $t = 0^-$  to  $t = 0^+$  is considered. In other words, the Laplace transform of a derivative of a function is a combination of the transform of the function, multiplicated by  $s$ , and its initial value. This property is quite useful as a differential equation in time can be turned into an algebraic equation in the Laplace domain, where it can be solved and mapped back into the time-domain (Inverse Laplace Transform).

The initial- and final-value theorems show that, the initial and the final values of a signal in the time-domain can be obtained from its Laplace transform without knowing its expression in the time-domain. *Initial-value theorem:* Considering stable the LTI system, that generated the signal  $x(t)$ , then

$$\lim_{s \rightarrow \infty} sX(s) = \lim_{t \rightarrow 0^+} x(t). \quad (2.50)$$

From the differentiation in time property, we have that  $\dot{x}(t) \xleftrightarrow{\mathcal{L}} sX(s) - x(0^-)$ . By taking the limit when  $s \rightarrow \infty$  of the first expression of (2.49), we find

$$\begin{aligned} \lim_{s \rightarrow \infty} \int_{0^-}^{\infty} \frac{dx(t)}{dt} e^{-st} dt &= \lim_{s \rightarrow \infty} \left( \int_{0^-}^{0^+} \frac{dx(t)}{dt} e^0 dt + \int_{0^+}^{\infty} \frac{dx(t)}{dt} e^{-st} dt \right) \\ &= \lim_{t \rightarrow 0^+} x(t) - \lim_{t \rightarrow 0^-} x(t) + 0 \\ &= \lim_{t \rightarrow 0^+} x(t) - \lim_{t \rightarrow 0^-} x(t). \end{aligned} \quad (2.51)$$

If we take the limit of  $s$  to  $\infty$  in the result of (2.49), we have

$$\lim_{s \rightarrow \infty} (sX(s) - x(0^-)) = \lim_{s \rightarrow \infty} sX(s) - \lim_{t \rightarrow 0^-} x(t), \quad (2.52)$$

then we can equal the results of (2.52) and (2.51) and get

$$\lim_{s \rightarrow \infty} sX(s) - \lim_{t \rightarrow 0^-} x(t) = \lim_{t \rightarrow 0^+} x(t) - \lim_{t \rightarrow 0^-} x(t). \quad (2.53)$$

As the right-hand side of (2.53) is obtained taking the limit when  $s \rightarrow \infty$  of the result of (2.49), we can see from (2.51) and (2.53) that

$$\lim_{s \rightarrow \infty} sX(s) = \lim_{t \rightarrow 0^+} x(t). \quad (2.54)$$

The initial-value theorem provides the behavior of a signal in the time-domain for small time intervals, i.e., for  $t \rightarrow 0^+$ . In other words, it determines the initial values of a function in time from its expression in the  $s$ -domain, which is particularly useful in circuits and systems.

*Final-value theorem:* Considering that the LTI system that generated the signal  $x(t)$  is stable, then

$$\lim_{s \rightarrow 0} sX(s) = \lim_{t \rightarrow \infty} x(t). \quad (2.55)$$

From the differentiation in time property, we have that  $\dot{x}(t) \xleftrightarrow{\mathcal{L}} sX(s) - x(0^-)$ . By taking the limit when  $s \rightarrow 0$  of the first expression of (2.49), we have

$$\begin{aligned} \lim_{s \rightarrow 0} \int_{0^-}^{\infty} \frac{dx(t)}{dt} e^{-st} dt &= \int_{0^-}^{\infty} \frac{dx(t)}{dt} dt \\ &= \lim_{t \rightarrow \infty} x(t) - \lim_{t \rightarrow 0^-} x(t). \end{aligned} \quad (2.56)$$

If we take the limit when  $s \rightarrow 0$  of the last expression of (2.49), we get

$$\lim_{s \rightarrow 0} (sX(s) - x(0^-)) = \lim_{s \rightarrow 0} sX(s) - \lim_{t \rightarrow 0^-} x(t); \quad (2.57)$$

then, we can write

$$\lim_{s \rightarrow 0} sX(s) - \lim_{t \rightarrow 0^-} x(t) = \lim_{t \rightarrow \infty} x(t) - \lim_{t \rightarrow 0^-} x(t). \quad (2.58)$$

As the right-hand side of (2.58) is obtained taking the limit when  $s \rightarrow 0$  of the result of (2.49), we can see from (2.56) and (2.58) that

$$\lim_{s \rightarrow 0} sX(s) = \lim_{t \rightarrow \infty} x(t). \quad (2.59)$$

The final-value theorem provides the behavior of a signal in the time-domain for large time intervals, i.e., for  $t \rightarrow \infty$ . In other words, it obtains the final value of a function in time, assuming it is stable and well defined when  $t \rightarrow \infty$ , from its expression in the  $s$ -domain. A LTI system, as will be seen in the next section, is considered stable if all of its poles lie within the left side of the  $s$ -plane. As  $t \rightarrow \infty$ ,  $x(t)$  must reach a steady value, thus it is not possible to apply the final-value theorem to signals such as sine, cosine or ramp.

A list of one-sided Laplace transform properties is summarized in Table 2.3.

The inverse Laplace transform is given by

$$\mathcal{L}^{-1}\{X(s)\} = x(t) = \int_{\sigma-j\infty}^{\sigma+j\infty} X(s)e^{st} ds. \quad (2.60)$$

The integration is performed along a line, parallel to the imaginary axis,  $(\sigma - j\infty, \sigma + j\infty)$  that lies in the ROC. However, the inverse transform can be calculated using partial fractions expansion with the method of residues. In this method, the  $s$ -domain signal  $X(s)$  is decomposed into partial fractions, thus expressing  $X(s)$  as a sum of simpler rational functions. Hence, as each term in the partial fraction is expected to have a known inverse transform, each transform may be obtained from a table like Table 2.2. Please recall that the ROC of a signal that is non-zero for  $t \geq 0$  is located to the right hand side of its poles. Conversely, for a signal that is non-zero for  $t \leq 0$ , the ROC of its Laplace transform lies to the left-hand side of its poles. In other words, if  $X(s) = 1/(s + a)$ ,  $a \in \mathcal{R}$ , its inverse Laplace transform is given as follows:

$$\mathcal{L}^{-1}\left\{\frac{1}{s+a}\right\} = \begin{cases} -e^{-at}, & \text{Re}(s) < -a, \\ e^{-at}, & \text{Re}(s) > -a, \end{cases} \quad (2.61)$$

In practice, the inverse Laplace transform is found recursing to tables of transform pairs (e.g., Table 2.2). Given a function  $X(s)$  in the  $s$ -domain and a region of convergence, its inverse Laplace transform is given by  $\mathcal{L}^{-1}\{X(s)\} = x(t)$  such that  $X(s) = \mathcal{L}\{x(t)\}$ ,  $s \in \text{ROC}$ .

An immediate application of the Laplace transform is on circuit analysis. Assuming that all initial conditions are equal to zero, i.e., there is no initial charge on the capacitor, the response  $y(t)$  of the circuit with input given by  $x(t) = u(t)$  displayed by Figure 2.4 could be obtained in the Laplace domain to be further transformed into the time-domain.

**Table 2.3** Properties of (One-Sided) Laplace Transform

Property	time-domain	s-domain	ROC
Linearity	$x(t)$	$X(s)$	$R$
	$x_1(t)$	$X_1(s)$	$R_1$
	$x_2(t)$	$X_2(s)$	$R_2$
	$a_1x_1(t) + a_2x_2(t)$	$a_1X_1(s) + a_2X_2(s)$	$\supseteq R_1 \cap R_2$
	$ax(t)$	$aX(s)$	$R$
	$x(at)$	$\frac{1}{a}X\left(\frac{s}{a}\right)$	$\frac{R}{ a }$
	$x(t - t_0)$	$e^{-st_0}X(s)$	$R$
	$e^{at}x(t)$	$X(s - a)$	$R + \text{Re}(a)$
	$\frac{dx(t)}{dt}$	$sX(s) - x(0^-)$	$\supseteq R$
Differentiation	$\frac{d^2x(t)}{dt^2}$	$s^2X(s) - sx(0^-) - \dot{x}(0^-)$	
	$\frac{d^n x(t)}{dt^n}$	$s^n X(s) - \sum_{k=1}^n s^{n-k} x^{(k-1)}(0^-)$	
	$-tx(t)$	$\frac{dX(s)}{ds}$	$R$
	$x_1(t) * x_2(t)$	$X_1(s)X_2(s)$	$\supseteq R_1 \cap R_2$
Convolution (in the s-domain)	$x_1(t)x_2(t)$	$\frac{1}{2\pi j}X_1(s) * X_2(s)$	$\supseteq R_1 \cap R_2$

First, the circuit elements are transformed from the time domain into the  $s$ -domain, thus creating an  $s$ -domain equivalent circuit. Hence, the RLC elements in the time domain and  $s$ -domain are:

*Resistor* (voltage-current relationship):

$$v(t) = Ri(t) \xleftrightarrow{\mathcal{L}} V(s) = RI(s). \quad (2.62)$$

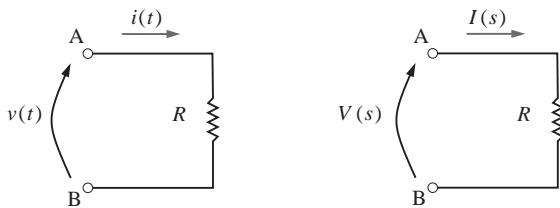
The equivalent circuit is depicted in Figure 2.12.

*Inductor* (initial current  $i(0^-)$ ):

$$v(t) = L \frac{di(t)}{dt} \xleftrightarrow{\mathcal{L}} V(s) = sLI(s) - Li(0^-); \quad (2.63)$$

applying the differentiation property (Table 2.3) leads to

$$I(s) = \frac{1}{sL}V(s) + \frac{i(0^-)}{s}. \quad (2.64)$$

**FIGURE 2.12**

Equivalent circuit of a resistor in the Laplace domain.

The equivalent circuit is depicted in Figure 2.13. The inductor,  $L$ , is an impedance  $sL$  in the  $s$ -domain in series with a voltage source,  $Li(0^-)$ , or in parallel with a current source,  $\frac{i(0^-)}{s}$ .  
*Capacitor* (initial voltage  $v(0^-)$ ):

$$i(t) = C \frac{dv(t)}{dt} \xleftrightarrow{\mathcal{L}} I(s) = sCV(s) - v(0^-); \quad (2.65)$$

applying the differentiation property (Table 2.3) leads to

$$V(s) = \frac{I(s)}{sC} + \frac{v(0^-)}{s}. \quad (2.66)$$

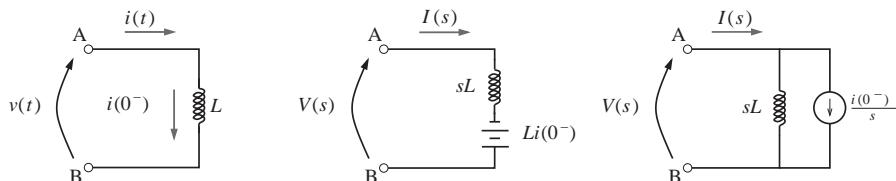
The capacitor,  $C$ , is an impedance  $\frac{1}{sC}$  in the  $s$ -domain in series with a voltage source,  $\frac{v(0^-)}{s}$ , or in parallel with a current source,  $Cv(0^-)$ . The voltage across the capacitor in the time-domain corresponds to the voltage across both the capacitor and the voltage source in the frequency domain.

As an example, the system's response given by the voltage across the capacitor,  $y(t)$ , depicted in Figure 2.4, is obtained in the Laplace domain as follows (see Figure 2.14):

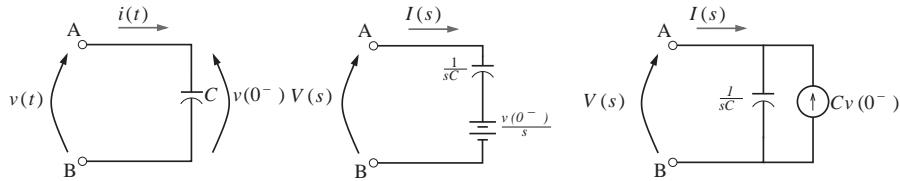
$$X(s) = \frac{1}{s} = I(s) \left( R + \frac{1}{sC} \right). \quad (2.67)$$

From (2.65), we have

$$Y(s) = \frac{I(s)}{sC} \quad (2.68)$$

**FIGURE 2.13**

Equivalent circuit of an inductor in the Laplace domain.

**FIGURE 2.14**

Equivalent circuit of a capacitor in the Laplace domain.

with

$$I(s) = \frac{X(s)}{R + \frac{1}{sC}}. \quad (2.69)$$

Substituting (2.69) in (2.68) and applying the inverse Laplace transform (Table 2.2) and the linearity property (Table 2.3), we get

$$\begin{aligned} \mathcal{L}^{-1}\{Y(s)\} &= \mathcal{L}^{-1}\left\{\frac{1}{s} - \frac{1}{s + \frac{1}{RC}}\right\}, \\ y(t) &= u(t) - e^{-t\frac{1}{RC}}u(t), \end{aligned} \quad (2.70)$$

which is the same result presented in (2.5).

Another example, where the Laplace transform is useful, is the RLC circuit displayed by Figure 2.8 where the desired response is the voltage across the capacitor  $C$ ,  $v_o(t) = v_C(t)$ . Note that the initial conditions are part of the transform, as well as the transient and steady-state responses. Given the input signal  $v_i(t) = e^{-t}u(t)$  with initial conditions  $v_o(0^-) = 1$ ,  $\dot{v}_o(0^-) = 0$ , and  $i_o(0^-) = 0$ , applying the Kirchhoff's voltage law to the circuit, we have:

$$v_i(t) = v_R(t) + v_L(t) + v_C(t). \quad (2.71)$$

Directly substituting (2.62), (2.63) and (2.65) in (2.71), and applying the Laplace transform to the input signal,  $v_i(t) = e^{-t}u(t)$ , we get

$$\begin{aligned} \frac{1}{s+1} &= \left( RI(s) + sLI(s) - Li_o(0^-) + \frac{1}{sC}I(s) + \frac{v_o(0^-)}{s} \right) \\ &= I(s) \left( R + sL + \frac{1}{sC} \right) - Li_o(0^-) + \frac{v_o(0^-)}{s} \\ &= \frac{I(s)}{sC} (s^2LC + sRC + 1) - Li_o(0^-) + \frac{v_o(0^-)}{s}, \end{aligned} \quad (2.72)$$

which, from (2.65),  $\frac{I(s)}{sC} = V_o(s) - \frac{v_o(0^-)}{s}$ , we have

$$\begin{aligned}\frac{1}{s+1} &= \left(V_o(s) - \frac{v_o(0^-)}{s}\right)(s^2LC + sRC + 1) - Li_o(0^-)\frac{v_o(0^-)}{s} \\ &= V_o(s)(s^2LC + sRC + 1) - \frac{v_o(0^-)}{s}(s^2LC + sRC + 1) \\ &\quad - Li_o(0^-) + \frac{v_o(0^-)}{s}.\end{aligned}\tag{2.73}$$

Substituting the values of  $R = 2.0$  (in  $\Omega$ ),  $L = 2.0$  (in H),  $C = 1/2$  (in F), the equation becomes

$$\begin{aligned}\frac{1}{s+1} &= V_o(s)(s^2 + s + 1) - \frac{v_o(0^-)}{s}(s^2 + s + 1) \\ &\quad - i_o(0^-) + \frac{v_o(0^-)}{s};\end{aligned}\tag{2.74}$$

considering that the initial conditions are  $v_0(0^-) = 1$ ,  $\dot{v}_o(0^-) = 0$ ,  $i_o(0^-) = 0$ , we get

$$\begin{aligned}\frac{1}{s+1} &= V_o(s)(s^2 + s + 1) - \frac{1}{s}(s^2 + s + 1) - 0 + \frac{1}{s} \\ &= V_o(s)(s^2 + s + 1) - s - 1 - \frac{1}{s} - 0 + \frac{1}{s} \\ &= V_o(s)(s^2 + s + 1) - s - 1 \\ V_o(s) &= \left(\frac{1}{s+1} + s + 1\right) \left(\frac{1}{s^2 + s + 1}\right) \\ &= \left(\frac{s^2 + 2s + 2}{s+1}\right) \left(\frac{1}{s^2 + s + 1}\right).\end{aligned}\tag{2.75}$$

After decomposing (2.75) into partial fractions and finding the poles and residues, the inverse Laplace transform is applied in order to find the expression of  $v_o(t)$ :

$$V_o(s) = \frac{1}{(s+1)} - j \frac{\frac{\sqrt{3}}{3}}{\left(s + \frac{1}{2} - j \frac{\sqrt{3}}{2}\right)} + j \frac{\frac{\sqrt{3}}{3}}{\left(s + \frac{1}{2} + j \frac{\sqrt{3}}{2}\right)}\tag{2.76}$$

$$\mathcal{L}^{-1}\{V_o(s)\} = \mathcal{L}^{-1} \left\{ \frac{1}{(s+1)} - j \frac{\frac{\sqrt{3}}{3}}{\left(s + \frac{1}{2} - j \frac{\sqrt{3}}{2}\right)} + j \frac{\frac{\sqrt{3}}{3}}{\left(s + \frac{1}{2} + j \frac{\sqrt{3}}{2}\right)} \right\}.\tag{2.77}$$

Applying the linearity property (Table 2.3) and recursing to the Laplace transform pair table (Table 2.2), we get

$$v_o(t) = e^{-t} - j \frac{\sqrt{3}}{3} e^{-\frac{t}{2} + jt \frac{\sqrt{3}}{2}} + j \frac{\sqrt{3}}{3} e^{-\frac{t}{2} - jt \frac{\sqrt{3}}{2}},\tag{2.78}$$

considering that the inverse Laplace transforms of  $\frac{1}{s+a}$ , for  $\text{Re}(s) > -\text{Re}(a)$ , given by Table 2.2, with  $a = \frac{1}{2} - j\frac{\sqrt{3}}{2}$  and  $a = \frac{1}{2} + j\frac{\sqrt{3}}{2}$  are

$$\begin{aligned}\mathcal{L}^{-1} \left\{ \frac{1}{(s + \frac{1}{2} - j\frac{\sqrt{3}}{2})} \right\} &= e^{-(\frac{1}{2} - j\frac{\sqrt{3}}{2})t} \text{ and,} \\ \mathcal{L}^{-1} \left\{ \frac{1}{(s + \frac{1}{2} + j\frac{\sqrt{3}}{2})} \right\} &= e^{-(\frac{1}{2} + j\frac{\sqrt{3}}{2})t},\end{aligned}$$

respectively.

Algebraically manipulating (2.78) and substituting the terms of complex exponentials by trigonometric identities, we have (for  $t \geq 0$ )

$$\begin{aligned}v_o(t) &= 2\frac{\sqrt{3}}{2}e^{-\frac{t}{2}} \sin\left(\frac{\sqrt{3}}{2}t\right) + e^{-t} \\ &= 1.1547 \sin(0.866t)e^{-\frac{t}{2}} + e^{-t}.\end{aligned}\tag{2.79}$$

The solution given by (2.79) is the same given by (2.33). However, in the solution obtained using the Laplace transform, the initial conditions were part of the transform. We could also have applied the Laplace transform directly to the ODE (2.30) assisted by Tables 2.3 and 2.2, as follows:

$$\mathcal{L}\{e^{-t}\} = \mathcal{L}\left\{ \frac{d^2v_o(t)}{dt^2} + \frac{dv_o(t)}{dt} + v_o(t) \right\},\tag{2.80}$$

where

$$\begin{aligned}\mathcal{L}\left\{ \frac{d^2v_o(t)}{dt^2} \right\} &= s^2V_o(s) - sv_o(0^-) - \dot{v}_o(0^-), \\ \mathcal{L}\left\{ \frac{dv_o(t)}{dt} \right\} &= sV_o(s) - v_o(0^-), \\ \mathcal{L}\{v_o(t)\} &= V_o(s), \text{ and} \\ \mathcal{L}\{e^{-t}\} &= \frac{1}{s+1}.\end{aligned}\tag{2.81}$$

Hence, substituting the expressions from (2.81) into (2.80) and considering the initial conditions, we obtain the same final expression presented in (2.75):

$$\begin{aligned}
 \frac{1}{s+1} &= V_o(s)(s^2 + s + 1) - sv_o(0^-) - v'_o(0^-) - v_o(0^-) \\
 &= V_o(s)(s^2 + s + 1) - s - 0 - 1 \\
 &= V_o(s)(s^2 + s + 1) - s - 1 \\
 V_o(s) &= \left( \frac{1}{s+1} + s + 1 \right) \left( \frac{1}{s^2 + s + 1} \right) \\
 &= \left( \frac{s^2 + 2s + 2}{s+1} \right) \left( \frac{1}{s^2 + s + 1} \right). \tag{2.82}
 \end{aligned}$$

In Section 1.02.2 the importance of working with a LTI system was introduced. The convolution integral in (2.9), that expresses the input signal,  $x(t)$ , as a sum of scaled and shifted impulse functions, uses delayed unit impulses as its basic signals. Therefore, a LTI system response is the same linear combination of the responses to the basic inputs.

Considering that complex exponentials, such as  $e^{-st}$ , are eigensignals of LTI systems, Eq. (2.36) is defined if one uses  $e^{-st}$  as a basis for the set of all input functions in a linear combination made of infinite terms (i.e., an integral). As the signal is being decomposed in basic inputs (complex exponentials), the LTI system's response could be characterized by weighting factors applied to each component in that representation.

In Section 1.02.7, the Fourier transform is introduced and it is shown that the Fourier integral does not converge for a large class of signals. The Fourier transform may be considered a subset of the Laplace transform, or the Laplace transform could be considered a generalization (or expansion) of the Fourier transform. The Laplace integral term  $e^{-st}$ , from (2.36), forces the product  $x(t)e^{-st}$  to zero as time  $t$  increases. Therefore, it could be regarded as the exponential weighting term that provides convergence to functions for which the Fourier integral does not converge.

Following the concept of representing signals as a linear combination of eigensignals, instead of choosing  $e^{-(\sigma+j\Omega)t}$  as the eigensignals, one could choose  $e^{-j\Omega t}$  as the eigensignals. The latter representation leads to the Fourier transform equation, and any LTI system's response could be characterized by the amplitude scaling applied to each of the basic inputs  $e^{-j\Omega t}$ .

Laplace transform and applications are discussed in greater detail in [2,4,5,8,10,11].

## 1.02.5 Transfer function and stability

The Laplace transform, seen in the previous section, is an important tool to solve differential equations and therefore to obtain, once given the input, the output of a linear and time invariant (LTI) system. For a LTI system, the Laplace transform of its mathematical representation, given as a differential equation—with constant coefficients and null initial conditions—as in (2.22) or, equivalently, given by a convolution integral as in (2.10), leads to the concept of *transfer function*, the ratio

$$H(s) = \frac{Y(s)}{X(s)} \tag{2.83}$$

between the Laplace transform of the output signal and the Laplace transform of the input signal. This representation of a LTI system also corresponds to the Laplace transform of its impulse response, i.e.,  $H(s) = \mathcal{L}\{h(t)\}$ .

Assuming that all initial conditions are equal to zero, solving a system using the concept of transfer function is usually easier for the convolution integral is replaced by a multiplication in the transformed domain:

$$Y(s) = \mathcal{L}\{h(t) * x(t)\} = H(s)X(s) \quad (2.84)$$

such that

$$y(t) = \mathcal{L}^{-1}\{H(s)X(s)\}, \quad (2.85)$$

the zero-state response of this system.

A simple example is given from the circuit in Figure 2.4 where

$$H(s) = \frac{1/RC}{s + 1/RC}.$$

If  $x(t) = u(t)$ ,  $X(s) = 1/s$ , and

$$\begin{aligned} Y(s) &= H(s)X(s) = \frac{1/RC}{(s + 1/RC)s} \\ &= \frac{1}{s} - \frac{1}{s + 1/RC}. \end{aligned}$$

Therefore,  $y(t) = \mathcal{L}^{-1}\{Y(s)\}$  corresponds to

$$y(t) = u(t) - e^{-\frac{1}{RC}t}u(t)$$

which is the same solution given in (2.5).

Given (2.10) and provided that all initial conditions are null, the transfer function  $H(s)$  corresponds to a ratio of two polynomials in  $s$ :

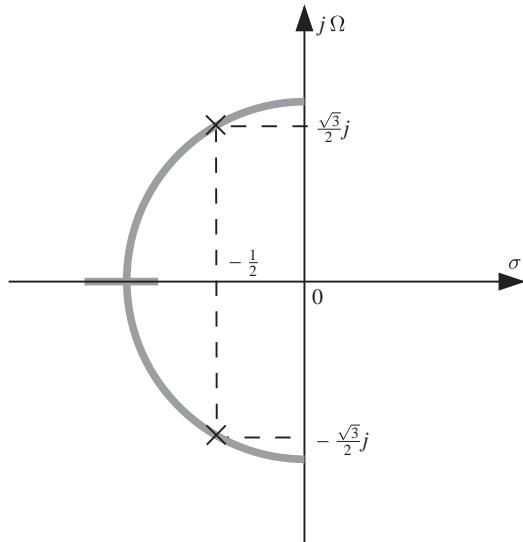
$$H(s) = \frac{P(s)}{Q(s)}. \quad (2.86)$$

The roots of the numerator  $P(s)$  are named *zeros* while the roots of the denominator  $Q(s)$  are known as *poles*. Both are usually represented in the complex plane  $s = \sigma + j\Omega$  and this plot is referred to as *pole-zero plot* or *pole-zero diagram*.

We use the ODE (2.29) to provide an example:

$$H(s) = \frac{V_o(s)}{V_i(s)} = \frac{1}{LCs^2 + RCs + 1}. \quad (2.87)$$

Using the same values,  $R = 2.0$  (in  $\Omega$ ),  $L = 2.0$  (in H),  $C = 1/2$  (in F), we obtain the poles (roots of the characteristic equation)  $s = \frac{-1 \pm j\sqrt{3}}{2}$  as seen in Figure 2.15.



**FIGURE 2.15**

An example of a pole-zero diagram of the transfer function in (2.87) for  $R = 2.0$  (in  $\Omega$ ),  $L = 2.0$  (in H),  $C = 1/2$  (in F). The gray curve corresponds to the root-locus of the poles when  $R$  varies from  $R = 0$  (poles at  $\pm j$ ) to  $R = 4.1$  (poles at  $-1.25$  and at  $-0.8$ ). See [Video 4](#) to watch animation.

For a LTI system, a complex exponential  $e^{st}$  can be considered an eigensignal:

$$\begin{aligned} x(t) = e^{st} \rightarrow y(t) &= e^{st} * h(t) = \int_{-\infty}^{\infty} h(\tau) e^{s(t-\tau)} d\tau \\ &= e^{st} \underbrace{\int_{-\infty}^{\infty} h(\tau) e^{-s\tau} d\tau}_{H(s)} \\ &= H(s)e^{st} = \frac{P(s)}{Q(s)}e^{st}, \end{aligned} \quad (2.88)$$

which is valid only for those values of  $s$  where  $H(s)$  exists.

If there is no common root between  $P(s)$  and  $Q(s)$ , the denominator of  $H(s)$  corresponds to the characteristic polynomial and, therefore, poles  $p_i$  of a LTI system correspond to their *natural frequencies*.

As mentioned in Section 1.02.2, the *stability* of a system, in a *bounded-input bounded-output* (BIBO) sense, implies that if  $|x(t)| < B_x$  then  $|y(t)| < B_y$ ,  $B_x$  and  $B_y < \infty$ , for all values of  $t$ . This corresponds to its input response being absolutely integrable, i.e.,

$$\int_{-\infty}^{\infty} |h(t)| dt = \|h(t)\|_1 < \infty. \quad (2.89)$$

As seen previously, the natural response of a linear system corresponds to a linear combination of complex exponentials  $e^{p_i t}$ . Let us assume, for convenience, that  $H(s)$  has  $N$  distinct poles and that the order of  $P(s)$  is lower than the order of  $Q(s)$ ; in that case, we can write

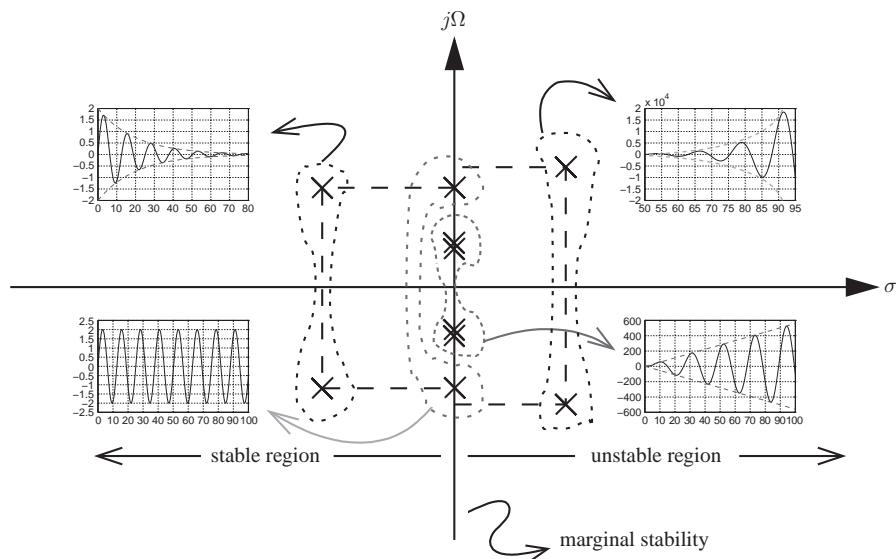
$$h(t) = \mathcal{L}^{-1} \left\{ \frac{P(s)}{\prod_{i=1}^N (s - p_i)} \right\} = \sum_{i=1}^N A_i e^{p_i t} u(t), \quad (2.90)$$

where constants  $A_i$  are obtained from simple partial fraction expansion.

In order for the system to be stable, each exponential should tend to zero as  $t$  tends to zero. Considering a complex pole  $p_i = \sigma_i + j\Omega_i$ ,  $\lim_{t \rightarrow \infty} |h(t)| \rightarrow 0$  should imply that  $\lim_{t \rightarrow \infty} |e^{(\sigma_i + j\Omega_i)t}| = \lim_{t \rightarrow \infty} |e^{\sigma_i t}| = 0$  or  $\sigma_i < 0$ ; that is, the real part of  $p_i$  should be negative.

While the location of the *zeros* of  $H(s)$  is irrelevant for the stability of a LTI system, their *poles* are of paramount importance. We summarize this relationship in the following:

1. A causal LTI system is BIBO stable if and only if all poles of its transfer function have negative real part (belong to the left-half of the  $s$ -plane).
2. A causal LTI system is unstable if and only if at least one of the following conditions occur: one pole of  $H(s)$  has a positive real part and repeated poles of  $H(s)$  have real part equal to zero (belong to the imaginary axis of the  $s$ -plane).
3. A causal LTI system is marginally stable if and only if there are no poles on the right-half of the  $s$ -plane and non-repeated poles occur on its imaginary axis.

**FIGURE 2.16**

Pole location on the  $s$ -plane and system stability: examples of impulse responses of causal LTI systems. Visit the “exploring the  $s$ -plane” <http://www.jhu.edu/signals/explore/index.html> website.

We can also state system stability from the region of convergence of  $H(s)$ : a LTI system is stable if and only if the ROC of  $H(s)$  includes the imaginary axis ( $s = j\Omega$ ). Figure 2.16 depicts examples of impulse response for BIBO stable, marginally stable, and unstable systems.

In this section, we have basically addressed BIBO stability which, although not sufficient for asymptotic stability (a system can be BIBO stable without being stable when initial conditions are not null), is usually employed for linear systems. A more thorough stability analysis could be carried out by using Lyapunov criteria [13].

The Laplace transform applied to the input response of the differential equation provides the transfer function which poles (roots of its denominator) determines the system stability: a causal LTI system is said to be stable when all poles lie in the left half of the complex  $s$ -plane. In this case, the system is also known as asymptotically stable for its output always tend to decrease, not presenting permanent oscillation.

Whenever distinct poles have their real part equal to zero (poles on the imaginary axis), permanent oscillation will occur and the system is marginally stable, the output signal does not decay nor grows indefinitely. Figure 2.16 shows the impulse response growing over time when two repeated poles are located on the imaginary axis. Although not shown in Figure 2.16, decaying exponential will take place of decaying oscillation when poles are real (and negative) while growing exponentials will appear for the case of real positive poles.

Although we have presented the key concepts of stability related to a linear system, much more could be said about stability theory. Hence, further reading is encouraged: [2, 13].

### 1.02.6 Frequency response

We have mentioned in the previous section that the complex exponential  $e^{st}$  is an eigensignal of a continuous-time linear system. A particular choice of  $s$  being  $j\Omega_0 = j2\pi f_0$ , i.e., the input signal

$$x(t) = e^{j\Omega_0 t} = \cos(\Omega_0 t) + j \sin(\Omega_0 t) \quad (2.91)$$

has a single frequency and the output, from (2.88), is

$$y(t) = e^{j\Omega_0 t} \underbrace{\int_{-\infty}^{\infty} h(\tau) e^{-j\Omega_0 \tau} d\tau}_{H(j\Omega_0)} \quad (2.92)$$

Being  $e^{j\Omega_0 t}$  an eigensignal,  $H(j\Omega_0)$  in (2.92) corresponds to its eigenvalue. Allowing a variable frequency  $\Omega = 2\pi f$  instead of a particular value  $\Omega_0 = 2\pi f_0$ , we define the *frequency response* of a linear system having impulse response  $h(t)$  as

$$H(j\Omega) = \int_{-\infty}^{\infty} h(\tau) e^{-j\Omega\tau} d\tau. \quad (2.93)$$

We note that the frequency response corresponds to the Laplace transform of  $h(t)$  on a specific region of the  $s$ -plane, the vertical (or imaginary) axis  $s = j\Omega$ :

$$H(j\Omega) = H(s)|_{s=j\Omega}. \quad (2.94)$$

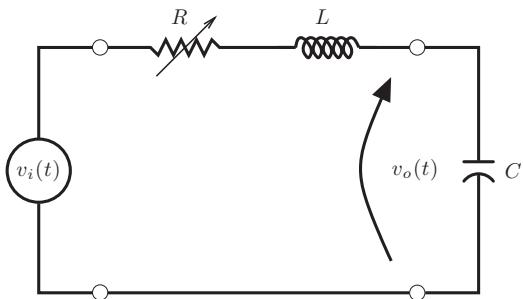
It is clear from (2.93) that the frequency response  $H(j\Omega)$  of a linear system is a complex function of  $\Omega$ . Therefore, it can be represented in its polar form as

$$H(j\Omega) = |H(j\Omega)|e^{\angle H(j\Omega)}. \quad (2.95)$$

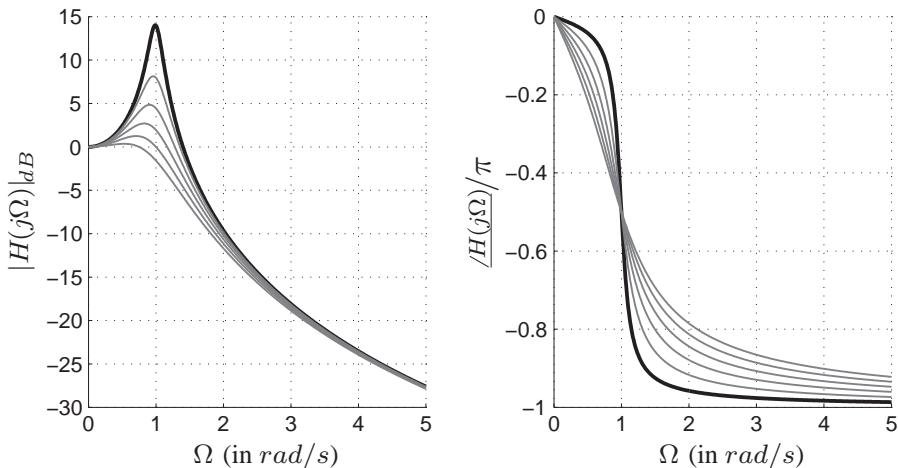
As an example, we use the RLC series circuit given in Figure 2.17 with  $L = 1.0$  (in H),  $C = 1.0$  (in F), and  $R$  varying from  $0.2\Omega$  to  $1.2\Omega$  (in  $\Omega$ ).

The frequency response, magnitude or absolute value (in dB) and argument or phase (in radians), are depicted in Figure 2.18. For this example, we consider the voltage applied to the circuit as input and the voltage measured at the capacitor as output. This is worth mentioning since the output could be, for instance, the current in the circuit or the voltage across the inductor.

In low frequencies, the capacitor tends to become an open circuit such that  $v_o(t) = v_i(t)$  and the gain tends to 1 or 0 dB. On the other hand, as the frequency  $\Omega$  goes towards infinity, the capacitor tends to become a short circuit such that  $v_o(t)$  goes to zero, gain in dB tending to minus infinity. The circuit behaves like a low-pass filter allowing low frequencies to the output while blocking high frequencies. Specially, when  $R$  has low values (when tending to zero), we observe a peak in  $|H(j\Omega)|$  at  $\Omega \approx \Omega_0 = 1$  rad/s. This *frequency*,  $\Omega_0 = 2\pi f_0$ , is termed the resonance frequency and corresponds to the absolute value of the pole responsible for this *oscillation*.

**FIGURE 2.17**

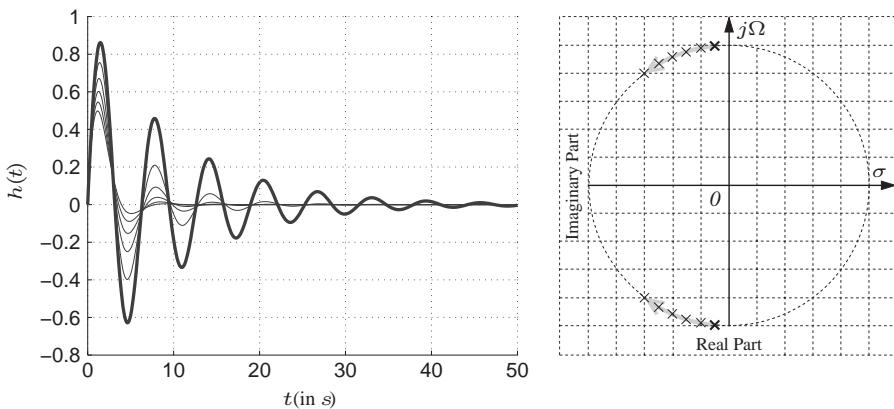
RLC series circuit: the transfer function is  $H(s) = \frac{V_o(s)}{V_i(s)}$ ,  $V_o(s) = \mathcal{L}\{v_0(t)\}$  and  $V_i(s) = \mathcal{L}\{v_i(t)\}$ , and the frequency response is given by  $H(j\Omega) = H(s)|_{s=j\Omega}$ .

**FIGURE 2.18**

Frequency response in magnitude ( $|H(j\Omega)|$  in dB) and normalized phase (argument of  $H(j\Omega)$  divided by  $\pi$ , i.e.,  $-1$  corresponds to  $-\pi$ ) for the circuit in Figure 2.17 with  $L = 1$  (in H),  $C = 1$  (in F), and  $R$  varying from 0.2 (in  $\Omega$ ) (highlighted highest peak in magnitude) to 1.2 (in  $\Omega$ ). Download the <http://www.ime.eb.br/~apolin/CTSS/Figs18and19.m> Matlab<sup>®</sup> code.

Let us write the frequency response from  $H(s)$ , as given in (2.87):

$$H(j\Omega) = H(s)|_{s=j\Omega} = \frac{1}{(1 - \Omega^2 LC) + j\Omega RC}. \quad (2.96)$$

**FIGURE 2.19**

Impulse responses and poles location in the  $s$ -plane (6 different values of  $R$ ) for the circuit in Figure 2.17. Note that, for lower values of  $R$ ,  $h(t)$  tends to oscillate and the poles are closer to the vertical axis (marginal stability region). Download the <http://www.ime.eb.br/~apolin/CTSS/Figs18and19.m> Matlab<sup>®</sup> code.

We note that, when  $R \rightarrow 0$ , the peak amplitude of  $|H(j\Omega)|$  occurs for  $1 - \Omega^2 LC \rightarrow 0$  (it tends to infinity as  $R$  goes to zero), i.e., the resonance frequency corresponds to  $\Omega_0 = \frac{1}{\sqrt{LC}}$ .

The impulse response as well as the pole diagram are shown in Figure 2.19; from this figure, we observe the oscillatory behavior of the circuit as  $R$  tends to zero in both time- and  $s$ -domain (poles approaching the vertical axis). The impulse response and the poles location for the minimum value of the resistance,  $R = 0.2$  (in  $\Omega$ ), are highlighted in this figure.

Before providing more details about the magnitude plot of the frequency response as a function of  $\Omega$ , let us show the (zero state) response of a linear system to a real single frequency excitation  $x(t) = A \cos(\Omega t + \theta)$ . We know that the complex exponential is an eigensignal to a linear system such that, making  $s = j\Omega$ ,

$$y(t)|_{x(t)=e^{j\Omega t}} = H(j\Omega)e^{j\Omega t}. \quad (2.97)$$

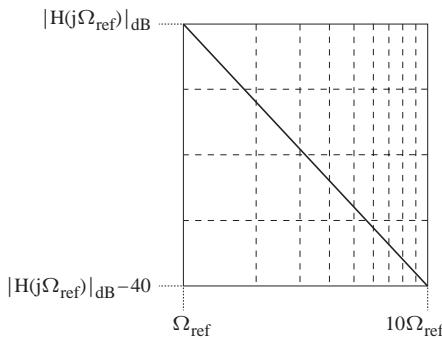
Since we can write  $A \cos(\Omega t + \theta)$  as  $A(e^{j(\Omega t + \theta)} + e^{-j(\Omega t + \theta)})/2$ , the output shall be given as

$$y(t) = \frac{A}{2}e^{j\theta}e^{j\Omega t}H(j\Omega) + \frac{A}{2}e^{-j\theta}e^{-j\Omega t}H(-j\Omega). \quad (2.98)$$

Assuming that  $h(t)$  is real, it is possible to assure that all poles of  $H(s)$  will occur in complex conjugate pairs, leading (as shall be also addressed in the next section) to  $H^*(j\Omega) = H(-j\Omega)$ . This conjugate symmetry property of the frequency response, i.e.,  $|H(j\Omega)| = |H(-j\Omega)|$  and  $\underline{H(j\Omega)} = (\underline{-H(j\Omega)})$ , when applied in (2.98), results in

$y(t)|_{x(t)=A \cos(\Omega t + \theta)} = |H(j\Omega)|A \cos(\Omega t + \theta + \angle H(j\Omega)).$

(2.99)

**FIGURE 2.20**

Asymptotical decay of 40 dB per decade.

The previous expression is also valid as regime solution when the input is a sinusoid, that is non-zero for  $t \geq 0$ , such as  $x(t) = A \cos(\Omega t + \theta)u(t)$ .

Now, back to our example of a frequency response given in (2.96), let us express its magnitude squared:

$$|H(j\Omega)|^2 = \frac{1}{1 + \Omega^2(R^2C^2 - 2LC) + \Omega^4L^2C^2}. \quad (2.100)$$

It is easy to see that, when  $\Omega$  tends to zero, the magnitude squared tends to one while, when  $\Omega$  tends to infinity, the dominant term becomes  $\Omega^4L^2C^2$ :

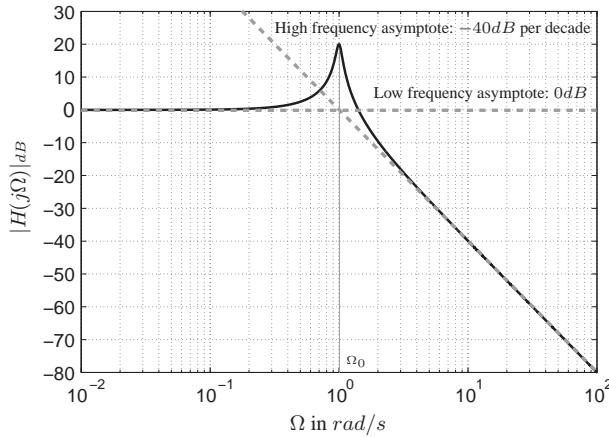
- (a)  $\Omega \rightarrow 0 \Rightarrow |H(j\Omega)| \rightarrow 1$  or 0 dB;
- (b)  $\Omega \rightarrow \infty \Rightarrow |H(j\Omega)| \rightarrow \frac{1}{\Omega^2LC}$  or, in dB,  $-40 \log(\Omega/\Omega_0)$ ,  $\Omega_0 = \frac{1}{\sqrt{LC}}$ .

These two regions of  $|H(j\Omega)|$ , close to zero and tending to infinity, could be visualized by lines in a semi-log plot, the frequency axis, due to its large range of values, is plotted using a logarithmic scale. Particularly, when  $\Omega \rightarrow \infty$ , the approximation in (b) tells us that, in an interval from  $\Omega_{ref}$  to  $10\Omega_{ref}$ ,  $\Omega_{ref} \gg \Omega_0$  (the resonance frequency), we have an attenuation of 40 dB as shown in Figure 2.20.

Magnitude in dB and phase of  $H(j\Omega)$  as a function of  $\Omega = 2\pi f$ , plotted in a logarithmic frequency scale, is known as *Bode Plots* or *Bode Diagrams*. A Bode Diagram may be sketched from lines (asymptotes) which are drawn from the structure of  $H(s)$ , its poles and zeros. For the case of a transfer function with two conjugate complex roots, we have

$$H(s) = \frac{\Omega_0^2}{s^2 + 2\xi\Omega_0s + \Omega_0^2}, \quad (2.101)$$

where, in our example in (2.87),  $\Omega_0 = \frac{1}{\sqrt{LC}}$  and  $\xi = \frac{R}{2}\sqrt{\frac{C}{L}}$ . Also note that (in order to have two conjugate complex roots)  $0 < \xi < 1$ .

**FIGURE 2.21**

Example of Bode Plot for  $H(s) = \frac{\Omega_0^2}{s^2 + 2\xi\Omega_0 s + \Omega_0^2}$  with  $\Omega_0 = 1$  and  $\xi = 0.05$ . Note that, for this case, the approximated height of the peak  $-20 \log(2\xi) = 20$  dB works well.

The magnitude of the frequency response in dB is given as

$$|H(j\Omega)|_{dB} = -10 \log \left( \left( 1 - \left( \frac{\Omega}{\Omega_0} \right)^2 \right)^2 + \left( 2\xi \frac{\Omega}{\Omega_0} \right)^2 \right). \quad (2.102)$$

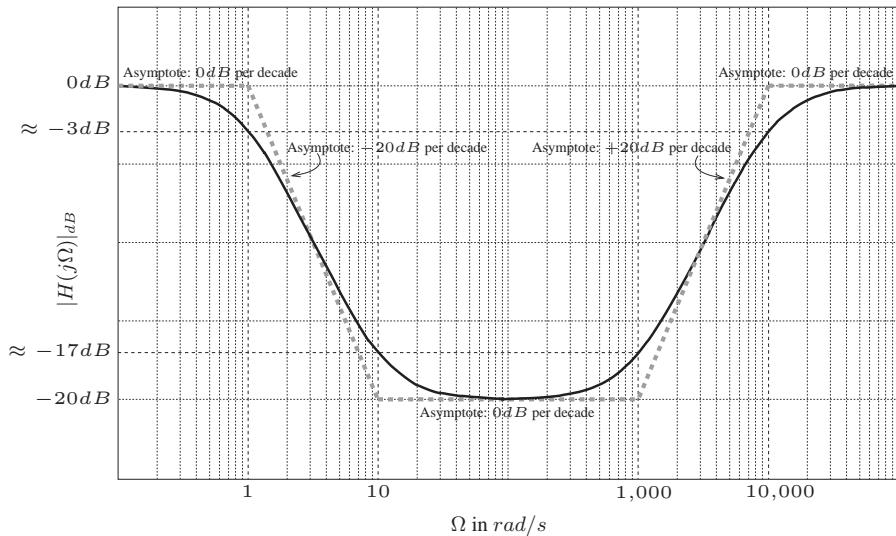
When  $\Omega = \Omega_0$ ,  $|H(j\Omega)| = -20 \log(2\xi)$ ; this value being a good approximation for the peak magnitude (see Figure 2.21) when  $\xi$  tends to zero (we have an error lower than 0.5% when  $\xi < 0.1$ ). The peak actually occurs in  $\Omega = \Omega_0\sqrt{1 - 2\xi^2}$ , having a peak height equal to  $\frac{1}{2\xi\sqrt{1-\xi^2}}$ ; we could also add that the peak is basically observable only when  $0 < \xi < 0.5$ . The Bode Plot of (2.87), with  $R = 0.1$  (in  $\Omega$ ),  $L = 1.0$  (in  $H$ ), and  $C = 1.0$  (in  $F$ ) showing the (low frequency and high frequency) asymptotes is depicted in Figure 2.21.

Another example with first order poles and zeros will end our discussion on Bode Plots. Let transfer function  $H(s)$ , with poles at  $-1$  and  $-10,000$ , and zeros at  $-10$  and  $-1000$ , be represented as follows:

$$\begin{aligned} H(s) &= \frac{s^2 + 1.01s + 10000}{s^2 + 10,001s + 10000} \\ &= \frac{(s + 1,000)(s + 10)}{(s + 10000)(s + 1)} \\ &= \frac{\left(1 + \frac{s}{1,000}\right)\left(1 + \frac{s}{10}\right)}{\left(1 + \frac{s}{10,000}\right)\left(1 + s\right)}. \end{aligned} \quad (2.103)$$

**Table 2.4** Asymptotes for the Bode Plot of Figure 2.22

$\Omega$ (rad/s)	event	asymptotic $ H(j\Omega) _{dB}$ (dB)
1	New asymptote starts ( $-20$ dB per decade)	0
10	New asymptote starts ( $+20$ dB per decade)	$-20$
1000	New asymptote starts ( $+20$ dB per decade)	$-20$
10,000	New asymptote starts ( $-20$ dB per decade)	0

**FIGURE 2.22**

Bode Magnitude (dB) Plot for  $H(s) = \frac{(1+\frac{s}{1,000})(1+\frac{s}{10})}{(1+\frac{s}{10,000})(1+\frac{s}{1})}$ . Download the <http://www.ime.eb.br/~apolin/CTSS/Fig22.m> Matlab<sup>®</sup> code.

The magnitude in  $dB$  of its frequency response could then be written as

$$\begin{aligned} |H(j\Omega)|_{dB} &= 20 \log \left| 1 + j \frac{\Omega}{1,000} \right| + 20 \log \left| 1 + j \frac{\Omega}{10} \right| \\ &\quad - 20 \log \left| 1 + j \frac{\Omega}{10,000} \right| - 20 \log |1 + j\Omega|. \end{aligned} \quad (2.104)$$

The first term, as in the previous example, can be checked for low and high frequencies:

- (a)  $\Omega \ll 1000 \Rightarrow 0_{dB}$ ;

(b)  $\Omega \gg 1000 \Rightarrow$  increases 20 dB per *decade* (from  $\Omega_{ref}$  to  $10\Omega_{ref}$ ) or, equivalently, 6 dB per *octave* (from  $\Omega_{ref}$  to  $2\Omega_{ref}$ ).

When  $\Omega = 1000$ , the value of this term is  $10 \log(2) = 3.0103$  dB which is the usual error at the these frequencies (*cutoff frequencies*).

We could, from the analysis of the first term, extend the results to all cutoff frequencies as shown in Table 2.4. The resulting asymptotic sketch and real curves are shown in Figure 2.22 where we observe the 3 dB errors at the cutoff frequencies.

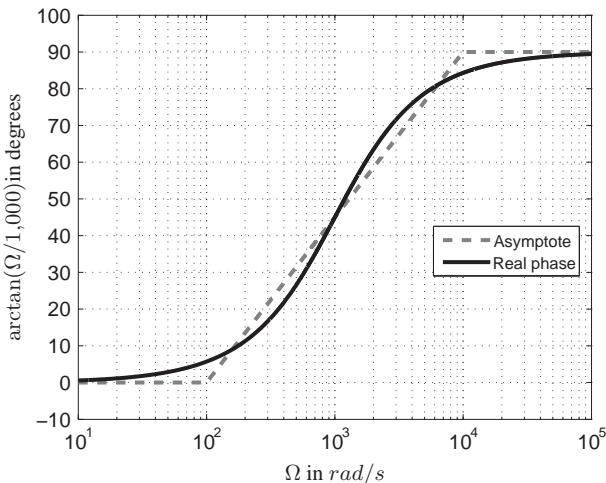
A final observation is regarded to the phase of the frequency response: similarly to the magnitude, a sketch of the phase can also be obtained from the poles and zeros of  $H(s)$ :

$$\angle H(j\Omega) = \angle 1+j\Omega/1000 + \angle 1+j\Omega/10 - \angle 1+j\Omega/10,000 - \angle 1+j\Omega. \quad (2.105)$$

The first term on the right hand side of (2.105),  $\angle 1+j\Omega/1000 = \arctan(\Omega/1000)$ , can be checked for low and high frequencies:

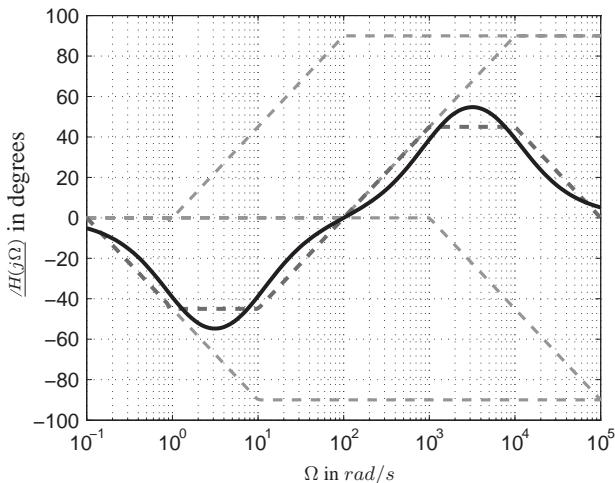
- (a)  $\Omega \ll 1000 \Rightarrow \arctan(\Omega/1000) \approx 0$ ;
- (b)  $\Omega \gg 1000 \Rightarrow \arctan(\Omega/1000) \approx 90^\circ$ .

Considering only this zero (corresponding cutoff frequency  $\Omega_0 = 1000$  rad/s), one may assume  $0.1\Omega_0$  (100 rad/s) and  $10\Omega_0$  (10,000 rad/s) sufficiently small and sufficiently large such that an asymptote could be drawn between points  $(10^2, 0^\circ)$  and  $(10^4, 90^\circ)$  as in Figure 2.23. For a pole, the asymptote is mirrored with respect to  $0^\circ$ .



**FIGURE 2.23**

Asymptotical behavior of the phase for a single zero.

**FIGURE 2.24**

Bode Phase Plot for  $H(s) = \frac{(1+\frac{s}{1.000})(1+\frac{s}{10})}{(1+\frac{s}{10.000})(1+\frac{s}{1})}$ . Lighter gray dashed lines represent the asymptotes and the darker dashed line corresponds to the asymptotic sketch (sum of the asymptotes); the real phase is the continuous curve in black. Download the <http://www.ime.eb.br/~apolin/CTSS/Fig24.m> Matlab<sup>®</sup> code.

The asymptotic sketch and real phase curves for (2.105) is shown in Figure 2.24; note that we have two negative slope asymptotes, starting in  $\Omega = 10^{-1}$  rad/s and  $\Omega = 10^3$  rad/s, and two positive slope asymptotes, starting at  $10^0$  rad/s and  $10^2$  rad/s.

The topics discussed in this section are found in the following books: [1, 3, 8, 14, 15].

## 1.02.7 The Fourier series and the Fourier transform

In electronic circuits we often deal with periodic and non-periodic signals. In addition, circuits could be driven by non-sinusoidal functions. Hence, methods that decompose continuous-time periodic (non-sinusoidal) and non-periodic signals into contributions from sinusoidal ( $\sin(\Omega t)$  and  $\cos(\Omega t)$ ) signals are extremely valuable for electronic system analysis; this is due to the fact that a LTI system only changes the amplitude and phase of a real sinusoid. Several properties make sinusoids an ideal choice to be the elementary basis functions for signal analysis and synthesis. For example, Eq. (2.99) shows that the LTI system changed the amplitude to  $|H(j\Omega)|A$  and the phase to  $\theta + \angle H(j\Omega)$  of the given sinusoidal input,  $x(t) = A \cos(\Omega t + \theta)$ . In addition, as sine and cosine are mutually orthogonal, sinusoidal basis functions are independent and can be processed independently. Usually, any sinusoidal function, such as the one mentioned above, can be expressed as a linear combination of sine and cosine functions:  $x(t) = C \cos(\Omega t) - D \sin(\Omega t)$ . If a signal is a sum of sinusoids, such as

$$x(t) = A_1 \cos(\Omega_1 t + \theta_1) + A_2 \cos(\Omega_2 t + \theta_2) \quad (2.106)$$

its impulse response, considering it is a linear system, would be given by

$$y(t) = A_1|H(j\Omega_1)| \cos(\Omega_1 t + \theta_1 + \angle H(j\Omega_1)) + A_2|H(j\Omega_2)| \cos(\Omega_2 t + \theta_2 + \angle H(j\Omega_2)) \quad (2.107)$$

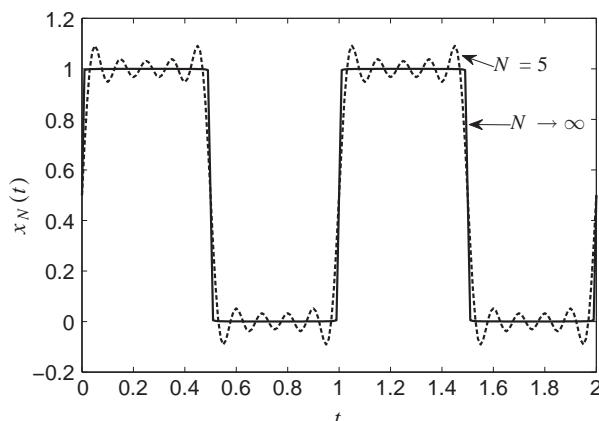
A periodic signal,  $x(t) = x(t+nT)$ ,  $\forall n = \pm 1, \pm 2, \pm 3, \dots$ , with period  $T$ , may contain components at frequencies  $\Omega = 2\pi n/T$ , where the fundamental frequency is given by  $\Omega_0 = 2\pi f_0$ ,  $f_0 = 1/T$ . The frequencies  $n\Omega_0$  are the harmonics, and any pair of signals are harmonically related if their periods are related by a integer ratio. In general, the resulting signal from a sum of harmonically related waveforms is also periodic, and its repetition period is equal to the fundamental period. The harmonic components exist only at discrete frequencies.

Consider that we add the harmonically related signals,  $x_n(t)$ , with  $\Omega_0 = 2\pi/T$ , over  $T = 1$  such that  $x(t) = \frac{1}{2} + \frac{2}{\pi} \sin(2\pi t) + \frac{2}{3\pi} \sin(6\pi t) + \frac{2}{5\pi} \sin(10\pi t) + \frac{2}{7\pi} \sin(14\pi t) + \dots$ , and get  $x_N(t)$  as the result when  $1 \leq N \leq 5$  and when  $N \rightarrow \infty$ , meaning that infinite terms are added:

$$x_N(t) = \frac{1}{2} + \sum_{n=1}^N x_n(t) = \frac{1}{2} + \frac{2}{\pi} \sum_{n=1}^N \frac{1}{2n-1} \sin(2\pi(2n-1)t). \quad (2.108)$$

The resulting waveforms,  $x_N(t)$  for  $1 \leq N \leq 5$  and  $x_N(t)$  when  $N$  is large, i.e.,  $N \rightarrow \infty$ , are displayed by Figure 2.25. It is clear that as the number of terms becomes infinite in (2.108) the result converges at every value of the period  $T$  (to the square wave) except at the discontinuities. Actually, the ringing effect at the discontinuities (not shown in Figure 2.25 for the curve with the large value of  $N$ ) never dies out as  $N$  becomes larger, yet reaching a finite limit (Gibbs phenomenon).

Hence, for a period  $T$ , for a fundamental frequency  $\Omega_0$  with frequencies that are integer multiples of the fundamental frequency such that  $n\Omega_0$ ,  $\forall n \in \mathbb{Z}$  (the set of all integers), the representation of the



**FIGURE 2.25**

Synthesized square wave from Eq. (2.108).

harmonic components,  $x_n(t)$ , of a signal  $x(t)$  can be written as

$$\begin{aligned}x_n(t) &= A_n \sin(n\Omega_0 t + \theta_n) \\&= a_n \cos(n\Omega_0 t) + b_n \sin(n\Omega_0 t) \\&= \frac{1}{2}(a_n - jb_n)e^{jn\Omega_0 t} + \frac{1}{2}(a_n + jb_n)e^{-jn\Omega_0 t}.\end{aligned}\quad (2.109)$$

Equation (2.109) presents three equivalent forms to express the harmonic components  $x_n(t)$ : the first one is written as a sine function with amplitude  $A_n$  and phase  $\theta_n$ ; the second one as sum of sine and cosine functions with real coefficients  $a_n$  and  $b_n$ , respectively; and the last form writes  $x_n(t)$  in terms of complex exponentials.

The derivation, providing proofs, of the expressions for computing the coefficients in a Fourier series is beyond the scope of this chapter. Hence, we simply state, without proof, that if  $x(t)$  is periodic with period  $T$  and fundamental frequency  $\Omega_0$ , any arbitrary real periodic signal  $x(t)$  could be represented by the infinite summation of harmonically related sinusoidal components, known as the Fourier series:

$$\begin{aligned}x(t) &= \frac{1}{2}a_0 + \sum_{n=1}^{\infty} A_n \sin(n\Omega_0 t + \theta_n) \\&= \frac{1}{2}a_0 + \sum_{n=1}^{\infty} (a_n \cos(n\Omega_0 t) + b_n \sin(n\Omega_0 t)),\end{aligned}\quad (2.110)$$

where for  $n = 0$ , we have the term  $\frac{a_0}{2}$  corresponding to the mean value of the signal, with frequency of 0 Hz, known as the DC value (or level), and the other terms are as follows:

$$\begin{aligned}a_n &= A_n \sin(\theta_n), \\b_n &= A_n \cos(\theta_n), \\A_n &= \sqrt{a_n^2 + b_n^2}, \\\theta_n &= \arctan\left(\frac{a_n}{b_n}\right).\end{aligned}\quad (2.111)$$

If in the last form presented in (2.109), we substitute

$$\begin{aligned}c_n &= \frac{1}{2}(a_n - jb_n), \\c_{-n} &= \frac{1}{2}(a_n + jb_n),\end{aligned}\quad (2.112)$$

to have a third representation of the Fourier series, given by

$$\begin{aligned}x(t) &= c_0 + \sum_{n=1}^{\infty} c_n e^{jn\Omega_0 t} + \sum_{n=-1}^{-\infty} c_n e^{jn\Omega_0 t} \\&= \sum_{n=-\infty}^{\infty} c_n e^{jn\Omega_0 t},\end{aligned}\quad (2.113)$$

where  $c_0$  is real and  $c_n$  are complex, for  $n \neq 0$ , coefficients. The coefficients of terms for positive ( $c_n$ ) and negative ( $c_{-n}$ ) values of  $n$  are complex conjugates ( $c_{-n} = c_n^*$ ). This Fourier series form demands the summation to be performed over all  $n \in \mathbb{Z}$ , as the continuous-time complex sinusoids,  $e^{jn\Omega_0 t}$ , present an infinite number of terms (specific frequencies  $n\Omega_0$ ). Moreover, for every integer value of  $n$ ,  $e^{jn\Omega_0 t} = \cos(n\Omega_0 t) + j \sin(n\Omega_0 t)$  is periodic with  $T = 2\pi$ . Hence, the periodic signal  $x(t)$  must have  $T = 2\pi$  in order to compute the Fourier series expansion. However, the analysis of the Fourier cosine series of a function  $x$  in the interval  $[0, \pi]$  is equivalent to the analysis of its extension first to  $[-\pi, \pi]$  as an even function, then to all of components with period  $T = 2\pi$ .

The complex exponential representation of the Fourier series, given by (2.113) generates a *two-sided spectrum*, as the summation requires all  $n \in \mathbb{Z}$ . The first and second representations of the Fourier series, presented in (2.110) produce a *one-sided spectra*, as the sum is computed only for values of  $n > 0$ .

Equations (2.110) and (2.113) are called Fourier synthesis equations as it is possible to synthesize any periodic (time) signal,  $x(t)$ , from an infinite set of harmonically related signals.

If  $x(t)$  is periodic with fundamental frequency  $\Omega_0$  and period  $T$ , the coefficients of the three Fourier series equivalent representations are given by

$$c_n = \frac{1}{T} \int_{t_1}^{t_1+T} x(t) e^{-jn\Omega_0 t} dt, \quad (2.114)$$

and by manipulating (2.112) we obtain

$$\begin{aligned} a_n &= \frac{2}{T} \int_{t_1}^{t_1+T} x(t) \cos(n\Omega_0 t) dt, \text{ and} \\ b_n &= \frac{2}{T} \int_{t_1}^{t_1+T} x(t) \sin(n\Omega_0 t) dt, \end{aligned} \quad (2.115)$$

where the integrals in (2.114) and (2.115) are obtained over any interval that corresponds to the period  $T$ , with the initial time  $t_1$  for the integration arbitrarily defined.

Equations (2.114) and (2.115) are known as Fourier analysis or Fourier decomposition as the signal is being decomposed into its spectral components (basic inputs).

The continuous-time Fourier series expands any continuous-time periodic signal into a series of sine waves. The Fourier analysis methods are named after the French mathematician Jean-Baptiste Joseph Fourier. In general, almost any periodic signal can be written as an infinite sum of complex exponential functions (or real sinusoidal functions) and thus be represented as a Fourier series. Therefore, finding the response to a Fourier series, means finding the response to any periodic function, and its effect on its spectrum. When computing trigonometric coefficients, one could multiply a random signal by sinusoids of different frequencies to disclose all the (hidden) frequency components of the signal. In this way, it becomes easier to isolate the signal of the television station we want to watch from the others that are being simultaneously received by our television set.

In addition, the multiplication of a signal by  $e^{j\Omega_0 t}$  means that the signal if being frequency shifted to  $\Omega_0$ . Which is equivalent to the Frequency Shift Laplace Property (Table 2.3).

Consider  $x(t) = 3/4 + \cos(\Omega_0 t) + \cos(2\Omega_0 t + \pi/3) + 1/3 \cos(5\Omega_0 t + \pi/4)$ , where  $\Omega_0 = 2\pi$ ,  $3/4$  is the DC level, and three non-null harmonics ( $n = 1, 2$ , and  $5$ ) with the time-domain representation displayed by Figure 2.1. The signal has several (hidden) frequencies, and no clue is provided when visually inspecting Figure 2.1. The Fourier series representations are very helpful in this case, showing the amplitudes and phases of each harmonic in the frequency-domain.

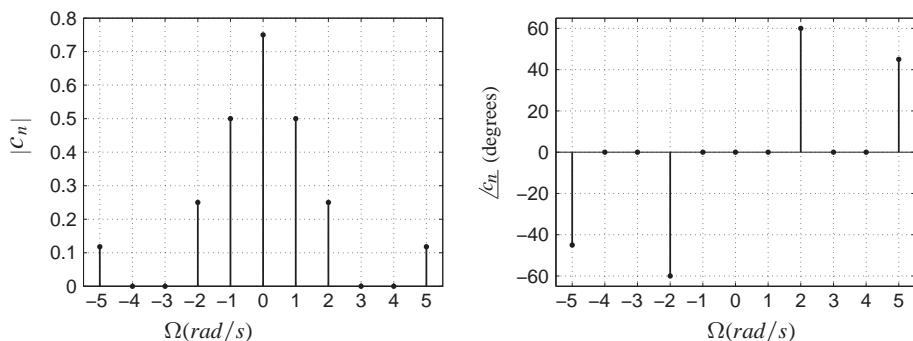
The exponential form, recursing to trigonometric identities, is given by:

$$x(t) = \underbrace{\frac{3}{4}}_{c_0} + \left( \underbrace{\frac{1}{2} e^{j\Omega_0 t}}_{c_1} + \underbrace{\frac{1}{2} e^{-j\Omega_0 t}}_{c_{-1}} \right) + \left( \underbrace{\frac{1}{2} e^{j\frac{\pi}{3}} e^{j2\Omega_0 t}}_{c_2} + \underbrace{\frac{1}{2} e^{-j\frac{\pi}{3}} e^{-j2\Omega_0 t}}_{c_{-2}} \right) + \left( \underbrace{\frac{1}{6} e^{j\frac{\pi}{4}} e^{j5\Omega_0 t}}_{c_5} + \underbrace{\frac{1}{6} e^{-j\frac{\pi}{4}} e^{-j5\Omega_0 t}}_{c_{-5}} \right), \quad (2.116)$$

and the Fourier series exponential form coefficients are (all other terms are zero).

The complex-valued coefficient  $c_n$  conveys both the amplitude ( $|c_n|$ ) and phase ( $\angle c_n$ ) of the frequency content of the signal  $x(t)$  at each value  $n\Omega_0$  rad/s as pictured in Figure 2.26.

The sinusoidal basis functions of the Fourier series are smooth and infinitely differentiable but they only represent periodic signals. The Fourier representation for a non-periodic (time-domain) signal has to treat the frequency spectrum of non-periodic signals as a continuous function of frequency. This representation could be obtained by considering a non-periodic signal as a special case of a periodic signal with an infinite period. The idea is that if the period of a signal is infinite ( $T \rightarrow \infty$ ), then it is never repeated. Hence, it is non-periodic. As  $T \rightarrow \infty$ ,  $\Omega_0 \rightarrow 0$ , thus decreasing the spacing between each adjacent line spectra (representing each harmonic contribution in the spectrum) towards a continuous representation of frequency.



**FIGURE 2.26**

Two-sided spectrum of periodic signal  $x(t)$  depicted in Figure 2.1.

The generalization to non-periodic signals is known as the *Fourier transform* and is given by (2.117). It is analogous to the Fourier series analysis Eq. (2.114). It expresses the time-domain function  $x(t)$  as a continuous function of frequency  $X(j\Omega)$  defined as follows:

$$X(j\Omega) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t} dt. \quad (2.117)$$

The inverse Fourier transform is analogous to the Fourier series synthesis equation (2.113). It obtains the time-domain signal,  $x(t)$ , from its frequency-domain representation  $X(j\Omega)$ :

$$x(t) = \mathcal{F}^{-1}\{X(j\Omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega)e^{j\Omega t} d\Omega. \quad (2.118)$$

The notation  $X(j\Omega) = \mathcal{F}\{x(t)\}$  denotes that  $X(j\Omega)$  is the Fourier transform of  $x(t)$ . Conversely,  $x(t) = \mathcal{F}^{-1}\{X(j\Omega)\}$  denotes that  $x(t)$  is the inverse Fourier transform of  $X(j\Omega)$ . This relationship is expressed with  $x(t) \xrightarrow{\mathcal{F}} X(j\Omega)$ .

The Fourier series coefficients have distinct units of amplitude, whereas  $X(j\Omega)$  has units of amplitude density. The magnitude of  $X(j\Omega)$  is termed as the *magnitude spectrum* ( $|X(j\Omega)|$ ) and its phase as the *phase spectrum* ( $\angle X(j\Omega)$ ).

Following the concept of representing signals as a linear combination of eigensignals, instead of choosing  $e^{-(\sigma+j\Omega)t}$  as the eigensignals, one could choose  $e^{-j\Omega t}$  as the eigensignals. The latter representation leads to the Fourier transform equation, and any LTI system's response could be characterized by the amplitude scaling applied to each of the basic inputs  $e^{-j\Omega t}$ .

This transform is very useful as it produces the frequency complex content of a signal,  $X(j\Omega)$ , from its temporal representation,  $x(t)$ . Moreover, the mapping provides a representation of the signal as combination of complex sinusoids (or exponentials) for frequency-domain behavior analysis, where the (computationally consuming) computation of the convolution in the time-domain is replaced by the (simple) operation of multiplication in the frequency-domain.

Nevertheless, functions such as the unit-step (Figure 2.2a), discontinuous at time  $t = 0$ , does not have forward Fourier transform as its integral does not converge.

An analysis of convergence is beyond the scope of this chapter, hence we assume that the existence of the Fourier transform is assured by three (modified) Dirichlet conditions (for non-periodic signals), requiring the temporal signal,  $x(t)$ : to have a finite number of discontinuities, with the size of each discontinuity being finite; to have a finite number of local maxima and minima; and to be absolutely integrable, i.e.,

$$\int_{-\infty}^{\infty} |x(t)| dt < \infty. \quad (2.119)$$

These are sufficient conditions, although not strictly necessary as several common signals, such as the unit-step, are not absolutely integrable. Nevertheless, we can define a transform pair that satisfies the Fourier transform properties by the usage of impulses when dealing with this class of signals.

The Fourier transform converts a 1-D time-domain signal,  $x(t)$ , into its 1-D complex spectrum  $X(j\Omega)$  representation in frequency-domain. In Section 1.02.4, the Laplace transform maps a time-domain 1-D

**Table 2.5** Fourier Transform Pairs

$x(t) = \mathcal{F}^{-1}\{X(j\Omega)\}$	$X(j\Omega) = \mathcal{F}\{x(t)\}$
$\delta(t)$	1
1	$2\pi\delta(\Omega)$
$u(t)$	$\pi\delta(\Omega) + \frac{1}{j\Omega}$
$ t $	$-\frac{2}{\Omega^2}$
$e^{-at}$	$\frac{1}{a+j\Omega}$
$e^{-a t }, a > 0$	$\frac{2a}{a^2+\Omega^2}$
$e^{j\Omega_0 t}$	$2\pi\delta(\Omega - \Omega_0)$
$sgn(t)$	$\frac{2}{j\Omega}$
$\sin \Omega_0 t$	$-j\pi[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)]$
$\cos \Omega_0 t$	$\pi[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)]$

signal,  $x(t)$ , to a complex representation,  $X(s)$ , defined over a complex plane ( $s$ -plane), spanned by its two variables  $\sigma$  and  $\Omega$ , as  $s = \sigma + j\Omega$ . The Laplace and the Fourier transforms are closely related, when  $s = j\Omega$ , i.e.,  $\sigma = 0$ , the Laplace integral becomes the Fourier transform. Referring to Figure 2.10 (Section 1.02.4), as the imaginary axis ( $j\Omega$ ) lies within the ROC, the Fourier transform exists only if  $a < 0$ . For a certain class of signals, there is a simple relationship between the Fourier and the Laplace transforms given by

$$\mathcal{F}\{x(t)\} = \mathcal{L}\{x(t)\}|_{s=j\Omega}. \quad (2.120)$$

The Laplace transform for  $x(t) = e^{-at}u(t), a > 0$  is  $X(s) = 1/(s+a)$  for  $\sigma > -a$  (Table 2.2).  $X(s)$  has a pole at  $s = -a$ , which lies at the left-half  $s$ -plane. The ROC is located at the right side of  $-a$  (right side of Figure 2.11), thus containing the imaginary axis ( $j\Omega$ ) of the  $s$ -plane. Hence, the signal  $x(t)$  has a Fourier transform given by  $X(j\Omega) = 1/(j\Omega + a)$ . However, the Fourier transform does not converge for  $\text{Re}(s) < -a$  because  $x(t)$  is not absolutely integrable, whereas  $x(t)$  has a Laplace transform with the ROC pictured in the left side of Figure 2.11.

Common Fourier transform pairs are summarized in Table 2.5.

As the Laplace transform, the Fourier transform presents a set of properties. A comprehensive description of the properties of the Fourier transform is beyond the intent of this chapter, and the properties we examine are due to their importance in the study of electronic system analysis. Much of the usefulness of the Fourier transform arises from its properties, that have important interpretations in the context of continuous-time signal processing. Understanding the relationship between time- and frequency-domains (symmetry between operations) is a key point in the study (and usage) of the Fourier transform.

(Time and frequency) scaling: If  $x(t) \xrightarrow{\mathcal{F}} X(j\Omega)$ , then  $x(at) \xleftrightarrow{\mathcal{F}} \frac{1}{|a|}X\left(j\frac{\Omega}{a}\right)$ , with  $a \neq 0 \in \mathbb{R}$ :

$$\mathcal{F}\{x(at)\} = \int_{-\infty}^{\infty} x(at)e^{-j\Omega t} dt, \quad (2.121)$$

substituting  $u = at$  we get

$$\mathcal{F}\{x(at)\} = \begin{cases} \int_{-\infty}^{\infty} x(u)e^{-j\frac{u}{a}\Omega} \frac{du}{a}, & a > 0; \\ \int_{\infty}^{\infty} x(u)e^{-j\frac{u}{a}\Omega} \frac{du}{a}, & a < 0. \end{cases} \quad (2.122)$$

The two previous expressions can be combined in one single expression given by

$$\begin{aligned} \mathcal{F}\{x(at)\} &= \frac{1}{|a|} \int_{-\infty}^{\infty} x(u)e^{-j\frac{\Omega}{a}u} du \\ &= \frac{1}{|a|} X\left(j\frac{\Omega}{a}\right), \end{aligned} \quad (2.123)$$

implying that a linear expansion of the time axis in the time-domain leads to linear compression of the frequency axis in the frequency-domain, and vice versa. This property indicates an important trade-off between the two domains, as narrow time-domain signals have wide Fourier representations (narrow pulses have wide bandwidth).

*Symmetry:* The symmetry properties are quite useful as for a real-valued, a real-valued and even, a real-valued and odd, and an imaginary temporal signal,  $x(t)$ , we have the following relations:

$x(t)$ real	$\xleftrightarrow{\mathcal{F}}$	$X^*(j\Omega) = X(-j\Omega)$
$x(t)$ real and even	$\xleftrightarrow{\mathcal{F}}$	$\text{Im}(X(j\Omega)) = 0$
$x(t)$ real and odd	$\xleftrightarrow{\mathcal{F}}$	$\text{Re}(X(j\Omega)) = 0$
$x(t)$ imaginary	$\xleftrightarrow{\mathcal{F}}$	$X^*(j\Omega) = -X(-j\Omega)$

These properties are important as for a real-valued function in time its Fourier transform is conjugate symmetric,  $X^*(j\Omega) = X(-j\Omega)$ . This means that its real part (the magnitude) is an even function of frequency, and that its imaginary part (phase) is an odd function of frequency. Therefore, when we obtain the Fourier transform of a real-valued time function, it is only necessary to display the transform for positive values of  $\Omega$ . If the signal is even, i.e.,  $x(t) = x(-t)$ , it has a Fourier transform that is a purely real function,  $X(j\Omega) = |X(j\Omega)|$ . Conversely, the transform of an odd function is an purely imaginary function.

Furthermore, other properties deserve to be highlighted. We start with the Frequency Shifting property presented in Table 2.6. As its representation in the frequency-domain indicates,  $X(j(\Omega - \Omega_0))$ , a multiplication by  $e^{j\Omega_0 t}$  in the time-domain, modulates the signal  $x(t)$  onto a different frequency. As most modulation and demodulation techniques involve multiplication, it is important to stress the importance of the convolution property (Table 2.6). A simple operation of multiplication in time becomes a computationally intensive operation of convolution in frequency. Conversely, a convolution operation in the time-domain is much easier to analyze in the frequency-domain (multiplication in the frequency-domain). Hence, as the convolution in the time-domain defines the output of a time-invariant filter to a given input, the analysis and design of filters are typically performed in the frequency-domain.

Another important relation is the Parseval's Relationship. It indicates that the information (energy) contained in the time-domain is preserved when representing the signal in the frequency-domain. If the

**Table 2.6** Properties of Fourier Transform

Property	Time-domain	Frequency-domain
Linearity	$a_1 x_1(t) + a_2 x_2(t)$	$a_1 X_1(j\Omega) + a_2 X_2(j\Omega)$
Scaling	$x(at)$	$\frac{1}{ a } X(j\frac{\Omega}{a})$
Time shifting	$x(t - t_0)$	$e^{-j\Omega t_0} X(j\Omega)$
Frequency shifting	$e^{j\Omega_0 t} x(t)$	$X(j(\Omega - \Omega_0))$
Differentiation in time	$\frac{dx(t)}{dt}$	$j\Omega X(j\Omega)$
	$\frac{d^n x(t)}{dt^n}$	$(j\Omega)^n X(j\Omega)$
Integration in time	$\int_{-\infty}^t x(\tau) d\tau$	$\frac{1}{j\Omega} X(j\Omega) + \pi X(0)\delta(\Omega)$
Differentiation in frequency	$-jtx(t)$	$\frac{dX(j\Omega)}{d\Omega}$
Convolution (in time)	$x_1(t) * x_2(t)$	$X_1(j\Omega) X_2(j\Omega)$
Convolution (in frequency)	$x_1(t)x_2(t)$	$\frac{1}{2\pi} X_1(j\Omega) * X_2(j\Omega)$

power associated with a (complex) signal  $x(t)$  is  $|x(t)|^2$ , Parseval's Relationship states that the signal is represented equivalently in either the time- or frequency-domain without lost or gain of energy:

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(j\Omega)|^2 d\Omega. \quad (2.124)$$

Hence, we can compute average power in either the time- or frequency-domain. The term  $|X(j\Omega)|^2$  is known as the *energy spectrum* or *energy density spectrum* of the signal and shows how the energy of  $x(t)$  is distributed across the spectrum.

A list of Fourier transform properties are summarized in Table 2.6.

One more time, we take, as an example, the RLC circuit displayed by Figure 2.8, where the desired response is the voltage,  $v_o(t)$ , across the capacitor  $C$ ,  $v_C(t)$ . If we use the Fourier transform, assisted by its differentiation property (Table 2.6), to solve the ODE in (2.30) we have

$$\mathcal{F}\{e^{-t}\} = V_i(j\Omega) = \mathcal{F}\left\{LC \frac{d^2 v_o(t)}{dt^2} + RC \frac{dv_o(t)}{dt} + v_o(t)\right\} \quad (2.125)$$

and we get the following result

$$V_i(j\Omega) = -LC\Omega^2 V_o(j\Omega) + jRC\Omega V_o(j\Omega) + V_o(j\Omega), \quad (2.126)$$

where the output in the frequency-domain,  $V_o(j\Omega)$ , is given by

$$V_o(j\Omega) = \frac{V_i(j\Omega)}{(1 - LC\Omega^2) + jRC\Omega}, \quad (2.127)$$

which exhibits a resonant behavior exactly as in (2.96). Also note that the frequency response, defined in (2.93), corresponds to the Fourier transform of the impulse response.

Fourier methods and applications are presented in [1, 4, 5, 16, 17].

### 1.02.8 Conclusion and future trends

This chapter provided an introductory overview on the general concepts of signals, systems, and analysis tools used in the continuous-time domain.

We believe that the knowledge of continuous-time signals and systems, even for a book directed for methods used in the discrete-time domain, is extremely important for, while exploring their differences, the reader can enhance the understanding of the natural phenomena.

A few papers discuss the relevance of teaching (analog) circuits and systems, as well its importance to an engineering career [18].

Analog circuit engineers have to deal with the entire circuit, and many parameters must be considered in their design. In order to simplify circuit models, designers have to make approximations about the analog components, which requires expertise and years of education.

This chapter dealt with traditional electrical components and their fundamental variables: resistors having a direct relationship between voltage and current ( $v = Ri$ ,  $R$  the resistance), inductors relating voltage and current with its flux ( $d\phi = v dt$  and  $d\phi = L di$ ,  $L$  the inductance), and capacitors relating voltage and current with charge ( $dq = C dv$  and  $dq = i dt$ ,  $C$  the capacitance). After first theorized in the early seventies, the *memristor* regained the media in 2008 when a first real implementation was announced: the fourth fundamental circuit element relating charge and flux as in its axiomatic definition,  $d\phi = M dq$  ( $M$  the *memristance*) [19]. The use of this component in usual electronic gadgets as well as teaching its theory even in undergraduate courses seems to be an upcoming technology trend.

In addition, most electronic devices have to interface with the external, analog, world, where data conversion is needed at the input and output sides. Analog technologies are used along with digital technologies, spanning from human interface components to analog circuits in wireless components. Furthermore, analog design is also present in the mixed-signal integrated circuits, and it is becoming increasingly critical due to high-speed circuitry [20]. Analog circuits are also used to process signals in very high frequency ranges, such as microwave and RF. In addition, electronic digital circuits are supplied with power, and the trend is to reduce power consumption of digital circuitry. Hence, knowledge of analog circuits is required if an engineer is designing a high performance digital circuit.

We have tried to condense in few pages, all pertinent information regarding analog signals and systems in order to help a complete understanding of the forthcoming chapters. Nevertheless, we have not exhausted the description of all details. For more interested readers, we outline in the following suggested references for further reading. We hope you enjoy the rest of the book.

---

## Glossary

BIBO	refers to a certain class of stability known as <i>bounded-input bounded-output</i>
Convolution	operation between two functions: the output (zero-state solution) of a linear and time-invariant system corresponds to the convolution integral between the input signal and the impulse response, i.e., $y(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau$

Frequency response	of a linear system corresponds to the Fourier transform of its impulse response: $H(j\Omega) = \int_{-\infty}^{\infty} h(\tau)e^{-j\Omega\tau} d\tau$
Impulse signal	$\delta(t)$ is a pulse of infinite amplitude and infinitesimal time characterized by being equal to zero when $t \neq 0$ and $\int_{-\infty}^{\infty} \delta(t)dt = 1$
Impulse response	$h(t)$ is the output of a linear and time-invariant system when the input corresponds to the impulse $\delta(t)$
LTI	linear and time-invariant system: when the output of a continuous-time system corresponds to a linear operation on the input signal while not depending on a particular instant but remaining the same as time goes by
ODE	ordinary differential equation: a differential equation having only one independent variable
RLC	is usually related to a circuit with resistor (R), inductor (L) and capacitor (C)
ROC	region of convergence of the Laplace transform of a given signal $x(t)$ : in the Laplace transform domain, it corresponds to the set of values of the complex variable $s$ that ensures the existence of $\mathcal{L}\{x(t)\}$
Transfer function	is the representation in the $s$ -domain of the input-output relation for a linear system; it corresponds to the Laplace transform of the impulse response, i.e., $H(s) = \frac{Y(s)}{X(s)} = \mathcal{L}\{h(t)\}$
Unit-step signal	$u(t)$ is equal to 0 when $t < 0$ , equal to 1 when $t > 0$ and relates to the impulse signal as follows: $u(t) = \int_{-\infty}^t \delta(t)dt$

---

## 1.02.9 Relevant Websites:

<<http://www.britannica.com/EBchecked/topic/330320/Pierre-Simon-marquis-de-Laplace/>> (article from the Encyclopedia Britannica about Laplace)

<<http://www.genealogy.ams.org/id.php?id=108295>> (American Mathematical Society Genealogy Project, Pierre-Simon Laplace)

<<http://www.academie-francaise.fr/immortels/base/academiciens/fiche.asp?param=336>> (mention to Pierre-Simon de Laplace, immortal of Académie-Française)

<<http://www.britannica.com/EBchecked/topic/215097/Joseph-Baron-Fourier/>> (article from the Encyclopedia Britannica about Fourier)

<<http://www.genealogy.ams.org/id.php?id=17981>> (American Mathematical Society Genealogy Project, Jean-Baptiste Joseph Fourier)

<<http://www.ieeeghn.org/>> (IEEE Global History Network)

<<http://www.coe.ufrrj.br/~acmq/>> (this site contains many tools for circuit analysis and design as well as a number of interesting links)

*Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 1](#) Introduction: Signal Processing Theory

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

### 1.02.10 Supplementary data

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/B978-0-12-396502-8.00002-4>.

## References

- [1] A. Oppenheim, A. Willsky, S. Nawab, Signals and Systems, Prentice-Hall Signal Processing Series, second ed., Prentice Hall, 1996.
- [2] B.P. Lathi, Linear Systems and Signals, second ed., Oxford University Press, 2004.
- [3] B.P. Lathi, Signal Processing and Linear Systems, second ed., Oxford University Press, Incorporated, 2009.
- [4] S. Haykin, B. Veen, Signals and Systems, second ed., John Wiley & Sons, 2003.
- [5] B. Girod, R. Rabenstein, A. Stenger, Signals and Systems, first ed., John Wiley & Sons, 2001.
- [6] W. Boyce, R. DiPrima, Elementary differential equations, ninth ed., John Wiley & Sons, USA, 2008.
- [7] A.C. Fischer-Cripps, The Mathematics Companion: Mathematical Methods for Physicists and Engineers, Taylor & Francis, New York, USA, 2005.
- [8] R. Dorf, J. Svoboda, Introduction to Electric Circuits, eighth ed., John Wiley & Sons, USA, 2010.
- [9] G. Strang, Computational Science and Engineering, Wellesley-Cambridge Press, Massachusetts, USA, 2007.
- [10] W. Thomson, Laplace Transformation, Prentice-Hall Electrical Engineering Series, second ed., Prentice-Hall Inc., 1960.
- [11] J. Holbrook, Laplace transforms for electronic engineers, International Series of Monographs on Electronics and Instrumentation, second ed., Pergamon Press, 1966.
- [12] F. Oberhettinger, L. Badii, Tables of Laplace transforms, Grundlehren der mathematischen wissenschaften, Springer-Verlag, 1973.
- [13] L. Padulo, M.A. Arbib, System Theory: A Unified State-Space Approach to Continuous and Discrete Systems, W.B. Saunders Company, Philadelphia, USA, 1974.
- [14] C. Close, Analysis of Linear Circuits, International edition, Harcourt Publishers Ltd., California, USA, 1974.
- [15] C. Desoer, E. Kuh, Basic circuit theory, McGraw Hill International Editions: Electrical and Electronic Engineering Series, New York, USA, 1969.
- [16] R. Bracewell, The Fourier transform and its applications, McGraw-Hill Series in Electrical and Computer Engineering, third ed., McGraw Hill, 2000.
- [17] A. Papoulis, The Fourier integral and its applications, McGraw-Hill Electronic Sciences Series, McGraw-Hill, 1962.
- [18] P. Diniz, Teaching circuits, systems, and signal processing, in: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems 1998, ISCAS'98, vol. 1, 1998, pp. 428 –431, doi: <<http://dx.doi.org/10.1109/ISCAS.1998.704468>>.
- [19] G.E. Pazienza, J. Albo-Canals, Teaching memristors to EE undergraduate students [class notes], IEEE Circ. Syst. Mag. 11 (4) (2011) 36–44.
- [20] R. Gray, History and trends in analog circuit challenges at the system level, in: 2010 International Symposium on Electronic System Design (ISED), 2010, p. 24, doi: <<http://dx.doi.org/10.1109/ISED.2010.62>>.

# Discrete-Time Signals and Systems

# 3

**Leonardo G. Baltar and Josef A. Nossek**

*Institute for Circuit Theory and Signal Processing, Technische Universität München,  
München, Germany*

---

## 1.03.1 Introduction

Discrete-time systems are signal processing entities that process discrete-time signals, i.e., sequences of signal values that are generally obtained as equidistant samples of continuous-time waveforms along the time axis. Usually a clock signal will determine the period  $T$  (sampling interval) in which the input signal values enter the system and, respectively, the output samples leave the system. The interval  $T$  also determines the cycle in which the internal signal values within the system are processed. Typical representatives of this class of systems are the analog discrete-time signal processing systems (switched capacitor circuits, charge-coupled devices (CCD), bucket-brigade devices (BBD)) as well as digital systems. In the case of the last, the sequence value for each sampling interval will also be discretized, i.e., it will be represented with a finite set of amplitudes, usually in a binary representation. [Chapter 5](#) covers the topic of sampling and quantization of continuous-time signals.

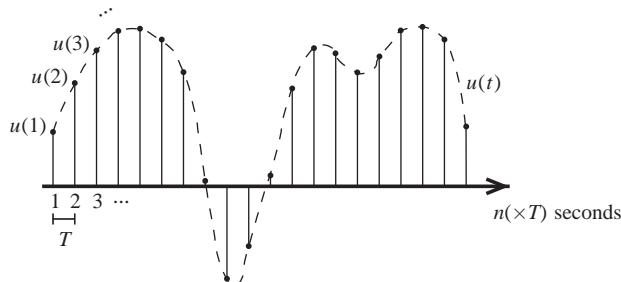
It is worth mentioning, that in the case of multi-rate discrete-time systems, the interval in which the internal signals and possibly the output signal are processed will usually be different from  $T$ . The topic of multi-rate discrete-time systems is covered in [Chapter 9](#).

It is important to emphasize that the discussion in this chapter considers the special case of discrete-time signals with unquantized values. This means that the amplitude of all the signals and the value of the parameters (coefficients) that describe the discrete-time systems are represented with infinite word length. This is an abstraction from real digital signals and systems, where the finite word length representation of both signals and parameters have to be taken into account to fully cover all important phenomena, such as limit cycles. Design of digital filters and their implementation will be discussed in [Chapter 7](#).

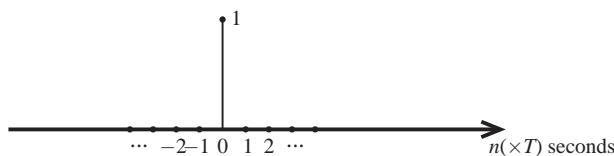
---

## 1.03.2 Discrete-time signals: sequences

Discrete-time signals can arise naturally in situations where the data to be processed is inherently discrete in time, like in financial or social sciences, economy, etc. Generally, the physical quantities that the discrete-time signal represents evolve continuously over time (for example, voice, temperature, voltages, currents or scattering variables), but inside a sampling interval of  $T$  seconds, can be completely characterized by means of a single value only, see [Figure 3.1](#). In the last case, the continuous-time signals are first sampled or discretized over time and possibly also quantized in amplitude.

**FIGURE 3.1**

Discrete-time signal as a sampled version of a continuous-time waveform.

**FIGURE 3.2**

Discrete-time unit impulse.

Still images are usually described as two dimensional discrete signals, where the two dimensions are spatial dimensions. For moving images, a third dimension is added.

Important examples of discrete-time signals that are used in practice are addressed in the following.

Analogously to the continuous-time unit impulse, known as the Dirac delta function, we can also define a discrete-time unit impulse, known as the Kronecker delta function, as

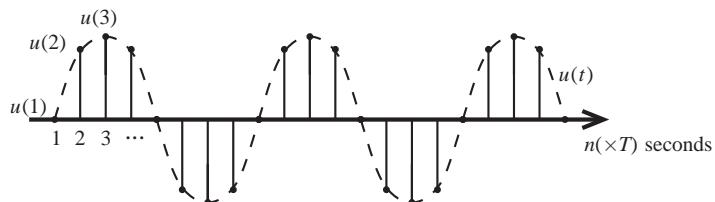
$$\delta(n) = \begin{cases} 1, & n = 0, \\ 0, & \text{else,} \end{cases} \quad (3.1)$$

where  $n$  is an integer number. The discrete time unit impulse is graphically shown in Figure 3.2. Any arbitrary discrete time sequence can be represented as a sum of weighted and delayed unit impulses. Also analogous to the continuous-time systems, the input–output behavior of a discrete-time system can also be described by the impulse response, as we will see in details in a later section.

Another commonly used test signal is a sequence, where the samples are equidistantly taken from a sinusoid

$$u(n) = U \sin(nT\omega + \varphi), \quad (3.2)$$

with  $U$  and  $\varphi$  being real constants.  $\omega$  is the frequency of the sinusoid in rad/s. In some cases it is important to analyze a system for certain frequencies and in this case a sinusoid input with the frequencies of interest can be applied at the input. An example of a discrete-time sinusoid and the corresponding continuous-time waveform are depicted in Figure 3.3

**FIGURE 3.3**

Discrete-time sinusoid and corresponding continuous-time.

We can also define a random sequence, an example being the case where the samples are distributed according to a normal distribution with zero mean and given variance  $\sigma^2$  and the individual samples are statistically independent, i.e.,

$$E [u(n)u(n - k)] = \begin{cases} \sigma^2, & k = 0, \\ 0, & \text{else.} \end{cases} \quad (3.3)$$

This is the discrete equivalent of the white Gaussian noise and it is useful if the behavior of the discrete time system is to be analyzed for all frequencies at once. Also in the case where some noise sources are to be included during the analysis or modeling of a discrete-time system implementation, this random sequence is the most used one. See [Chapter 4](#) for more details on random sequences.

### 1.03.3 Discrete-time systems

A discrete-time system with discrete-time input sequence  $u(n)$  and output sequence  $y(n)$  is an operator  $\mathcal{T}$  that performs the mapping

$$y(n) = \mathcal{T}\{u(n)\}. \quad (3.4)$$

It is important to note that this mapping is applied to the entire input sequence  $u(n)$ , beginning with some starting instant in the far distant past, including the present instant  $n$  and up to a far future instant, to obtain the present output  $y(n)$ .

#### 1.03.3.1 Classification

There are many different classes of discrete-time systems. They can be categorized according to various important properties that will allow us to analyze and design systems using specific mathematical tools. All the properties explained here find an equivalent for continuous-time systems.

##### 1.03.3.1.1 Memoryless systems

One sample of the output sequence  $y(n)$  of a memoryless system for a specific index  $n_0$  depends exclusively on the input value  $u(n_0)$ . The output depends neither on any past or future value of the input  $\dots, u(n_0 + 1), u(n_0 - 1), u(n_0 - 2), \dots$  nor on any past or future value of the output  $\dots, y(n_0 + 1),$

$y(n_0 - 1), y(n_0 - 2), \dots$ , i.e.,

$$y(n_0) = \mathcal{T}\{u(n_0)\}. \quad (3.5)$$

### 1.03.3.1.2 Dynamic systems

In contrast, a memory or dynamic system has its output dependent on at least one past value, either the output or the input

$$y(n) = \mathcal{T}\{u(n), u(n-1), \dots, u(n-N), y(n-1), \dots, y(n-N)\}. \quad (3.6)$$

The positive integer constant  $N$  is usually called the degree of the system. We have assumed here, without loss of generality, that the same number of past inputs and outputs influence a certain output sample.

### 1.03.3.1.3 Linear systems

If the system is a linear system, the superposition principle should hold. Let us consider an input  $u_1$  for the system. Then the output is

$$y_1(n) = \mathcal{T}\{u_1(n)\}. \quad (3.7)$$

Similarly, with another input  $u_2$ , the output is

$$y_2(n) = \mathcal{T}\{u_2(n)\}. \quad (3.8)$$

The superposition principle tells us that if each input is scaled by any constant  $\alpha$ , and moreover, the sum of the inputs is employed as a new input, then for the new output, it holds that

$$\begin{aligned} \mathcal{T}\{\alpha_1 u_1(n) + \alpha_2 u_2(n)\} &= \mathcal{T}\{\alpha_1 u_1(n)\} + \mathcal{T}\{\alpha_2 u_2(n)\} \\ &= \alpha_1 \mathcal{T}\{u_1(n)\} + \alpha_2 \mathcal{T}\{u_2(n)\} = \alpha_1 y_1(n) + \alpha_2 y_2(n). \end{aligned} \quad (3.9)$$

This can be extended for any number of different inputs and any value of the scaling factors. One of the simplest and probably most popular linear dynamic system is the recurrence used to generate Fibonacci sequences [1]:

$$y(n) = u(n) + y(n-1) + y(n-2), \quad (3.10)$$

where  $u(n) = \delta(n)$ .

### 1.03.3.1.4 Non-linear systems

In the case of non-linear systems, the superposition principle does not apply. An example of this is the famous logistic map [2] or discrete logistic equation, that arise in many contexts in the biological, economical and social sciences.

$$y(n) = u(n) + \alpha y(n-1)(1 - y(n-1)), \quad (3.11)$$

where  $\alpha$  is a positive real number,  $u(n) = \gamma \delta(n)$  and  $\gamma$  represents a initial ratio of population to a maximum population. The logistic map is a discrete-time demographic model analogous to the continuous-time logistic equation [3] and is a simple example of how chaotic behavior can arise. In this chapter we are mainly interested in linear systems, since there exist many well established analysis tools.

### 1.03.3.1.5 Time-invariant systems

If the samples of the input sequence are delayed by  $n_1$  and the output samples are delayed by the same interval, a discrete-time system is called time-invariant.

$$y(n - n_1) = T\{u(n - n_1)\}. \quad (3.12)$$

Both the Fibonacci recurrence and the logistic map are Time-invariant systems.

### 1.03.3.1.6 Time-variant systems

In the case of time-variant systems, internal parameters of the system, like multiplier coefficients, can also vary with time. This is the case for the so-called adaptive systems or adaptive filters (see [Chapter 11](#) for more details). A simple example of a time-variant system is the amplitude modulation

$$y(n) = \sin(nT\omega + \varphi)u(n). \quad (3.13)$$

There exists a special case of time-variant systems where the internal multiplier coefficients do not necessarily vary with time, but the sampling rate is changed within the system. Those systems are called multi-rate systems and they frequently present a cyclical output. This means that for a certain multiple delay of the input signal, the output sequence will have the same samples as for the non-delayed input.

### 1.03.3.1.7 Causal systems

In a causal system, the output samples only depend on the actual or past samples of the input sequence, and on past samples of the output itself. If the system is non-causal, it means that the output sample depends on an input value that has not yet entered the system. The definition of dynamic systems in [\(1.03.3.1\)](#) is already considered a causal case. Non-causal systems are not relevant for almost all practical applications.

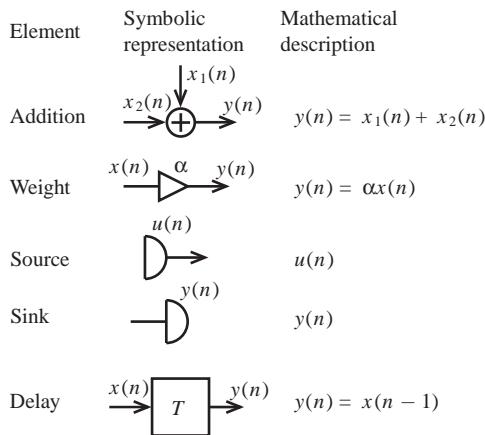
## 1.03.4 Linear Time-Invariant (LTI) Systems

This is a technically important class of discrete-time systems, for which a rich mathematical framework exists.

Linear time-invariant continuous-time systems in the electrical/electronic domain can be built with memoryless (resistive) elements such as resistors, independent and controlled sources and memory possessing (reactive) elements, for example, capacitors or inductors. In linear time-invariant discrete-time systems, we also have memoryless and memory-possessing elements such as

- adders and multipliers, signal source and sink,
- delay elements.

The symbolic representations of these aforementioned elements are depicted in [Figure 3.4](#). It is important to note that in this context, nothing is said about how these elements are realized—neither with which electronic components nor with which technology.

**FIGURE 3.4**

Basic building blocks of a discrete time system.

#### 1.03.4.1 State-space description

Any linear time-invariant system can be structured into a memoryless linear  $(2N + 2)$  terminal network with  $N$  delay elements and a single input and a single output. Although the system can easily extend to multiple inputs and/or multiple outputs, we will consider only single input/single output (SISO) systems, with which we can study all important phenomena.

Because everything within the linear time-invariant memoryless box in Figure 3.5 can only perform weighted sums, the  $(N + 1)$  dimensional input vector (state vector  $\mathbf{x}(n)$  stacked with scalar input  $u(n)$ ) is mapped onto the  $(N + 1)$  dimensional output vector (state vector  $\mathbf{x}(n + 1)$  stacked with scalar output  $y(n)$ ) through multiplication with the  $(N + 1) \times (N + 1)$  matrix  $\mathcal{S}$

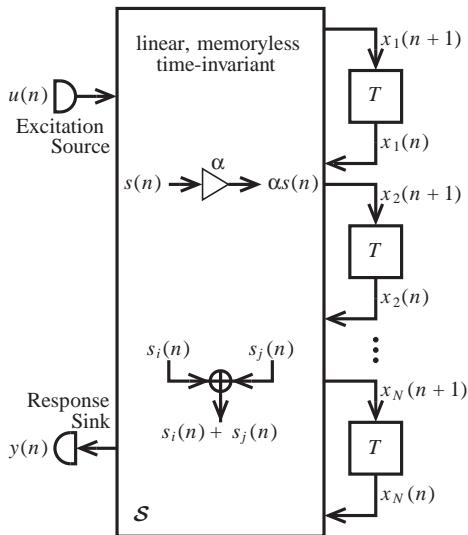
$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \\ \vdots \\ x_N(n+1) \\ y(n) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c}^T & d \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_N(n) \\ u(n) \end{bmatrix} = \mathcal{S} \cdot \begin{bmatrix} \mathbf{x}(n) \\ u(n) \end{bmatrix}. \quad (3.14)$$

The matrix  $\mathcal{S}$  can obviously be partitioned into the well-known state space description

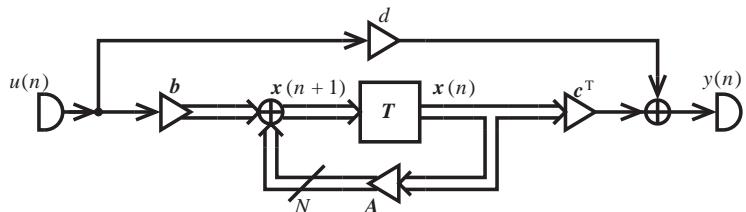
$$\begin{aligned} \mathbf{x}(n+1) &= \mathbf{A}\mathbf{x}(n) + \mathbf{b}u(n), \\ y(n) &= \mathbf{c}^T\mathbf{x}(n) + du(n), \end{aligned} \quad (3.15)$$

where  $\mathbf{A} \in \mathbb{R}^{N \times N}$ ,  $\mathbf{b} \in \mathbb{R}^N$ ,  $\mathbf{c}^T \in \mathbb{R}^{1 \times N}$ ,  $d \in \mathbb{R}$ ,  $\mathbf{x}(n), \mathbf{x}(n+1) \in \mathbb{R}^N$  and  $u(n), y(n) \in \mathbb{R}$ .

Now we can take (3.15) and refine the block diagram depicted in Figure 3.5 to show Figure 3.6, which is the general state-space realization. The structure in Figure 3.6 is still rather general, because for each element of  $\mathbf{A}$ ,  $\mathbf{b}$ ,  $\mathbf{c}$  and  $d$ , a multiplication has to be performed, especially if we assume all of their elements with non-trivial values. The total number of multiplications is in this case,  $N(N + 2) + 1$ .

**FIGURE 3.5**

Discrete-time system.

**FIGURE 3.6**

General state space realization a discrete system.

But, as we will see later, that  $\mathbf{A}$ ,  $\mathbf{b}$  and  $\mathbf{c}$  are not uniquely determined by a given input–output mapping. Consequently there is room for optimization, e.g., to reduce the number of actual multiplications to be performed per sample.

Let us compute the output signal  $y(n)$ , using an initial state vector  $\mathbf{x}(0)$ , and the input signal sequence from the time instant  $n = 0$  onwards

$$\begin{aligned}
 \mathbf{x}(n) &= \mathbf{A}\mathbf{x}(n-1) + \mathbf{b}u(n-1) \\
 &= \mathbf{A}(\mathbf{A}\mathbf{x}(n-2) + \mathbf{b}u(n-2)) + \mathbf{b}u(n-1) \\
 &= \dots \\
 &= \mathbf{A}^n\mathbf{x}(0) + \sum_{k=1}^n \mathbf{A}^{k-1}\mathbf{b}u(n-k),
 \end{aligned} \tag{3.16}$$

$$y(n) = \underbrace{\mathbf{c}^T \mathbf{A}^n \mathbf{x}(0)}_{\text{zero-input response}} + \underbrace{\sum_{k=1}^n (\mathbf{c}^T \mathbf{A}^{k-1} \mathbf{b} u(n-k))}_{\text{zero-state response}} + d u(n). \quad (3.17)$$

From this general response we can also derive the so-called impulse response, for which we set  $\mathbf{x}(0) = \mathbf{0}$  and  $u(n) = \delta(n)$ , the unit impulse which was defined in (3.1). This leads to

$$y(n) = h(n) = \begin{cases} d & n = 0, \\ \mathbf{c}^T \mathbf{A}^{n-1} \mathbf{b} & n \geq 1. \end{cases} \quad (3.18)$$

By making use of the impulse response  $h(n)$ , we can reformulate the zero-state response with a general input sequence  $u(n)$

$$\begin{aligned} y(n) &= \sum_{k=1}^n (\mathbf{c}^T \mathbf{A}^{k-1} \mathbf{b} u(n-k)) + d u(n) \\ &= \sum_{k=0}^n h(n) u(n-k) = h(n) * u(n), \end{aligned} \quad (3.19)$$

which is the so-called convolution sum. This summation formulation is corresponding to the so-called convolution integral for continuous-time signals and systems.

Now it is time to show that we can generate a whole class of equivalent systems (equivalent in the sense that they have the same zero-state response and the same input-output mapping respectively) with the aid of a so-called similarity transform.

With a non-singular  $N \times N$  matrix  $\mathbf{T}$  we can define a new state vector as the linear transformation of the original one as  $\mathbf{x}'(n) = \mathbf{T}^{-1} \mathbf{x}(n)$ , then we can rewrite the state-space equations with the new state vector  $\mathbf{x}'(n)$  as

$$\begin{aligned} \mathbf{T} \mathbf{x}'(n+1) &= \mathbf{A} \mathbf{T} \mathbf{x}'(n) + \mathbf{b} u(n), \\ y(n) &= \mathbf{c}^T \mathbf{T} \mathbf{x}'(n) + d u(n), \end{aligned} \quad (3.20)$$

and by multiplying the first equation with  $\mathbf{T}^{-1}$  from the left side we get

$$\begin{aligned} \mathbf{x}'(n+1) &= \mathbf{T}^{-1} \mathbf{A} \mathbf{T} \mathbf{x}'(n) + \mathbf{T}^{-1} \mathbf{b} u(n), \\ y(n) &= \mathbf{c}^T \mathbf{T} \mathbf{x}'(n) + d u(n). \end{aligned} \quad (3.21)$$

The new state-space representation can now be formulated as

$$\begin{aligned} \mathbf{x}'(n+1) &= \mathbf{A}' \mathbf{x}'(n) + \mathbf{b}' u(n), \\ y(n) &= \mathbf{c}'^T \mathbf{x}'(n) + d' u(n), \end{aligned} \quad (3.22)$$

where the equations

$$\mathbf{A}' = \mathbf{T}^{-1} \mathbf{A} \mathbf{T}, \quad \mathbf{b}' = \mathbf{T}^{-1} \mathbf{b}, \quad \mathbf{c}'^T = \mathbf{c}^T \mathbf{T} \text{ and } d' = d \quad (3.23)$$

hold.

By inserting  $\mathbf{A}'$ ,  $\mathbf{b}'$ ,  $\mathbf{c}'$  and  $d'$  into the zero-state response, we see that this response is invariant to the above transformation.

Depending on the choice of the transformation matrix, the matrix  $\mathbf{A}'$  can have different forms. A particularly interesting form is the so-called normal form, where the state matrix is diagonalized, allowing a much lower complexity than a dense matrix. First we apply an eigenvalue decomposition [4] to  $\mathbf{A}$ :

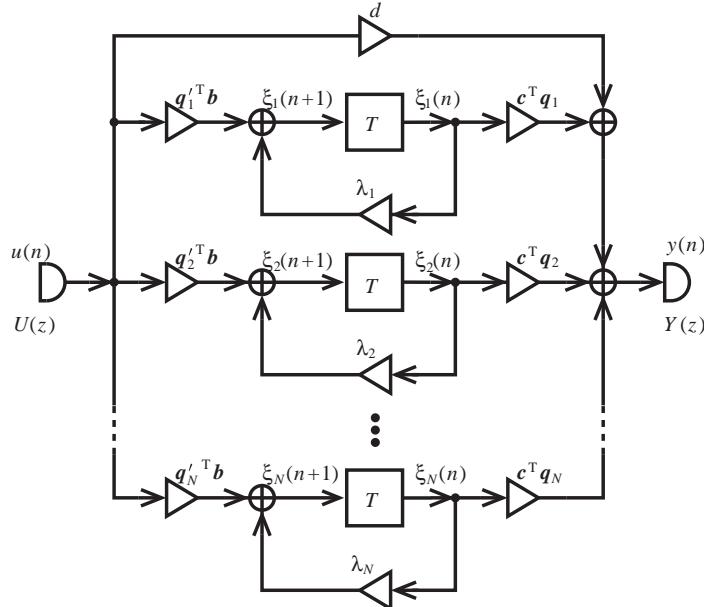
$$\begin{aligned}\mathbf{A} &= \mathbf{Q} \Lambda \mathbf{Q}^{-1} && \text{Eigenvalue decomposition of } \mathbf{A}, \\ \mathbf{Q} &= [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n] && \text{Modal matrix with the eigenvectors } \mathbf{q}_k, \\ \Lambda &= \text{diag}(\lambda_k)_{k=1}^n && \text{Eigenvalue } \lambda_k \text{ with } \mathbf{A}\mathbf{q}_k = \lambda_k\mathbf{q}_k,\end{aligned}$$

where we have assumed that  $\mathbf{A}$  is diagonalizable and we do not need to resort to the Jordan form. The transformation matrix and the new state vector are  $\mathbf{T} = \mathbf{Q}$  and  $\xi(n) = \mathbf{Q}^{-1}\mathbf{x}(n)$ . The system can then be described by

$$\begin{aligned}\dot{\xi}(n+1) &= \Lambda \xi(n) + \mathbf{Q}^{-1} \mathbf{b} u(n), \\ y(n) &= \mathbf{c}^T \mathbf{Q} \xi(n) + d u(n),\end{aligned}\tag{3.24}$$

which leads to the new specific implementation of Figure 3.7. With this new implementation, the number of coefficients, and consequently the number of multiplications, is reduced to  $3N + 1$ .

If some of the eigenvalues are complex valued (if this is the case they will always come in complex conjugate pairs provided  $\mathbf{A}$  is real valued), we always will merge the two corresponding first order



**FIGURE 3.7**

Normal form for real valued eigenvalues.

systems to one second order system with real valued multiplier coefficients only. This is equivalent to have a block diagonal state matrix (with 2-by-2 blocks) instead of a diagonal one.

Now we can compute the transfer function by transforming (3.15) to the frequency domain with the  $\mathcal{Z}$ -transform.

#### 1.03.4.2 Transfer function of a discrete-time LTI system

Similar to continuous-time systems we can also analyze a discrete-time system with the help of a functional transform to make a transition to the frequency domain. However, we have only sequences of signal values in the time domain and not continuous waveforms. Therefore, we have to apply the  $\mathcal{Z}$ -transform [5]:

$$\begin{aligned}\mathcal{Z}\{u(n)\} &= \sum_{n=-\infty}^{\infty} u(n)z^{-n} = U(z), \\ \mathcal{Z}\{y(n)\} &= \sum_{n=-\infty}^{\infty} y(n)z^{-n} = Y(z), \\ \mathcal{Z}\{\mathbf{x}(n)\} &= \sum_{n=-\infty}^{\infty} \mathbf{x}(n)z^{-n} = \mathbf{X}(z).\end{aligned}\quad (3.25)$$

The delay in the time-domain is represented in the frequency domain by a multiplication by  $z^{-1}$ .

$$\begin{aligned}\mathcal{Z}\{\mathbf{x}(n+1)\} &= \sum_{n=-\infty}^{\infty} \mathbf{x}(n+1)z^{-n} = z \sum_{n=-\infty}^{\infty} \mathbf{x}(n+1)z^{-(n+1)} \\ &\stackrel{n'=n+1}{=} z \sum_{n'=-\infty}^{\infty} \mathbf{x}[n']z^{-n'} = z\mathcal{Z}\{\mathbf{x}(n)\} = z\mathbf{X}(z).\end{aligned}\quad (3.26)$$

Now we can transform the state-space equations to the frequency domain

$$\begin{aligned}z\mathbf{X}(z) &= \mathbf{A}\mathbf{X}(z) + \mathbf{b}U(z), \\ Y(z) &= \mathbf{c}^T\mathbf{X}(z) + dU(z),\end{aligned}\quad (3.27)$$

and after we eliminate the state vector  $\mathbf{X}(z)$ , we obtain the transfer function

$$H(z) = \frac{Y(z)}{U(z)} = \mathbf{c}^T(z\mathbf{1} - \mathbf{A})^{-1}\mathbf{b} + d.\quad (3.28)$$

Since

$$z = e^{j\omega T} \quad \text{and hence for technical frequencies } z|_{p=j\omega} = e^{j\omega T} \quad (3.29)$$

holds,  $H(e^{j\omega T})$  is obviously a periodical function of  $\omega$  resp.  $f$  with frequency-period  $2\pi/T$  resp.  $1/T$ .

The question that arises here is: What type of function is (3.28)? To answer that we have to consider the expression  $(z\mathbf{1} - \mathbf{A})^{-1}$  more closely. We assume of course that  $(z\mathbf{1} - \mathbf{A})^{-1}$  is nonsingular and

therefore the inverse exists (this means that the inversion will only be performed for the values of  $z$  at which it is possible). The inverse reads

$$(z\mathbf{1} - \mathbf{A})^{-1} = \frac{\text{adj}(z\mathbf{1} - \mathbf{A})}{\det(z\mathbf{1} - \mathbf{A})}, \quad (3.30)$$

where the determinant is a polynomial in  $z$  of degree  $N$ , also called the characteristic polynomial

$$\det(z\mathbf{1} - \mathbf{A}) = z^N + \alpha_1 z^{N-1} + \cdots + \alpha_{N-1} z + \alpha_N, \quad \alpha_k \in \mathcal{R} \quad (3.31)$$

and the elements of the adjugate matrix are the transposed cofactors of the matrix. The cofactors on the other hand are sub-determinants of  $(z\mathbf{1} - \mathbf{A})$  and therefore polynomials of at most degree  $(N - 1)$ .

The numerator of  $\mathbf{c}^T(z\mathbf{1} - \mathbf{A})^{-1}\mathbf{b}$  is hence a linear combination of cofactors, i.e., a linear combination of polynomials of degree  $(N - 1)$  and therefore a polynomial of at most degree  $(N - 1)$ . If we apply this result in Eq. (3.28), we will obtain a rational fractional function of  $z$

$$H(z) = \frac{\beta_0 z^N + \beta_1 z^{N-1} + \cdots + \beta_{N-1} z + \beta_N}{z^N + \alpha_1 z^{N-1} + \cdots + \alpha_{N-1} z + \alpha_N}, \quad (3.32)$$

where  $\beta_0 \neq 0$  only for  $d \neq 0$ , since  $\beta_0 = d$ .

We usually divide the nominator and the denominator by  $z^n$  and obtain in the numerator and in the denominator polynomials in  $z^{-1}$ :

$$H(z) = \frac{\beta_N z^{-N} + \beta_{N-1} z^{-(N-1)} + \cdots + \beta_1 z^{-1} + \beta_0}{\alpha_N z^{-N} + \alpha_{N-1} z^{-(N-1)} + \cdots + \alpha_1 z^{-1} + 1} = \frac{\sum_{k=0}^N \beta_k z^{-k}}{1 + \sum_{k=1}^N \alpha_k z^{-k}} = \frac{Y(z)}{U(z)}. \quad (3.33)$$

By multiplying both sides of (3.33) with the denominator we get

$$Y(z) \left( 1 + \sum_{k=1}^N \alpha_k z^{-k} \right) = U(z) \sum_{k=0}^N \beta_k z^{-k}. \quad (3.34)$$

Now we go back to the time domain with help of the inverse  $\mathcal{Z}$ -transform and obtain a global difference equation

$$\begin{aligned} y(n) + \sum_{k=1}^N \alpha_k y(n-k) &= \sum_{k=0}^N \beta_k u(n-k), \\ y(n) &= \sum_{k=0}^N \beta_k u(n-k) - \sum_{k=1}^N \alpha_k y(n-k). \end{aligned} \quad (3.35)$$

The difference equation is equivalent to the description of a discrete-time system like a differential equation is for the description of a continuous-time system.

The representation of Eq. (3.35) leads us to an important special case, for which we have no equivalent in continuous-time systems built with lumped elements: the so-called finite impulse (FIR) response systems.

### 1.03.4.3 Finite duration impulse response systems

Those are systems with, as the name says, a finite duration impulse response (FIR), i.e., the  $\alpha_i$ 's in (3.35) are equal to zero such that

$$y(n) = \sum_{k=0}^N \beta_k u(n-k) \quad (3.36)$$

and with a unit impulse as excitation  $u(n) = \delta(n)$  we get

$$y(n) = h(k) = \sum_{k=0}^N \beta_k \delta(n-k) = \begin{cases} 0, & n < 0, \\ \beta_n, & 0 \leq n \leq N, \\ 0, & n > N. \end{cases} \quad (3.37)$$

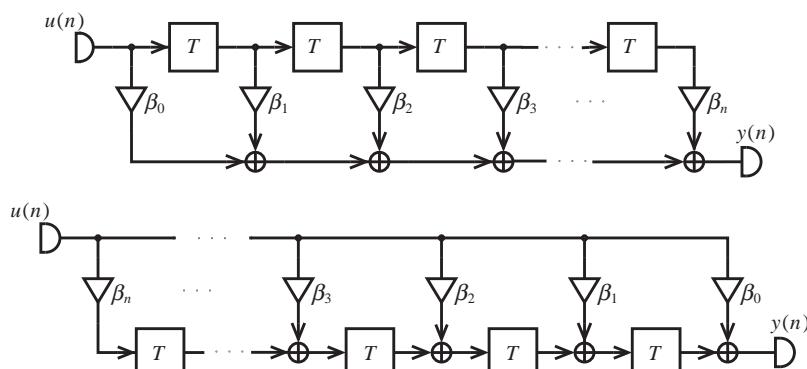
We can see that the coefficients of the polynomial transfer function directly give then values of the impulse response and so that it takes us directly to an implementation, see Figure 3.8. Note that the two structures are inter-convertible if the principle of flow reversal is applied to the block diagram.

Let us now write down the state space description for the uppermost structure in Figure 3.8. By calling the input of the delay elements  $x_k(n+1)$  and their outputs  $x_k(n)$  indexing them from the left most to the rightmost we can see that

$$\begin{aligned} x_1(k+1) &= u(n), \\ x_2(k+1) &= x_1(n), \\ &\vdots = \vdots \\ x_N(k+1) &= x_{N-1}(n), \end{aligned} \quad (3.38)$$

and for the output

$$y(n) = \beta_1 x_1(n) + \beta_2 x_2(n) + \cdots + \beta_N x_N(n) + \beta_0 u(n) \quad (3.39)$$



**FIGURE 3.8**

Two FIR realizations inter-convertible through the principle of flow reversal.

holds. In matrix vector notation we get

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \\ \vdots \\ x_N(n+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_N(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(n), \quad (3.40)$$

$$y(n) = [\beta_1 \beta_2 \cdots \beta_N] \mathbf{x}(n) + \beta_0 u(n). \quad (3.41)$$

It can be easily demonstrated that the second structure in Figure 3.8 is obtained by a transposition of the state-space description of the first structure, the matrix and vectors for it are then given by

$$\mathbf{A}' = \mathbf{A}^T, \quad \mathbf{b}' = \mathbf{c}, \quad \mathbf{c}'^T = \mathbf{b}^T, \quad d' = d. \quad (3.42)$$

Other usual names for FIR systems are:

- Non-recursive systems,
- Moving Average (MA),
- Transversal filter.

#### 1.03.4.4 Infinite duration impulse response systems

On the other hand if the  $\alpha_i$ 's are different from zero, then the impulse response will have infinite duration, as the name says. We can again clearly derive a realization structure from the corresponding time-domain description, in this case from Eq. (3.35), see Figure 3.9. This per se inefficient realization (two times the number of necessary delay elements) can be transformed into the well known direct form through simple rearranging the two blocks like in Figure 3.10. This is called a canonical realization because it possesses a minimum number of delay elements.

Now we can setup the state-space equations for the IIR system structure of Figure 3.10. As already mentioned we label the output of the delays the states  $x_k(n)$  and their inputs  $x_k(n+1)$ . By numbering from the topmost to the lowest delay we can see that

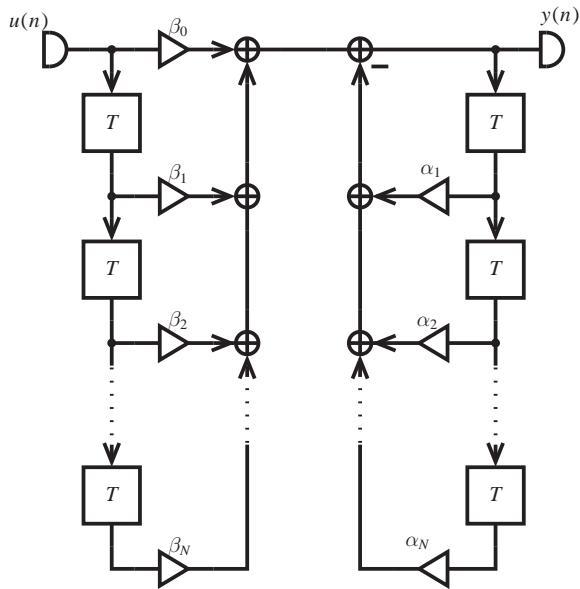
$$x_1(n+1) = -\alpha_1 x_1(n) - \alpha_2 x_2(n) - \dots - \alpha_N x_N(n) + u(n) \quad (3.43)$$

holds and from the second until the  $N$ th-state we get

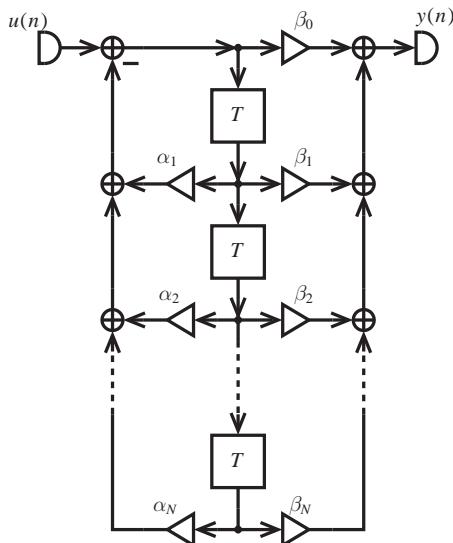
$$\begin{aligned} x_2(n+1) &= x_1(n), \\ x_3(n+1) &= x_2(n), \\ &\vdots = \vdots \\ x_N(n+1) &= x_{N-1}(n). \end{aligned} \quad (3.44)$$

For the output we can see that

$$\begin{aligned} y(n) &= \beta_0 x_1(n+1) + \beta_1 x_1(n) + \beta_2 x_2(n) + \dots + \beta_N x_N(n) \\ &= (\beta_1 - \beta_0 \alpha_1) x_1(n) + (\beta_2 - \beta_0 \alpha_2) x_2(n) + \dots + (\beta_N - \beta_0 \alpha_N) x_N(n) + \beta_0 u(n), \end{aligned} \quad (3.45)$$

**FIGURE 3.9**

IIR realization according to Eq. (3.35).

**FIGURE 3.10**

Canonical direct realization of an IIR system.

where we have used the definition of  $x_1(n+1)$  of (3.43). In matrix vector notation we get

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \\ \vdots \\ x_N(n+1) \end{bmatrix} = \begin{bmatrix} -\alpha_1 & -\alpha_2 & \cdots & -\alpha_N \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(n) \\ x_2(n) \\ \vdots \\ x_N(n) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(n), \quad (3.46)$$

$$y(n) = [\beta_1 - \beta_0\alpha_1, \beta_2 - \beta_0\alpha_2, \dots, \beta_N - \beta_0\alpha_N] \mathbf{x}(n) + \beta_0 u(n), \quad (3.47)$$

where because of its particular structure,  $\mathbf{A}$  is called a companion matrix.

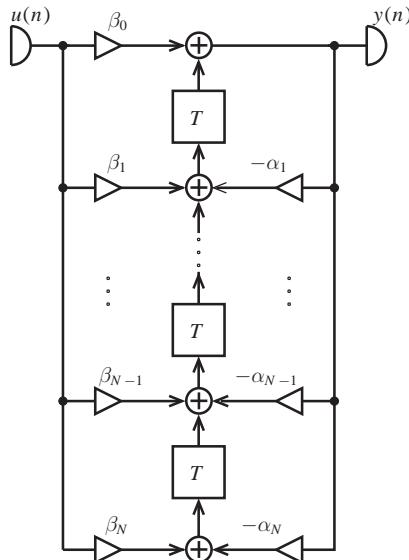
If we apply the principle of flow reversal to the diagram of Figure 3.10 we will end up with the realization shown in Figure 3.11. This is again equivalent to transform the state-space description into the transposed system, i.e., the transformed system is represented by

$$\mathbf{A}' = \mathbf{A}^T, \quad \mathbf{b}' = \mathbf{c}, \quad \mathbf{c}'^T = \mathbf{b}^T, \quad d' = d. \quad (3.48)$$

If we apply these definitions to (3.28) we can see that

$$\begin{aligned} H'(z) &= \mathbf{c}'^T (z\mathbf{1} - \mathbf{A}')^{-1} \mathbf{b}' + d' = \mathbf{b}^T (z\mathbf{1} - \mathbf{A}^T)^{-1} \mathbf{c} + d = (H'(z))^T \\ &= \mathbf{c}^T ((z\mathbf{1} - \mathbf{A}^T)^{-1})^T (\mathbf{b}^T)^T + d = \mathbf{c}^T (z\mathbf{1} - \mathbf{A})^{-1} \mathbf{b} + d = H(z). \end{aligned} \quad (3.49)$$

This means that the transposed realization has the same transfer function as the original system.



**FIGURE 3.11**

Transposed realization of the system from Figure 3.10.

Other usual names for IIR systems are:

- Recursive systems,
- Autoregressive Moving Average (ARMA),
- Systems with feedback.

#### 1.03.4.5 Observability and controllability

As we have seen in the previous section every LTI system can be completely characterized by a state-space description

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{b}u(n), \quad (3.50)$$

$$y(n) = \mathbf{c}^T\mathbf{x}(n) + du(n), \quad (3.51)$$

from which we can uniquely derive the corresponding transfer function

$$H(z) = \mathbf{c}^T (z\mathbf{1} - \mathbf{A})^{-1} \mathbf{b} + d. \quad (3.52)$$

It is important to note that although the way from a given state-space description to the input–output transfer function is unique, the same is not true for the other direction. For one given transfer function there are infinitely many different state-space realizations, which all may have a number of different properties, with the very same input–output relation. One of those properties is the stability of the so-called LTI system.

An often applied stability criterion is to check whether the zeroes of the denominator polynomial of the transfer function  $H(z)$  lie within the unit circle of the  $z$ -plane or equivalently the impulse response

$$h(n) = \mathcal{Z}^{-1}\{H(z)\} \quad (3.53)$$

decays over time. In the following we will see that this is not telling us the complete truth about the stability of LTI systems. Because stability depends on the actual implementation, which in detail is reflected in the state-space description and not only in transfer function or impulse response. To get a clear view, we have to introduce the concept of observability and controllability, and based on these concepts define different types of stability.

Let us start with the definition of observability.

Starting with the state-space description (3.50) we assume that the excitation  $u(n) = 0$  for  $n \geq 0$  and an initial state  $\mathbf{x}(0) \neq \mathbf{0}$ . Let us now compute the system output in time domain:

$$\begin{aligned} y(0) &= \mathbf{c}^T \mathbf{x}(0), \\ y(1) &= \mathbf{c}^T \mathbf{x}(1) = \mathbf{c}^T (\mathbf{A}\mathbf{x}(0)), \\ &\vdots \\ y(N-1) &= \mathbf{c}^T \mathbf{A}^{N-1} \mathbf{x}(0). \end{aligned} \quad (3.54)$$

Let us stack these subsequent  $N$  output values in one vector

$$\mathbf{y}(0) = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{c}^T \\ \mathbf{c}^T \mathbf{A} \\ \mathbf{c}^T \mathbf{A}^2 \\ \vdots \\ \mathbf{c}^T \mathbf{A}^{N-1} \end{bmatrix} \mathbf{x}(0) = \mathcal{O}(\mathbf{c}^T, \mathbf{A}) \mathbf{x}(0). \quad (3.55)$$

From (3.55) we see that this output vector  $\mathbf{y}(0)$  is given by the product of the so called observability matrix  $\mathcal{O}(\mathbf{c}^T, \mathbf{A})$ , which is completely determined by  $\mathbf{A}$  and  $\mathbf{c}^T$  and the system state at the time instant  $n = 0 : \mathbf{x}(0)$ . Now if the observability matrix is invertible, we can compute the state vector

$$\mathbf{x}(0) = \mathcal{O}(\mathbf{c}^T, \mathbf{A})^{-1} \mathbf{y}(0). \quad (3.56)$$

If this is possible, the system (3.50) is completely observable.

Next we define controllability in a corresponding way by assuming an initial zero state  $\mathbf{x}(0) = \mathbf{0}$  and let us look for a sequence  $u(n)$ ,  $n \geq 0$  to drive the system into some desired state  $\mathbf{x}(k)$ . We look at the evolution of the state as controlled by the excitation

$$\begin{aligned} \mathbf{x}(1) &= \mathbf{b}u(0), \\ \mathbf{x}(2) &= \mathbf{Ax}(1) + \mathbf{bu}(1) = \mathbf{Ab}u(0) + \mathbf{bu}(1), \\ \mathbf{x}(3) &= \mathbf{Ax}(2) + \mathbf{bu}(2) = \mathbf{A}^2\mathbf{b}u(0) + \mathbf{Ab}u(1) + \mathbf{bu}(2), \\ &\vdots \\ \mathbf{x}(N) &= \mathbf{A}^{N-1}\mathbf{b}u(0) + \mathbf{A}^{N-2}\mathbf{b}u(1) + \dots + \mathbf{A}^2\mathbf{b}u(N-3) + \mathbf{Ab}u(N-2) + \mathbf{bu}(N-1). \end{aligned} \quad (3.57)$$

This can be put together as

$$\mathbf{x}(N) = [\mathbf{b}, \mathbf{Ab}, \mathbf{A}^2\mathbf{b}, \dots, \mathbf{A}^{N-1}\mathbf{b}] \begin{bmatrix} u(N-1) \\ u(N-2) \\ u(N-3) \\ \vdots \\ u(0) \end{bmatrix} = \mathcal{C}(\mathbf{b}, \mathbf{A})\mathbf{u}(N), \quad (3.58)$$

where  $\mathcal{C}(\mathbf{b}, \mathbf{A})$  is the so called controllability matrix and  $\mathbf{u}(N)$  is the vector, in which the subsequent excitation samples have been stacked. If the controllability matrix is invertible, we can solve for the excitation

$$\mathbf{u}(N) = \mathcal{C}(\mathbf{b}, \mathbf{A})^{-1} \mathbf{x}(N), \quad (3.59)$$

which will steer the system from the zero initial state into a specified desired state  $\mathbf{x}(N)$ . If this is possible, the system (3.50) is completely controllable.

### 1.03.4.6 Stability

Based on the concepts of observability and controllability we can now define different concepts of stability:

- Internal stability,
- Output-stability,
- Input-stability,
- Input–output stability.

We will start with **internal stability** by requiring that the Euclidean norm of the state vector hat to decay over time from any initial value with zero excitation:

$$\forall \mathbf{x}(0) \in \mathcal{R}^N \text{ and } u(n) = 0, \text{ for } n \geq 0 : \lim_{n \rightarrow \infty} \|\mathbf{x}(n)\|_2 = 0. \quad (3.60)$$

This is equivalent to require the state vector converge to the zero vector.

By looking at the evolution of the state vector over time

$$\begin{aligned} \mathbf{x}(1) &= \mathbf{A}\mathbf{x}(0), \\ \mathbf{x}(2) &= \mathbf{A}\mathbf{x}(1) = \mathbf{A}^2\mathbf{x}(0), \\ &\vdots \\ \mathbf{x}(n) &= \mathbf{A}^n\mathbf{x}(0), \end{aligned} \quad (3.61)$$

and making use of the eigenvalue decomposition (EVD) of  $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$ ,  $\Lambda = \text{diag}\{\lambda_i\}_{i=1}^N$  we get

$$\mathbf{Q}^{-1}\mathbf{x}(n) = \Lambda^n \mathbf{Q}^{-1}\mathbf{x}(0). \quad (3.62)$$

From (3.62) we can conclude that the vector  $\mathbf{Q}^{-1}\mathbf{x}(n)$  will converge to the zero vector, i.e.,  $\lim_{n \rightarrow \infty} \|\mathbf{Q}^{-1}\mathbf{x}(n)\|_2 = 0$  if and only if all eigenvalues  $\lambda_i$  of  $\mathbf{A}$  are smaller than one in magnitude

$$|\lambda_i| < 1 \iff \lim_{n \rightarrow \infty} \|\mathbf{Q}^{-1}\mathbf{x}(n)\|_2 = 0 \iff \lim_{n \rightarrow \infty} \|\mathbf{x}(n)\|_2 = 0. \quad (3.63)$$

This equivalent to say that the poles of the transfer function are localized inside the unity circle, since the eigenvalues of  $\mathbf{A}$  equal to the poles of the transfer function. In the above derivation we have assumed that  $\mathbf{A}$  is a diagonalizable matrix. If this is not the case because of multiple eigenvalues, we have to refrain to the so called Jordan form. Although this is slightly more complicated, it leads to the same result.

A somewhat weaker stability criterion is to check only if the system output decays to zero from any initial state and zero excitation:

$$\forall \mathbf{x}(0) \in \mathcal{R}^N \text{ and } u(n) = 0 \text{ for } n \geq 0 : \lim_{n \rightarrow \infty} |y(n)|^2 = 0. \quad (3.64)$$

Computing the output we get

$$\begin{aligned} y(n) &= \mathbf{c}^T \mathbf{x}(n) = \mathbf{c}^T \mathbf{A}^n \mathbf{x}(0) = \mathbf{c}^T \mathbf{Q} \Lambda^n \mathbf{Q}^{-1} \mathbf{x}(0) = \eta^T \Lambda^n (\mathbf{Q}^{-1} \mathbf{x}(0)) \\ &= [\eta_1 \ \eta_2 \cdots \eta_N] \begin{bmatrix} \lambda_1^n & 0 & \cdots & 0 \\ 0 & \lambda_2^n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_N^n \end{bmatrix} (\mathbf{Q}^{-1} \mathbf{x}(0)) = \sum_{i=1}^N \eta_i \lambda_i^n (\mathbf{Q}^{-1} \mathbf{x}(0))_i. \end{aligned} \quad (3.65)$$

$\eta^T = \mathbf{c}^T \mathbf{Q}$  is the transformed output vector. If an entry  $\eta_i$  is equal to zero, then the eigenmode (normal mode)  $\lambda_i$  will not contribute to the system output and, therefore, a non-decaying eigenmode  $|\lambda_i| \geq 1$  will not violate the **output-stability** criterion. This can only happen if the observability matrix is not full rank. From (3.55) we have

$$\begin{aligned} \mathbf{y}(0) &= \mathcal{O}(\mathbf{c}^T, \mathbf{A})\mathbf{x}(0) = \mathcal{O}(\mathbf{c}^T, \mathbf{A})\mathbf{Q}(\mathbf{Q}^{-1}\mathbf{x}(0)) = \mathcal{O}(\eta^T, \Lambda)(\mathbf{Q}^{-1}\mathbf{x}(0)) \\ &= \begin{bmatrix} \eta_1 & \eta_2 & \dots & \eta_N \\ \eta_1\lambda_1 & \eta_2\lambda_2 & \dots & \eta_N\lambda_N \\ \eta_1\lambda_1^2 & \eta_2\lambda_2^2 & \dots & \eta_N\lambda_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \eta_1\lambda_1^{N-1} & \eta_2\lambda_2^{N-1} & \dots & \eta_N\lambda_N^{N-1} \end{bmatrix} (\mathbf{Q}^{-1}\mathbf{x}(0)) \end{aligned} \quad (3.66)$$

and that  $\mathcal{O}(\eta^T, \Lambda)$  is rank deficient if at least one  $\eta_i$  is zero, and since  $\mathbf{Q}$  is full rank  $\mathcal{O}(\mathbf{c}^T, \mathbf{A})$  must also be rank deficient. Therefore, a system can be output stable without being internally stable, if it is not completely observable.

A complementary criterion is that of **input-stability**, requiring the state vector converging to the zero vector, but not from any arbitrary initial state, but only from states which can be controlled from the system input. Now we start with an initial state  $\mathbf{x}(N)$ , which can be generated from an input sequence (see (3.58))

$$\mathbf{x}(N) = \mathcal{C}(\mathbf{b}, \mathbf{A})\mathbf{u}(N). \quad (3.67)$$

The transformed state  $\mathbf{Q}^{-1}\mathbf{x}(N)$  then reads

$$\mathbf{Q}^{-1}\mathbf{x}(N) = \mathbf{Q}^{-1}\mathcal{C}(\mathbf{b}, \mathbf{A})\mathbf{u}(N) = \mathcal{C}(\mu, \Lambda)\mathbf{u}(N) \quad (3.68)$$

with the transformed input vector  $\mu = \mathbf{Q}^{-1}\mathbf{b}$ . After the input  $\mathbf{u}(N)$  has produced the state  $\mathbf{x}(N)$  it is switched off, i. e.  $u(n) = 0$  for  $n \geq N$ .

We ask whether the state vector can converge to the zero vector although the system may not be internally stable. The answer is given by the structure of

$$\mathcal{C}(\mu, \Lambda) = \begin{bmatrix} \mu_1 & \mu_2 & \dots & \mu_N \\ \mu_1\lambda_1 & \mu_2\lambda_2 & \dots & \mu_N\lambda_N \\ \vdots & \vdots & \ddots & \vdots \\ \mu_1\lambda_1^{N-1} & \mu_2\lambda_2^{N-1} & \dots & \mu_N\lambda_N^{N-1} \end{bmatrix} = \mathbf{Q}^{-1}\mathcal{C}(\mathbf{b}, \mathbf{A}). \quad (3.69)$$

If some  $\mu_i = 0$ , then  $\mathbf{Q}^{-1}\mathbf{x}(0)$  will converge to the zero vector, even if the corresponding  $|\lambda_i| \geq 1$ , because this non-decaying eigenmode cannot be excited from the regular input. But this is only possible, if  $\mathcal{C}(\mathbf{b}, \mathbf{A})$  is not full rank and the system is, therefore, not completely controllable.

Finally we look at the concept of the commonly used **input-output-stability**, i.e., we excite the system with zero initial state with a unit pulse and require the output to converge to zero over time. In this setting the state and the output evolves as

$$\mathbf{x}(n) = \mathbf{A}^n \mathbf{b}, \quad (3.70)$$

$$y(n) = \mathbf{c}^T \mathbf{A}^n \mathbf{b} = \mathbf{c}^T \mathbf{Q} \Lambda^n \mathbf{Q}^{-1} \mathbf{b} = \eta^T \Lambda^n \mu = \sum_{i=1}^N \eta_i \mu_i \lambda_i^n. \quad (3.71)$$

The output will converge to zero, even if there are non-decaying eigenmodes  $\lambda_i$ , as long as for every such  $\lambda_i$  there is a  $\eta_i = 0$  or a  $\mu_i = 0$ , or both are zero. We see that a system could be input–output stable, although it is neither internally stable nor input- or output-stable.

Only for systems, which are completely observable and controllable, the four different stability criteria coincide.

**Example 3.1.** The four stability concepts will now be illustrated with an example: the Cascade Integrator Comb (CIC) filter [6]. These filters are attractive in situations where either a narrow-band signal should be extracted from a wideband source or a wideband signal should be constructed from a narrowband source. The first situation is associated with the concept of sampling rate decrease or decimation and the second one leads to the concept of sampling rate increase or interpolation. CIC filters are very efficient from a hardware point of view, because they need no multipliers. In the first case, i.e., decimation, the integrators come first, then the sampling rate will be decreased by a factor of  $R$  followed by a subsequent cascade of comb filters (Figure 3.12a), while in the second case, i.e., interpolation, the comb filters come first, then the sampling rate increase is followed by a subsequent cascade of integrators (Figure 3.12b).

The stability problem in these CIC filters stems from the integrators, which obviously exhibit a non-decaying eigenmode (pole with unit magnitude). To analyze the stability of such CIC filters we set the sampling rate decrease/increase factor  $R = 1$  and the number of integrators and comb filters stages to  $L = 1$  without loss of generality. In addition we assume a differential delay of  $M = 3$  samples per stage. This leads us to the following two implementations to be analyzed which are shown in Figures 3.13a and b.

Let us start with an analysis of the filter in Figure 3.13a, the state space description of which reads

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{b}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, d_1 = 1 \quad (3.72)$$

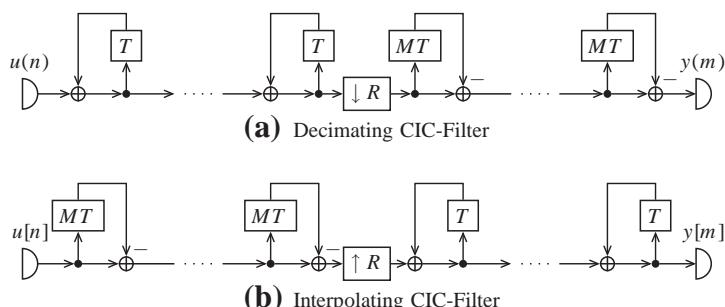
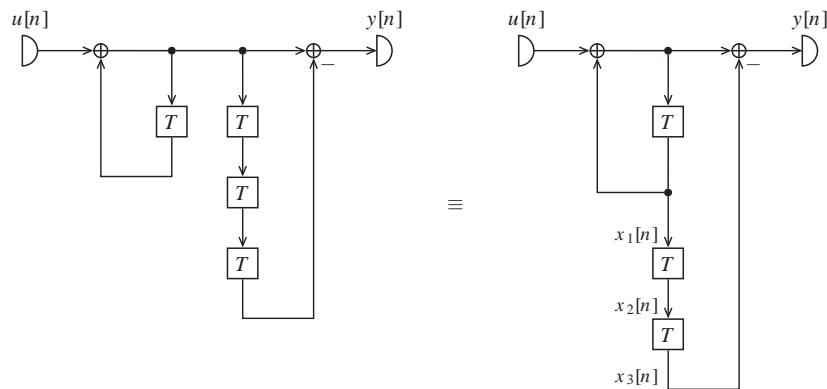
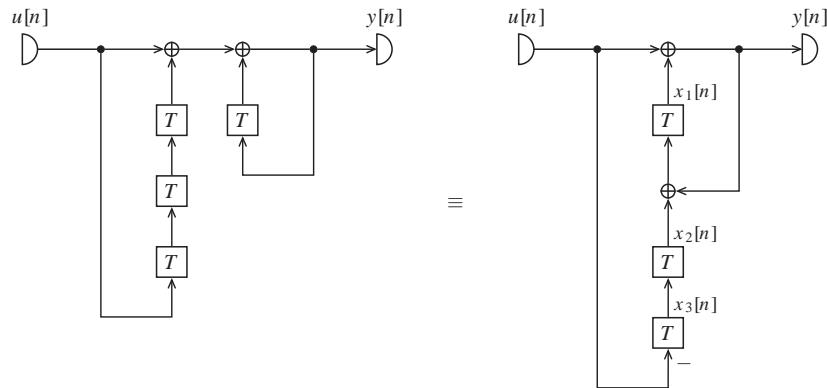


FIGURE 3.12

Multiplierless decimator and integrator.



(a) Cascade of Integrator and Comb Filter



(b) Cascade of Comb and Integrator Filter

**FIGURE 3.13**

CIC Filters without sampling rate alteration with  $R = 1$ ,  $L = 1$  and  $M = 3$ .

and leads to the transfer function

$$H_1(z^{-1}) = \frac{1 - z^{-3}}{1 - z^{-1}} = 1 + z^{-1} + z^{-2}. \quad (3.73)$$

The observability matrix

$$\mathcal{O}(\mathbf{c}_1^T, \mathbf{A}_1) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \det(\mathcal{O}(\mathbf{c}_1^T, \mathbf{A}_1)) = 0 \quad (3.74)$$

is obviously rank deficient and the system is therefore not completely observable.

Since the state matrix  $\mathbf{A}$  is not diagonalizable, we have to refrain to a Jordan form

$$\mathbf{A}_1 = \mathbf{Q}_1 \mathbf{J} \mathbf{Q}_1^{-1} = \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{3}} \\ 0 & 1 & \frac{1}{\sqrt{3}} \\ 1 & 0 & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 1 & 0 \\ \sqrt{3} & 0 & 0 \end{bmatrix}, \quad (3.75)$$

which shows the three eigenvalues are  $\lambda_1 = 0$ ,  $\lambda_2 = 0$  and  $\lambda_3 = 1$ . Not all eigenvalues are less than one in magnitude. Therefore, the system of Figure 3.13a is not internally stable. But transforming the observability matrix according to (3.66), we get

$$\mathcal{O}(\boldsymbol{\eta}^T, \mathbf{J}) = \mathcal{O}(\mathbf{c}_1^T, \mathbf{A}_1) \mathbf{Q}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{3}} \\ 0 & 1 & \frac{1}{\sqrt{3}} \\ 1 & 0 & \frac{1}{\sqrt{3}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.76)$$

with the third component  $\eta_3$  of the transformed output vector  $\boldsymbol{\eta}^T = \mathbf{c}_1^T \mathbf{Q}_1$  equal to zero. This clearly means that the non-vanishing third eigenmode is not observable at the system output. Therefore the system is output stable, i.e., the output converges in the limit  $\lim_{n \rightarrow \infty} \mathbf{y}(n) = 0$ , although the state vector does not converge to the zero vector. But the controllability matrix

$$\mathcal{C}(\mathbf{b}_1, \mathbf{A}_1) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad \det(\mathcal{C}(\mathbf{b}_1, \mathbf{A}_1)) = 1 \quad (3.77)$$

is full rank, i.e., the system is completely controllable, and an arbitrary input sequence may excite the non-decaying eigenmode. Therefore, the system is not input-stable.

Now let us analyze the filter in Figure 3.13b, which has the following state-space description

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad d_2 = 1 \quad (3.78)$$

and the same transfer function as the previous filter from Figure 3.13a. It is interesting to note that the second filter can be obtained from the first one by applying transposition

$$\mathbf{A}_2 = \mathbf{A}_1^T, \quad \mathbf{b}_2 = \mathbf{c}_1, \quad \mathbf{c}_2 = \mathbf{b}_1, \quad d_2 = d_1. \quad (3.79)$$

The controllability matrix of the second system

$$\mathcal{C}(\mathbf{b}_2, \mathbf{A}_2) = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \det(\mathcal{C}(\mathbf{b}_2, \mathbf{A}_2)) = 0 \quad (3.80)$$

is obviously rank deficient and the system is therefore not completely controllable.

Since the state matrix  $\mathbf{A}_2 = \mathbf{A}^T$  is not diagonalizable we have again to refrain to a Jordan form

$$\mathbf{A}_2 = \mathbf{Q}_2 \mathbf{J} \mathbf{Q}_2^{-1} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 \\ -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -\sqrt{2} & 0 \\ 0 & 0 & -\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix}, \quad (3.81)$$

which again shows obviously the same eigenvalues  $\lambda_1 = 0$ ,  $\lambda_2 = 0$  and  $\lambda_3 = 1$ . The second system is again not internally stable. Transforming the above controllability matrix according to (3.68) we get

$$\mathcal{C}(\boldsymbol{\mu}, \mathbf{J}) = \mathbf{Q}_2^{-1} \mathcal{C}(\mathbf{b}_2, \mathbf{A}_2) = \begin{bmatrix} 0 & -\sqrt{2} & 0 \\ 0 & 0 & -\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \sqrt{2} & 0 \\ \sqrt{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (3.82)$$

with the third component  $\mu_3$  of the transformed input vector  $\boldsymbol{\mu} = \mathbf{Q}_2^{-1} \mathbf{b}_2$  being equal to zero. This shows that the non-decaying third eigenmode will not be excited by any possible input signal. Therefore, the system is input-stable, although it is not internally stable.

The observability matrix of the second system

$$\mathcal{O}(\mathbf{c}_2^T, \mathbf{A}_2) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \det(\mathcal{O}(\mathbf{b}_2, \mathbf{A}_2)) = 1 \quad (3.83)$$

is full rank and the second system is completely observable. Therefore the non-decaying eigenmode, although not excited by any input sequence, is observable at the output. This can always happen because of an unfavorable initial state accidentally occurring in the switch-on transient of the power supply or because of some disturbance entering the system not through the regular input path. Therefore, the system is not output-stable.

But both systems are input–output-stable, because both have the same impulse response with finite duration.

This example shows, that the standard input–output stability criterion, i.e., requiring a decaying impulse response, is not always telling the whole story. Judging whether a system is stable or not needs a detailed knowledge about the system structure, or in other words the realization, and not only about input–output-mapping.

## 1.03.5 Discrete-time signals and systems with MATLAB

In this section we will provide some examples how to generate discrete-time signals in MATLAB and how to represent and implement basic discrete-time systems. We assume here a basic MATLAB installation. The commands and programs shown here are not unique in the way they perform the analysis and implement discrete-time signals and systems. Our objective here is to give an introduction with very simple commands and functions. With this introduction the reader will get more familiar with this powerful tool that is widely employed to analyze, simulate and implement digital signal processing systems. If one has access to the full palette of MATLAB toolboxes, like e.g., Signal Processing Toolbox, Control System Toolbox, Communications System Toolbox and DSP System Toolbox, many of the scripts included here can be substituted by functions encountered in those libraries.

### 1.03.5.1 Discrete-time signals

Discrete-time signals are defined in MATLAB as one dimensional arrays, i.e., as vectors. They can be row or column vectors and their entries can be either real or complex valued. The signals we will generate and analyze in this section are always row vectors and their entries are always real valued. The length of the vectors will be defined according to the time span in which the system is supposed to be analyzed.

#### 1.03.5.1.1 Unit impulse

The unit impulse  $\delta(n)$  can be generated with the commands

```
L = 100;
delta = [1 zeros(1,L-1)];
```

where  $L$  is the length of the desired input and the function `zeros(1,L-1)` generates a row vector with  $L-1$  zeros.

The unit impulse can be further weighted and delayed. For example,  $2*\delta(n - 5)$  can be generated by

```
L = 100;
N = 5;
delta = [zeros(1,N) 2 zeros (1,L-1-N)];
```

where we have assumed a delay of 5 samples or sampling periods.

The length of the vector containing only a unit impulse depends on the objective of the analysis. For example, if the impulse response of a discrete-time system is to be studied, there should be enough elements in the input array so that a significant part of the impulse response is contained in the output vector.

#### 1.03.5.1.2 Sinusoid

A sinusoid can be generated with the help of the function `sin`. Below is an example of a sinusoid where all its parameters are first defined

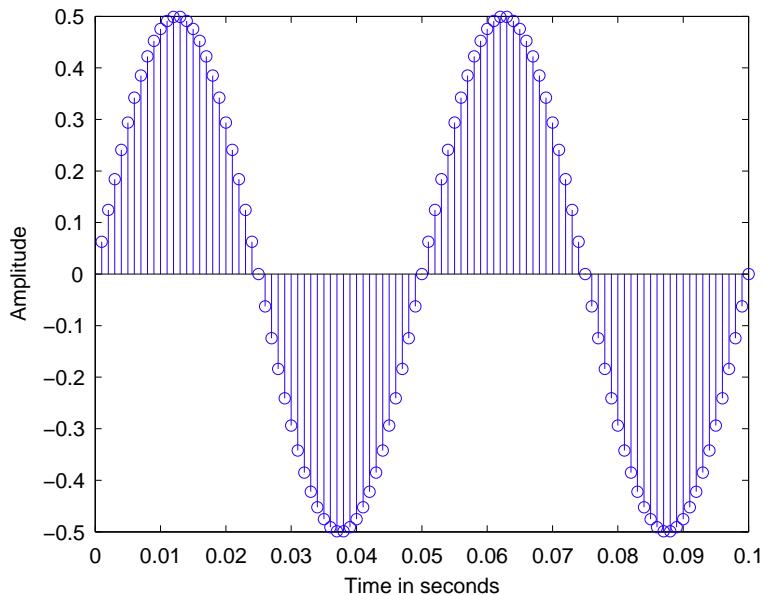
```
fs = 1000; % Sampling frequency in Hz
T = 1/fs; % Sampling period
n = 1:100; % Time indexes
U = 0.5; % Amplitude of the sinusoid
fc = 20; % Sinusoid frequency in Herz
omega = 2*pi*fc; % Sinusoid angular frequency in rad/s
phi = 0; % phase in radians
x = U* sin(omega*n*T+phi);
```

We can see that the comments after each command explain what the parameter means.

If we would like to graphically represent the sinusoid, we could use

```
stem(n*T,x);
xlabel('Time in seconds');
ylabel('Amplitude');
```

and we get the plot in Figure 3.14, where we can see two periods of the sinusoid. An alternative to `stem` is the command `plot`. If we employ it using the same parameters and the same syntax we obtain

**FIGURE 3.14**

Discrete sinusoid with  $f_c = 20$  Hz and  $f_s = 1$  kHz plotted in MATLAB.

Figure 3.15, where it should be noted that MATLAB graphically connects the amplitude samples with a straight line and as a consequence the discrete-time sinusoid looks like a continuous-time one. It is important to keep in mind that the true discrete-time sinusoid is only defined for certain time instants and the amplitude between two true samples in the Figure 3.15 is only an approximation of the amplitude of the equivalent continuous-time sinusoid.

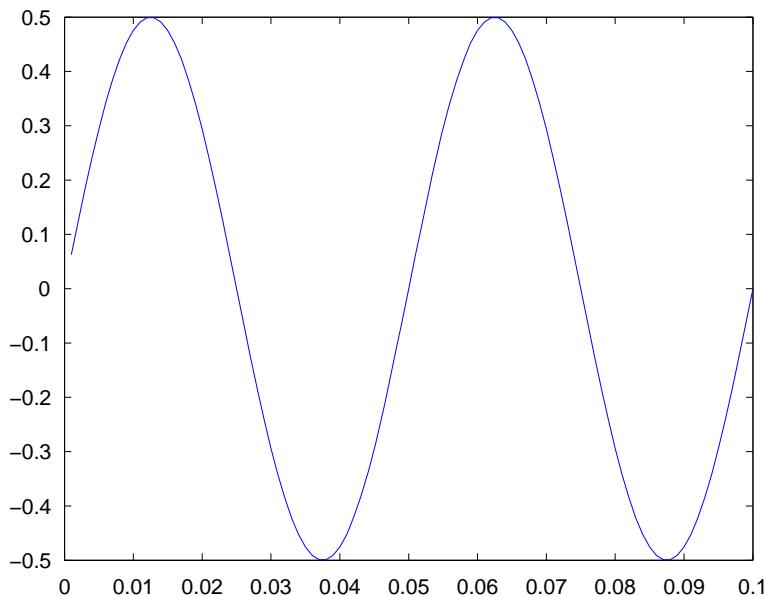
As an exercise the reader could play around with the parameters of the sinusoid. For example, he or she could change the phase  $\phi$  to  $\pi/2$  and plot the sinusoid again, then try to use the function `cos` instead of `sin` and set  $\phi$  to 0 again and compare with the previous plots. Change the sinusoid frequency until you reach the Nyquist frequency, which is 500 Hz for the example above. Compare the use of the command `plot` for different sampling rates and different sinusoid frequencies to see how the approximation of the continuous-time sinusoid gets worse.

### 1.03.5.1.3 White gaussian noise

To generate a WGN signal we have to employ the function `randn` that generates pseudo-random numbers following the normal or Gaussian distribution with zero mean and unit variance. By using

```
L = 100;
sigma2 = 0.5; % Variance of the WGN
u = sqrt(sigma2)*randn(1,L);
```

we generate a white Gaussian noise vector with zero mean and variance  $\sigma^2 = 0.5$ .

**FIGURE 3.15**

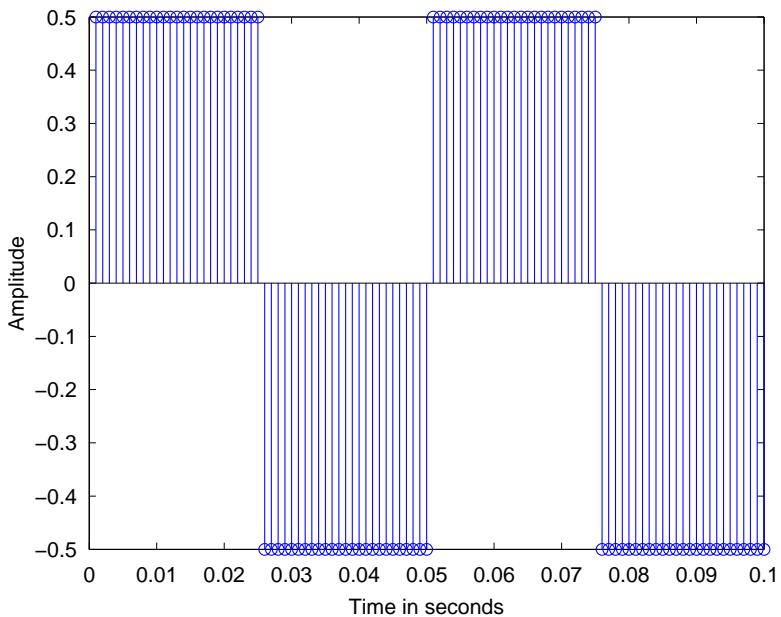
Discrete-time sinusoid represented in MATLAB with the command `plot`.

#### 1.03.5.1.4 Elaborated signal model

Many operations can be further performed with the generated signals, they can be, for example, added to each other or be multiplied by a constant or by another signal. As a last example let us generate a typical signal model used in communications systems. Let us consider an input signal composed by a sum of weighted and delayed unit impulses. This could be a data carrying signal generated at the transmitter. Then we multiply it by a sinusoid, that could represent a modulation to a higher frequency, for example the radio frequency (RF), to allow the propagation of the signal as an electromagnetic wave in a physical medium between transmitter and receiver. Finally, we add white Gaussian noise that represents the thermal noise generated by the analog components used in the transmitter and receiver.

The program to generate this signal is the following

```
L = 100;
u_in = [0.5*ones(1,L/4) -0.5*ones(1,L/4) 0.5*ones(1,L/4)
         -0.5*ones(1,L/4)]; % Sum of weighted and delayed unit impulses
fs = 1000; % Sampling frequency in Hz
T = 1/fs; % Sampling period
n = 1:L; % Time indexes
U = 1; % Amplitude of the sinusoid
fc = 50; % Sinusoid frequency in Herz
```

**FIGURE 3.16**

Weighted and delayed sum of unit impulses  $u_{in}$ .

```

omega = 2*pi*fc; % Sinusoid frequency in rad/s
phi = 0; % Phase in radians
u_sin = U*sin(omega*n*T+phi); % Sinusoid
sigma2 = 0.02; % Variance of the White Gaussian Noise
u_awgn = sqrt(sigma2)*randn(1,L); % White Gaussian Noise Vector
u_mod = u_in.*u_sin+u_awgn; % Modulation and noise addition

```

In Figure 3.16 we can see the signal that is composed by a sum of weighted and delayed unit impulses.

In Figure 3.17 the modulated sinusoid is depicted.

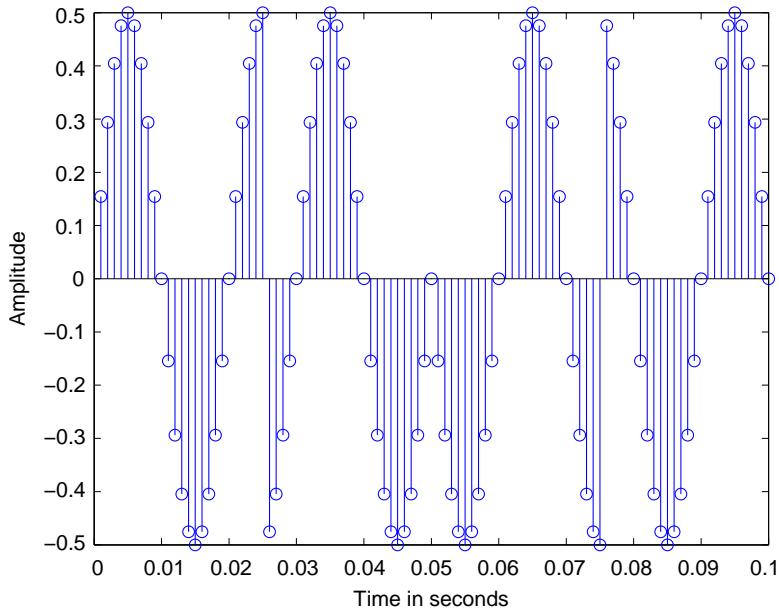
We can see in Figure 3.18 the modulated sinusoid with the additive white Gaussian noise.

### 1.03.5.2 Discrete-time systems representation and implementation

There are different ways to represent a discrete-time system in MATLAB, like, for example, the space-state or the transfer function. To start we revisit the elaborated signal model introduced in the last section. If we would like to, at least approximately, recover the input signal represented by the sum of delayed and weighted unit impulses, we should first demodulate it by multiplying it by a sinusoid with the same frequency of the one used to modulate

```
u_demod=u_mod.*u_sin
```

We will then obtain the sequence shown in Figure 3.19.

**FIGURE 3.17**

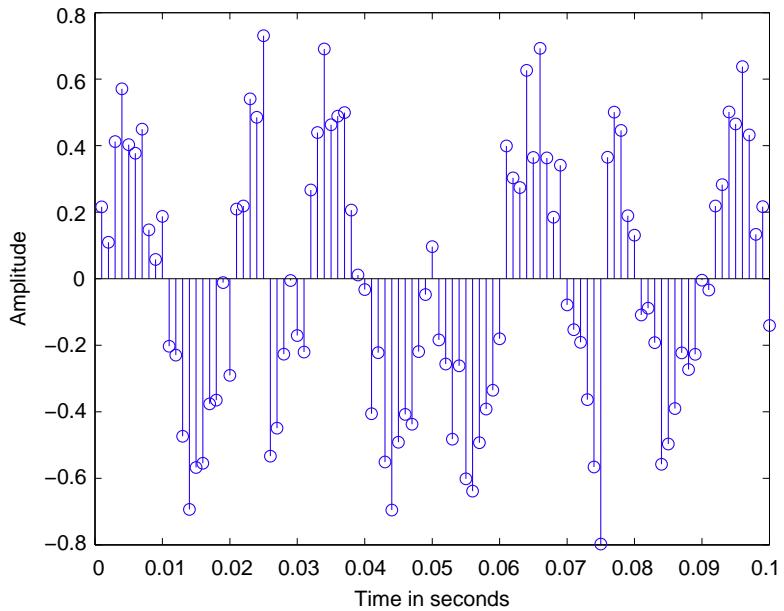
Modulated sinusoid  $u_{\text{in}} * u_{\text{sin}}$ .

The only problem with the new sequence is that not only an approximation of the desired signal is obtained, but also another modulated version of it, but this time with a sinusoid with the double of the original modulation frequency, and we still have the additive noise. If the noise variance is low enough we are still able to approximately recover the original signal without any further processing. But we have first to eliminate the modulated signal component and for this we will apply a discrete-time filter specific designed for this task. We assume here that the filter is given and we only care on how to represent and implement it. In [7] and the references therein many methods for the design of discrete-time filters can be encountered.

Let us say that a state-space description of the filter is already known and is defined as

```
A = [ 0.6969    0.8263   -0.6089   0.0111;
      1           0           0           0 ;
      0           1           0           0 ;
      0           0           1           0 ];
b = [1;
      0;
      0;
      0];
c = [-0.4272    1.0189   -0.1298   0.0160];
d = [0.9738];
```

where we can identify the companion matrix structure of  $\mathbf{A}$ .

**FIGURE 3.18**

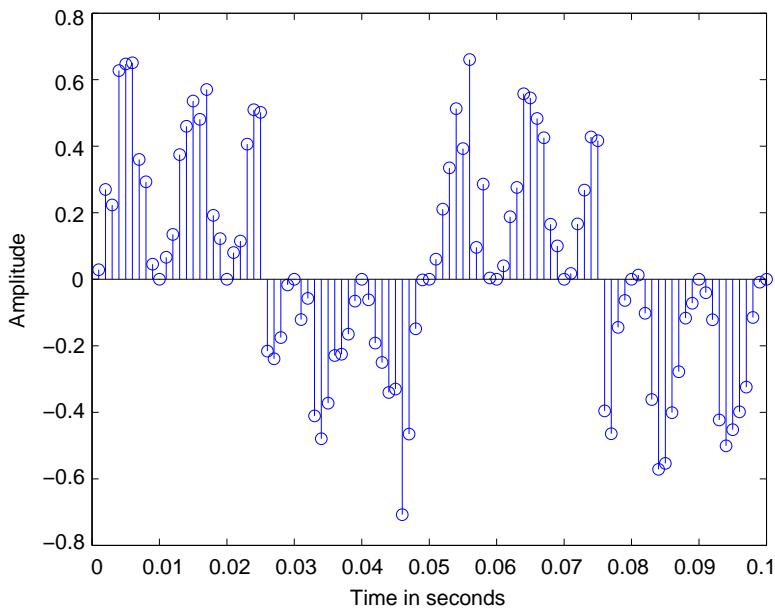
Modulated sinusoid with additive white Gaussian noise  $u_{\text{mod}}$ .

To compute the output of the filter given an initial state and the input signal, the state equations can be directly employed and iteratively calculated as

```
L = 100;
u= [1 zeros (1,L-1)]; % Input signal, here a unit impulse
x= [0; 0; 0; 0]; % Initial state vector
y = zeros(1,L); % Initialization of the output vector
for n = 1:L
    y(n)=c*x+d*u(n);
    x = A*x+b*u(n);
end
```

where we can see that the values of the state-vector are not stored for all time instants. If one is interested in looking at the internal states evolution, a matrix could be defined with so many rows as the number of states and so many columns as the length of the input/output signal, and the state vector for each time instant can be saved in its columns.

If we substitute the unit impulse by the demodulated signal as the input  $u = u_{\text{demod}}$  we obtain the signal depicted in Figure 3.20. One can see that we do not obtain exactly the input signal  $u_{\text{in}}$  but an approximation, particularly in communications systems that is usually the case, where it is important to recover the information contained in  $u_{\text{in}}$ , even if the waveform has been distorted, and not the waveform itself.

**FIGURE 3.19**

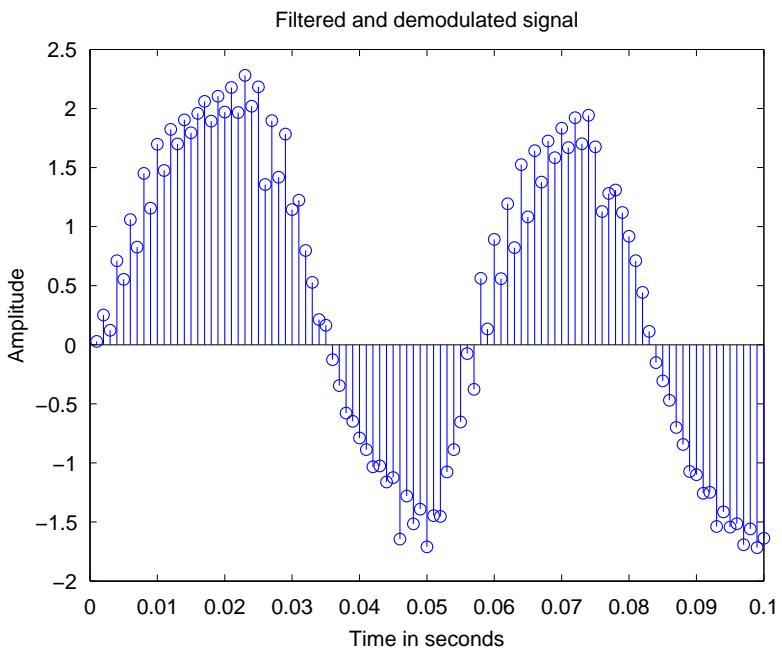
Demodulated signal  $u_{\text{demod}}$ .

But if one would like to implement Eq. (3.17) the program becomes more complicated as shown below

```

L=100;
u=[1 zeros(1,L-1)]; % Input signal, here a unit impulse
x_0z=[0; 0; 0; 0]; % Initial state vector
y=zeros(1,L); % Allocation of the output vector
cAb_array=zeros(1,L); % Allocation of the array c*A^(k-1)*b
A_power=eye(4); % Allocation of the powers of A matrix
cAb_array(1)=c*b; % First element of the array cAb_array
y(1)=d*u(1); % Output at sample n=0
for n=2:L
    cAb_array(n)=c*A_power*b; % Inclusion of a new element in cAb_array
    y(n)=c*(A*A_power)*x_0+sum(cAb_array(1:n).*u(n:-1:1))+d*u(n);
    % Output
    A_power=A*A_power; % New power of A
end

```

**FIGURE 3.20**

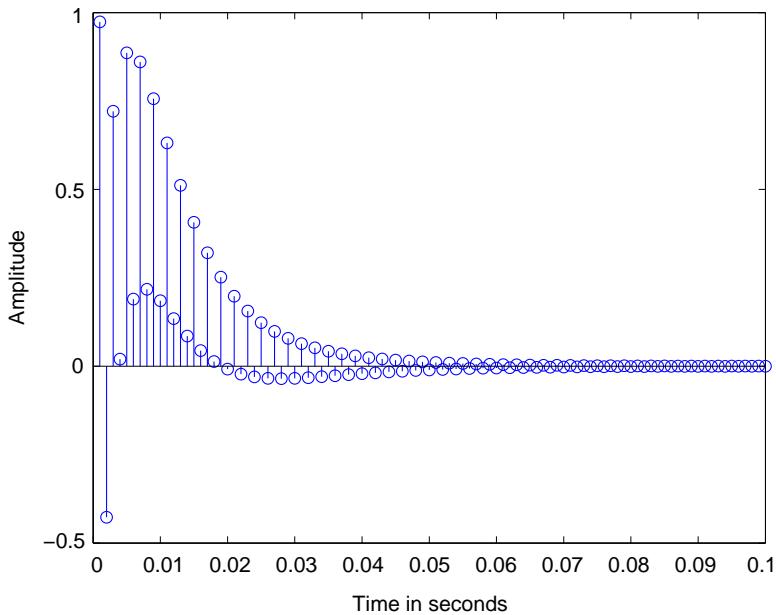
Demodulated and filtered signal  $u_{\text{demod}}$ .

In both programs presented above the impulse response can be obtained and its first 100 samples are plotted in Figure 3.21.

The similarity transformation can also be easily implemented. Let us consider the transformation into the normal form. The code to perform it is

```
[Q,Lambda]=eig(A); % Eigenvalue decomposition:  
% Lambda -> Eigenvalues in main diagonal  
% Q -> Eigenvectors in the columns  
T=Q;  
A_norm=inv(T)*A*T;  
b_norm=inv(T)*b;  
c_norm=c*T;  
d_norm=d;
```

The resulting state-space description for our example is

**FIGURE 3.21**

First 100 samples of the impulse response.

```

A_norm = [-0.9242      0      0      0
           0      0.8057      0      0
           0      0      0.7968      0
           0      0      0     0.0186];
b_norm = [ 0.6380;
           126.6454;
           127.3681;
           1.7322];
c_norm = [ -0.7502   0.2283   -0.2268   0.0140];
d_norm = 0.9738;

```

From the state-space representation it is possible to calculate the coefficients of the numerator and of the denominator polynomials of the transfer function with help of the Faddeev-Leverrier algorithm [8]

```

beta = [d zeros(1,3)];
alpha = [1 zeros(1,3)];
F = eye(4);
for i=1:4
    alpha(i+1) = -(1/i)*trace(A*F);

```

```

beta(i+1) = c*F*b+d*alpha(i+1);
F = A*F+alpha(i+1)*eye(4);
end

```

where the arrays beta and alpha contain the coefficients of the nominator and denominator polynomials.

With the coefficients of the transfer function it is also possible to calculate the output given the input of the system and some past values of the output by using the difference equation that can be implemented as

```

L=100;
u=[1 zeros(1,L-1)];
y=zeros(1,L);           % We assume a relaxed initial condition
y(1)=beta(1).*u(1);    % Transition phase or first initial output
y(2)=sum(beta(1:2).*u(2:-1:1))-alpha(2).*y(1);
y(3)=sum(beta(1:3).*u(3:-1:1))-sum(alpha(2:3).*y(2:-1:1));
y(4)=sum(beta(1:4).*u(4:-1:1))-sum(alpha(2:4).*y(3:-1:1));
for n=5:L
    y(n)=sum(beta.*u(n:-1:n-4))-sum(alpha(2:end).*y(n-1:-1:n-4));
end

```

As we saw before this equivalent to a state-space realization in direct form.

## 1.03.6 Conclusion

In this chapter we have introduced an important and fundamental topic in electrical engineering that provides the basics for digital signal processing. We have started with the presentation of some frequently used discrete-time signals. Then we have showed how to classify discrete-time systems and started the study of the widely employed class of linear time-invariant systems. We have showed the most basic way of representing LTI systems in a way that not only the input–output behavior is described, but also important internal signals, the so-called state-space representation. We have seen that the state-space representation is not unique and, consequently, there is room for optimization in the way discrete-time systems are realized. After that, the transfer function was derived from the state-space description and the two main classes of discrete-time systems, namely FIR and IIR were introduced. We formulated the concepts of controllability and observability, and showed how they can be used to evaluate the stability of the system. We finally gave some examples how to implement and analyze simple discrete-time signals and systems with MATLAB.

*Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 4](#) Random Signals and Stochastic Processes

See this Volume, [Chapter 5](#) Sampling and Quantization

See this Volume, [Chapter 6](#) Digital Filter Structures and Implementations

See this Volume, [Chapter 7](#) Multirate Signal Processing

See this Volume, [Chapter 12](#) Adaptive Filters

## References

- [1] Wikipedia.Fibonacci number—Wikipedia, the free encyclopedia, 2012 (accessed 16 March 2012).
- [2] Robert M. May, Simple mathematical models with very complicated dynamics, *Nature*, 261 (5560) (1976) 459–467.
- [3] Pierre-François Verhulst, Notice sur la loi que la population poursuit dans son accroissement, *Correspondance mathématique et physique*, 10 (1838) 113–121.
- [4] Gilbert Strang, *Linear Algebra and Its Applications*, Brooks Cole, third ed., February 1988.
- [5] E.I. Jury, *Theory and application of the z-transform method*. John Wiley, New York, USA, 1964.
- [6] E.B. Hogenauer, An economical class of digital filters for decimation and interpolation, *IEEE Trans. Acoust. Speech Signal Process.* 29 (2) (1981) 155–162.
- [7] E.A.B. da Silva, P.S.R. Diniz, S. Lima Netto, *Digital Signal Processing: System Analysis and Design*, Cambridge University Press, Cambridge, UK, second ed., 2010.
- [8] R.A. Roberts, C.T. Mullis, *Digital Signal Processing*, Number Bd. 1 in Addison-Wesley Series in Electrical Engineering, Addison-Wesley, 1987.

# Random Signals and Stochastic Processes

# 4

**Luiz Wagner Pereira Biscainho**

*DEL/Poli & PEE/COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil*

## 1.04.1 Introduction

Probability is an abstract concept useful to model chance experiments. The definition of a numerical representation for the outcomes of such experiments (the random variable) is essential to build a complete and general framework for probabilistic models. Such models can be extended to non-static outcomes in the form of time signals,<sup>1</sup> leading to the so-called stochastic process, which can evolve along continuous or discrete time. Its complete description is usually too complicated to be applied to practical situations. Fortunately, a well-accepted set of simplifying properties like stationarity and ergodicity allows the modeling of many problems in so different areas as Biology, Economics, and Communications.

There are plenty of books on random processes,<sup>2</sup> each one following some preferred order and notation, but covering essentially the same topics. This chapter is just one more attempt to present the subject in a compact manner. It is structured in the usual way: probability is first introduced, then described through random variables; within this framework, stochastic processes are presented, then associated with processing systems. Due to space limitations, proofs were avoided, and examples were kept at a minimum. No attempt was made to cover many families of probability distributions, for example. The preferred path was to define clear and unambiguously the concepts and entities associated to the subject and whenever possible give them simple and intuitive interpretations. Even risking to seem redundant, the author decided to explicitly duplicate the formulations related to random processes and sequences (i.e., continuous- and discrete-time random processes); the idea was to provide always a direct response to a consulting reader, instead of suggesting modifications in the given expressions.

Writing technical material always poses a difficult problem: which level of detail and depth will make the text useful? Our goal was making the text approachable for an undergraduate student as well as a consistent reference for more advanced students or even researchers (re)visiting random processes. The author tried to be especially careful with the notation consistency throughout the chapter in order to avoid confusion and ambiguity (which may easily occur in advanced texts). The choices of covered topics and order of presentation reflect several years of teaching the subject, and obviously match those of some preferred books. A selected list of didactic references on statistics [1,2], random variables

<sup>1</sup>Of course, other independent variables (e.g., space) can substitute for time in this context.

<sup>2</sup>Even if the literature sometimes prefers to employ “stochastic” for processes and “random” for signals, the expression “random processes” became common use and will be used throughout the chapter.

and processes [3,4], and applications [5,6] is given at the end of the chapter. We hope you enjoy the reading.

### 1.04.2 Probability

Probabilistic models are useful to describe and study phenomena that cannot be precisely predicted. To establish a precise framework for the concept of probability, one should define a chance **experiment**, of which each **trial** yields an **outcome**  $s$ . The set of all possible outcomes is the **sample space**  $S$ . In this context, one speaks of the **probability** that one trial of the experiment yields an outcome  $s$  that belongs to a desired set  $A \subset S$  (when one says the **event**  $A$  has occurred), as illustrated in Figure 4.1.

There are two different views of probability: the subjectivist and the objectivist. For subjectivists, the probability measures someone's degree of belief on the occurrence of a given event, while for objectivists it results from concrete reality. Polemics aside, objectivism can be more rewarding didactically.

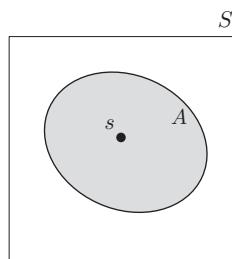
From the objectivist view point, one of the possible ways to define probability is the so-called **classical** or **a priori** approach: given an experiment whose possible outcomes  $s$  are equally likely, the probability of an event  $A$  is defined as the ratio between the number  $N(A)$  of acceptable outcomes (elements of  $A$ ) and the number  $N(S)$  of possible outcomes (elements of  $S$ ):

$$P(A) = \frac{N(A)}{N(S)}. \quad (4.1)$$

The experiment of flipping a fair coin, with  $S = \{\text{head}, \text{tail}\}$ , is an example in which the probabilities of both individual outcomes can be theoretically set:  $P(\{\text{head}\}) = P(\{\text{tail}\}) = 0.5$ .

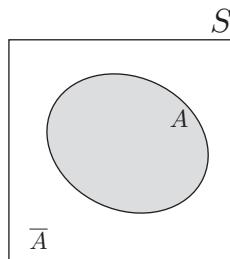
Another way to define probability is the **relative frequency** or **a posteriori** approach: the probability that the event  $A$  occurs after one trial of a given experiment can be obtained by taking the limit of the ratio between the number  $n_A$  of successes (i.e., occurrences of  $A$ ) and the number  $n$  of experiment trials, when the repeats go to infinity:

$$P(A) = \lim_{n \rightarrow \infty} \frac{n_A}{n}. \quad (4.2)$$



**FIGURE 4.1**

Sample space  $S$ , event  $A$ , and outcome  $s$ .

**FIGURE 4.2**

Event  $A$  and its complement  $\bar{A}$ .

In the ideal coin flip experiment, one is expected to find equal probabilities for head and tail. On the other hand, this pragmatic approach allows modeling the non-ideal case, provided the experiment can be repeated.

A pure **mathematical** definition can provide a sufficiently general framework to encompass every conceptual choice: the **axiomatic** approach develops a complete probability theory from three **axioms**:

1.  $P(A) \geq 0$ .
2.  $P(S) = 1$ .
3. Given  $A_m$ ,  $1 \leq m \leq M$ , such that  $A_m \cap A_{m'} = \emptyset \forall m' \neq m$ ,  $1 \leq m' \leq M$ , then  $P\left(\bigcup_{m=1}^M A_m\right) = \sum_{m=1}^M P(A_m)$ .

Referring to an experiment with sample space  $S$ :

- The events  $A_m$ ,  $1 \leq m \leq M$ , are said to be **mutually exclusive** when the occurrence of one prevents the occurrence of the others. They are the subject of the third axiom.
- The **complement**  $\bar{A}$  of a given event  $A$ , illustrated in Figure 4.2, is determined by the non-occurrence of  $A$ , i.e.,  $\bar{A} = S - A$ . From this definition,  $P(\bar{A}) = 1 - P(A)$ . Complementary events are also mutually exclusive.
- An event  $A = \emptyset$  is called **impossible**. From this definition,  $P(A) = 0$ .
- An event  $A = S$  is called **certain**. From this definition,  $P(A) = 1$ .

It should be emphasized that all events  $A$  related to a given experiment are completely determined by the sample space  $S$ , since by definition  $A \subset S$ . Therefore, a set  $B$  of outcomes not in  $S$  is mapped to an event  $B \cap S = \emptyset$ . For instance, for the experiment of rolling a fair die,  $S = \{1, 2, 3, 4, 5, 6\}$ ; the event corresponding to  $B = \{7\}$  (showing a 7) is  $B \cap S = \emptyset$ .

According to the experiment, sample spaces may be **countable** or **uncountable**.<sup>3</sup> For example, the sample space of the coin flip experiment is countable. On the other hand, the sample space of the experiment that consists in sampling with no preference any real number from the interval  $S = (0, 100]$

<sup>3</sup>One could think of mixed cases, but this discussion is better conveyed in the random variable framework, which comprises every possible mapping from the original experiment to a subset of the real numbers.

is uncountable. Given an individual outcome  $s \in S$ , defining  $A = \{s\}$ ,

$$P(A) \begin{cases} \neq 0, & \text{if } S \text{ is countable;} \\ = 0, & \text{if } S \text{ is uncountable.} \end{cases} \quad (4.3)$$

As a consequence, one should not be surprised to find an event  $A \neq \emptyset$  with  $P(A) = 0$  or an event  $A \neq S$  with  $P(A) = 1$ . In the  $(0, 100]$  interval sampling experiment,  $A = \{50\} \neq \emptyset$  has  $P(A) = 0$ ; and  $\bar{A} \neq S$  has  $P(\bar{A}) = 1$ .

### 1.04.2.1 Joint, conditional, and total probability—Bayes' rule

The **joint probability** of a set of events  $A_m$ ,  $1 \leq m \leq M$ , is the probability of their simultaneous occurrence  $\bigcap_{m=1}^M A_m$ . Referring to Figure 4.3, given two events  $A$  and  $B$ , their joint probability can be found as

$$P(A \cap B) = P(A) + P(B) - P(A \cup B). \quad (4.4)$$

By rewriting this equation as

$$P(A \cup B) = P(A) + P(B) - P(A \cap B), \quad (4.5)$$

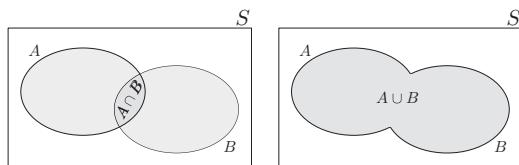
one finds an intuitive result: the term  $P(A \cap B)$  is included twice in  $P(A) + P(B)$ , and should thus be discounted. Moreover, when  $A$  and  $B$  are mutually exclusive, we arrive at the third axiom. In the die experiment, defining  $A = \{s \in S | s \text{ is even}\} = \{2, 4, 6\}$  and  $B = \{s \in S | s > 3\} = \{4, 5, 6\}$ :  $P(A) = 1/2$ ,  $P(B) = 1/2$ ,  $P(A \cap B) = P(\{4, 6\}) = 1/3$ , and  $P(A \cup B) = P(\{2, 4, 5, 6\}) = 2/3$ .

The **conditional probability**  $P(A|B)$  is the probability of event  $A$  conditioned to the occurrence of event  $B$ , and can be computed as

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0. \quad (4.6)$$

The value of  $P(A \cap B)$  accounts for the uncertainty of both  $A$  and  $B$ ; the term  $P(B)$  discounts the uncertainty of  $B$ , since it is certain in this context. In fact, the conditioning event  $B$  is the new (reduced) sample space for the experiment. By rewriting this equation as

$$P(A \cap B) = P(A|B)P(B), \quad (4.7)$$



**FIGURE 4.3**

$A \cap B$  and  $A \cup B$ .

one gets another interpretation: the joint probability of  $A$  and  $B$  combines the uncertainty of  $B$  with the uncertainty of  $A$  when  $B$  is known to occur. Using the example in the last paragraph,  $P(A|B) = [P(\{4, 6\}) \text{ in sample space } B] = 2/3$ .

Since  $P(A \cap B) = P(B \cap A) = P(B|A)P(A)$ , the **Bayes Rule** follows straightforwardly:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (4.8)$$

This formula allows computing one conditional probability  $P(A|B)$  from its reverse  $P(B|A)$ . Using again the example in the last paragraph, one would arrive at the same result for  $P(A|B)$  using  $P(B|A) = [P(\{4, 6\}) \text{ in sample space } A] = 2/3$  in the last equation.

If the sample space  $S$  is partitioned into  $M$  disjoint sets  $A_m$ ,  $1 \leq m \leq M$ , such that  $A_m \cap A_{m'} = \emptyset \forall m' \neq m$ ,  $1 \leq m' \leq M$ , then any event  $B \subset S$  can be written as  $B = \bigcup_{m=1}^M (B \cap A_m)$ . Since  $B \cap A_m$ ,  $1 \leq m \leq M$ , are disjoint sets,

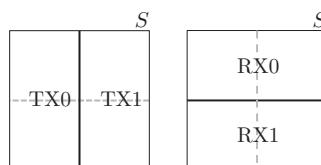
$$P(B) = \sum_{m=1}^M P(B|A_m)P(A_m), \quad (4.9)$$

which is called the **total probability** of  $B$ .

For a single event of interest  $A$ , for example, the application of Eq. (4.9) to Eq. (4.8) yields

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})}. \quad (4.10)$$

Within the Bayes context,  $P(A)$  and  $P(A|B)$  are usually known as the **a priori** and **a posteriori probabilities** of  $A$ , respectively;  $P(B|A)$  is called **likelihood** in the estimation context, and also **transition probability** in the communications context. In the latter case,  $A = \{a\}$  could refer to a symbol  $a$  sent by the transmitter and  $B = \{b\}$ , to a symbol  $b$  recognized by the receiver. Knowing  $P(B|A)$ , the probability of recognizing  $b$  when  $a$  is sent (which models the communication channel), and  $P(A)$ , the a priori probability of the transmitter to send  $a$ , allows to compute  $P(A|B)$ , the probability of  $a$  having been sent given that  $b$  has been recognized. In the case of binary communication, we could partition the sample space either in the events TX0 (0 transmitted) and TX1 (1 transmitted), or in the events RX0 (0 recognized) and RX1 (1 recognized), as illustrated in Figure 4.4; the event “error” would be  $E = (TX0 \cap RX1) \cup (TX1 \cap RX0)$ .



**FIGURE 4.4**

Binary communication example: partitions of  $S$  regarding transmission and reception.

### 1.04.2.2 Probabilistic independence

The events  $A_m$ ,  $1 \leq m \leq M$  are said to be **mutually independent** when the occurrence of one does not affect the occurrence of any combination of the others.

For two events  $A$  and  $B$ , three equivalent tests can be employed: they are independent if and only if.

1.  $P(A|B) = P(A)$ , or
2.  $P(B|A) = P(B)$ , or
3.  $P(A \cap B) = P(A)P(B)$ .

The first two conditions follow directly from the definition of independence. Using any of them in Eq. (4.6) one arrives at the third one. The reader is invited to return to Eq. (4.7) and conclude that  $B$  and  $A|B$  are independent events. Consider the experiment of rolling a special die with the numbers 1, 2, and 3 stamped in black on three faces and stamped in red on the other three; the events  $A = \{1\}$  and  $B = \{s \in S | s \text{ is a red number}\}$  are mutually independent.

Algorithmically, the best choice for testing the mutual independence of more than two events is using the third condition for every combination of  $2, 3, \dots, M$  events among  $A_1, A_2, \dots, A_M$ .

As a final observation, mutually exclusive events are not mutually independent; on the contrary, one could say they are maximally dependent, since the occurrence of one precludes the occurrence of the other.

### 1.04.2.3 Combined experiments—Bernoulli trials

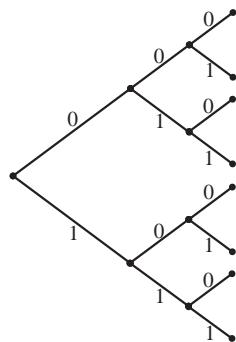
The theory we have discussed so far can be applied when more than one experiment is performed at a time. The generalization to the **multiple experiment** case can be easily done by using cartesian products.

Consider the experiments  $E_m$  with their respective sample spaces  $S_m$ ,  $1 \leq m \leq M$ . Define the combined experiment  $E$  such that each of its trials is composed by one trial of each  $E_m$ . An outcome of  $E$  is the  $M$ -tuple  $(s_1, s_2, \dots, s_M)$ , where  $s_m \in S_m$ ,  $1 \leq m \leq M$ , and the sample space of  $E$  can be written as  $S = S_1 \times S_2 \times \dots \times S_M$ . Analogously, any event of  $E$  can be expressed as  $A = A_1 \times A_2 \times \dots \times A_M$ , where  $A_m$  is a properly chosen event of  $E_m$ ,  $1 \leq m \leq M$ .

In the special case when the sub-experiments are mutually independent, i.e., the outcomes of one do not affect the outcomes of the others, we have  $P(A) = \prod_{m=1}^M P(A_m)$ . However, this is not the general case. Consider the experiment of randomly selecting a card from a 52-card deck, repeated twice: if the first card drawn is replaced, the sub-experiments are independent; if not, the first outcome affects the second experiment. For example, for  $A = \{\text{ace of spades, ace of spades}\}$ ,  $P(A) = (1/52)^2$  if the first card is replaced, and  $P(A) = 0$  if not.

At this point, an interesting counting experiment (called **Bernoulli trials**) can be defined. Take a random experiment  $E$  with sample space  $S$  and a desired event  $A$  (success) with  $P(A) = p$ , which also defines  $\bar{A}$  (failure) with  $P(\bar{A}) = 1 - p$ . What is the probability of getting exactly  $k$  successes in  $N$  independent repeats of  $E$ ? The solution can be easily found by noticing that the desired result is composed by  $k$  successes and  $N - k$  failures, which may occur in  $\binom{N}{k}$  different orders. Then,

$$P(k \text{ successes after } N \text{ trials}) = \binom{N}{k} p^k (1-p)^{N-k}. \quad (4.11)$$

**FIGURE 4.5**

Three consecutive bits sent through a binary communication system.

When  $N \rightarrow \infty$ ,  $p \rightarrow 0$ , and  $Np \rightarrow$  a constant,

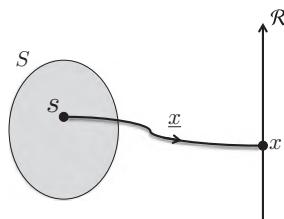
$$P(k \text{ successes after } N \text{ trials}) \rightarrow \frac{(Np)^k e^{-Np}}{k!}. \quad (4.12)$$

Returning to the card deck experiment (with replacement): the probability of selecting exactly 2 aces of spades after 300 repeats is approximately 5.09% according to Eq. (4.11), while Eq. (4.12) provides the approximate value 5.20%. In a binary communication system where 0 and 1 are randomly transmitted with equal probabilities, the probability that exactly 2 bits 1 are sent among 3 bits transmitted is 0.25%; this result can be easily checked by inspection of Figure 4.5.

### 1.04.3 Random variable

Mapping each outcome of a random experiment to a real number provides a different framework for the study of probabilistic models, amenable to simple interpretation and easy mathematical manipulation. This mapping is performed by the so-called random variable.

Given a random experiment with sample space  $S$ , a **random variable** is any function  $\underline{x}(s)$  that maps each  $s \in S$  into some  $x \in \mathcal{R}$  (see Figure 4.6). The image of this transformation with domain  $S$  and

**FIGURE 4.6**

Sample space  $S$  mapped into  $\mathcal{R}$  via random variable  $\underline{x}(s)$ .

co-domain  $\mathcal{R}$ , which results of the convenient choice of the mapping function, can be seen as the sample space of  $\underline{x}$ ; any event  $A$  of  $\underline{x}$  can be described as a subset of  $\mathcal{R}$ ; and each mapped outcome  $x$  is called a sample of  $\underline{x}$ . The following conditions should be satisfied by a random variable  $\underline{x}$ :

1.  $\{\underline{x} \leq x\}$  is an event  $\forall x \in \mathcal{R}$ ;
2.  $P(\{\underline{x} = -\infty\}) = P(\{\underline{x} = \infty\}) = 0$ .

We will see later that these conditions allow the proper definition of the cumulative probability distribution function of  $\underline{x}$ .

As seen in Section 1.04.2, sample spaces (and events) can be countable or uncountable, according to the nature of the random experiment's individual outcomes. After mapped into subsets of  $\mathcal{R}$ , countable events remain countable—e.g., one could associate with the coin flip experiment a random variable  $\underline{v}$  such that  $\underline{v}(\text{head}) = 0$  and  $\underline{v}(\text{tail}) = 1$ . On the other hand, uncountable events may or may not remain uncountable. Consider the following four distinct definitions of a random variable  $\underline{i}$  associated with the  $(0, 100]$  interval experiment described immediately before Eq. (4.3):

1.  $\underline{i}_1(s) = s$ ;
2.  $\underline{i}_2(s) = \begin{cases} -1, & s \leq 50; \\ 1, & s > 50; \end{cases}$
3.  $\underline{i}_3(s) = \begin{cases} s, & s \leq 50; \\ 100, & s > 50; \end{cases}$
4.  $\underline{i}_4(s) = \min[s, 50]$ .

The sample space of:

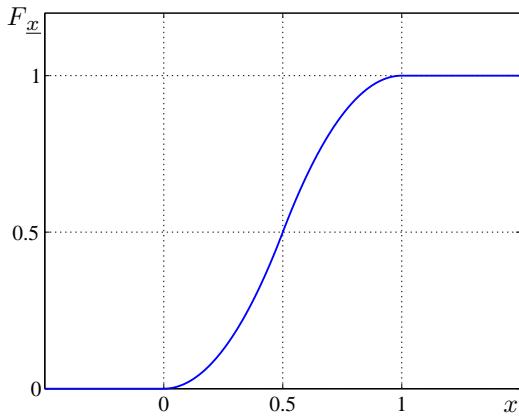
1.  $\underline{i}_1, (0, 100]$ , is uncountable;
2.  $\underline{i}_2, \{-1, 1\}$ , is countable;
3.  $\underline{i}_3, (0, 50] \cup \{100\}$ , is part uncountable, part countable;
4.  $\underline{i}_4, (0, 50]$ , even if continuous, may be difficult to classify.

The classification of sample spaces as countable or uncountable leads directly to the classification of random variables as discrete, continuous, or mixed. A **discrete** random variable  $\underline{d}$  has a countable sample space  $S_{\underline{d}}$ , and has  $P(\{\underline{d} = d\}) > 0, \forall d \in S_{\underline{d}}$ —this is the case of  $\underline{v}$  and  $\underline{i}_2$  defined above. A **continuous** random variable  $\underline{c}$  has an uncountable sample space  $S_{\underline{c}}$ , and (to avoid ambiguity) has  $P(\{\underline{c} = c\}) = 0, \forall c \in S_{\underline{c}}$ —this is the case of  $\underline{i}_1$  defined above. A **mixed** variable  $\underline{m}$  has a sample space  $S_{\underline{m}}$  composed by the union of real intervals with continuously distributed probabilities, within which  $P(\{\underline{m} = m\}) = 0$ , and discrete real values with finite probabilities  $P(\{\underline{m} = m\}) > 0$ —this is the case of  $\underline{i}_3$  and  $\underline{i}_4$  defined above. Specifically, since  $P(0 < \underline{i}_4 < 50) = P(\underline{i}_4 = 50) = 50\%$ ,  $S_{\underline{i}_4}$  should rather be treated as part uncountable, part countable than as simply uncountable.

### 1.04.3.1 Probability distributions

From the conditions that must be satisfied by any random variable  $\underline{x}$ , an overall description of its probability distribution can be provided by the so-called **cumulative probability distribution function** (shortened to **CDF**),

$$F_{\underline{x}}(x) = P(\{\underline{x} \leq x\}). \quad (4.13)$$

**FIGURE 4.7**

Example of cumulative probability distribution function.

Since  $P(\underline{x} = -\infty) = 0$ ,  $F_{\underline{x}}(-\infty) = 0$ . It is obvious that  $F_{\underline{x}}(\infty) = 1$ ; but since  $P(\{\underline{x} = \infty\}) = 0$ ,  $P(\{\underline{x} < \infty\}) = 1$ . Moreover,  $0 \leq F_{\underline{x}}(x) \leq 1$  and is a non-decreasing function of  $x$ , i.e.,  $F_{\underline{x}}(x_1) \leq F_{\underline{x}}(x_2)$  if  $x_1 < x_2$ . Also,  $F_{\underline{x}}(x^+) = F_{\underline{x}}(x)$ , i.e.,  $F_{\underline{x}}(x)$  is continuous from the right; this will be important in the treatment of discrete random variables. A typical CDF is depicted in Figure 4.7. One can use the CDF to calculate probabilities by noticing that

$$P(\{x_1 < \underline{x} \leq x_2\}) = F_{\underline{x}}(x_2) - F_{\underline{x}}(x_1). \quad (4.14)$$

For the random variable  $\underline{x}$  whose distribution is described in Figure 4.7, one can easily check by inspection that  $P(\underline{x} \leq \frac{1}{2}) = 50\%$ .

The CDF of the random variable  $i_1$  associated above with the  $(0, 100]$  interval sampling experiment is

$$F_{i_1}(i_1) = \begin{cases} 0, & i_1 \leq 0; \\ i_1, & 0 < i_1 \leq 100; \\ 1, & i_1 > 100. \end{cases} \quad (4.15)$$

The CDF of the random variable  $v$  associated above with the coin flip experiment is

$$F_v(v) = \begin{cases} 0, & v < 0; \\ 0.5, & 0 \leq v < 1; \\ 1, & v \geq 1. \end{cases} \quad (4.16)$$

Given a random variable  $\underline{x}$ , any single value  $x_0$  such that  $P(\{\underline{x} = x_0\}) = p > 0$  contributes a step<sup>4</sup> of amplitude  $p$  to  $F_{\underline{x}}(x)$ . Then, it is easy to conclude that for a discrete random variable  $\underline{d}$  with

<sup>4</sup>Unit step function:  $u(x) = \begin{cases} 0, & x < 0; \\ 1, & x \geq 0. \end{cases}$

$$S_{\underline{d}} = \{\dots, d_{i-1}, d_i, d_{i+1}, \dots\},$$

$$F_{\underline{d}}(d) = \sum_i P(\{\underline{d} = d_i\}) u(d - d_i). \quad (4.17)$$

An even more informative function that can be derived from the CDF to describe the probability distribution of a random variable  $\underline{x}$  is the so-called **probability density function** (shortened to **PDF**),

$$f_{\underline{x}}(x) = \frac{dF_{\underline{x}}(x)}{dx}. \quad (4.18)$$

Since  $F_{\underline{x}}(x)$  is a non-decreasing function of  $x$ , it follows that  $f_{\underline{x}}(x) \geq 0$ . From the definition,

$$F_{\underline{x}}(x) = \int_{-\infty}^{x^+} f_{\underline{x}}(\tilde{x}) d\tilde{x}. \quad (4.19)$$

Then,

$$\int_{-\infty}^{\infty} f_{\underline{x}}(x) dx = 1. \quad (4.20)$$

The PDF corresponding to the CDF shown in Figure 4.7 is depicted in Figure 4.8. One can use the PDF to calculate probabilities by noticing that

$$P(\{x_1 < \underline{x} \leq x_2\}) = \int_{x_1^+}^{x_2^+} f_{\underline{x}}(x) dx. \quad (4.21)$$

Again, for the random variable  $\underline{x}$  whose distribution is described in Figure 4.8, one can easily check by inspection that  $(\underline{x} \leq \frac{1}{2}) = 50\%$ .

The PDF of the random variable  $\underline{i}_1$  associated above with the  $(0, 100]$  interval sampling experiment is

$$f_{\underline{i}_1}(i_1) = \begin{cases} 1, & 0 < i_1 \leq 100; \\ 0, & \text{otherwise.} \end{cases} \quad (4.22)$$

The PDF of the random variable  $\underline{v}$  associated above with the coin flip experiment is

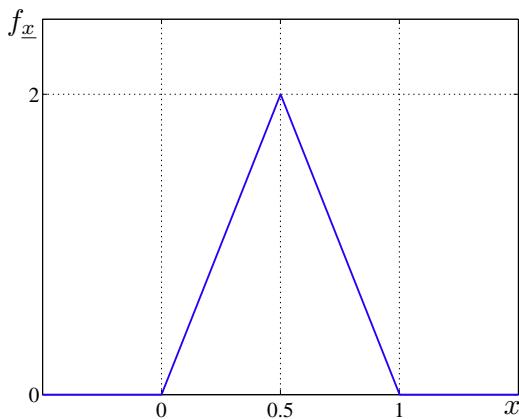
$$f_{\underline{v}}(v) = \begin{cases} \infty, & v = 0 \text{ or } v = 1; \\ 0, & \text{otherwise,} \end{cases} \quad (4.23)$$

which is not well-defined. But coherently with what has been seen for the CDF, given a random variable  $\underline{x}$ , any single value  $x_0$  such that  $P(\{\underline{x} = x_0\}) = p > 0$  contributes an impulse<sup>5</sup> of area  $p$  to  $f_{\underline{x}}(x)$ . Then, it is easy to conclude that for a discrete random variable  $\underline{d}$  with  $S_{\underline{d}} = \{\dots, d_{i-1}, d_i, d_{i+1}, \dots\}$ ,

$$f_{\underline{d}}(d) = \sum_i P(\{\underline{d} = d_i\}) \delta(d - d_i). \quad (4.24)$$

---

<sup>5</sup>Unit impulse distribution, or Dirac delta: For  $x \in \mathcal{R}$ ,  $\delta(x) = 0 \forall x \neq 0$  and  $\int_{-\infty}^{\infty} \delta(x) dx = 1$ .

**FIGURE 4.8**

Example of probability density function.

In particular, for the coin flip experiment the PDF is  $f_v(v) = 0.5\delta(v) + 0.5\delta(v - 1)$ .

In the case of discrete random variables, in order to avoid the impulses in the PDF, one can operate directly on the so-called **mass probability function**<sup>6</sup>:

$$P_d(d) = P(\{d = d\}). \quad (4.25)$$

In this case,

$$F_d(d) = \sum_{i|d_i \leq d} P_d(d_i). \quad (4.26)$$

This chapter favors an integrate framework for continuous and discrete variables, based on CDFs and PDFs.

It is usual in the literature referring (for short) to the CDF as “the distribution” and to the PDF as “the density” of the random variable. This text avoids this loose terminology, since the word “distribution” better applies to the overall probabilistic behavior of the random variable, no matter in the form of a CDF or a PDF.

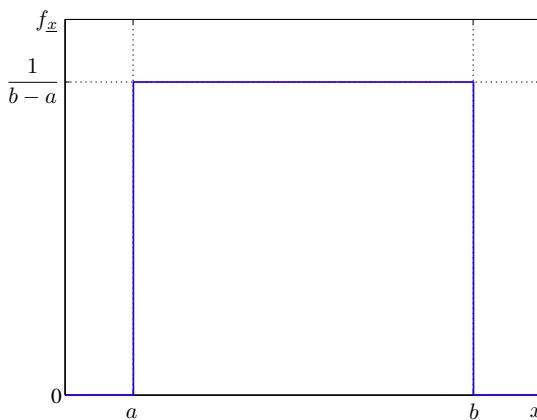
### 1.04.3.2 Usual distributions

The simplest continuous distribution is the so-called **uniform**. A random variable  $\underline{x}$  is said to be uniformly distributed between  $a$  and  $b > a$  if its PDF is

$$f_{\underline{x}}(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b; \\ 0, & \text{otherwise,} \end{cases} \quad (4.27)$$

---

<sup>6</sup>This unusual notation is employed here for the sake of compatibility with the other definitions.

**FIGURE 4.9**

Uniform probability distribution.

depicted in Figure 4.9. Notice that the inclusion or not of the interval bounds is unimportant here, since the variable is continuous. The error produced by uniform quantization of real numbers is an example of uniform random variable.

Perhaps the most recurrent continuous distribution is the so-called **Gaussian** (or **normal**). A Gaussian random variable  $\underline{x}$  is described by the PDF

$$f_{\underline{x}}(x) = \frac{1}{\sqrt{2\pi\sigma_{\underline{x}}^2}} e^{-\frac{(x-\bar{x})^2}{2\sigma_{\underline{x}}^2}}. \quad (4.28)$$

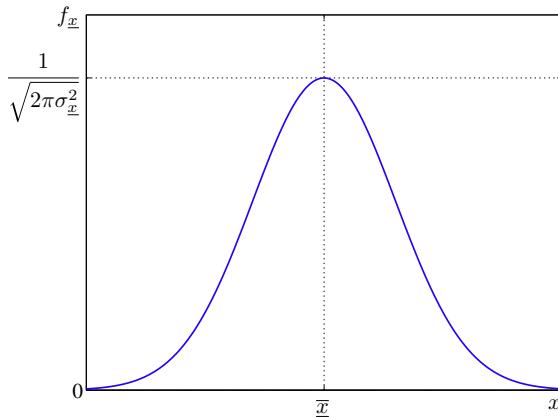
As seen in Figure 4.10, this function is symmetrical around  $\bar{x}$ , with spread controlled by  $\sigma_{\underline{x}}$ . These parameters, respectively called statistical mean and standard deviation of  $\underline{x}$ , will be precisely defined in Section 1.04.3.4. The Gaussian distribution arises from the combination of several independent random phenomena, and is often associated with noise models.

There is no closed expression for the Gaussian CDF, which is usually tabulated for a normalized Gaussian random variable  $\tilde{x}$  with  $\bar{x} = 0$  and  $\sigma_{\tilde{x}}^2 = 1$  such that

$$f_{\tilde{x}}(\tilde{x}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\tilde{x}^2}{2}}. \quad (4.29)$$

In order to compute  $p = P(\{x_1 < \underline{x} \leq x_2\})$  for a Gaussian random variable  $\underline{x}$ , one can build an auxiliary variable  $\tilde{x} = \frac{\underline{x}-\bar{x}}{\sigma_{\underline{x}}}$  such that  $p = P(\{\tilde{x}_1 < \tilde{x} \leq \tilde{x}_2\}) = F_{\tilde{x}}(\tilde{x}_2) - F_{\tilde{x}}(\tilde{x}_1)$ , which can then be approximated by tabulated values of  $F_{\tilde{x}}(\tilde{x})$ .

**Example 1.** The values  $r$  of a 10%-resistor series produced by a component industry can be modeled by a Gaussian random variable  $\underline{r}$  with  $\sigma_{\underline{r}} = \frac{1}{30}\bar{r}$ . What is the probability of producing a resistor within  $\pm 1\%$  of  $\bar{r}$ ?

**FIGURE 4.10**

Gaussian probability distribution.

**Solution 1.** The normalized counterpart of  $r$  is  $\tilde{r} = \frac{r - \bar{r}}{\frac{1}{30}\bar{r}}$ . Then,

$$P(\{0.99\bar{r} \leq r \leq 1.01\bar{r}\}) = P(\{-0.3 \leq \tilde{r} \leq 0.3\}) = F_{\tilde{r}}(0.3) - F_{\tilde{r}}(-0.3) \approx 0.2358. \quad (4.30)$$

Notice once more that since the variable is continuous, the inclusion or not of the interval bounds has no influence on the result.

In Section 1.04.2.3, an important discrete random variable was implicitly defined. A random variable  $\underline{x}$  that follows the so-called **binomial** distribution is described by

$$f_{\underline{x}}(x) = \sum_{k=0}^N \binom{N}{k} p^k (1-p)^{N-k} \delta(x - k), \quad (4.31)$$

and counts the number of occurrences of an event  $A$  which has probability  $p$ , after  $N$  independent repeats of a random experiment. Games of chance are related to binomial random variables.

The so-called **Poisson** distribution, described by

$$f_{\underline{x}}(x) = e^{-\lambda T} \sum_{k=0}^{\infty} \frac{(\lambda T)^k}{k!} \delta(x - k), \quad (4.32)$$

counts the number of occurrences of an event  $A$  that follows a mean rate of  $\lambda$  occurrences per unit time, during the time interval  $T$ . Traffic studies are related to Poisson random variables.

It is simple to derive the Poisson distribution from the binomial distribution. If an event  $A$  may occur with no preference anytime during a time interval  $(t_0, t_0 + T_0)$ , the probability that  $A$  occurs within the time interval  $[t, t + T] \subset (t_0, t_0 + T_0)$  is  $\frac{T}{T_0}$ . If  $A$  occurs  $N$  times in  $(t_0, t_0 + T_0)$ , the probability that

it falls exactly  $k$  times in  $[t, t + T] \subset (t_0, t_0 + T_0)$  is  $\binom{N}{k} p^k (1-p)^{N-k}$ . If  $T_0 \rightarrow \infty$ ,  $p \rightarrow 0$ ; if  $A$

follows a mean rate of  $\lambda = \frac{N}{T_0}$  occurrences per unit time, then  $N \rightarrow \infty$  and the probability of exact  $k$  occurrences in a time interval of duration  $T$  becomes  $e^{-\lambda T} \sum_{k=0}^{\infty} \frac{(\lambda T)^k}{k!}$ , where  $\lambda T$  substituted for  $Np$ . If a central office receives 100 telephone calls per minute in the mean, the probability that more than 1 call arrive during 1s is  $1 - e^{-\frac{100}{60}} (1 - e^{-\frac{100}{60}}) \approx 49.6\%$ .

Due to space restrictions, this chapter does not detail other random distributions, which can be easily found in the literature.

### 1.04.3.3 Conditional distribution

In many instances, one is interested in studying the behavior of a given random variable  $\underline{x}$  under some constraints. A **conditional distribution** can be built to this effect, if the event  $B \subset \mathcal{R}$  summarizes those constraints. The corresponding CDF of  $\underline{x}$  conditioned to  $B$  can be computed as

$$F_{\underline{x}}(x|B) = P(\{\underline{x} \leq x | B\}) = \frac{P(\{\underline{x} \leq x\} \cap B)}{P(B)}. \quad (4.33)$$

The related PDF is simply

$$f_{\underline{x}}(x|B) = \frac{dF_{\underline{x}}(x|B)}{dx}. \quad (4.34)$$

Conditional probabilities can be straightforwardly computed from conditional CDFs or PDFs.

When the conditioning event is an interval  $B = (a, b]$ , it can be easily deduced that

$$F_{\underline{x}}(x|B) = \begin{cases} 0, & x \leq a; \\ \frac{F_{\underline{x}}(x) - F_{\underline{x}}(a)}{F_{\underline{x}}(b) - F_{\underline{x}}(a)}, & a < x \leq b; \\ 1, & x > b; \end{cases} \quad (4.35)$$

and

$$f_{\underline{x}}(x|B) = \begin{cases} \frac{f_{\underline{x}}(x)}{\int_a^b f_{\underline{x}}(x) dx}, & a < x \leq b; \\ 0, & \text{otherwise.} \end{cases} \quad (4.36)$$

Both sentences in Eq. (4.36) can be easily interpreted:

- Within  $(a, b]$ , the conditional PDF has the same shape of the original one; the normalization factor  $\int_a^b f_{\underline{x}}(x) dx$  ensures  $P((a, b]) = 100\%$  in the restricted sample space  $(a, b]$ .
- By definition, there is null probability of getting  $x$  outside  $(a, b]$ .

As an example, the random variable  $H$  defined by the non-negative outcomes of a normalized Gaussian variable follows the PDF

$$f_{\underline{h}}(h) = \sqrt{\frac{2}{\pi}} e^{-\frac{h^2}{2}} u(h). \quad (4.37)$$

The results shown in this section can be promptly generalized to any conditioning event.

### 1.04.3.4 Statistical moments

The concept of mean does not need any special introduction: the single value that substituted for each member in a set of numbers produces the same total. In the context of probability, following a frequentist path, one could define the arithmetic mean of infinite samples of a random variable  $\underline{x}$  as its **statistical mean**  $\bar{x}$  or its **expected value**  $E[\underline{x}]$ .

Recall the random variable  $\underline{v}$  associated with the fair coin flip experiment, with  $P_{\underline{v}}(0) = P_{\underline{v}}(1) = 0.5$ . After infinite repeats of the experiment, one gets 50% of heads ( $\underline{v} = 0$ ) and 50% of tails ( $\underline{v} = 1$ ); then, the mean outcome will be<sup>7</sup>  $E[\underline{v}] = 0.5$ . If another variable  $\underline{v}'$  is associated with an unfair coin with probabilities  $P_{\underline{v}'}(0) = 0.4$  and  $P_{\underline{v}'}(1) = 0.6$ , the same reasoning leads to  $E[\underline{v}'] = 0.6$ . Instead of averaging infinite outcomes, just summing the possible values of the random variable weighted by their respective probabilities also yields its statistical mean. Thus we can state that for any discrete random variable  $\underline{d}$  with sample space  $S = \{\dots, d_{i-1}, d_i, d_{i+1}, \dots\}$ ,

$$E[\underline{d}] = \sum_i d_i P_{\underline{d}}(d_i). \quad (4.38)$$

This result can be generalized. Given a continuous random variable  $\underline{x}$ , the probability of drawing a value in the interval  $dx$  around  $x_0$  is given by  $f_{\underline{x}}(x_0)dx$ . The weighted sum of every  $x \in \mathcal{R}$  is simply

$$E[\underline{x}] = \int_{-\infty}^{\infty} x f_{\underline{x}}(x) dx. \quad (4.39)$$

By substituting the PDF of a discrete variable  $\underline{d}$  (see Eq. (4.24)) into this expression, one arrives at Eq. (4.38). Then, Eq. (4.39) is the analytic expression for the expected value of any random variable  $\underline{x}$ .

Suppose another random variable  $g(\underline{x})$  is built as a function of  $\underline{x}$ . Since the probability of getting the value  $x_0$  is the same as getting the respective  $g(x_0)$ , we can deduce that

$$E[g(\underline{x})] = \int_{-\infty}^{\infty} g(x) f_{\underline{x}}(x) dx. \quad (4.40)$$

A complete family of measures based on expected values can be associated with a random variable. The so-called *n*th-order **moment** of  $\underline{x}$  (about the origin) is defined as

$$m_n = E[\underline{x}^n], \quad n \in \mathbb{Z}. \quad (4.41)$$

The first two moments of  $\underline{x}$  are:

- $m_1 = \bar{x}$ , i.e., the **mean** of  $\underline{x}$ , given by the centroid of  $f_{\underline{x}}(x)$ .
- $m_2 = \underline{x}^2$ , i.e., the **mean square value** of  $\underline{x}$ .

A modified family of parameters can be formed by computing the moments about the mean. The so-called *n*th-order **central moment** of  $\underline{x}$  is defined as

$$\mu_n = E[(\underline{x} - \bar{x})^n], \quad n \in \mathbb{Z}. \quad (4.42)$$

Subtracting the statistical mean from a random variable can be interpreted as disregarding its “deterministic” part (represented by its statistical mean) Three special cases:

---

<sup>7</sup>Notice that the expected value of a random variable is not necessarily meaningful for the associated experiment.

- $\mu_2 = \sigma_{\underline{x}}^2$ , known as the **variance** of  $\underline{x}$ , which measures the spread of  $f_{\underline{x}}(x)$  around  $\bar{x}$ . The so-called **standard deviation**  $\sigma_{\underline{x}} = \sqrt{\mu_2}$  is a convenient measure with the same dimension of  $\underline{x}$ .
- $\mu_3$ , whose standardized version  $\frac{\mu_3}{\sigma_{\underline{x}}^3}$  is the so-called **skewness** of  $\underline{x}$ , which measures the asymmetry of  $f_{\underline{x}}(x)$ .
- $\mu_4$ , whose standardized version<sup>8</sup> minus three  $\frac{\mu_4}{\sigma_{\underline{x}}^4} - 3$  is the so-called **kurtosis** of  $\underline{x}$ , which measures the peakedness of  $f_{\underline{x}}(x)$ . One can say the distributions are measured against the Gaussian, which has a null kurtosis.

Of course, analogous definitions apply to the moments computed over conditional distributions.

<b>Table 4.1</b> Statistical Distribution of $\underline{d}$	
$d$	$P_d$
-1	0.2
0	0.5
1	0.3

A useful expression, which will be recalled in the context of random processes, relates three of the measures defined above:

$$\sigma_{\underline{x}}^2 = \bar{x}^2 - \underline{x}^2. \quad (4.43)$$

Rewritten as

$$\bar{x}^2 = \underline{x}^2 + \sigma_{\underline{x}}^2, \quad (4.44)$$

it allows to split the overall “intensity” of  $\underline{x}$ , measured by its mean square value, into a deterministic part, represented by  $\bar{x}^2$ , and a random part, represented by  $\sigma_{\underline{x}}^2$ .

As an example, consider a discrete random variable  $\underline{d}$  distributed as shown in Table 4.1. Their respective parameters are:

- mean  $\bar{d} = 0.1$ , indicating the PDF of  $\underline{d}$  is shifted to the right of the origin;
- mean square value  $\bar{d}^2 = 1$ , which measures the intensity of  $\underline{d}$ ;
- variance  $\sigma_{\underline{d}}^2 = 0.49$ , which measures the random part of the intensity of  $\underline{d}$ ;
- standard deviation  $\sigma_{\underline{d}} = 0.7$ , which measures the spread of  $\underline{d}$  around its mean;
- skewness  $\frac{\mu_3}{\sigma_{\underline{d}}^3} \approx -0.14$ , indicating the PDF of  $\underline{d}$  is left-tailed;
- kurtosis  $\frac{\mu_4}{\sigma_{\underline{d}}^4} - 3 \approx -0.96$ , indicating the PDF of  $\underline{d}$  is less peaky than the PDF of a Gaussian variable.

At this point, we can discuss two simple transformations of a random variable  $\underline{x}$  whose effects can be summarized by low-order moments:

- A random variable  $y = \underline{x} + x_0$  can be formed by adding a fixed offset  $x_0$  to each sample of  $\underline{x}$ . As a consequence of this operation, the new PDF is a shifted version of the original one:  $f_y(y) = f_{\underline{x}}(y - x_0)$ , thus adding  $x_0$  to the mean of  $\underline{x}$ :  $\bar{y} = \bar{x} + x_0$ .

---

<sup>8</sup>The classical definition does not subtract three.

- A random variable  $\underline{y} = \alpha \underline{x}$  can be formed by scaling by  $\alpha$  each sample of  $\underline{x}$ . As a consequence of this operation, the new PDF is a scaled version of the original one:  $f_{\underline{y}}(y) = \frac{1}{\alpha} f_{\underline{x}}\left(\frac{y}{\alpha}\right)$ , thus scaling by  $\alpha$  the standard deviation of  $\underline{x}$ :  $\sigma_{\underline{y}} = \alpha \sigma_{\underline{x}}$ .

Such transformations do not change the shape (and therefore the type) of the original distribution. In particular, one can generate:

- a zero-mean version of  $\underline{x}$  by making  $\underline{y} = \underline{x} - \bar{\underline{x}}$ , which disregards the deterministic part of  $\underline{x}$ ;
- a unit-standard deviation version of  $\underline{x}$  by making  $\underline{y} = \frac{\underline{x}}{\sigma_{\underline{x}}}$ , which enforces a standard statistical variability to  $\underline{x}$ ;
- a normalized version of  $\underline{x}$  by making  $\underline{y} = \frac{\underline{x} - \bar{\underline{x}}}{\sigma_{\underline{x}}}$ , which combines both effects.

**Table 4.2** Statistical Distribution of  $\underline{d}$

$\tilde{d}$	$P_{\tilde{d}}$
-11/7	0.2
-1/7	0.5
9/7	0.3

We already defined a normalized Gaussian distribution in Section 1.04.3.2. The normalized version  $\underline{d}$  of the random variable  $\underline{d}$  of the last example would be distributed as shown in Table 4.2.

The main importance of these expectation-based parameters is to provide a partial description of an underlying distribution without the need to resource to the PDF. In a practical situation in which only a few samples of a random variable are available, as opposed to its PDF and related moments, it is easier to get more reliable estimates for the latter (especially low-order ones) than for the PDF itself. The same rationale applies to the use of certain auxiliary inequalities that avoid the direct computation of probabilities on a random variable (which otherwise would require the knowledge or estimation of its PDF) by providing upper bounds for them based on low-order moments. Two such inequalities are:

- Markov's inequality for a non-negative random variable  $\underline{x}$ :  $P(\{\underline{x} \geq a\}) \leq \frac{\bar{\underline{x}}}{a}$ ,  $\forall a > 0$ .
- Chebyshev's inequality for a random variable  $\underline{x}$ :  $P(\{|\underline{x} - \bar{\underline{x}}| \geq a\}) \leq \frac{\sigma_{\underline{x}}^2}{a^2}$ ,  $\forall a > 0$ .

The derivation of Markov's inequality is quite simple:

$$\begin{aligned} P(\underline{x} \geq a) &= \int_a^{\infty} f_{\underline{x}}(x) dx = \int_{-\infty}^{\infty} f_{\underline{x}}(x) u(x-a) dx \\ &\leq \int_{-\infty}^{\infty} \frac{x}{a} f_{\underline{x}}(x) u(x-a) dx \leq \int_{-\infty}^{\infty} \frac{x}{a} f_{\underline{x}}(x) dx = \frac{E[\underline{x}]}{a}. \end{aligned} \quad (4.45)$$

Chebyshev's inequality follows directly by substituting  $(x - \bar{x})^2 \geq a^2$  for  $x \geq a$  in Markov's inequality. As an example, the probability of getting a sample  $g > 3$  from the normalized Gaussian random variable  $\underline{g}$  is  $P(\{\underline{g} > 3\}) \approx 0.0013$ ; Chebyshev's inequality predicts  $P(\{\underline{g} > 3\}) \leq \frac{1}{9}$ , an upper bound almost 100 times greater than the actual probability.

Two representations of the distribution of a random variable in alternative domains provide interesting links with its statistical moments. The so-called **characteristic function** of a random variable  $\underline{x}$  is defined as

$$\Phi_{\underline{x}}(\omega) = E[e^{j\omega\underline{x}}], \quad \omega \in \mathcal{R}, \quad (4.46)$$

and inter-relates the moments  $m_n$  by successive differentiations:

$$m_n = (-j)^n \left. \frac{d^n \Phi_{\underline{x}}(\omega)}{d\omega^n} \right|_{\omega=0}. \quad (4.47)$$

The so-called **moment generating function** of a random variable  $\underline{x}$  is defined as

$$M_{\underline{x}}(v) = E[e^{v\underline{x}}], \quad v \in \mathcal{R}, \quad (4.48)$$

and provides a more direct way to compute  $m_n$ :

$$m_n = \left. \frac{d^n M_{\underline{x}}(v)}{dv^n} \right|_{v=0}. \quad (4.49)$$

### 1.04.3.5 Transformation of random variables

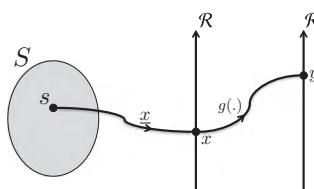
Consider the problem of describing a random variable  $\underline{y} = g(\underline{x})$  obtained by transformation of another random variable  $\underline{x}$ , illustrated in Figure 4.11. By definition,

$$F_{\underline{y}}(y) = P(\{\underline{y} \leq y\}) = P(\{g(\underline{x}) \leq y\}). \quad (4.50)$$

If  $\underline{x}$  is a continuous random variable and  $g(\cdot)$  is a differentiable function, the sentence  $g(x) \leq y$  is equivalent to a set of sentences in the form  $h_1(y) < x \leq h_2(y)$ . Since  $P(\{h_1(y) < x \leq h_2(y)\}) = F_{\underline{x}}(h_2(y)) - F_{\underline{x}}(h_1(y))$ , then  $F_{\underline{y}}(y)$  can be expressed as a function of  $F_{\underline{x}}(x)$ .

Fortunately, an intuitive formula expresses  $f_{\underline{y}}(y)$  as a function of  $f_{\underline{x}}(x)$ :

$$f_{\underline{y}}(y) = \sum_n \frac{f_{\underline{x}}(x_n)}{\left| \frac{dg(x)}{dx} \right|_{x=x_n}}, \quad \text{with } y = g(x_n). \quad (4.51)$$



**FIGURE 4.11**

Random variable  $\underline{x}$  mapped into random variable  $\underline{y}$ .

Given that the transformation  $g(\cdot)$  is not necessarily monotonic,  $x_n$  are all possible values mapped to  $y$ ; therefore, their contributions must be summed up. It is reasonable that  $f_y(y)$  must be directly proportional to  $f_x(x)$ : the more frequent a given  $x_0$ , the more frequent its respective  $y_0 = g(x_0)$ . By its turn, the term  $\left| \frac{dg(x)}{dx} \right|$  accounts for the distortion imposed to  $f_x(x)$  by  $g(x)$ . For example, if this transformation is almost constant in a given region, no matter if increasing or decreasing, then  $\left| \frac{dg(x)}{dx} \right|$  in the denominator will be close to zero; this just reflects the fact that a wide range of  $x$  values will be mapped into a narrow range of values of  $y$ , which will then become denser than  $x$  in that region.

As an example, consider that a continuous random variable  $x$  is transformed into a new variable  $y = x^2$ . For the CDF of  $y$ ,

$$\underline{y} \leq y \Rightarrow \underline{x}^2 \leq y \Rightarrow -\sqrt{y} \leq \underline{x} \leq \sqrt{y} \Rightarrow F_y(y) = F_x(\sqrt{y}) - F_x(-\sqrt{y}). \quad (4.52)$$

By Eq. (4.51), or by differentiation of Eq. (4.52), one arrives at the PDF of  $y$ :

$$f_y(y) = \frac{f_x(\sqrt{y}) + f_x(-\sqrt{y})}{\sqrt{y}}. \quad (4.53)$$

The case when real intervals of  $x$  are mapped into single values of  $y$  requires an additional care to treat the nonzero individual probabilities of the resulting values of the new variable. However, the case of a discrete random variable  $x$  with  $S = \{\dots, x_{i-1}, x_i, x_{i+1}, \dots\}$  being transformed is trivial:

$$f_y(y) = \sum_i P_y(y_i) \delta(y - y_i), \text{ with } y_i = g(x_{i_n}) \text{ and } P_y(y_i) = \sum_n P_x(x_{i_n}). \quad (4.54)$$

Again,  $x_{i_n}$  are all possible values mapped to  $y_i$ .

An interesting application of transformations of random variables is to obtain samples of a given random variable  $y$  from samples of another random variable  $x$ , both with known distributions. Assume that exists  $y = g(x)$  such that  $F_y(y = g(x)) = F_x(x)$ . Then,  $y = F_y^{-1}F_x(x)$ , which requires only the invertibility of  $F_y(y)$ .

### 1.04.3.6 Multiple random variable distributions

A single random experiment  $E$  (composed or not) may give rise to a set of  $N$  random variables, if each individual outcome  $s \in S$  is mapped into several real values  $x_1, x_2, \dots, x_N$ . Consider, for example, the random experiment of sampling the climate conditions at every point on the globe; daily mean temperature  $t$  and relative air humidity  $h$  can be considered as two random variables that serve as numerical summarizing measures. One must generalize the probability distribution descriptions to cope with this multiple random variable situation: it should be clear that the set of  $N$  individual probability distribution descriptions, one for each distinct random variable, provides less information than their joint probability distribution description, since the former cannot convey information about their mutual influences.

We start by defining a **multiple or vector random variable** as the function  $\underline{\mathbf{x}}$  that maps  $s \in S$  into  $\underline{\mathbf{x}} \in \mathcal{R}^N$ , such that

$$\underline{\mathbf{x}} = \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \vdots \\ \underline{x}_N \end{bmatrix} \quad (4.55)$$

and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}. \quad (4.56)$$

Notice that  $x_1, x_2, \dots, x_N$  are jointly sampled from  $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N$ .

The **joint cumulative probability distribution function** of  $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N$ , or simply the CDF of  $\underline{\mathbf{x}}$ , can be defined as

$$F_{\underline{\mathbf{x}}}(\mathbf{x}) = F_{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N}(x_1, x_2, \dots, x_N) = P(\{\underline{x}_1 \leq x_1\} \cap \{\underline{x}_2 \leq x_2\} \cap \dots \cap \{\underline{x}_N \leq x_N\}). \quad (4.57)$$

The following relevant properties of the joint CDF can be easily deduced:

- $F_{\underline{\mathbf{x}}}(\dots, x_{n-1}, -\infty, x_{n+1}, \dots) = 0$ , since  $P(\{x_n \leq -\infty\}) = 0$  for any  $1 \leq n \leq N$ .
- $F_{\underline{\mathbf{x}}}(\infty, \infty, \dots, \infty) = 1$ , since this condition encompasses the complete  $S$ .
- $0 \leq F_{\underline{\mathbf{x}}}(\mathbf{x}) \leq 1$  and is a nondecreasing function of  $x_n$ ,  $1 \leq n \leq N$ , by construction.

Define  $\underline{\mathbf{x}}_i$  containing  $N_i$  variables of interest among the random variables in  $\underline{\mathbf{x}}$ , and leave the remaining  $N - N_i$  **nuisance** variables in  $\underline{\mathbf{x}}_n$ . The so-called **marginal CDF** of  $\underline{x}_{i_1}, \underline{x}_{i_2}, \dots, \underline{x}_{i_{N_i}}$  separately describes these variables' distribution from the knowledge of the CDF of  $\underline{\mathbf{x}}$ :

$$F_{\underline{\mathbf{x}}_i}(\mathbf{x}_i) = F_{\underline{\mathbf{x}}}(\mathbf{x})|_{x_{n_1}=x_{n_2}=\dots=x_{n_{N-N_i}}=\infty}. \quad (4.58)$$

One says the variables  $x_{n_1}, x_{n_2}, \dots, x_{n_{N-N_i}}$  have been **marginalized** out: the condition  $x_{n_m} \leq \infty$  simply means they must be in the sample space, thus one does not need to care about them anymore.

The CDF of a discrete random variable  $\underline{\mathbf{d}}$  with  $S_{\underline{\mathbf{d}}} = \{\dots, \mathbf{d}_{i-1}, \mathbf{d}_i, \mathbf{d}_{i+1}, \dots\}$ , analogously to the single variable case, is composed by steps at the admissible  $N$ -tuples:

$$F_{\underline{\mathbf{d}}}(\mathbf{d}) = \sum_i P_{\underline{\mathbf{d}}}(\mathbf{d}_i) u(d_1 - d_{1i}) u(d_2 - d_{2i}) \dots u(d_N - d_{Ni}). \quad (4.59)$$

The **joint probability density function** of  $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N$ , or simply the PDF of  $\underline{\mathbf{x}}$ , can be defined as

$$f_{\underline{\mathbf{x}}}(\mathbf{x}) = f_{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N}(x_1, x_2, \dots, x_N) = \frac{\partial^N F_{\underline{\mathbf{x}}}(\mathbf{x})}{\partial x_1 \partial x_2 \dots \partial x_N}. \quad (4.60)$$

Since  $F_{\underline{\mathbf{x}}}(\mathbf{x})$  is a non-decreasing function of  $x_n$ ,  $1 \leq n \leq N$ , it follows that  $f_{\underline{\mathbf{x}}}(\mathbf{x}) \geq 0$ . From the definition,

$$F_{\underline{\mathbf{x}}}(\mathbf{x}) = \int_{-\infty}^{x_N} \dots \int_{-\infty}^{x_2} \int_{-\infty}^{x_1} f_{\underline{\mathbf{x}}}(\tilde{\mathbf{x}}) d\tilde{x}_1 d\tilde{x}_2 \dots d\tilde{x}_N. \quad (4.61)$$

Then,

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{\underline{x}}(\mathbf{x}) dx_1 dx_2 \cdots dx_N = 1. \quad (4.62)$$

The probability of any event  $B \in \mathcal{R}^N$  can be computed as

$$P(B) = \int_{\mathbf{x} \in B} f_{\underline{x}}(\mathbf{x}) dx_1 dx_2 \cdots dx_N. \quad (4.63)$$

Once more we can marginalize the nuisance variables  $x_{n_1}, x_{n_2}, \dots, x_{n_{N-N_i}}$  to obtain the **marginal PDF** of  $\underline{x}_i$  from the PDF of  $\underline{x}$ :

$$f_{\underline{x}_i}(\mathbf{x}_i) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{\underline{x}}(\mathbf{x}) dx_{n_1} dx_{n_2} \cdots dx_{n_{N-N_i}}. \quad (4.64)$$

Notice that if  $\underline{x}_i$  and  $\underline{x}_n$  are statistically dependent, the marginalization of  $\underline{x}_n$  does not “eliminate” the effect of  $\underline{x}_n$  on  $\underline{x}_i$ : the integration is performed over  $\underline{x}$ , which describes their mutual influences. For a discrete random variable  $\underline{d}$  with  $S_{\underline{d}} = \{\dots, \mathbf{d}_{i-1}, \mathbf{d}_i, \mathbf{d}_{i+1}, \dots\}$  consists of impulses at the admissible  $N$ -tuples:

$$f_{\underline{d}}(\mathbf{d}) = \sum_i P_{\underline{d}}(\mathbf{d}_i) \delta(d_1 - d_{1i}) \delta(d_2 - d_{2i}) \cdots \delta(d_N - d_{Ni}). \quad (4.65)$$

Suppose, for example, that we want to find the joint and marginal CDFs and PDFs of two random variables  $\underline{x}$  and  $\underline{y}$  jointly uniformly distributed in the region  $0 \leq y \leq x \leq 1$ , i.e., such that

$$f_{\underline{x}, \underline{y}}(x, y) = \begin{cases} 2, & 0 \leq y \leq x \leq 1; \\ 0, & \text{otherwise.} \end{cases} \quad (4.66)$$

The marginalization of  $\underline{y}$  yields

$$f_{\underline{x}}(x) = \int_{-\infty}^{\infty} f_{\underline{x}, \underline{y}}(x, y) dy = \int_0^x 2 dy = \begin{cases} 2x, & 0 \leq x \leq 1; \\ 0, & \text{otherwise.} \end{cases} \quad (4.67)$$

The marginalization of  $\underline{x}$  yields

$$f_{\underline{y}}(y) = \int_{-\infty}^{\infty} f_{\underline{x}, \underline{y}}(x, y) dx = \int_y^1 2 dx = \begin{cases} 2(1-y), & 0 \leq y \leq 1; \\ 0, & \text{otherwise.} \end{cases} \quad (4.68)$$

By definition,

$$F_{\underline{x}, \underline{y}}(x, y) = \int_{-\infty}^y \int_{-\infty}^x f_{\underline{x}, \underline{y}}(\tilde{x}, \tilde{y}) d\tilde{x} d\tilde{y} = \begin{cases} 0, & x < 0 \text{ or } y < 0; \\ y(2x - y), & 0 \leq y \leq x \leq 1; \\ x^2, & 0 \leq x \leq 1 \text{ and } y > x; \\ y(2 - y), & 0 \leq y \leq 1 \text{ and } x > 1; \\ 1, & x > 1 \text{ and } y > 1. \end{cases} \quad (4.69)$$

The reader is invited to sketch the admissible region  $0 \leq y \leq x \leq 1$  on the  $(x, y)$  plane and try to solve Eq. (4.69) by visual inspection. The marginal CDF of  $\underline{x}$  is given by

$$F_{\underline{x}}(x) = F_{\underline{x}, \underline{y}}(x, \infty) = \begin{cases} 0, & x < 0; \\ x^2, & 0 \leq x \leq 1; \\ 1, & x > 1, \end{cases} \quad (4.70)$$

which could also have been obtained by direct integration of  $f_{\underline{x}}(x)$ . The marginal CDF of  $\underline{y}$  is given by

$$F_{\underline{y}}(y) = F_{\underline{x}, \underline{y}}(\infty, y) = \begin{cases} 0, & y < 0; \\ y(2-y), & 0 \leq y \leq 1; \\ 1, & y > 1, \end{cases} \quad (4.71)$$

which could also have been obtained by direct integration of  $f_{\underline{y}}(y)$ .

Similarly to the univariate case discussed in Section 1.04.3.3, the **conditional distribution** of a vector random variable  $\underline{x}$  restricted by a conditioning event  $B \in \mathcal{R}^N$  can be described by its respective CDF

$$F_{\underline{x}}(\underline{x}|B) = \frac{P(\{\underline{x}_1 \leq x_1\} \cap \{\underline{x}_2 \leq x_2\} \cap \dots \cap \{\underline{x}_N \leq x_N\} \cap B)}{P(B)} \quad (4.72)$$

and PDF

$$f_{\underline{x}}(\underline{x}|B) = \frac{\partial^N F_{\underline{x}}(\underline{x}|B)}{\partial x_1 \partial x_2 \dots \partial x_N}. \quad (4.73)$$

A special case arises when  $B$  imposes a point conditioning: Define  $\underline{x}_B$  containing  $N_B$  variables among those in  $\underline{x}$ , such that  $B = \{\underline{x}_B = \underline{x}_B\}$ . It can be shown that

$$f_{\underline{x}}(\underline{x}|B) = \frac{f_{\underline{x}}(\underline{x})}{f_{\underline{x}_B}(\underline{x}_B)}, \quad (4.74)$$

an intuitive result in light of Eq. (4.6)—to which one may directly refer in the case of discrete random variables. Returning to the last example, the point-conditioned PDFs can be shown to be

$$f_{\underline{x}}(x|\underline{y} = y) = \begin{cases} \frac{1}{1-y}, & 0 \leq y \leq x \leq 1; \\ 0, & \text{otherwise}; \end{cases} \quad (4.75)$$

and

$$f_{\underline{y}}(y|\underline{x} = x) = \begin{cases} \frac{1}{x}, & 0 \leq y \leq x \leq 1; \\ 0, & \text{otherwise}. \end{cases} \quad (4.76)$$

### 1.04.3.7 Statistically independent random variables

Based on the concept of statistical independence, discussed in Section 1.04.2.2, the mutual **statistical independence** of a set of random variables means that the probabilistic behavior of each one is not affected by the probabilistic behavior of the others. Formally, the random variables  $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N$  are independent if any of the following conditions is fulfilled:

- $F_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) = \prod_{n=1}^N F_{x_n}(x_n)$ , or
- $f_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) = \prod_{n=1}^N f_{x_n}(x_n)$ , or
- $F_{x_n}(x_n | \text{any condition on the remaining variables}) = F_{x_n}(x_n)$ ,  $\forall 1 \leq n \leq N$ , or
- $f_{x_n}(x_n | \text{any condition on the remaining variables}) = f_{x_n}(x_n)$ ,  $\forall 1 \leq n \leq N$ .

Returning to the example developed in Section 1.04.3.6,  $x$  and  $y$  are clearly statistically dependent. However, if they were jointly uniformly distributed in the region defined by  $0 \leq x \leq 1$ ,  $0 \leq y \leq 1$ , they would be statistically independent—this verification is left to the reader.

The PDF of a random variable  $\underline{y} = \sum_{n=1}^N f_{x_n}(x_n)$  composed by the sum of  $N$  independent variables  $x_n$  can be computed as<sup>9</sup>

$$f_{\underline{y}}(y) = (f_{x_1} * f_{x_2} * \dots * f_{x_N})(y). \quad (4.77)$$

### 1.04.3.8 Joint statistical moments

The definition of **statistical mean** can be directly extended to a vector random variable  $\underline{\mathbf{x}}$  with known PDF. The **expected value** of a real scalar function  $g(\underline{\mathbf{x}})$  is given by:

$$E[g(\underline{\mathbf{x}})] = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} g(\underline{\mathbf{x}}) f_{\underline{\mathbf{x}}}(\underline{\mathbf{x}}) dx_1 dx_2 \dots dx_N. \quad (4.78)$$

We can analogously proceed to the definition of  $N$ -variable joint statistical moments. However, due to the more specific usefulness of the  $N > 2$  cases, only 2-variable moments will be presented.

An  $(n+k)$ th-order **joint moment** of  $\underline{x}$  and  $\underline{y}$  (about the origin) has the general form

$$m_{n,k} = E[\underline{x}^n \underline{y}^k], \quad n, k \in \mathbb{Z}. \quad (4.79)$$

The most important case corresponds to  $m = n = 1$ :  $m_{11} = R_{\underline{x}\underline{y}}$  is called the **statistical correlation** between  $\underline{x}$  and  $\underline{y}$ . Under a frequentist perspective,  $R_{\underline{x}\underline{y}}$  is the average of infinite products of samples  $x$  and  $y$  jointly drawn from  $f_{\underline{x},\underline{y}}(x, y)$ , which links the correlation to an inner product between the random variables. Indeed, when  $R_{\underline{x}\underline{y}} = 0$ ,  $\underline{x}$  and  $\underline{y}$  are called **orthogonal**. One can say the correlation quantifies the overall linear relationship between the two variables. Moreover, when  $R_{\underline{x}\underline{y}} = \bar{x}\bar{y}$ ,  $\underline{x}$  and  $\underline{y}$  are called mutually **uncorrelated**; this “separability” in the mean is weaker than the distribution separability implied by the independence. In fact, if  $\underline{x}$  and  $\underline{y}$  are independent,

$$\begin{aligned} R_{\underline{x}\underline{y}} &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy f_{\underline{x},\underline{y}}(x, y) dx dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy f_{\underline{x}}(x) f_{\underline{y}}(y) dx dy \\ &= \int_{-\infty}^{\infty} xf_{\underline{x}}(x) dx \int_{-\infty}^{\infty} yf_{\underline{y}}(y) dy = \bar{x}\bar{y}, \end{aligned} \quad (4.80)$$

i.e., they are also uncorrelated. The converse is not necessarily true.

An  $(n+k)$ th-order **joint central moment** of  $\underline{x}$  and  $\underline{y}$ , computed about the mean, has the general form

$$\mu_{nk} = E[((\underline{x} - \bar{x})^n (\underline{y} - \bar{y})^k)], \quad n, k \in \mathbb{Z}. \quad (4.81)$$

---

<sup>9</sup>Convolution integral:  $(a * b)(y) = \int_{-\infty}^{\infty} a(\tilde{y})b(y - \tilde{y})d\tilde{y}$ .

Once more, the case  $m = n = 1$  is specially important:  $\mu_{11} = C_{\underline{x}\underline{y}}$  is called the **statistical covariance** between  $\underline{x}$  and  $\underline{y}$ , which quantifies the linear relationship between their random parts. Since<sup>10</sup>  $C_{\underline{x}\underline{y}} = R_{\underline{x}\underline{y}} - \bar{x}\bar{y}$ ,  $\underline{x}$  and  $\underline{y}$  are uncorrelated when  $C_{\underline{x}\underline{y}} = 0$ . If  $C_{\underline{x}\underline{y}} > 0$ , one says  $\underline{x}$  and  $\underline{y}$  are positively correlated, i.e., the variations of their statistical samples tend to occur in the same direction; if  $C_{\underline{x}\underline{y}} < 0$ , one says  $\underline{x}$  and  $\underline{y}$  are negatively correlated, i.e., the variations of their statistical samples tend to occur in opposite directions. For example, the age and the annual medical expenses of an individual are expected to be positively correlated random variables. A normalized covariance can be computed as the correlation between the normalized versions of  $\underline{x}$  and  $\underline{y}$ :

$$\rho_{\underline{x}\underline{y}} = \frac{C_{\underline{x}\underline{y}}}{\sigma_{\underline{x}}\sigma_{\underline{y}}} = E \left[ \frac{\underline{x} - \bar{x}}{\sigma_{\underline{x}}} \cdot \frac{\underline{y} - \bar{y}}{\sigma_{\underline{y}}} \right], \quad (4.82)$$

known as the **correlation coefficient** between  $\underline{x}$  and  $\underline{y}$ ,  $-1 \leq \rho_{\underline{x}\underline{y}} \leq 1$ , and can be interpreted as the percentage of correlation between  $x$  and  $y$ . Recalling the inner product interpretation, the correlation coefficient can be seen as the cosine of the angle between the statistical variations of the two random variables.

Table 4.3 Statistical Distribution of $\underline{d}$		
$d_1$	$d_2$	$P_{d_1, d_2}$
-1	1	0.3
1	-1	0.6
1	1	0.1

Consider, for example, the discrete variables  $\underline{d}_1$  and  $\underline{d}_2$  jointly distributed as shown in Table 4.3. Their joint second-order parameters are:

- correlation  $R_{\underline{x}\underline{y}} = -0.80$ , indicating  $\underline{d}_1$  and  $\underline{d}_2$  are not orthogonal;
- covariance  $C_{\underline{x}\underline{y}} = -0.72$ , indicating  $\underline{d}_1$  and  $\underline{d}_2$  tend to evolve in opposite directions;
- correlation coefficient  $\rho_{\underline{x}\underline{y}} \approx -0.80$ , indicating this negative correlation is relatively strong.

Returning to  $N$ -dimensional random variables, the **characteristic function** of a vector random variable  $\underline{x}$  is defined as

$$\Phi_{\underline{x}}(\omega_1, \omega_2, \dots, \omega_N) = E[e^{j(\omega_1\underline{x}_1 + \omega_2\underline{x}_2 + \dots + \omega_N\underline{x}_N)}], \quad \omega_1, \omega_2, \dots, \omega_N \in \mathcal{R}, \quad (4.83)$$

and inter-relates the moments of order  $m_{n_1, n_2, \dots, n_N}$  by:

$$m_{n_1, n_2, \dots, n_N} = (-j)^{n_1+n_2+\dots+n_N} \left. \frac{\partial^{n_1+n_2+\dots+n_N} \Phi_{\underline{x}}(\omega_1, \omega_2, \dots, \omega_N)}{\partial \omega_1^{n_1} \partial \omega_2^{n_2} \dots \partial \omega_N^{n_N}} \right|_{\omega_1=\omega_2=\dots=\omega_N=0}. \quad (4.84)$$

The **moment generating function** of a vector random variable  $\underline{x}$  is defined as

$$M_{\underline{x}}(v_1, v_2, \dots, v_N) = E[e^{v_1\underline{x}_1 + v_2\underline{x}_2 + \dots + v_N\underline{x}_N}], \quad v_1, v_2, \dots, v_N \in \mathcal{R}, \quad (4.85)$$

<sup>10</sup>Equation (4.43) can be seen as the next expression computed for  $\underline{x} = \underline{y}$ , since  $R_{\underline{x}\underline{x}} = \bar{x^2}$  and  $C_{\underline{x}\underline{x}} = \sigma_x^2$ .

and allows the computation of  $m_{n_1, n_2, \dots, n_N}$  as:

$$m_{n_1, n_2, \dots, n_N} = \frac{\partial^{n_1+n_2+\dots+n_N} M_{\underline{x}}(\nu_1, \nu_2, \dots, \nu_N)}{\partial \nu_1^{n_1} \nu_2^{n_2} \cdots \partial \nu_N^{n_N}} \Bigg|_{\nu_1=\nu_2=\dots=\nu_N=0}. \quad (4.86)$$

### 1.04.3.9 Central limit theorem

A result that partially justifies the ubiquitous use of Gaussian models, the **Central Limit Theorem** (CLT) states that (under mild conditions in practice) the distribution of a sum  $\underline{y} = \sum_{n=1}^N \underline{x}_n$  of  $N$  independent variables  $\underline{x}_n$  approaches a Gaussian distribution as  $N \rightarrow \infty$ . Having completely avoided the CLT proof, we can at least recall that in this case,  $f_{\underline{y}}(y) = \lim_{N \rightarrow \infty} f_{\underline{y}}(y) = (f_{\underline{x}_1} * f_{\underline{x}_2} * \cdots * f_{\underline{x}_N})(y)$  (see Eq. (4.77)). Of course,  $\bar{y} = \sum_{n=1}^N \bar{x}_n$  and, by the independence property,  $\sigma_{\underline{y}}^2 = \sum_{n=1}^N \sigma_{\underline{x}_n}^2$ .

Gaussian approximations for finite- $N$  models are useful for sufficiently high  $N$ , but due to the shape of the Gaussian distribution, the approximation error grows as  $y$  distances from  $\bar{y}$ .

The reader is invited to verify the validity of the approximation provided by the CLT for successive sums of independent  $\underline{x}_n$  uniformly distributed in  $[0, 1]$ .

Interestingly, the CLT also applies to the sum of discrete distributions: even if  $f_{\underline{y}}(y)$  remains impulsive, the shape of  $F_{\underline{y}}(y)$  approaches the shape of the CDF of a Gaussian random variable as  $N$  grows.

### 1.04.3.10 Multivariate Gaussian distribution

The PDF of an  $N$ -dimensional Gaussian variable is defined as

$$f_{\underline{x}}(\underline{x}) = \frac{1}{\sqrt{(2\pi)^N \det(\mathbf{C}_{\underline{x}})}} e^{-\frac{(\underline{x}-\bar{\underline{x}})^T \mathbf{C}_{\underline{x}}^{-1} (\underline{x}-\bar{\underline{x}})}{2}}, \quad (4.87)$$

where  $\bar{\underline{x}}$  is an  $N \times 1$  vector with elements  $\{\bar{\underline{x}}\}_n = \bar{x}_n$ ,  $1 \leq n \leq N$ , and  $\mathbf{C}_{\underline{x}}$  is an  $N \times N$  matrix with elements  $\{\mathbf{C}_{\underline{x}}\}_{mn} = C_{\underline{x}_m \underline{x}_n}$ ,  $1 \leq m, n \leq N$ , such that any related marginal distribution remains Gaussian. As a consequence, by Eq. (4.74), any conditional distribution  $f_{\underline{x}}(\underline{x}|B)$  with point conditioning  $B$  is also Gaussian.

By this definition, we can see the Gaussian distribution is completely defined by its first- and second-order moments, which means that if  $N$  jointly distributed random variables are known to be Gaussian, estimating their **mean-vector**  $\bar{\underline{x}}$  and their **covariance-matrix**  $\mathbf{C}_{\underline{x}}$  is equivalent to estimate their overall PDF. Moreover, if  $\underline{x}_m$  and  $\underline{x}_n$  are mutually uncorrelated,  $\forall m \neq n$ , then  $\mathbf{C}_{\underline{x}}$  is diagonal, and the joint PDF becomes  $f_{\underline{x}}(\underline{x}) = \prod_{n=1}^N f_{\underline{x}_n}(x_n)$ , i.e.,  $\underline{x}_n$  will be mutually independent. This is a strong result inherent to Gaussian distributions.

### 1.04.3.11 Transformation of vector random variables

The treatment of a general multiple random variable  $\underline{y} = \mathbf{T}(\underline{x})$  resulting from the application of the transformation  $\mathbf{T}(\cdot)$  to the multiple variable  $\underline{x}$  always starts by enforcing  $F_{\underline{x}}(\underline{x}) = F_{\underline{y}}(\underline{y})$ , with  $\underline{y} = \mathbf{T}(\underline{x})$ .

The special case when  $\mathbf{T}$  is invertible, i.e.,  $\underline{\mathbf{x}} = \mathbf{T}^{-1}(\underline{\mathbf{y}})$  (or  $\underline{x}_n = T_n^{-1}(\underline{\mathbf{y}})$ ,  $1 \leq n \leq N$ ), follows a closed expression:

$$f_{\underline{\mathbf{y}}}(\underline{\mathbf{y}}) = f_{\underline{\mathbf{x}}}(\underline{\mathbf{x}} = \mathbf{T}^{-1}(\underline{\mathbf{y}}))|J|, \quad (4.88)$$

where

$$J = \det \left( \begin{bmatrix} \frac{\partial T_1^{-1}}{\partial y_1} & \frac{\partial T_1^{-1}}{\partial y_2} & \cdots & \frac{\partial T_1^{-1}}{\partial y_N} \\ \frac{\partial T_2^{-1}}{\partial y_1} & \frac{\partial T_2^{-1}}{\partial y_2} & \cdots & \frac{\partial T_2^{-1}}{\partial y_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial T_N^{-1}}{\partial y_1} & \frac{\partial T_N^{-1}}{\partial y_2} & \cdots & \frac{\partial T_N^{-1}}{\partial y_N} \end{bmatrix} \right). \quad (4.89)$$

The reader should notice that this expression with  $N = 1$  reduces to Eq. (4.51) particularized to the invertible case.

Given the multiple random variable  $\underline{\mathbf{x}}$  with known mean-vector  $\bar{\mathbf{x}}$  and covariance-matrix  $\mathbf{C}_{\underline{\mathbf{x}}}$ , its linear transformation  $\underline{\mathbf{y}} = \mathbf{T}\underline{\mathbf{x}}$  will have:

- a mean-vector  $\bar{\mathbf{y}} = \mathbf{T}\bar{\mathbf{x}}$ ,
- a covariance-matrix  $\mathbf{C}_{\underline{\mathbf{y}}} = \mathbf{T}\mathbf{C}_{\underline{\mathbf{x}}}\mathbf{T}^T$ .

It is possible to show that the linear transformation  $\underline{\mathbf{y}}$  of an  $N$ -dimensional Gaussian random variable  $\underline{\mathbf{x}}$  is also Gaussian. Therefore, in this case the two expressions above completely determine the PDF of the transformed variable.

The generation of samples that follow a desired multivariate probabilistic distribution from samples that follow another known multivariate probabilistic distribution applies the same reasoning followed in the univariate case to the multivariate framework. The reader is invited to show that given the pair  $x_1, x_2$  of samples from the random variables  $\underline{x}_1, \underline{x}_2$ , jointly uniform in the region  $0 \leq x \leq 1, 0 \leq y \leq 1$ , it is possible to generate the pair  $y_1, y_2$  of samples from the random variables  $\underline{y}_1, \underline{y}_2$ , jointly Gaussian with the desired parameters  $\bar{y}_1, \sigma_{y_1}^2, \bar{y}_2, \sigma_{y_2}^2$ , and  $\rho_{y_1 y_2}$  by making

$$\begin{cases} y_1 = \bar{y}_1 + \sigma_{y_1} \sqrt{-2 \ln x_1} \cos(2\pi x_2), \\ y_2 = \bar{y}_2 + \rho_{y_1 y_2} \sigma_{y_2} \sqrt{-2 \ln x_1} \cos(2\pi x_2) + \sigma_{y_2} \sqrt{1 - \rho_{y_1 y_2}^2} \sqrt{-2 \ln x_1} \sin(2\pi x_2). \end{cases} \quad (4.90)$$

### 1.04.3.12 Complex random variables

At first sight, defining a complex random variable  $\underline{z}$  may seem impossible, since the seminal event  $\{\underline{z} \leq z\}$  makes no sense. However, in the vector random variable framework this issue can be circumvented if one jointly describes the real and imaginary parts (or magnitude and phase) of  $\underline{z}$ .

The single complex random variable  $\underline{z} = \underline{x} + j\underline{y}$ ,  $x, y \in \mathcal{R}$ , is completely represented by  $f_{\underline{x}, \underline{y}}(x, y)$ , which allows one to compute the expected value of a scalar function  $g(\underline{z})$  as  $E[g(\underline{z})] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(z) f_{\underline{x}, \underline{y}}(x, y) dx dy$ . Moreover, we can devise general definitions for the mean and variance of  $\underline{z}$  as, respectively:

- $\bar{\underline{z}} = \bar{\underline{x}} + j\bar{\underline{y}}$ ;
- $\sigma_{\underline{z}}^2 = E[|\underline{z} - \bar{\underline{z}}|^2] = \sigma_{\underline{x}}^2 + \sigma_{\underline{y}}^2$ , which measures the spread of  $\underline{z}$  about its mean in the complex plan.

An  $N$ -dimensional complex random variable  $\underline{z} = \underline{x} + j\underline{y}$  can be easily tackled through a properly defined joint distribution, e.g.,  $f_{\underline{x}, \underline{y}}(\underline{x}, \underline{y})$ . The individual variables  $z_n$ ,  $1 \leq n \leq N$ , will be independent when

$$f_{\underline{x}, \underline{y}}(\underline{x}, \underline{y}) = \prod_{n=1}^N f_{x_n, y_n}(x_n, y_n). \quad (4.91)$$

The following definitions must be generalized to cope with complex random variables:

- correlation:  $R_{z_1 z_2} = E[z_1^* z_2]$ ;
- covariance:  $C_{z_1 z_2} = E[(z_1 - \bar{z}_1)^*(z_2 - \bar{z}_2)]$ .

Now,  $C_{z_1 z_2} = R_{z_1 z_2} - \bar{z}_1^* \bar{z}_2$ . As before,  $z_1$  and  $z_2$  will be:

- orthogonal when  $R_{z_1 z_2} = 0$ ;
- uncorrelated when  $C_{z_1 z_2} = 0$ .

### 1.04.3.13 An application: estimators

One of the most encompassing applications of vector random variables is the so-called **parameter estimation**, per se an area supported by its own theory. The typical framework comprises using a finite set of measured data from a given phenomenon to estimate the parameters<sup>11</sup> of its underlying model.

A set  $\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N$  of  $N$  independent identically distributed (**iid**) random variables such that

$$f_{\underline{x}}(\underline{x}) = \prod_{n=1}^N f_{\underline{x}}(x_n) \quad (4.92)$$

can describe an  $N$ -size **random sample** of a population modeled by  $f_{\underline{x}}(x)$ . Any function  $g(\underline{x})$  (called a **statistic** of  $\underline{x}$ ) can constitute a **point estimator**  $\hat{\theta}$  of some parameter  $\theta$  such that given an  $N$ -size sample  $x_1, x_2, \dots, x_N$  of  $\underline{x}$ , one can compute an estimate  $\hat{\theta} = g(\underline{x})$  of  $\theta$ .

A classical example of point estimator for a fixed parameter is the so-called **sample mean**, which estimates  $\theta = \bar{x}$  by the arithmetic mean of the available samples  $x_n$ ,  $n = 1, 2, \dots, N$ :

$$\hat{\bar{x}}_N = \frac{1}{N} \sum_{n=1}^N x_n. \quad (4.93)$$

Not always defining a desired estimator is trivial task as in the previous example. In general, resorting to a proper analytical criterion is necessary. Suppose we are given the so-called likelihood function of  $\theta$ ,  $L(\theta) = f_{\underline{x}}(\underline{x}|\theta)$ , which provides a probabilistic model for  $\underline{x}$  as an explicit function of  $\theta$ . Given the sample  $\underline{x}$ , the so-called **maximum likelihood** (ML) estimator  $\hat{\theta}_{\text{ML}}$  computes an estimate of  $\theta$  by direct maximization of  $L(\theta)$ . This operation is meant to find the value of  $\theta$  that would make the available sample  $\underline{x}$  most probable.

---

<sup>11</sup>We will tackle only scalar parameters, without loss of generality.

Sometimes the parameter  $\theta$  itself can be a sample of a random variable  $\underline{\theta}$  described by an a priori PDF  $f_{\underline{\theta}}(\theta)$ . For example, suppose one wants to estimate the mean value of a shipment of components received from a given factory; since the components may come from several production units, we can think of a statistical model for their nominal values. In such situations, any reliable information on the parameter distribution can be taken in account to better tune the estimator formulation; these so-called **Bayesian estimators** rely on the a posteriori PDF of  $\theta$ :

$$f_{\underline{\theta}}(\theta|\mathbf{x}) = \frac{f_{\underline{\mathbf{x}}, \underline{\theta}}(\mathbf{x}, \theta)}{f_{\underline{\mathbf{x}}}(\mathbf{x})} \propto L(\theta) f_{\underline{\theta}}(\theta). \quad (4.94)$$

Given a sample  $\mathbf{x}$ , the so-called **maximum a posteriori** (MAP) estimator  $\hat{\underline{\theta}}_{\text{MAP}}$  computes an estimate of  $\theta$  as the mode of  $f_{\underline{\theta}}(\theta|\mathbf{x})$ . This choice is meant to find the most probable  $\theta$  that would have produced the available data  $\mathbf{x}$ . Several applications favor the **posterior mean** estimator, which for a given sample  $\mathbf{x}$  computes  $\hat{\underline{\theta}}_{\text{PM}} = E[\underline{\theta}|\mathbf{x}]$ , over the MAP estimator.

The quality of an estimator  $\underline{\theta}$  can be assessed through some of its statistical properties.<sup>12</sup> An overall measure of the estimator performance is its **mean square error**  $\text{MSE}(\hat{\underline{\theta}}) = E[(\theta - \hat{\underline{\theta}})^2]$ , which can be decomposed in two parts:

$$\text{MSE}(\hat{\underline{\theta}}) = b^2(\hat{\underline{\theta}}) + \sigma_{\hat{\underline{\theta}}}^2, \quad (4.95)$$

where  $b(\hat{\underline{\theta}}) = E[\theta - \hat{\underline{\theta}}]$  is the estimator **bias** and  $\sigma_{\hat{\underline{\theta}}}^2$  is its **variance**. The bias measures the deterministic part of the error, i.e., how much the estimates deviate from the target in the mean, thus providing an **accuracy** measure: the smaller its bias, the more accurate the estimator. The variance, by its turn, measures the random part of the error, i.e., how much the estimates spread about their mean, thus providing a **precision** measure: the smaller its variance, the more precise the estimator. Another property attributable to an estimator is **consistency**<sup>13</sup>:  $\hat{\underline{\theta}}$  is said to be consistent when  $\lim_{N \rightarrow \infty} P(|\hat{\underline{\theta}} - \theta| \geq \epsilon) = 0, \forall \epsilon > 0$ . Using the Chebyshev inequality (see Section 1.04.3.4), one finds that an unbiased estimator with  $\lim_{N \rightarrow \infty} \sigma_{\hat{\underline{\theta}}}^2 = 0$  is consistent. It can be shown that the sample mean defined in Eq. (4.93) has  $b(\hat{\underline{x}}_N) = 0$  (thus is unbiased) and  $\sigma_{\hat{\underline{x}}_N}^2 = \frac{\sigma_x^2}{N}$  (thus is consistent). The similarly built estimator for  $\sigma_x^2$ , the **unbiased sample variance**, computes

$$\widehat{\sigma}_{\underline{x}}^2 = \frac{1}{N-1} \sum_{n=1}^N \left( x_n - \hat{\underline{x}}_N \right)^2. \quad (4.96)$$

The reader is invited to prove that this estimator is unbiased, while its more intuitive form with denominator  $N$  instead of  $N-1$  is not.

One could argue how much reliable a point estimation is. In fact, if the range of  $\theta$  is continuous, one can deduce that  $P(\{\hat{\underline{\theta}} = \theta\}) = 0$ . However, perhaps we would feel safer if the output of an estimator were something like “ $\theta_1 \leq \theta \leq \theta_2$  with probability  $p$ .”  $\theta_1$  and  $\theta_2$  are the confidence limits and  $p$  is the confidence of this **interval estimate**.

<sup>12</sup>For the sake of simplicity, we refer only to the estimation of a fixed parameter  $\theta$ .

<sup>13</sup>There are other definitions of consistency.

**Example 2.** Suppose we use a 10-size sample mean to estimate  $\bar{x}$  of a unit-variance Gaussian random variable  $\underline{x}$ . Determine the 95%-confidence interval for the estimates.

**Solution 2.** It is easy to find that  $\hat{\bar{x}}_{10}$  is a Gaussian random variable with  $E[\hat{\bar{x}}_{10}] = \bar{x}$  and  $\sigma_{\hat{\bar{x}}_{10}}^2 = 0.1$ . We should find  $a$  such that  $P(\{\hat{\bar{x}}_{10} - a \leq \bar{x} \leq \hat{\bar{x}}_{10} + a\}) = 0.95$ , or  $P(|\bar{x} - \hat{\bar{x}}_{10}| \leq a) = 0.95$ . For the associated normalized Gaussian random variable,  $P\left(\left\{-1.96 \leq \frac{\bar{x} - \hat{\bar{x}}_{10}}{1/\sqrt{10}} \leq 1.96\right\}\right) \approx 0.95$ . Then,  $a \approx 0.62$ .

## 1.04.4 Random process

If the outcomes  $s$  of a given random experiment are amenable to variation in time  $t$ , each one can be mapped by some  $\underline{x}(s(t))$  into a real function  $x(t)$ . This is a direct generalization of the random variable. As a result, we get an infinite **ensemble** of time-varying **sample functions** or **realizations**  $x(t)$  which form the so-called **stochastic** or **random process**  $\underline{x}(t)$ . For a fixed  $t_0$ ,  $x(t_0)$  reduces to a single random variable, thus a random process can be seen as a time-ordered multiple random variable. An example of random process is the simultaneous observation of air humidity at every point of the globe,  $\underline{h}(t)$ : each realization  $h(t)$  describes the humidity variation in time at a randomly chosen place on the earth whereas the random variable  $\underline{h}(t_0)$  describes the distribution of air humidity over the earth at instant  $t_0$ . A similar construction in the discrete time  $n$  domain would produce  $\underline{x}[n]$  composed of realizations  $x[n]$ . If the hourly measures of air humidity substitute for its continuous measure in the former example, we get a new random process  $\underline{h}[n]$ .

As seen, there can be **continuous-** or **discrete-time** random processes. Since random variables have already been classified as continuous or discrete, one analogously refers to **continuous-valued** and **discrete-valued** random processes. Combining these two classifications according to time and value is bound to result ambiguous or awkwardly lengthy; when this complete categorization is necessary, one favors the nomenclature **random process** for the continuous-time case and **random sequence** for the discrete-time case, using the words “continuous” and “discrete” only with reference to amplitude values.<sup>14</sup> In the former examples, the continuous random process  $\underline{h}(t)$  and random sequence  $\underline{h}[n]$  would model the idealized actual air humidity as a physical variable, whereas the practical digitized measures of air humidity would be modeled by their discrete counterparts.

### 1.04.4.1 Distributions of random processes and sequences

One could think of the complete description of a random process or sequence as a joint CDF (or PDF) encompassing every possible instant of time  $t$  or  $n$ , respectively, which is obviously impractical. However, their partial representation by  $L$ th-order distributions (employing a slightly modified notation to include the reference to time) can be useful. The CDF of a random process  $\underline{x}(t)$  can be defined as

$$F_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = P(\{\underline{x}(t_1) \leq x_1, \underline{x}(t_2) \leq x_2, \dots, \underline{x}(t_L) \leq x_L\}), \quad (4.97)$$

<sup>14</sup>In this text, for completeness we opted for always explicitly writing both formulations, for processes and sequences.

with an associated PDF

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = \frac{\partial^L F_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, \dots, t_L)}{\partial x_1 \partial x_2 \dots \partial x_L}. \quad (4.98)$$

The CDF of a random sequence  $\underline{x}[n]$  can be defined as

$$F_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L) = P(\{x[n_1] \leq x_1, x[n_2] \leq x_2, \dots, x[n_L] \leq x_L\}), \quad (4.99)$$

with an associated PDF

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L) = \frac{\partial^L F_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L)}{\partial x_1 \partial x_2 \dots \partial x_L}. \quad (4.100)$$

This convention can be easily extended to the joint distributions of  $M$  random processes  $x_m(t)$ ,  $m = 1, 2, \dots, M$  (sequences  $x_m[n]$ ,  $m = 1, 2, \dots, M$ ), as will be seen in the next subsection. The notation for point-conditioned distributions can also be promptly inferred (see Section 1.04.4.12).

### 1.04.4.2 Statistical independence

As with random variables, the mutual independence of random processes or sequences is tested by the complete separability of their respective distributions. The random processes  $x_m(t)$ ,  $m = 1, 2, \dots, M$ , are independent when

$$\begin{aligned} & f_{x_1, x_2, \dots, x_M}(x_{11}, x_{12}, \dots, x_{1L_1}, x_{21}, x_{22}, \dots, x_{2L_2}, \dots, x_{M1}, x_{M2}, \dots, x_{ML_M}; \\ & t_{11}, t_{12}, \dots, t_{1L_1}, t_{21}, t_{22}, \dots, t_{2L_2}, \dots, t_{M1}, t_{M2}, \dots, t_{ML_M}) \\ &= \prod_{m=1}^M f_{x_m}(x_{m1}, x_{m2}, \dots, x_{mL_m}; t_{m1}, t_{m2}, \dots, t_{mL_m}) \end{aligned} \quad (4.101)$$

for any choice of time instants. Similarly, the random sequences  $x_m[n]$ ,  $m = 1, 2, \dots, M$ , are independent when

$$\begin{aligned} & f_{x_1, x_2, \dots, x_M}(x_{11}, x_{12}, \dots, x_{1L_1}, x_{21}, x_{22}, \dots, x_{2L_2}, \dots, x_{M1}, x_{M2}, \dots, x_{ML_M}; \\ & n_{11}, n_{12}, \dots, n_{1L_1}, n_{21}, n_{22}, \dots, n_{2L_2}, \dots, n_{M1}, n_{M2}, \dots, n_{ML_M}) \\ &= \prod_{m=1}^M f_{x_m}(x_{m1}, x_{m2}, \dots, x_{mL_m}; n_{m1}, n_{m2}, \dots, n_{mL_m}) \end{aligned} \quad (4.102)$$

for any choice of time instants.

### 1.04.4.3 First- and second-order moments for random processes and sequences

It is not necessary to redefine moments in the context of random processes and sequences, since we will always be tackling a set of random variables as in Section 1.04.3.8. But it is useful to revisit the first- and second-order cases with a slightly modified notation to include time information.

For a random process  $\underline{x}(t)$ , the following definitions apply:

- the **mean**  $\bar{x}(t) = E[\underline{x}(t)]$ ;
- the **mean square value or mean instantaneous power**  $\bar{x}^2(t) = E[\underline{x}^2(t)]$ ;
- the **variance**  $\sigma_{\underline{x}}^2(t) = E[(\underline{x}(t) - \bar{x}(t))^2]$ ;
- the **auto-correlation**  $R_{xx}(t_1, t_2) = E[\underline{x}(t_1)\underline{x}(t_2)]$ ;
- the **auto-covariance**  $C_{xx}(t_1, t_2) = E[(\underline{x}(t_1) - \bar{x}(t_1))(\underline{x}(t_2) - \bar{x}(t_2))]$ .

As seen for random variables,

$$\sigma_{\underline{x}}^2(t) = \bar{x}^2(t) - \bar{x}^2(t) \Rightarrow \bar{x}^2(t) = \bar{x}^2(t) + \sigma_{\underline{x}}^2(t), \quad (4.103)$$

which means that the mean instantaneous power of the process is the sum of a deterministic parcel with a random parcel.

Analogously, for a random sequence  $\underline{x}[n]$ , the following definitions apply:

- the **mean**  $\bar{x}[n] = E[\underline{x}[n]]$ ;
- the **mean square value or mean instantaneous power**  $\bar{x}^2[n] = E[\underline{x}^2[n]]$ ;
- the **variance**  $\sigma_{\underline{x}}^2[n] = E[(\underline{x}[n] - \bar{x}[n])^2]$ ;
- the **auto-correlation**  $R_{xx}[n_1, n_2] = E[\underline{x}[n_1]\underline{x}[n_2]]$ ;
- the **auto-covariance**  $C_{xx}[n_1, n_2] = E[(\underline{x}[n_1] - \bar{x}[n_1])(\underline{x}[n_2] - \bar{x}[n_2])]$ .

Again,

$$\sigma_{\underline{x}}^2[n] = \bar{x}^2[n] - \bar{x}^2[n] \Rightarrow \bar{x}^2[n] = \bar{x}^2[n] + \sigma_{\underline{x}}^2[n], \quad (4.104)$$

i.e., the mean instantaneous power of the sequence is the sum of a deterministic parcel with a random parcel.

Given two random processes  $\underline{x}(t)$  and  $\underline{y}(t)$ , one defines:

- the **cross-correlation**  $R_{xy}(t_1, t_2) = E[\underline{x}(t_1)\underline{y}(t_2)]$ ;
- the **cross-covariance**  $C_{xy}(t_1, t_2) = E[(\underline{x}(t_1) - \bar{x}(t_1))(\underline{y}(t_2) - \bar{y}(t_2))]$ .

The processes  $\underline{x}(t)$  and  $\underline{y}(t)$  are said to be mutually:

- **orthogonal** when  $R_{xy}(t_1, t_2) = 0, \forall t_1 \neq t_2$ ;
- **uncorrelated** when  $C_{xy}(t_1, t_2) = 0$ , which is the same as  $R_{xy}(t_1, t_2) = \bar{x}(t_1)\bar{y}(t_2) \forall t_1 \neq t_2$ .

The mean instantaneous power of  $(\underline{x}(t) + \underline{y}(t))$  is given by

$$E[(\underline{x}(t) + \underline{y}(t))^2] = \bar{x}^2(t) + 2E[\underline{x}(t)\underline{y}(t)] + \bar{y}^2(t), \quad (4.105)$$

which suggests the definition of  $E[\underline{x}(t)\underline{y}(t)]$  as the **mean instantaneous cross power** between  $\underline{x}(t)$  and  $\underline{y}(t)$ .

Analogously, given two random sequences  $\underline{x}[n]$  and  $\underline{y}[n]$ , one defines:

- the **cross-correlation**  $R_{xy}[n_1, n_2] = E[\underline{x}[n_1]\underline{y}[n_2]]$ ;
- the **cross-covariance**  $C_{xy}[n_1, n_2] = E[(\underline{x}[n_1] - \bar{x}[n_1])(\underline{y}[n_2] - \bar{y}[n_2])]$ .

The sequences  $\underline{x}[n]$  and  $\underline{y}[n]$  are said to be mutually:

- **orthogonal** when  $R_{xy}[n_1, n_2] = 0, \forall n_1 \neq n_2$ ;

- **uncorrelated** when  $C_{\underline{x}\underline{y}}[n_1, n_2] = 0$ , which is the same as  $R_{\underline{x}\underline{y}}[n_1, n_2] = \bar{\underline{x}}[n_1] \bar{\underline{y}}[n_2] \forall n_1 \neq n_2$ .

The mean instantaneous power of  $(\underline{x}[n] + \underline{y}[n])$  is given by

$$E[(\underline{x}[n] + \underline{y}[n])^2] = \bar{\underline{x}}^2[n] + 2E[\underline{x}[n]\underline{y}[n]] + \bar{\underline{y}}^2[n], \quad (4.106)$$

which suggests the definition of  $E[\underline{x}[n]\underline{y}[n]]$  as the **mean instantaneous cross power** between  $\underline{x}[n]$  and  $\underline{y}[n]$ .

The interpretation of this “cross power” is not difficult: it accounts for the constructive or destructive interaction of the two involved processes (sequences) as dictated by their mutual correlation.

We will dedicate some space later to the properties of second-order moments.

#### 1.04.4.4 Stationarity

The stationarity of random processes and sequences refers to the time-invariance of their statistical properties, which turns their treatment considerably easier.

**Strict-sense** is the strongest class of stationarity. A strict-sense stationary (SSS) random process  $\underline{x}(t)$  (sequence  $\underline{x}[n]$ ) bears exactly the same statistical properties as  $\underline{x}(t + \Delta t)$ ,  $\forall \Delta t \in \mathcal{R}$  ( $\underline{x}[n + \Delta n]$ ,  $\forall \Delta n \in \mathcal{Z}$ ), which means that its associated joint distribution for any set of time instants does not change when they are all shifted by the same time lag. A random process (sequence) in which each realization is a constant sampled from some statistical distribution in time is SSS.

Under this perspective, we can define that a random process  $\underline{x}(t)$  is **Lth-order** stationary when

$$F_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = F_{\underline{x}}(x_1, x_2, \dots, x_L; t_1 + \Delta t, t_2 + \Delta t, \dots, t_L + \Delta t) \quad (4.107)$$

or, alternatively,

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = f_{\underline{x}}(x_1, x_2, \dots, x_L; t_1 + \Delta t, t_2 + \Delta t, \dots, t_L + \Delta t). \quad (4.108)$$

Analogously, a random sequence  $\underline{x}[n]$  is **Lth-order** stationary when

$$F_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L) = F_{\underline{x}}(x_1, x_2, \dots, x_L; n_1 + \Delta n, n_2 + \Delta n, \dots, n_L + \Delta n) \quad (4.109)$$

or, alternatively,

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L) = f_{\underline{x}}(x_1, x_2, \dots, x_L; n_1 + \Delta n, n_2 + \Delta n, \dots, n_L + \Delta n). \quad (4.110)$$

We can say that an SSS random process or sequence is stationary for any order  $L$ . Moreover,  $L$ th-order stationarity implies  $(L - 1)$ th-order stationarity, by marginalization at both sides of any of the four equations above.

First-order stationarity says the statistical distribution at any individual instant of time  $t$  ( $n$ ) is the same, i.e.,  $f_{\underline{x}}(\underline{x}; t)$  does not depend on  $t$  ( $f_{\underline{x}}(\underline{x}; n)$  does not depend on  $n$ ). A natural consequence is the time-invariance of its statistical mean:  $\bar{\underline{x}}(t) = \bar{\underline{x}}$  ( $\bar{\underline{x}}[n] = \bar{\underline{x}}$ ). Second-order stationarity says the joint statistical distribution at any two time instants  $t$  and  $t + \tau$  for a fixed  $\tau$  ( $n$  and  $n + k$  for a fixed  $k$ ) is the same. A natural consequence is the time-invariance of its auto-correlation, which can depend only of the time lag between the two time instants:  $R_{\underline{x}\underline{x}}(t, t + \tau) = R_{\underline{x}\underline{x}}(\tau)$  ( $R_{\underline{x}\underline{x}}[n, n + k] = R_{\underline{x}\underline{x}}[k]$ ).

A weaker class of stationarity called **wide-sense** is much used in practice for its simple testability. A random process  $\underline{x}(t)$  (sequence  $\underline{x}[n]$ ) is said to be wide-sense stationary (WSS) when its mean and auto-correlation are time-invariant. Such conditions do not imply stationarity of any order, although second-order stationarity implies wide-sense stationarity. As an extension of such definition, one says that two processes  $\underline{x}(t)$  and  $\underline{y}(t)$  (sequences  $\underline{x}[n]$  and  $\underline{y}[n]$ ) are jointly WSS when they are individually WSS and their cross-correlation is time-invariant, i.e., can depend only of the time lag between the two time instants:  $R_{\underline{x}\underline{y}}(t, t + \tau) = R_{\underline{x}\underline{y}}(\tau)$  ( $R_{\underline{x}\underline{y}}[n, n + k] = R_{\underline{x}\underline{y}}[k]$ ).

#### 1.04.4.5 Properties of correlation functions for WSS processes and sequences

Wide-sense stationarity in random processes and sequences induces several interesting properties, some of which are listed below. For processes,

- $|R_{\underline{x}\underline{x}}(\tau)| \leq R_{\underline{x}\underline{x}}(0);$
- $R_{\underline{x}\underline{x}}(-\tau) = R_{\underline{x}\underline{x}}(\tau);$
- If  $\lim_{\tau \rightarrow \infty} R_{\underline{x}\underline{x}}(\tau) = B$ ,  $B = \bar{x}^2$ ;
- If  $\underline{x}(t)$  has a periodic component,  $R_{\underline{x}\underline{x}}(\tau)$  has the same periodic component;
- $R_{\underline{x}\underline{y}}(-\tau) = R_{\underline{y}\underline{x}}(\tau);$
- $R_{\underline{x}\underline{y}}(\tau) \leq \sqrt{R_{\underline{x}\underline{x}}(0)R_{\underline{y}\underline{y}}(0)};$
- $R_{\underline{x}\underline{y}}(\tau) \leq \frac{R_{\underline{x}\underline{x}}(0) + R_{\underline{y}\underline{y}}(0)}{2}.$

For sequences,

- $|R_{\underline{x}\underline{x}}[k]| \leq R_{\underline{x}\underline{x}}[0];$
- $R_{\underline{x}\underline{x}}[-k] = R_{\underline{x}\underline{x}}[k];$
- If  $\lim_{k \rightarrow \infty} R_{\underline{x}\underline{x}}[k] = B$ ,  $B = \bar{x}^2$ ;
- If  $\underline{x}[n]$  has a periodic component,  $R_{\underline{x}\underline{x}}[k]$  has the same periodic component;
- $R_{\underline{x}\underline{y}}[-k] = R_{\underline{y}\underline{x}}[k];$
- $R_{\underline{x}\underline{y}}[k] \leq \sqrt{R_{\underline{x}\underline{x}}[0]R_{\underline{y}\underline{y}}[0]};$
- $R_{\underline{x}\underline{y}}[k] \leq \frac{R_{\underline{x}\underline{x}}[0] + R_{\underline{y}\underline{y}}[0]}{2}.$

Even if the properties are not proven here, some interesting observations can be traced about them:

- The statistical behavior of the process (sequence) at a given instant of time is maximally correlated with itself, an intuitive result.
- The autocorrelation commutes.
- The statistical behaviors of a WSS process (sequence) with no periodic components at two time instants separated by  $\tau \rightarrow \infty$  ( $k \rightarrow \infty$ ) are uncorrelated. Then, the autocorrelation tends to the product of two identical statistical means.
- Periodic components in the process (sequence) cause the correlation maximum and surrounding behavior to repeat at each fundamental period.
- The arithmetic and geometric mean properties are in a certain sense linked to the first property. The second property is more stringent than the first.

#### 1.04.4.6 Time averages of random processes and sequences

The proper use of random processes and sequences to model practical problems depends on the careful understanding of both their statistical and time variations. The main difference between these two contexts, which must be kept in mind, is the fact that time is inexorably ordered, thus allowing the inclusion in the model of some deterministic (predictable) time-structure. When we examine the statistical auto-correlation between two time instants of a random process or sequence, we learn how the statistical samples taken at those instants follow each other. In a WSS process or sequence, this measure for different lags can convey an idea of statistical periodicity.<sup>15</sup> But what if one is concerned with the time structure and characteristics of each (or some) individual realization of a given process or sequence? The use of time averages can provide this complementary description.

The time average of a given continuous-time function  $f(t)$  can be defined as

$$A[f(t)] = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T f(t) dt. \quad (4.111)$$

Given the random process  $\underline{x}(t)$ , the **time average** of any of its realizations  $x(t)$  is

$$\bar{x} = A[x(t)]. \quad (4.112)$$

The **average power** of  $x(t)$  is

$$\mathcal{P}_{xx} = A[x^2(t)]. \quad (4.113)$$

The **time auto-correlation** of  $x(t)$  for a lag  $\tau$  is defined as

$$\mathcal{R}_{xx}(\tau) = A[x(t)x(t + \tau)]. \quad (4.114)$$

The **time cross-correlation** between the realizations  $x(t)$  of process  $\underline{x}(t)$  and  $y(t)$  of process  $\underline{y}(t)$  for a lag  $\tau$  is defined as

$$\mathcal{R}_{xy}(\tau) = A[x(t)y(t + \tau)]. \quad (4.115)$$

As in the statistical case, an **average cross power** between  $x(t)$  and  $y(t)$ , which accounts for their mutual constructive or destructive interferences due to their time structure, can be defined as

$$\mathcal{P}_{xy} = A[x(t)y(t)]. \quad (4.116)$$

Computed for the complete ensemble, such measures produce the random variables  $\bar{x}$ ,  $\mathcal{P}_{xx}$ ,  $\mathcal{R}_{xx}(\tau)$ ,  $\mathcal{R}_{xy}(\tau)$ , and  $\mathcal{P}_{xy}$ , respectively. Under mild convergence conditions,

$$E[\bar{x}] = E[A[\underline{x}(t)]] = A[E[\underline{x}(t)]] = A[\bar{x}(t)], \quad (4.117)$$

$$E[\mathcal{P}_{xx}] = E[A[\underline{x}^2(t)]] = A[E[\underline{x}^2(t)]] = A[\bar{x}^2(t)], \quad (4.118)$$

$$E[\mathcal{R}_{xx}(\tau)] = E[A[\underline{x}(t)\underline{x}(t + \tau)]] = A[E[\underline{x}(t)\underline{x}(t + \tau)]] = A[R_{xx}(t, t + \tau)], \quad (4.119)$$

$$E[\mathcal{R}_{xy}(\tau)] = E[A[\underline{x}(t)\underline{y}(t + \tau)]] = A[E[\underline{x}(t)\underline{y}(t + \tau)]] = A[R_{xy}(t, t + \tau)], \quad (4.120)$$

---

<sup>15</sup>We hinted this point when discussing the properties of second-order moments, in Section 1.04.4.5.

and

$$\mathbb{E}[\underline{\mathcal{P}}_{xy}] = \mathbb{E}[A[\underline{x}(t)\underline{y}(t)]] = A[\mathbb{E}[\underline{x}(t)\underline{y}(t)]], \quad (4.121)$$

which are respectively the overall mean value, mean power, and auto-correlation (for lag  $\tau$ ) of process  $\underline{x}(t)$ , and cross-correlation (for lag  $\tau$ ) and mean cross power between processes  $\underline{x}(t)$  and  $\underline{y}(t)$ .

The time average of a given discrete-time function  $f[n]$  can be defined as

$$A[f[n]] = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N f[n]. \quad (4.122)$$

Given the random sequence  $\underline{x}[n]$ , the **time average** of any of its realizations  $x[n]$  is

$$\bar{x} = A[x[n]]. \quad (4.123)$$

The **average power** of  $x[n]$  is

$$\underline{\mathcal{P}}_{xx} = A[x^2[n]]. \quad (4.124)$$

The **time auto-correlation** of  $x[n]$  for lag  $k$  is defined as

$$\mathcal{R}_{xx}[k] = A[x[n]x[n+k]]. \quad (4.125)$$

The **time cross-correlation** between the realizations  $x[n]$  of sequence  $\underline{x}[n]$  and  $y[n]$  of sequence  $\underline{y}[n]$  for lag  $k$  is defined as

$$\mathcal{R}_{xy} = A[x[n]y[n+k]]. \quad (4.126)$$

As in the statistical case, an **average cross power** between  $x[n]$  and  $y[n]$ , which accounts for their mutual constructive or destructive interferences due to their time structure, can be defined as

$$\underline{\mathcal{P}}_{xy} = A[x[n]y[n]]. \quad (4.127)$$

Computed for the complete ensemble, such measures produce the random variables  $\bar{x}$ ,  $\underline{\mathcal{P}}_{xx}$ ,  $\mathcal{R}_{xx}[k]$ ,  $\underline{\mathcal{R}}_{xy}[k]$ , and  $\underline{\mathcal{P}}_{xy}$ , respectively. Under mild convergence conditions,

$$\mathbb{E}[\bar{x}] = \mathbb{E}[A[\underline{x}[n]]] = A[\mathbb{E}[\underline{x}[n]]] = A[\bar{x}[n]], \quad (4.128)$$

$$\mathbb{E}[\underline{\mathcal{P}}_{xx}] = \mathbb{E}[A[\underline{x}^2[n]]] = A[\mathbb{E}[\underline{x}^2[n]]] = A[\bar{x}^2[n]], \quad (4.129)$$

$$\mathbb{E}[\mathcal{R}_{xx}[k]] = \mathbb{E}[A[\underline{x}[n]\underline{x}[n+k]]] = A[\mathbb{E}[\underline{x}[n]\underline{x}[n+k]]] = A[R_{xx}[n, n+k]], \quad (4.130)$$

$$\mathbb{E}[\underline{\mathcal{R}}_{xy}[k]] = \mathbb{E}[A[\underline{x}[n]\underline{y}[n+k]]] = A[\mathbb{E}[\underline{x}[n]\underline{y}[n+k]]] = A[R_{xy}[n, n+k]], \quad (4.131)$$

and

$$\mathbb{E}[\underline{\mathcal{P}}_{xy}] = \mathbb{E}[A[\underline{x}[n]\underline{y}[n]]] = A[\mathbb{E}[\underline{x}[n]\underline{y}[n]]], \quad (4.132)$$

which are respectively the overall mean value, mean power, and auto-correlation (for lag  $k$ ) of sequence  $\underline{x}[n]$ , and cross-correlation (for lag  $k$ ) and mean cross power between sequences  $\underline{x}[n]$  and  $\underline{y}[n]$ .

As an example, consider that a deterministic signal  $s(t) \neq 0$  (with  $A[s(t)] = 0$ ) is additively contaminated by random noise  $n(t)$  which can be modeled as a realization of the WSS random process  $\underline{n}(t)$  (with  $\bar{n}(t) = 0$  and  $\bar{n}^2(t) = \sigma_n^2$ ), thus yielding the random signal  $x(t) = s(t) + n(t)$ . The corresponding process  $\underline{x}(t) = s(t) + \underline{n}(t)$  is obviously not WSS, since  $\bar{x}(t) = s(t)$ . Furthermore, (following the nomenclature we have adopted) we can compute its:

- mean instantaneous power  $\overline{\underline{x}^2}(t) = s^2(t) + \sigma_n^2$ ;
- mean power  $\mathcal{P}_{xx} = A[s^2(t)] + A[n^2(t)]$ ;
- overall mean power  $E[\mathcal{P}_{xx}] = A[s^2(t)] + \sigma_n^2$ .

Defining the signal-to-noise ratio (SNR) as the ratio between signal and noise powers, we can conclude that  $\text{SNR}_{\underline{x}} = \frac{A[s^2(t)]}{\sigma_n^2}$ .

Now, suppose that two differently contaminated versions of  $s(t)$  are available:  $x_1(t) = s(t) + n_1(t)$  and  $x_2(t) = s(t) + n_2(t)$ , and that the underlying noise processes  $\underline{n}_1(t)$  and  $\underline{n}_2(t)$  are jointly WSS and uncorrelated, with  $\overline{\underline{n}_1}(t) = \overline{\underline{n}_2}(t) = 0$ ,  $\overline{\underline{n}_1^2}(t) = \sigma_{n_1}^2$ , and  $\overline{\underline{n}_2^2}(t) = \sigma_{n_2}^2$ . If an average signal  $x_m(t) = \frac{x_1(t)+x_2(t)}{2}$  is computed, we can guarantee  $\text{SNR}_{\underline{x}_m} > \text{SNR}_{\underline{x}_1}$  and  $\text{SNR}_{\underline{x}_m} > \text{SNR}_{\underline{x}_2}$  (i.e., noise is reduced) as long as  $\frac{1}{3} < \frac{\sigma_{n_1}^2}{\sigma_{n_2}^2} < 3$ .

#### 1.04.4.7 Ergodicity

The so-called **ergodicity** is a property that allows interchanging statistic and temporal characteristics of some random processes (sequences)—which are then called ergodic. There are several levels of ergodicity, some of which are discussed below.

A random process  $\underline{x}(t)$  with constant statistical mean is said to be mean-ergodic when any time average  $\overline{\underline{x}}$  is equal to  $\underline{\underline{x}}$  with probability 1, which requires  $\sigma_{\overline{\underline{x}}}^2 = 0$ . If  $\underline{x}(t)$  is WSS, a necessary and sufficient condition for mean-ergodicity is

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^{2T} C_{\underline{x}\underline{x}}(\tau) \left(1 - \frac{\tau}{2T}\right) d\tau = 0. \quad (4.133)$$

A random process  $\underline{x}(t)$  with time-invariant auto-correlation is said to be auto-correlation-ergodic when any time auto-correlation  $\mathcal{R}_{xx}(\tau)$  is equal to  $R_{\underline{x}\underline{x}}(\tau)$  with probability 1, which requires  $\sigma_{\mathcal{R}_{xx}(\tau)}^2 = 0$ . Two processes  $\underline{x}(t)$  and  $\underline{y}(t)$  with time-invariant cross-correlation are cross-correlation-ergodic when any time cross-correlation  $\mathcal{R}_{xy}(\tau)$  is equal to  $R_{\underline{x}\underline{y}}(\tau)$  with probability 1, which requires  $\sigma_{\mathcal{R}_{xy}(\tau)}^2 = 0$ . Conditions for correlation-ergodicity of random processes involve 4th-order moments.

A random sequence  $\underline{x}[n]$  with constant statistical mean is said to be mean-ergodic when any time average  $\overline{\underline{x}}$  is equal to  $\underline{\underline{x}}$  with probability 1, which requires  $\sigma_{\overline{\underline{x}}}^2 = 0$ . If  $\underline{x}[n]$  is WSS, a necessary and sufficient condition for mean-ergodicity is

$$\lim_{M \rightarrow \infty} \frac{1}{2M+1} \sum_{k=-M}^M C_{\underline{x}\underline{x}}[k] \left(1 - \frac{|k|}{2M+1}\right) = 0. \quad (4.134)$$

A random sequence  $\underline{x}[n]$  with time-invariant auto-correlation is said to be auto-correlation-ergodic when any time auto-correlation  $\mathcal{R}_{xx}[k]$  is equal to  $R_{\underline{x}\underline{x}}[k]$  with probability 1, which requires  $\sigma_{\mathcal{R}_{xx}[k]}^2 = 0$ . Two sequences  $\underline{x}[n]$  and  $\underline{y}[n]$  with time-invariant cross-correlation are cross-correlation-ergodic when any time cross-correlation  $\mathcal{R}_{xy}[k]$  is equal to  $R_{\underline{x}\underline{y}}[k]$  with probability 1, which requires  $\sigma_{\mathcal{R}_{xy}[k]}^2 = 0$ . Conditions for correlation-ergodicity of random sequences involve 4th-order moments.

A process (sequence) that is mean- and auto-correlation-ergodic is called **wide-sense ergodic**. Two wide-sense ergodic processes (sequences) that are cross-correlation-ergodic are called **jointly wide-sense ergodic**.

A process (sequence) is **distribution-ergodic** when is ergodic for every moment.

The SSS random process (sequence) formed by random constant realizations is not ergodic in any sense. From Eqs. (4.133) and (4.134), a WSS process  $\underline{x}(t)$  (sequence  $\underline{x}[n]$ ), statistically uncorrelated at any different time instants  $t_1$  and  $t_2$  ( $n_1$  and  $n_2$ ), i.e., with  $C_{\underline{x}\underline{x}}(t_2 - t_1) = \delta(t_2 - t_1)$  ( $C_{\underline{x}\underline{x}}[n_2 - n_1] = \delta[n_2 - n_1]$ ), is mean-ergodic.

In practical real-life situations, quite often just a single realization of an underlying random process (sequence) is available. In such cases, if the latter is known to be ergodic, we can make use of the complete powerful statistical modeling framework described in this chapter. But how can one guarantee the property is enjoyed by a process (sequence) of which an only sample function is known? This is not so stringent a requirement: in fact, one needs just to be sure that there can be an ergodic process (sequence) of which that time function is a realization. Strictly speaking, a given music recording  $s(t)$  additively contaminated by background noise  $d(t)$  cannot be modeled as a realization of a mean-ergodic process,<sup>16</sup> since each member of the ensemble would combine the same  $s(t)$  with a different  $d(t)$ . The random process which describes the noisy signal can be written as  $\underline{x}(t) = s(t) + \underline{d}(t)$ ; thus,  $\bar{x}(t) = s(t)$ , while in practice we know that  $\bar{x} = 0$ .

#### 1.04.4.8 An encompassing example

Define the random sequences  $\underline{x}_1[n] = \underline{a}_1 \cos(\Omega_0 n + \underline{\phi}_1)$  and  $\underline{x}_2[n] = \underline{a}_2 \sin(\Omega_0 n + \underline{\phi}_2)$ , such that:

- $\underline{a}_1$  is a random variable with mean  $\bar{\underline{a}}_1$  and variance  $\sigma_{\underline{a}_1}^2$ ;
- $\underline{a}_2$  is a random variable with mean  $\bar{\underline{a}}_2$  and variance  $\sigma_{\underline{a}_2}^2$ ;
- $(-\pi \leq \Omega_0 < \pi)$  rad/sample is a real constant;
- $\underline{\phi}_1$  is a random variable uniformly distributed between  $-\pi$  and  $\pi$  rad;
- $\underline{\phi}_2$  is a random variable uniformly distributed between  $-\pi$  and  $\pi$  rad;
- $\underline{a}_1, \underline{a}_2, \underline{\phi}_1$ , and  $\underline{\phi}_2$  are mutually independent.

Compute their time and statistical means, mean powers, auto-correlations, cross-correlations, and mean cross-powers; and discuss their correlation, orthogonality, stationarity, and ergodicity.

**Solution 3.** Time-averages:

$$\bar{x}_1 = A[x_1[n]] = a_1 A[\cos(\Omega_0 n + \phi_1)] = 0. \quad (4.135)$$

$$\bar{x}_2 = A[x_2[n]] = a_2 A[\sin(\Omega_0 n + \phi_2)] = 0. \quad (4.136)$$

$$\begin{aligned} R_{x_1 x_1}[k] &= A[x_1[n]x_1[n+k]] = a_1^2 A[\cos(\Omega_0 n + \phi_1)\cos(\Omega_0 n + \Omega_0 k + \phi_1)] \\ &= a_1^2 A \left[ \frac{\cos(2\Omega_0 n + \Omega_0 k + 2\phi_1) + \cos(\Omega_0 k)}{2} \right] = \frac{a_1^2}{2} \cos(\Omega_0 k); \end{aligned} \quad (4.137)$$

---

<sup>16</sup>Even if the author has done so many times.

then,

$$\mathcal{P}_{x_1 x_1} = R_{x_1 x_1}[0] = \frac{a_1^2}{2}. \quad (4.138)$$

$$\begin{aligned} \mathcal{R}_{x_2 x_2}[k] &= A[x_2[n]x_2[n+k]] = a_2^2 A[\sin(\Omega_0 n + \phi_2) \sin(\Omega_0 n + \Omega_0 k + \phi_2)] \\ &= a_2^2 A \left[ \frac{\cos(\Omega_0 k) - \cos(2\Omega_0 n + \Omega_0 k + 2\phi_2)}{2} \right] = \frac{a_2^2}{2} \cos(\Omega_0 k); \end{aligned} \quad (4.139)$$

then,

$$\mathcal{P}_{x_2 x_2} = \mathcal{R}_{x_2 x_2}[0] = \frac{a_2^2}{2}. \quad (4.140)$$

$$\begin{aligned} \mathcal{R}_{x_1 x_2}[k] &= A[x_1[n]x_2[n+k]] = a_1 a_2 A[\cos(\Omega_0 n + \phi_1) \sin(\Omega_0 n + \Omega_0 k + \phi_2)] \\ &= a_1 a_2 A \left[ \frac{\sin(2\Omega_0 n + \Omega_0 k + \phi_1 + \phi_2) - \sin(\phi_1 - \phi_2 - \Omega_0 k)}{2} \right] \\ &= \frac{a_1 a_2}{2} \sin(\Omega_0 k + \phi_2 - \phi_1); \end{aligned} \quad (4.141)$$

then,

$$\mathcal{P}_{x_1 x_2} = \mathcal{R}_{x_1 x_2}[0] = \frac{a_1 a_2}{2} \sin(\phi_2 - \phi_1). \quad (4.142)$$

Expected values:

$$\begin{aligned} \bar{x}_1 &= E[\underline{x}_1[n]] = E[\underline{a}_1] E[\cos(\Omega_0 n + \underline{\phi}_1)] \\ &= \bar{a}_1 \int_{-\pi}^{\pi} \frac{1}{2\pi} \cos(\Omega_0 n + \phi_1) d\phi_1 = \bar{a}_1 \cdot 0 = 0. \end{aligned} \quad (4.143)$$

$$\begin{aligned} \bar{x}_2 &= E[x_2[n]] = E[\underline{a}_2] E[\sin(\Omega_0 n + \underline{\phi}_2)] \\ &= \bar{a}_2 \int_{-\pi}^{\pi} \frac{1}{2\pi} \sin(\Omega_0 n + \phi_2) d\phi_2 = \bar{a}_2 \cdot 0 = 0. \end{aligned} \quad (4.144)$$

$$\begin{aligned} R_{\underline{x}_1 \underline{x}_1}[n, n+k] &= E[\underline{x}_1[n]\underline{x}_1[n+k]] = E[\underline{a}_1^2] E[\cos(\Omega_0 n + \underline{\phi}_1) \cos(\Omega_0 n + \Omega_0 k + \underline{\phi}_1)] \\ &= \frac{\bar{a}_1^2 + \sigma_{\underline{a}_1}^2}{2} E \left[ \frac{\cos(2\Omega_0 n + \Omega_0 k + 2\underline{\phi}_1) + \cos(\Omega_0 k)}{2} \right] \\ &= \frac{\bar{a}_1^2 + \sigma_{\underline{a}_1}^2}{2} \cos(\Omega_0 k); \end{aligned} \quad (4.145)$$

then,

$$\bar{x}_1^2 = R_{\underline{x}_1 \underline{x}_1}[n, n] = \frac{\bar{a}_1^2 + \sigma_{\underline{a}_1}^2}{2}. \quad (4.146)$$

$$\begin{aligned}
R_{\underline{x}_2 \underline{x}_2}[n, n+k] &= E[\underline{x}_2[n] \underline{x}_2[n+k]] = E[\underline{a}_2^2] E[\sin(\Omega_0 n + \underline{\phi}_2) \sin(\Omega_0 n + \Omega_0 k + \underline{\phi}_2)] \\
&= \frac{\underline{a}_2^2 + \sigma_{\underline{a}_2}^2}{2} E\left[\frac{\cos(\Omega_0 k) - \cos(2\Omega_0 n + \Omega_0 k + 2\underline{\phi}_2)}{2}\right] \\
&= \frac{\underline{a}_2^2 + \sigma_{\underline{a}_2}^2}{2} \cos(\Omega_0 k);
\end{aligned} \tag{4.147}$$

then,

$$\overline{\underline{x}_2^2} = R_{\underline{x}_2 \underline{x}_2}[n, n] = \frac{\underline{a}_2^2 + \sigma_{\underline{a}_2}^2}{2}. \tag{4.148}$$

$$\begin{aligned}
R_{\underline{x}_1 \underline{x}_2}[n, n+k] &= E[\underline{x}_1[n] \underline{x}_2[n+k]] = E[\underline{a}_1] E[\underline{a}_2] \\
&\quad E[\cos(\Omega_0 n + \underline{\phi}_1)] E[\sin(\Omega_0 n + \Omega_0 k + \underline{\phi}_2)] \\
&= \overline{\underline{a}_1 \underline{a}_2} \cdot 0.0 = 0.
\end{aligned} \tag{4.149}$$

Conclusions:

- Since  $\overline{\underline{x}_1}$  and  $R_{\underline{x}_1 \underline{x}_1}[n, n+k]$  do not depend on  $n$ ,  $\underline{x}_1[n]$  is WSS.
- Since  $\overline{\underline{x}_2}$  and  $R_{\underline{x}_2 \underline{x}_2}[n, n+k]$  do not depend on  $n$ ,  $\underline{x}_2[n]$  is WSS.
- Since  $\underline{x}_1[n]$  is WSS,  $\underline{x}_2[n]$  is WSS, and  $R_{\underline{x}_1 \underline{x}_2}[n, n+k]$  does not depend on  $n$ ,  $\underline{x}_1[n]$  and  $\underline{x}_2[n]$  are jointly WSS.
- Since  $R_{\underline{x}_1 \underline{x}_2}[n, n+k] = \overline{\underline{x}_1 \underline{x}_2}$ ,  $\underline{x}_1[n]$  and  $\underline{x}_2[n]$  are uncorrelated.
- Since  $R_{\underline{x}_1 \underline{x}_2}[n, n+k] = 0$ ,  $\underline{x}_1[n]$  and  $\underline{x}_2[n]$  are orthogonal.
- Since  $\overline{\underline{x}_1} = \overline{\underline{x}_1}$ ,  $\underline{x}_1[n]$  is mean-ergodic.
- Since  $\overline{\underline{x}_2} = \overline{\underline{x}_2}$ ,  $\underline{x}_2[n]$  is mean-ergodic.

#### 1.04.4.9 Gaussian processes and sequences

A very special case of random processes (sequences) are the Gaussian-distributed. The corresponding  $L$ th-order PDFs can be written (see Section 1.04.3.10).

- for processes as

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = \frac{1}{\sqrt{(2\pi)^L \det(\mathbf{C}_{\underline{x}})}} e^{-\frac{(\mathbf{x}-\bar{\mathbf{x}})^T \mathbf{C}_{\underline{x}}^{-1} (\mathbf{x}-\bar{\mathbf{x}})}{2}}, \tag{4.150}$$

where  $\bar{\mathbf{x}}$  is the  $L \times 1$  mean-vector with elements  $\{\bar{x}_l\}_l = \bar{x}(t_l)$ , for  $1 \leq l \leq L$ , and  $\mathbf{C}_{\underline{x}}$  is the  $L \times L$  covariance-matrix with elements  $\{\mathbf{C}_{\underline{x}}\}_{l_1 l_2} = C_{\underline{x} \underline{x}}(t_{l_1}, t_{l_2})$ , for  $1 \leq l_1, l_2 \leq L$ ;

- for sequences as

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; n_1, n_2, \dots, n_L) = \frac{1}{\sqrt{(2\pi)^L \det(\mathbf{C}_{\underline{x}})}} e^{-\frac{(\mathbf{x}-\bar{\mathbf{x}})^T \mathbf{C}_{\underline{x}}^{-1} (\mathbf{x}-\bar{\mathbf{x}})}{2}}, \tag{4.151}$$

where  $\underline{\bar{x}}$  is the  $L \times 1$  mean-vector with elements  $\{\underline{\bar{x}}\}_l = \bar{x}[n_l]$ , for  $1 \leq l \leq L$ , and  $\mathbf{C}_{\underline{x}}$  is the  $L \times L$  covariance-matrix with elements  $\{\mathbf{C}_{\underline{x}}\}_{l_1 l_2} = C_{\underline{x}\underline{x}}[n_{l_1}, n_{l_2}]$ , for  $1 \leq l_1, l_2 \leq L$ .

From the definition above:

- A WSS Gaussian random process (sequence) is SSS: Since the PDF of a Gaussian process (sequence) is entirely described by first- and second-order moments, if these moments do not depend on  $t(n)$ , the PDF itself does not depend on  $t(n)$ .
- Uncorrelated Gaussian processes (sequences) are independent: A joint PDF of two uncorrelated Gaussian processes (sequences) can be easily factorized as the product of their individual PDFs, since the covariance-matrix becomes block diagonal.

These two strong properties turn Gaussian models mathematically simpler to tackle, and the strict-sense stationarity and independence conditions easier to meet in practice.

The reader is invited to show that a stationary Gaussian process (sequence) whose auto-correlation is absolutely integrable in  $\tau$  (summable in  $k$ ) is ergodic.

#### 1.04.4.10 Poisson random process

Poisson is an example of discrete random process that we have already defined in Section 1.04.3.2. Each realization  $x(t)$  of  $\underline{x}(t)$  counts the occurrences along time of a certain event whose mean occurrence rate is  $\lambda$  per time unit, provided that.

- there are no simultaneous occurrences;
- occurrence times are independent.

It can model the entry of clients in a store, for example. By convention:

- $\underline{x}(0) = 0$ , i.e., the count starts in  $t = 0$ ;
- $\underline{x}(t_0 > 0)$  provides the count between  $t = 0$  and  $t = t_0$ ;
- $-\underline{x}(t_0 > 0)$  provides the count between  $t = t_0$  and  $t = 0$ .

Time origin can be shifted if necessary.

From Eq. (4.32), the first-order PDF of a Poisson process is given by

$$f_{\underline{x}}(x; t) = \sum_{k=0}^{\infty} \frac{(\lambda t)^k e^{-\lambda t}}{k!} \delta(x - k). \quad (4.152)$$

Applying the definition iteratively, one can show that for  $t_1 \leq t_2 \leq \dots \leq t_L$ , the corresponding  $L$ th-order PDF is

$$f_{\underline{x}}(x_1, x_2, \dots, x_L; t_1, t_2, \dots, t_L) = \begin{cases} \sum_{k_1=0}^{\infty} \sum_{k_2=k_1}^{\infty} \cdots \sum_{k_L=k_{L-1}}^{\infty} \frac{(\lambda t_1)^{k_1}}{k_1!} \prod_{l=2}^L \frac{[\lambda(t_l-t_{l-1})]^{k_l-k_{l-1}}}{(k_l-k_{l-1})!} \\ e^{-\lambda t_L} \delta(x_1 - k_1) \delta(x_2 - k_2) \cdots \delta(x_L - k_L), & k_1 \leq k_2 \leq \dots \leq k_L; \\ 0, & \text{otherwise.} \end{cases} \quad (4.153)$$

#### 1.04.4.11 Complex random processes and sequences

Analogously to what has been done for random variables, real random processes and sequences can be generalized to handle the complex case.

A complex random process can be described as  $\underline{z}(t) = \underline{x}(t) + j\underline{y}(t)$ , where  $\underline{x}(t)$  and  $\underline{y}(t)$  are real random processes. It is stationary in some sense as long as  $\underline{x}(t)$  e  $\underline{y}(t)$  are jointly stationary in that sense. The remaining definitions related to process stationarity are kept the same.

For a complex random process  $\underline{z}(t)$ , the following definitions apply:

- the **mean**  $\bar{\underline{z}}(t) = \bar{\underline{x}}(t) + \bar{\underline{y}}(t)$ ;
- the **mean instantaneous power**  $\overline{|\underline{z}|^2}(t) = \overline{|\underline{x}|^2}(t) + \overline{|\underline{y}|^2}(t)$ ;
- the **variance**  $\sigma_{\underline{z}}^2(t) = E[(|\underline{z}(t) - \bar{\underline{z}}(t)|^2)] = \sigma_{\underline{x}}^2(t) + \sigma_{\underline{y}}^2(t)$ ;
- the **auto-correlation**  $R_{\underline{z}\underline{z}}(t_1, t_2) = E[\underline{z}^*(t_1)\underline{z}(t_2)]$ ;
- the **auto-covariance**  $C_{\underline{z}\underline{z}}(t_1, t_2) = E[(\underline{z}(t_1) - \bar{\underline{z}}(t_1))^*(\underline{z}(t_2) - \bar{\underline{z}}(t_2))]$ .

Given two random processes  $\underline{z}_1(t)$  and  $\underline{z}_2(t)$ , one defines:

- the **mean instantaneous cross power**  $E[\underline{z}_1^*(t)\underline{z}_2(t)]$ ;
- the **cross-correlation**  $R_{\underline{z}_1\underline{z}_2}(t_1, t_2) = E[\underline{z}_1^*(t_1)\underline{z}_2(t_2)]$ ;
- the **cross-covariance**  $C_{\underline{z}_1\underline{z}_2}(t_1, t_2) = E[(\underline{z}_1(t_1) - \bar{\underline{z}}_1(t_1))^*(\underline{z}_2(t_2) - \bar{\underline{z}}_2(t_2))]$ .

The processes  $\underline{z}_1(t)$  and  $\underline{z}_2(t)$  are said to be mutually:

- **orthogonal** when  $R_{\underline{z}_1\underline{z}_2}(t_1, t_2) = 0, \forall t_1 \neq t_2$ ;
- **uncorrelated** when  $C_{\underline{z}_1\underline{z}_2}(t_1, t_2) = 0$ , which is the same as  $R_{\underline{z}_1\underline{z}_2}(t_1, t_2) = \bar{\underline{z}}_1^*(t_1)\bar{\underline{z}}_2(t_2) \forall t_1 \neq t_2$ .

For a random sequence  $\underline{z}[n]$ , the following definitions apply:

- the **mean**  $\bar{\underline{z}}[n] = \bar{\underline{x}}[n] + \bar{\underline{y}}[n]$ ;
- the **mean instantaneous power**  $\overline{|\underline{z}|^2}[n] = \overline{|\underline{x}|^2}[n] + \overline{|\underline{y}|^2}[n]$ ;
- the **variance**  $\sigma_{\underline{z}}^2[n] = E[(|\underline{z}[n] - \bar{\underline{z}}[n]|^2)] = \sigma_{\underline{x}}^2[n] + \sigma_{\underline{y}}^2[n]$ ;
- the **auto-correlation**  $R_{\underline{z}\underline{z}}[n_1, n_2] = E[\underline{z}^*[n_1]\underline{z}[n_2]]$ ;
- the **auto-covariance**  $C_{\underline{z}\underline{z}}[n_1, n_2] = E[(\underline{z}[n_1] - \bar{\underline{z}}[n_1])^*(\underline{z}[n_2] - \bar{\underline{z}}[n_2])]$ .

Given two random processes  $\underline{z}_1[n]$  and  $\underline{z}_2[n]$ , one defines:

- the **mean instantaneous cross power**  $E[\underline{z}_1^*[n]\underline{z}_2[n]]$ ;
- the **cross-correlation**  $R_{\underline{z}_1\underline{z}_2}[n_1, n_2] = E[\underline{z}_1^*[n_1]\underline{z}_2[n_2]]$ ;
- the **cross-covariance**  $C_{\underline{z}_1\underline{z}_2}[n_1, n_2] = E[(\underline{z}_1[n_1] - \bar{\underline{z}}_1[n_1])^*(\underline{z}_2[n_2] - \bar{\underline{z}}_2[n_2])]$ .

The processes  $\underline{z}_1[n]$  and  $\underline{z}_2[n]$  are said to be mutually:

- **orthogonal** when  $R_{\underline{z}_1\underline{z}_2}[n_1, n_2] = 0, \forall n_1 \neq n_2$ ;
- **uncorrelated** when  $C_{\underline{z}_1\underline{z}_2}[n_1, n_2] = 0$ , which is the same as  $R_{\underline{z}_1\underline{z}_2}[n_1, n_2] = \bar{\underline{z}}_1^*[n_1]\bar{\underline{z}}_2[n_2] \forall n_1 \neq n_2$ .

### 1.04.4.12 Markov chains

The simplest random processes (sequences) are those whose statistics at a given time are independent from every other time: a first-order distribution suffices to describe them. A less trivial model is found when the statistical time interdependency assumes a special recursive behavior such that the knowledge of a past conditioning state summarizes all previous history of the process (sequence). For the so-called **Markov** random process (sequence), the knowledge of the past does not affect the expectation of the future when the present is known.

Mathematically, a random process  $\underline{x}(t)$  is said to be a Markov process when for  $t_{n-1} < t_n$ ,

$$f_{\underline{x}}(x_{t_n}; t_n | x_t; \text{all } t \leq t_{n-1}) = f_{\underline{x}}(x_{t_n}; t_n | x_{t_{n-1}}; t_{n-1}), \quad (4.154)$$

or for  $t_1 < t_2 < \dots < t_{n-1} < t_n$ ,

$$f_{\underline{x}}(x_{t_n}; t_n | x_{t_{n-1}}, \dots, x_{t_2}, x_{t_1}; t_{n-1}, \dots, t_2, t_1) = f_{\underline{x}}(x_{t_n}; t_n | x_{t_{n-1}}; t_{n-1}). \quad (4.155)$$

We will restrict ourselves to the discrete-time case. A random sequence  $\underline{x}[n]$  is said to be a Markov sequence when

$$f_{\underline{x}}(x_n; n | x_{n-1}, \dots, x_1, x_0; n-1, \dots, 1, 0) = f_X(x_n; n | x_{n-1}; n-1), \quad (4.156)$$

which can be seen as a **transition PDF**. From the definition, one arrives at the **chain rule**

$$f_{\underline{x}}(x_0, x_1, \dots, x_n; 0, 1, \dots, n) = f_{\underline{x}}(x_n; n | x_{n-1}; n-1) \cdots f_{\underline{x}}(x_1; 1 | x_0; 0) f_{\underline{x}}(x_0; 0), \quad (4.157)$$

which means that the overall Markov sequence can be statistically described for  $n \geq 0$  from the knowledge of its distribution at  $n = 0$  and its subsequent transition distributions. Some interesting properties can be deduced from the expressions above:

- Since  $f_{\underline{x}}(x_n; n | x_{n+1}, x_{n+2}, \dots, x_{n+k}; n+1, n+2, \dots, n+k) = f_{\underline{x}}(x_n; n | x_{n+1}; n+1)$ , a time-reversed Markov sequence is also a Markov sequence.
- For  $n_1 < n_2 < n_3$ ,  $f_{\underline{x}}(x_{n_1}, x_{n_3}; n_1, n_3 | x_{n_2}; n_2) = f_{\underline{x}}(x_{n_3}; n_3 | x_{n_2}; n_2) f_{\underline{x}}(x_{n_1}; n_1 | x_{n_2}; n_2)$ .

A much important property of Markov sequences is the so-called **Chapman-Kolmogorov equation**: for  $n_1 < n_2 < n_3$ ,

$$f_{\underline{x}}(x_{n_3}; n_3 | x_{n_1}; n_1) = \int_{-\infty}^{\infty} f_{\underline{x}}(x_{n_3}; n_3 | x_{n_2}; n_2) f_{\underline{x}}(x_{n_2}; n_2 | x_{n_1}; n_1) dx_{n_2}, \quad (4.158)$$

which provides a recursive way to compute arbitrary transition PDFs.

We can say a Markov sequence  $\underline{x}[n]$  is **stationary** when  $f_{\underline{x}}(x_n; n)$  and  $f_{\underline{x}}(x_n; n | x_{n-1}; n-1)$  are shift-invariant. In this case, the overall sequence can be obtained from  $f_{\underline{x}}(x_1, x_2; 1, 2)$ . A less trivial (and quite useful) model is provided by **homogeneous** Markov sequences, which are characterized only by a shift-invariant transition distribution; they are not stationary in general, but can be asymptotically stationary (i.e., for  $n \rightarrow \infty$ ) under certain conditions.

Discrete Markov processes and sequences are called **Markov chains**, which can assume a countable number of random states  $a_i$  described by their **state probabilities** ( $P(\underline{x}[n] = a_i) = p_i[n]$  for sequences) and **transition probabilities** ( $P(\underline{x}[n_2] = a_j | \underline{x}[n_1] = a_i) = \Pi_{ij}[n_1, n_2]$ ,  $n_1 < n_2$  for sequences). Discrete-time Markov chains enjoy the following properties, for  $n_1 < n_2 < n_3$ :

- $\sum_j \Pi_{ij}[n_1, n_2] = 1$ , which totalizes all possible ways to left state  $a_i$  in  $n_1$ ;
- $\sum_i p_i[n_1] \Pi_{ij}[n_1, n_2] = p_j[n_2]$ , which totalizes all possible ways to arrive at state  $a_j$  in  $n_2$ ;
- $\Pi_{il}[n_1, n_3] = \sum_j \Pi_{ij}[n_1, n_2] \Pi_{jl}[n_2, n_3]$  (**Chapman-Kolmogorov equation**).

If the chain has a finite number of states, a matrix notation can be employed. The state probability vector  $\mathbf{p}[n]$ , with elements  $\{\mathbf{p}_i[n]\} = p_i[n]$ , and the transition probability matrix  $\boldsymbol{\Pi}[n_1, n_2]$ , with elements  $\{\boldsymbol{\Pi}_{ij}[n_1, n_2]\} = \Pi_{ij}[n_1, n_2]$ , are related by  $\mathbf{p}[n_2] = \boldsymbol{\Pi}^T[n_1, n_2] \mathbf{p}[n_1]$ .

The special class of homogeneous Markov chains enjoys the following additional properties:

- $\Pi_{ij}[n_1, n_2] = \Pi_{ij}[k], k = n_2 - n_1$ ;
- $\Pi_{il}[k_2 + k_1] = \sum_j \Pi_{ij}[k_1] \Pi_{jl}[k_2]$ .

Accordingly, the transition probability matrix becomes  $\boldsymbol{\Pi}[k]$ . Defining  $\boldsymbol{\Pi}[1] = \boldsymbol{\Pi}$ ,

$$\mathbf{p}[n] = \boldsymbol{\Pi}^T \mathbf{p}[n-1] = (\boldsymbol{\Pi}^T)^n \mathbf{p}[0], \quad (4.159)$$

i.e.,  $\boldsymbol{\Pi}[n] = \boldsymbol{\Pi}^n$ . When asymptotic stationarity is reachable, one can find the **steady-state probability vector**  $\mathbf{p} = \mathbf{p}[\infty]$  such that  $\mathbf{p} = \boldsymbol{\Pi} \mathbf{p}$ .

Consider, for example, the homogeneous Markov chain  $\underline{x}[n]$  depicted in Figure 4.12, with states 1 and 2, state probabilities  $P(\underline{x}[n] = 1) = p_1[n]$  and  $P(\underline{x}[n] = 2) = p_2[n]$ , and transition probabilities  $P(\underline{x}[n] = 2 | \underline{x}[n-1] = 1) = \Pi_{12}[1] = a$  and  $P(\underline{x}[n] = 1 | \underline{x}[n-1] = 2) = \Pi_{21}[1] = b$ . Its corresponding one-sample transition matrix is

$$\boldsymbol{\Pi}[1] = \boldsymbol{\Pi} = \begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix}, \quad (4.160)$$

such that

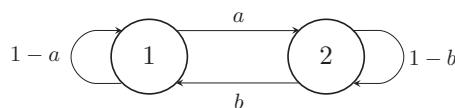
$$\boldsymbol{\Pi}[k] = \boldsymbol{\Pi}^k = \begin{bmatrix} \frac{b+a(1-a-b)^k}{a+b} & \frac{a-a(1-a-b)^k}{a+b} \\ \frac{b-b(1-a-b)^k}{a+b} & \frac{a+b(1-a-b)^k}{a+b} \end{bmatrix}. \quad (4.161)$$

It can also be shown that the chain reaches the steady-state probability vector

$$\lim_{n \rightarrow \infty} \mathbf{p}[n] = \lim_{n \rightarrow \infty} \begin{bmatrix} p_1[n] \\ p_2[n] \end{bmatrix} = \begin{bmatrix} \frac{b}{a+b} \\ \frac{a}{a+b} \end{bmatrix}. \quad (4.162)$$

#### 1.04.4.13 Spectral description of random processes and sequences

At first sight, the direct conversion of each realization  $x(t)$  ( $x[n]$ ) to the frequency domain seems the easiest way to spectrally characterize a random process  $\underline{x}(t)$  (sequence  $\underline{x}[n]$ ). However, it is difficult to



**FIGURE 4.12**

Homogeneous Markov chain with two states.

guarantee the existence of the Fourier transform

$$X(j\omega) = \mathcal{F}[x(t)] \triangleq \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt \quad (4.163)$$

(in the case of random processes) or

$$X(e^{j\Omega}) = \mathcal{F}[x[n]] \triangleq \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n} \quad (4.164)$$

(in the case of random sequences) for every realization. And even if possible, we would get a random spectrum of limited applicability. In Section 1.04.4.5, we have found that the auto-correlation conveys information about every sinusoidal component found in the process (sequence). A correct interpretation of the Fourier transform<sup>17</sup> suggests that the auto-correlation in fact conveys information about the overall spectrum of the process (sequence). Furthermore, it is a better behaved function than the individual realizations, and thus amenable to be Fourier transformed.

In Section 1.04.4.6, Eq. (4.118), we defined the overall mean power of the random process  $\underline{x}(t)$ , which for the general complex case, becomes

$$\overline{\underline{\mathcal{P}}_{xx}} = A[\mathbb{E}[|\underline{x}^2(t)|]]. \quad (4.165)$$

It can be shown that

$$\overline{\underline{\mathcal{P}}_{xx}} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lim_{T \rightarrow \infty} \frac{\mathbb{E}[|\underline{X}_T(j\omega)|^2]}{2T} d\omega, \quad (4.166)$$

where

$$\underline{X}_T(j\omega) = \mathcal{F}[\underline{x}_T(t)] \quad (4.167)$$

and

$$\underline{x}_T(t) = \begin{cases} x(t), & -T < t < T; \\ 0, & \text{otherwise.} \end{cases} \quad (4.168)$$

We can then define a **power spectral density**

$$S_{\underline{xx}}(j\omega) = \lim_{T \rightarrow \infty} \frac{\mathbb{E}[|\underline{X}_T(j\omega)|^2]}{2T}, \quad (4.169)$$

such that  $\overline{\underline{\mathcal{P}}_{xx}} = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{\underline{xx}}(j\omega) d\omega$ . Some additional algebraic manipulation yields

$$S_{\underline{xx}}(j\omega) = \mathcal{F}[A[R_{\underline{xx}}(t, t + \tau)]], \quad (4.170)$$

which directly relates the auto-correlation to the power spectral density of the process, as predicted. From the expressions above,  $S_{\underline{xx}}(j\omega)$  is a non-negative real function of  $\omega$ . Furthermore, if  $\underline{x}(t)$  is real, then  $S_{\underline{xx}}(j\omega)$  is an even function of  $\omega$ .

---

<sup>17</sup>The Fourier transform represents a time signal as a linear combination of continuously distributed sinusoids.

In Section 1.04.4.6, Eq. (4.121), we also defined the overall mean cross-power between the processes  $\underline{x}(t)$  and  $\underline{y}(t)$ , which for the general complex case, becomes

$$\overline{\mathcal{P}_{xy}} = A[\mathbb{E}[\underline{x}^*(t)\underline{y}(t)]] \quad (4.171)$$

Following the same steps as above, we arrive at

$$\overline{\mathcal{P}_{xy}} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lim_{T \rightarrow \infty} \frac{\mathbb{E}[X_T^*(j\omega)Y_T(j\omega)]}{2T} d\omega, \quad (4.172)$$

where  $X_T(j\omega)$  is defined as before,

$$Y_T(j\omega) = \mathcal{F}[\underline{y}_T(t)] \quad (4.173)$$

and

$$y_T(t) = \begin{cases} y(t), & -T < t < T; \\ 0, & \text{otherwise.} \end{cases} \quad (4.174)$$

We can then define the corresponding **cross-power spectral density**

$$S_{\underline{x}\underline{y}}(j\omega) = \lim_{T \rightarrow \infty} \frac{\mathbb{E}[X_T^*(j\omega)Y_T(j\omega)]}{2T}, \quad (4.175)$$

such that  $\overline{\mathcal{P}_{xy}} = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{\underline{x}\underline{y}}(j\omega) d\omega$ . In terms of the cross-correlation, the cross-power spectral density can be written as

$$S_{\underline{x}\underline{y}}(j\omega) = \mathcal{F}[A[R_{\underline{x}\underline{y}}(t, t + \tau)]] \quad (4.176)$$

As expected, the cross-power density of orthogonal processes is zero. From the expressions above,  $S_{\underline{x}\underline{y}}(j\omega) = S_{\underline{y}\underline{x}}^*(j\omega)$ . Furthermore, if  $x(t)$  and  $y(t)$  are real, then the real part of  $S_{\underline{x}\underline{y}}(j\omega)$  is even and the imaginary part of  $S_{\underline{x}\underline{y}}(j\omega)$  is odd.

It should be noticed that in the WSS case,  $\overline{\mathcal{P}_{xx}} = \mathbb{E}[|\underline{x}^2(t)|]$ ,  $\overline{\mathcal{P}_{xy}} = \mathbb{E}[\underline{x}^*(t)\underline{y}(t)]$ ,  $S_{\underline{x}\underline{x}}(j\omega) = \mathcal{F}[R_{\underline{x}\underline{x}}(\tau)]$ , and  $S_{\underline{x}\underline{y}}(j\omega) = \mathcal{F}[R_{\underline{x}\underline{y}}(\tau)]$ .

A similar development can be done for random sequences. In Section 1.04.4.6, Eq. (4.129), we defined the overall mean power of the random sequence  $\underline{x}[n]$ , which for the general complex case, becomes

$$\overline{\mathcal{P}_{xx}} = A[\mathbb{E}[|\underline{x}^2[n]|]] \quad (4.177)$$

It can be shown that

$$\overline{\mathcal{P}_{xx}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \lim_{N \rightarrow \infty} \frac{\mathbb{E}[|\underline{X}_N(e^{j\Omega})|^2]}{2N + 1} d\Omega, \quad (4.178)$$

where

$$\underline{X}_N(e^{j\Omega}) = \mathcal{F}[\underline{x}_N[n]] \quad (4.179)$$

and

$$x_N[n] = \begin{cases} x[n], & -N \leq n \leq N; \\ 0, & \text{otherwise.} \end{cases} \quad (4.180)$$

We can then define a **power spectral density**

$$S_{\underline{x}\underline{x}}(e^{j\Omega}) = \lim_{N \rightarrow \infty} \frac{E[|X_N(e^{j\Omega})|^2]}{2N + 1}, \quad (4.181)$$

such that  $\overline{\mathcal{P}_{xx}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{\underline{x}\underline{x}}(e^{j\Omega}) d\Omega$ . Some additional algebraic manipulation yields

$$S_{\underline{x}\underline{x}}(e^{j\Omega}) = \mathcal{F}[A[R_{\underline{x}\underline{x}}[n, n+k]]], \quad (4.182)$$

which directly relates the auto-correlation to the power spectral density of the random sequence, as expected. From the expressions above,  $S_{\underline{x}\underline{x}}(e^{j\Omega})$  is a non-negative real function of  $\Omega$ . Furthermore, if  $x[n]$  is real, then  $S_{\underline{x}\underline{x}}(e^{j\Omega})$  is an even function of  $\Omega$ .

In Section 1.04.4.6, Eq. (4.132), we also defined the overall mean cross-power between the random sequences  $\underline{x}[n]$  and  $\underline{y}[n]$ , which for the general complex case, becomes

$$\overline{\mathcal{P}_{xy}} = A[E[\underline{x}^*[n]\underline{y}[n]]]. \quad (4.183)$$

Following the same steps as above, we arrive at

$$\overline{\mathcal{P}_{xy}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \lim_{N \rightarrow \infty} \frac{E[X_N^*(e^{j\Omega})Y_N(e^{j\Omega})]}{2N + 1} d\Omega, \quad (4.184)$$

where  $X_N(e^{j\Omega})$  is defined as before,

$$Y_N(e^{j\Omega}) = \mathcal{F}[\underline{y}_N[n]] \quad (4.185)$$

and

$$y_N[n] = \begin{cases} y[n], & -N \leq n \leq N; \\ 0, & \text{otherwise.} \end{cases} \quad (4.186)$$

We can then define the corresponding **cross-power spectral density**

$$S_{\underline{x}\underline{y}}(e^{j\Omega}) = \lim_{N \rightarrow \infty} \frac{E[X_N^*(e^{j\Omega})Y_N(e^{j\Omega})]}{2N + 1}, \quad (4.187)$$

such that  $\overline{\mathcal{P}_{xy}} = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{\underline{x}\underline{y}}(e^{j\Omega}) d\Omega$ . In terms of the cross-correlation, the cross-power spectral density can be written as

$$S_{\underline{x}\underline{y}}(e^{j\Omega}) = \mathcal{F}[A[R_{\underline{x}\underline{y}}[n, n+k]]]. \quad (4.188)$$

As expected, the cross-power density of orthogonal processes is zero. From the expressions above,  $S_{\underline{x}\underline{y}}(j\omega) = S_{\underline{y}\underline{x}}^*(e^{j\Omega})$ . Furthermore, if  $\underline{x}[n]$  and  $\underline{y}[n]$  are real, then the real part of  $S_{\underline{x}\underline{y}}(e^{j\Omega})$  is even and the imaginary part of  $S_{\underline{x}\underline{y}}(e^{j\Omega})$  is odd.

It should be noticed that in the WSS case,  $\overline{\mathcal{P}_{xx}} = E[|\underline{x}[n]|^2]$ ,  $\overline{\mathcal{P}_{xy}} = E[\underline{x}^*[n]\underline{y}[n]]$ ,  $S_{\underline{x}\underline{x}}(e^{j\Omega}) = \mathcal{F}[R_{\underline{x}\underline{x}}[k]]$ , and  $S_{\underline{x}\underline{y}}(e^{j\Omega}) = \mathcal{F}[R_{\underline{x}\underline{y}}[k]]$ .

#### 1.04.4.14 White and colored noise

One usually refers to a disturbing signal  $d(t)$  ( $d[n]$ ) as noise. Since noise signals are typically unpredictable, they are preferably modeled as realizations of some noise random process  $\underline{d}(t)$  (sequence  $\underline{d}[n]$ ).

A very special case is the so-called **white noise** (by analogy with white light), which is a uniform combination of all frequencies.

Continuous-time white noise  $w(t)$  is sampled from a random process  $\underline{w}(t)$  characterized by the following properties:

- zero mean:  $\overline{\underline{w}}(t) = 0$ ;
- non-correlation between distinct time instants:  $R_{\underline{w}\underline{w}}(t, t + \tau) = 0, \tau \neq 0$ ;
- constant power spectral density:  $S_{\underline{w}\underline{w}}(j\omega) = S_{\underline{w}\underline{w}} \forall \omega$ ;
- infinite overall mean power:  $\overline{\mathcal{P}_{xx}} = \infty$ .

From the last property, continuous-time white noise is not physically realizable. Furthermore, WSS continuous-time white noise has  $R_{\underline{w}\underline{w}}(\tau) = S_{\underline{w}\underline{w}}\delta(\tau)$ .

Discrete-time white noise  $w[n]$  is sampled from a random sequence  $\underline{w}[n]$  characterized by the following properties:

- zero mean:  $\overline{\underline{w}}[n] = 0$ ;
- non-correlation between distinct time instants:  $R_{\underline{w}\underline{w}}[n, n + k] = 0, k \neq 0$ ;
- constant power spectral density:  $S_{\underline{w}\underline{w}}(e^{j\Omega}) = A[\sigma_{\underline{w}}^2[n]]$ ;
- overall mean power  $\overline{\mathcal{P}_{xx}} = A[\sigma_{\underline{w}}^2[n]]$ .

For WSS discrete-time white noise,  $S_{\underline{w}\underline{w}}(e^{j\Omega}) = \sigma_{\underline{w}}^2$ ,  $R_{\underline{w}\underline{w}}[k] = \sigma_{\underline{w}}^2\delta[k]$ , and  $\overline{\mathcal{P}_{xx}} = \sigma_{\underline{w}}^2$ .

Unless differently stated, white noise is implicitly assumed to be generated by a WSS random process (sequence). Notice, however, that the sequence  $d[n] = w(n) \cos(\Omega_0 n)$ , where  $0 < \Omega_0 < \pi$  rad and  $\underline{w}(n)$  is WSS white noise with unit variance, for example, satisfies all conditions to be called white noise, even if  $R_{\underline{d}\underline{d}}[n, n + k] = \delta[k] \cos(\Omega_0 n)$ .

A common (more stringent) model of white noise imposes that values at different time instants of the underlying process (sequence) be statistically i.i.d.

Any random noise whose associated power spectral density is not constant is said to be **colored noise**. One can find in the literature several identified colored noises (pink, grey, etc.), each one with a pre-specified spectral behavior. It should also be noticed that any band-limited approximation of white noise destroys its non-correlation property. Consider  $d(t)$  generated by a WSS random process  $\underline{d}(t)$  such that

$$S_{\underline{d}\underline{d}}(j\omega) = \begin{cases} \frac{P\pi}{W}, & -W < \omega < W; \\ 0, & \text{otherwise.} \end{cases} \quad (4.189)$$

It is common practice to call  $d(t)$  “white noise” of bandwidth  $W$  and overall mean power  $P$ . Since its auto-correlation is

$$R_{\underline{d}\underline{d}}(\tau) = \frac{P \sin(W\tau)}{W\tau}, \quad (4.190)$$

strictly speaking one cannot say  $d(t)$  is white noise.

#### 1.04.4.15 Applications: modulation, “Bandpass” and band-limited processes, and sampling

An interesting application of the spectral description of random processes is modeling of AM (amplitude modulation). Assuming a carrier signal  $c(t) = A_0 \cos(\omega_0 t)$  modulated by a real random signal  $m(t)$  sampled from a process  $\underline{m}(t)$ , the resulting modulated random process  $\underline{x}(t) = \underline{m}(t)A_0 \cos(\omega_0 t)$  has auto-correlation

$$R_{\underline{x}\underline{x}}(t, t + \tau) = \frac{A_0^2}{2} R_{\underline{m}\underline{m}}(t, t + \tau) [\cos(\omega_0 \tau) + \cos(2\omega_0 t + \omega_0 \tau)]. \quad (4.191)$$

If  $\underline{m}(t)$  is WSS, then

$$A[R_{\underline{x}\underline{x}}(t, t + \tau)] = \frac{A_0^2}{2} R_{\underline{m}\underline{m}}(\tau) \cos(\omega_0 \tau), \quad (4.192)$$

and the corresponding power spectral density is

$$S_{\underline{x}\underline{x}}(j\omega) = \frac{A_0^2}{4} [S_{\underline{m}\underline{m}}(j(\omega - \omega_0)) + S_{\underline{m}\underline{m}}(j(\omega + \omega_0))]. \quad (4.193)$$

We conclude that the AM constitution of  $\underline{x}(t)$  carries through its auto-correlation, which provides a simple statistical model for AM.

Quite often we find ourselves dealing with **band-limited** random signals. In the AM discussion above, typically  $\underline{m}(t)$  has bandwidth  $W \ll \omega_0$ . For a random process  $\underline{x}(t)$  whose spectrum is at least concentrated around  $\omega = 0$  (**baseband** process), we can attribute it an **RMS** (root-mean-squared) **bandwidth**  $W_{\text{RMS}}$  such that

$$W_{\text{RMS}}^2 = \frac{\int_{-\infty}^{\infty} \omega^2 S_{\underline{x}\underline{x}}(j\omega) d\omega}{\int_{-\infty}^{\infty} S_{\underline{x}\underline{x}}(j\omega) d\omega}. \quad (4.194)$$

If the spectrum is concentrated around a centroid

$$\bar{\omega}_0 = \frac{\int_0^{\infty} \omega S_{\underline{x}\underline{x}}(j\omega) d\omega}{\int_0^{\infty} S_{\underline{x}\underline{x}}(j\omega) d\omega}, \quad (4.195)$$

$W_{\text{RMS}}$  is given by

$$\left(\frac{W_{\text{RMS}}}{2}\right)^2 = \frac{\int_0^{\infty} (\omega - \bar{\omega}_0)^2 S_{\underline{x}\underline{x}}(j\omega) d\omega}{\int_0^{\infty} S_{\underline{x}\underline{x}}(j\omega) d\omega}. \quad (4.196)$$

If the process bandwidth excludes  $\omega = 0$ , it is called<sup>18</sup> a “**bandpass**” process.

It can be shown that if a band-limited baseband WSS random process  $\underline{x}(t)$  with bandwidth  $W$ , i.e., such that  $S_{\underline{x}\underline{x}}(j\omega) = 0$  for  $|\omega| > W$ , is sampled at rate  $\omega_s > 2W$  to generate the random sequence

$$\underline{x}[n] = \underline{x}\left(n \frac{2\pi}{\omega_s}\right), \quad (4.197)$$

then  $\underline{x}(t)$  can be recovered from  $\underline{x}[n]$  with zero mean-squared-error. This is the stochastic version of the **Nyquist criterion** for lossless sampling.

---

<sup>18</sup>With considerable freedom of nomenclature.

#### 1.04.4.16 Processing of random processes and sequences

Statistical signal modeling is often employed in the context of signal processing, and thus the interaction between random signals and processing systems calls for a systematic approach. In this chapter, we will restrict ourselves to **linear time-invariant** (LTI) systems.

A system defined by the operation  $y = L[x]$  between input  $x$  and output  $y$  is called **linear** if any linear combination of  $m$  inputs produce as output the same linear combination of their  $m$  corresponding outputs:

- for a continuous-time system,  $L\left[\sum_{m=1}^M \alpha_m x_m(t)\right] = \sum_{m=1}^M \alpha_m L[x_m(t)], \alpha_m \in \mathcal{C}$ ,
- for a discrete-time system,  $L\left[\sum_{m=1}^M \alpha_m x_m[n]\right] = \sum_{m=1}^M \alpha_m L[x_m[n]], \alpha_m \in \mathcal{C}$ .

For a linear system, one can define an **impulse response**  $h$  as the system output to a unit impulse  $\delta$  applied to the system input at a given time instant:

- for a continuous-time system, a unit impulse  $\delta(\cdot)$  applied at instant  $\tau$ , produces an impulse response  $h_\tau(t) = L[\delta(t - \tau)]$ ,
- for a discrete-time system, a unit impulse<sup>19</sup>  $\delta[\cdot]$  applied at instant  $k$ , produces an impulse response  $h_k[n] = L[\delta[n - k]]$ .

A system is called **time-invariant** if a given input  $x$  applied at different time instants produces the same output  $y$  accordingly time-shifted:

- for a continuous-time system, if  $L[x(t)] = y(t)$ , then  $L[x(t - \tau)] = y(t - \tau) \forall \tau$ ; thus,  $h_\tau(t) = h_0(t - \tau) \triangleq h(t - \tau)$ ,
- for a discrete-time system, if  $L[x[n]] = y[n]$ , then  $L[x[n - k]] = y[n - k] \forall k$ ; thus,  $h_\tau[n] = h_0[n - k] \triangleq h[n - k]$ .

The output of an LTI system to any input  $x$  is the **convolution**  $x * h$  between the input and the impulse response  $h$  of the system:

- for a continuous-time system,  $L[x(t)] = \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau \triangleq (x * h)(t)$ ,
- for a discrete-time system,  $L[x[n]] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] \triangleq (x * h)[n]$ .

In the frequency domain, the output  $Y$  is related to the input  $X$  by the system **frequency response**  $H$ :

- for a continuous-time system,  $Y(j\omega) = X(j\omega)H(j\omega), H(j\omega) = \mathcal{F}[h(t)]$ ,
- for a discrete-time system,  $Y(e^{j\Omega}) = X(e^{j\Omega})H(e^{j\Omega}), H(e^{j\Omega}) = \mathcal{F}[h[n]]$ .

The product  $XH$  brings the concept of **filtering**:  $H$  defines how much of each frequency component of input  $X$  passes to the output  $Y$ . From now on, when referring to the processing of a random process (sequence) by a system we imply that each process realization is filtered by that system.

The filtering of a random process  $\underline{x}(t)$  by an LTI system with impulse response  $h(t)$  results in another random process  $\underline{y}(t)$  such that  $\underline{y}(t) = (\underline{x} * h)(t)$ . Assuming  $\underline{x}(t)$  WSS and possibly complex, so as  $h(t)$ , then  $\underline{y}(t)$  will be also WSS and<sup>20</sup>:

- $\underline{\bar{y}} = \underline{\bar{x}}H(j0)$ ;

---

<sup>19</sup>Unit impulse function, or Kronecker delta: For  $x \in \mathcal{Z}$ ,  $\delta[x] = 0 \forall x \neq 0$ , and  $\delta[0] = 1$ .

<sup>20</sup>Where notation  $f_{-}(t) \triangleq f(-t)$  was used.

- $R_{\underline{xy}}(\tau) = R_{\underline{yx}}^*(-\tau) = (R_{\underline{xx}} * h)(\tau);$
- $R_{\underline{yy}}(\tau) = (R_{\underline{xy}} * h_-^*)(\tau) = (R_{\underline{xx}} * h * h_-^*)(\tau);$
- $S_{\underline{xy}}(j\omega) = S_{\underline{yx}}^*(j\omega) = S_{\underline{xx}}(j\omega)H(j\omega);$
- $S_{\underline{yy}}(j\omega) = S_{\underline{xy}}(j\omega)H^*(j\omega) = S_{\underline{xx}}(j\omega)|H(j\omega)|^2.$

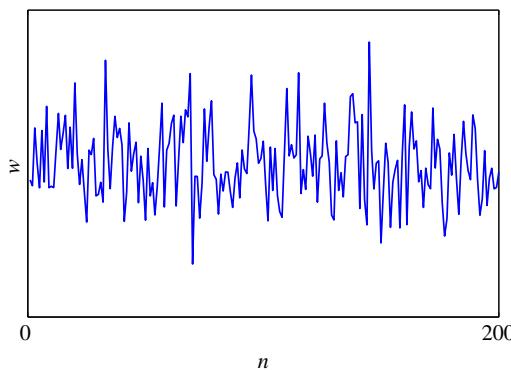
The processing of a random sequence  $\underline{x}[n]$  by an LTI system with impulse response  $h[n]$  results in another random sequence  $\underline{y}[n]$  such that  $\underline{y}[n] = (\underline{x} * h)[n]$ . Assuming  $\underline{x}[n]$  WSS and possibly complex, so as  $h[n]$ , then  $\underline{y}[n]$  will be also WSS and<sup>21</sup>:

- $\bar{\underline{y}} = \bar{\underline{x}}H(e^{j0});$
- $R_{\underline{xy}}[k] = R_{\underline{yx}}^*[-k] = (R_{\underline{xx}} * h)[k];$
- $R_{\underline{yy}}[k] = (R_{\underline{xy}} * h_-^*)[k] = (R_{\underline{xx}} * h * h_-^*)[k];$
- $S_{\underline{xy}}(e^{j\Omega}) = S_{\underline{yx}}^*(e^{j\Omega}) = S_{\underline{xx}}(e^{j\Omega})H(e^{j\Omega});$
- $S_{\underline{yy}}(e^{j\Omega}) = S_{\underline{xy}}(e^{j\Omega})H^*(e^{j\Omega}) = S_{\underline{xx}}(e^{j\Omega})|H(e^{j\Omega})|^2.$

Among the above results, one of the most important is the effect of filtering on the power spectral density  $S_{\underline{xx}}$  of the input WSS random process (sequence): it is multiplied by the squared magnitude  $|H|^2$  of the frequency response of the LTI system to produce the power spectral density  $S_{\underline{yy}}$  of the output WSS random process (sequence). As a direct consequence, any WSS random process (sequence) with known power spectral density  $S$  can be modeled by the output of an LTI system with squared magnitude response  $|H|^2 \propto S$  whose input is white noise.

In order to qualitatively illustrate this filtering effect:

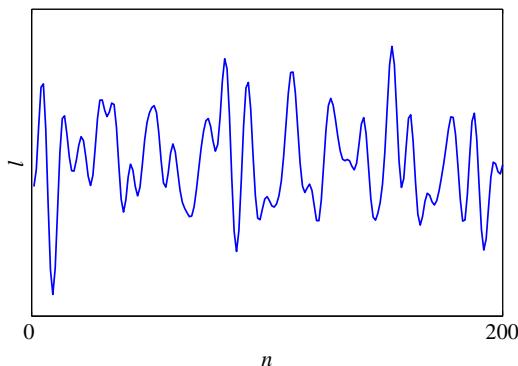
- Figure 4.13 shows 200 samples of white noise  $w[n]$ ;



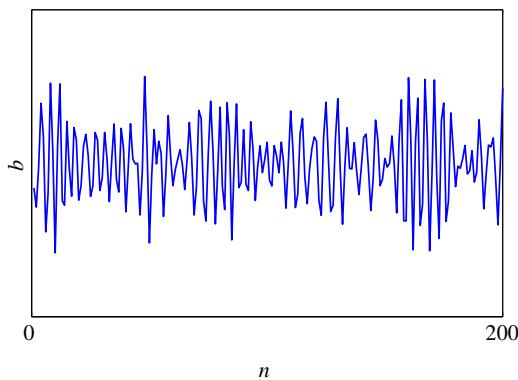
**FIGURE 4.13**

White noise.

<sup>21</sup>Where notation  $f_-[-n] \triangleq f[-n]$  was used.

**FIGURE 4.14**

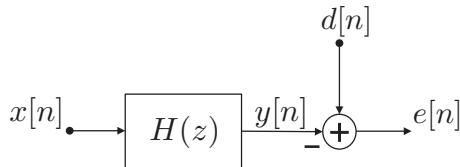
Realization of a “low-pass” sequence.

**FIGURE 4.15**

Realization of a “band-pass” sequence.

- Figure 4.14 shows 200 samples of  $l[n]$ , a slowly evolving signal obtained by low-pass filtering of  $w[n]$  to 20% of its original bandwidth;
- Figure 4.15 shows 200 samples of  $b[n]$ , an almost sinusoidal signal obtained by band-pass filtering of  $w[n]$  to the central 4% of its original bandwidth.

From the area of **optimum filtering**, whose overall goal is finding the best filter to perform a defined task, we can bring an important application of the concepts presented in this section. The general discrete **Wiener filter  $H$** , illustrated in Figure 4.16, is the linear time-invariant filter which minimizes the mean quadratic value  $E[e^2[n]]$  of the error  $e[n] = d[n] - y[n]$  between the desired signal  $d[n]$  and the filtered version  $y[n]$  of the input signal  $x[n]$ . Assuming that  $x[n]$  and  $d[n]$  are realizations of two jointly WSS random sequences  $\underline{x}[n]$  and  $\underline{d}[n]$ , if the optimum filter is not constrained to be causal, one finds that it

**FIGURE 4.16**

General discrete Wiener filter.

must satisfy the equation

$$(h * R_{\underline{xx}})[k] = R_{\underline{xd}}[k], \quad (4.198)$$

i.e., its frequency response is

$$H(e^{j\Omega}) = \frac{S_{\underline{xd}}(e^{j\Omega})}{S_{\underline{xx}}(e^{j\Omega})}. \quad (4.199)$$

If one needs a causal FIR solution, a different but equally simple solution can be found for  $h[n]$ .

We can solve a very simple example where we compute an optimum zero-order predictor for a given signal of which a noisy version is available. In this case,  $H(e^{j\Omega}) = c$  (a simple gain to be determined), the input  $x[n] = s[n] + v[n]$  (signal  $s[n]$  additively corrupted by noise  $v[n]$ ) and  $d[n] = x[n+1]$ . The solution is

$$c = \frac{R_{\underline{xx}}[1]}{R_{\underline{xx}}[0] + R_{\underline{vv}}[0]}, \quad (4.200)$$

which yields the minimum mean quadratic error

$$\mathbb{E}[e^2[n]] = \frac{R_{\underline{xx}}^2[0] + R_{\underline{xx}}[0]R_{\underline{vv}}[0] - R_{\underline{xx}}^2[1]}{R_{\underline{xx}}[0] + R_{\underline{vv}}[0]}. \quad (4.201)$$

Notice that if no noise is present,

$$c = \frac{R_{\underline{xx}}[1]}{R_{\underline{xx}}[0]} \quad (4.202)$$

and

$$\mathbb{E}[e^2[n]] = \frac{R_{\underline{xx}}^2[0] - R_{\underline{xx}}^2[1]}{R_{\underline{xx}}[0] + R_{\underline{vv}}[0]}. \quad (4.203)$$

The reader should be aware that this example studies a theoretical probabilistic model, which may therefore depend on several parameters which probably would not be available in practice and should rather be estimated. In fact, it points out the solution a practical system should pursue.

The Wiener filter is a kind of benchmark for adaptive filters [7], which optimize themselves recursively.

#### 1.04.4.17 Characterization of LTI systems and WSS random processes

A direct way to find the impulse response  $h(t)$  of an LTI system is to apply white noise  $\underline{w}(t)$  with power spectral density  $S_{\underline{w}\underline{w}}(\omega) = S_{\underline{w}\underline{w}}$   $\forall \omega$  to the system input, which yields  $R_{\underline{x}\underline{y}}(\tau) = S_{\underline{w}\underline{w}}h(\tau)$ . From an estimate  $\widehat{R}_{\underline{x}\underline{y}}(t)$  of the cross-correlation, one can obtain an estimate of the desired impulse response:  $\hat{h}(t) \approx \widehat{R}_{\underline{x}\underline{y}}(t)$ . In practice, assuming ergodicity, the following approximations are employed:

- $x(t) \approx \delta(t)$ ;
- $\widehat{R}_{\underline{x}\underline{y}}(\tau) = \widehat{\mathcal{R}_{xy}}(\tau) = \frac{1}{2T} \int_{-T}^T x(t)y(t + \tau)dt$  for sufficiently long  $T$ .

Furthermore, the operations are often performed in the discrete-time domain.

The effective bandwidth of an LTI system with frequency response  $H(j\omega)$  can be estimated by its **noise bandwidth**  $W_w$ . For a low-pass system, one looks for an equivalent ideal low-pass filter with bandwidth  $\pm W_w$  around  $\omega = 0$  with bandpass gain  $H(0)$ ;  $W_w$  is computed to guarantee that both systems deliver the same mean output power when the same white noise is applied to their both inputs. A straightforward algebraic development yields

$$W_w = \frac{\int_0^\infty |H(j\omega)|^2 d\omega}{|H(0)|^2}. \quad (4.204)$$

For a band-pass system with magnitude response centered about  $\omega_0$ , one looks for an ideal band-pass filter equivalent with bandwidth  $W_w$  around  $\pm\omega_0$  with bandpass gain  $H(j\omega_0)$ ;  $W_w$  is computed to guarantee that both systems deliver the same mean output power when the same equal white noise is applied to their both inputs. For this case,

$$W_w = \frac{\int_0^\infty |H(j\omega)|^2 d\omega}{|H(\omega_0)|^2}. \quad (4.205)$$

A real “bandpass” random process (see Section 1.04.4.15) about center frequency  $\omega_0$  can be expressed as  $\underline{n}(t) = \underline{m}(t) \cos[\omega_0 t + \underline{\theta}(t)]$ , where the envelope  $\underline{m}(t)$  and the phase  $\underline{\theta}(t)$  are both baseband random processes. We can write

$$\underline{n}(t) = \underline{n}_x(t) \cos(\omega_0 t) - \underline{n}_y(t) \sin(\omega_0 t), \quad (4.206)$$

where  $\underline{n}_x(t) = \underline{n} \cos(\underline{\theta}(t))$  and  $\underline{n}_y(t) = \underline{n} \sin(\underline{\theta}(t))$  are the quadrature components of  $\underline{n}(t)$ . In the typical situation where  $\underline{n}(t)$  is a zero-mean WSS random process with auto-correlation  $R_{nn}(\tau)$ , we can explicitly find the first and second-order moments of the quadrature components that validate the model. First define the **Hilbert transform** of a given signal  $n(t)$  as  $\hat{n}(t) = (n * h)(t)$ , where

$$H(j\omega) = \mathcal{F}[h(t)] = \begin{cases} -j, & \omega > 0; \\ j, & \omega < 0. \end{cases} \quad (4.207)$$

From  $R_{nn}(\tau)$ , we can find  $R_{\hat{n}\hat{n}}(\tau)$  and proceed to write:

- $\overline{\underline{n}_x} = \overline{\underline{n}_y} = 0$ ;
- $R_{\underline{n}_x \underline{n}_x}(\tau) = R_{\underline{n}_y \underline{n}_y}(\tau) = R_{nn}(\tau) \cos(\omega_0 \tau) + R_{\hat{n}\hat{n}}(\tau) \sin(\omega_0 \tau)$ ;
- $R_{\underline{n}_x \underline{n}_y}(\tau) = -R_{\underline{n}_y \underline{n}_x}(\tau) = R_{nn}(\tau) \sin(\omega_0 \tau) - R_{\hat{n}\hat{n}}(\tau) \cos(\omega_0 \tau)$ .

In the frequency domain, if we define

$$S_{\underline{n}\underline{n}}^+(\omega) \triangleq \begin{cases} S_{nn}(\omega), & \omega > 0 \\ 0, & \omega \leq 0, \end{cases} \quad (4.208)$$

then

- $S_{\underline{n}_x\underline{n}_x}(\omega) = S_{\underline{n}_y\underline{n}_y}(\omega) = S_{nn}^+(\omega + \omega_0) + S_{nn}^+(\omega - \omega + \omega_0);$
- $S_{\underline{n}_x\underline{n}_y}(\omega) = -S_{\underline{n}_y\underline{n}_x}(\omega) = j[S_{nn}^+(\omega + \omega_0) - S_{nn}^+(\omega - \omega + \omega_0)].$

In the particular case where  $\underline{n}(t)$  is Gaussian, it can be shown that  $\underline{n}_x(t)$  and  $\underline{n}_y(t)$  are Gaussian, and thus completely defined by their first- and second-order moments above.

#### 1.04.4.18 Statistical modeling of signals: random sequence as the output of an LTI system

A discrete-time signal  $x[n]$  with spectrum  $X(e^{j\Omega})$  can be modeled as the output of a linear time-invariant system whose frequency response is equal to  $X(e^{j\Omega})$  when excited by a unit impulse  $\delta[n]$ . In this case, the impulse response of the system is expected to be  $h[n] = x[n]$ .

In the context of statistic models, an analogous model can be built: now, we look for a linear time-invariant system which produces at its output the WSS random sequence  $\underline{x}[n]$  with power spectral density  $S_{\underline{x}\underline{x}}(e^{j\Omega})$ , when excited by a unit-variance random sequence  $\underline{w}[n]$  of white noise. As before, the frequency response of the system alone is expected to shape the output spectrum, yet this time in the mean power sense, i.e., the system must be such that  $S_{\underline{x}\underline{x}}(e^{j\Omega}) = |H(e^{j\Omega})|^2$ .

Assume the overall modeling system is described by the difference equation

$$x[n] = \sum_{k=1}^p a_p[k]x[n-k] + \sum_{k=0}^q b_q[k]w[n-k], \quad (4.209)$$

where  $w[n]$  is white noise with  $S_{ww}(e^{j\Omega}) = 1$ . The corresponding transfer function is

$$H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^q b_q[k]z^{-k}}{1 + \sum_{k=1}^p a_p[k]z^{-k}}. \quad (4.210)$$

The output  $\underline{x}[n]$  of this general system model with  $\underline{w}[n]$  at its input is called **ARMA (auto-regressive moving-average<sup>22</sup>) process** of order  $(p, q)$ . It can be described by the following difference equation in terms of its auto-correlation:

$$R_{\underline{x}\underline{x}}[k] = \sum_{l=1}^p a_p[l]R_{\underline{x}\underline{x}}[k-l] = \begin{cases} \sum_{l=0}^{q-k} b_k[l+k]h^*[l], & 0 \leq k \leq q; \\ 0, & k > q. \end{cases} \quad (4.211)$$

---

<sup>22</sup>Auto-regressive for its output feedback, moving average for its weighted sum of past input samples.

The values of  $R_{xx}[k]$  for  $k < 0$  can be easily found by symmetry.

If one restricts the system to be FIR (i.e., to have a finite-duration impulse response), then  $p = 0$ , i.e.,  $a_p[k] = 0$  for  $1 \leq k \leq p$ , and

$$H(z) = \sum_{k=0}^q b_q[k]z^{-k}. \quad (4.212)$$

The output of this “all-zero”<sup>23</sup> system model with  $\underline{w}[n]$  at its input is called an **MA (moving-average) process** of order  $q$ . Its auto-correlation can be found to be

$$R_{xx}[k] = \sum_{l=0}^{q-|k|} b_k[l + |k|]b_q^*[l]. \quad (4.213)$$

In this case,  $R_{xx}[k] = 0$  for  $k < -q$  or  $k > q$ . This model is more suited for modeling notches in the random sequence power spectrum.

If one restricts the system to be “all-pole,”<sup>24</sup> then  $q = 0$ , i.e.,  $b_q[k] = 0$  for  $1 \leq k \leq p$ , and

$$H(z) = \frac{b[0]}{1 + \sum_{k=1}^p a_p[k]z^{-k}}. \quad (4.214)$$

The output of this system model with  $\underline{w}[n]$  at its input is called an **AR (auto-regressive) process** of order  $p$ . Its auto-correlation follows the difference equation below:

$$R_{xx}[k] = \sum_{l=1}^p a_p[l]R_{xx}[k-l] = |b[0]|^2\delta[k]. \quad (4.215)$$

Again, the values of  $R_{xx}[k]$  for  $k < 0$  can be easily found by symmetry. This model is more suited for modeling peaks in the random sequence power spectrum, which is typical of quasi-periodic signals (as audio signals, for example). It should be noticed that, differently from the ARMA and MA cases, the equation for the AR process auto-correlation is linear in the system coefficients, which makes their estimation easier. If  $R_{xx}[k]$  is known for  $0 \leq k \leq p$ , and recalling that  $R_{xx}[-k] = R_{xx}^*[k]$ , one can:

- solve the  $p$ th-order linear equation system obtained by substituting  $k = 1, 2, \dots, p$  in Eq. (4.215) to find  $a_p[k]$ ,  $1 \leq k \leq p$ ;
- compute  $|b[0]|$  from Eq. (4.215) with  $k = 0$ .

All-pole modeling of audio signals is extensively used in restoration systems [8].

---

<sup>23</sup>This is a misname, since the poles of the system are at the origin. In fact, the zeros are the only responsible for shaping the frequency response of the system.

<sup>24</sup>This is another misname, since the zeros of the system are at the origin. In fact, the poles are the only responsible for shaping the frequency response of the system.

## Acknowledgments

The author thanks Dr. Paulo A.A. Esquef for carefully reviewing this manuscript and Leonardo de O. Nunes for kindly preparing the illustrations.

*Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 2](#) Continuous-Time Signals and Systems

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

---

## References

- [1] R. Deep, Probability and Statistics, Academic, San Diego, 2006.
- [2] G. Casella, R.L. Berger, Statistical Inference, second ed., Duxbury, Pacific Grove, 2001.
- [3] A. Papoulis, P. Unnikrishna, Probability, Random Variables and Stochastic Processes, fourth ed., McGraw-Hill, New York, 2002.
- [4] P.Z. Peebles Jr., Probability, Random Variables and Random Signal Principles, fourth ed., McGraw-Hill, New York, 2001.
- [5] K.S. Shanmugan, A.M. Breipohl, Random Signals: Detection, Estimation and Data Analysis, Wiley, New York, 1988.
- [6] M.H. Hayes, Statistical Digital Signal Processing and Modeling, Wiley, Hoboken, 1996.
- [7] P.S.R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation, Springer, New York, 2012.
- [8] S.J. Godsill, J.W. Rayner, Digital Audio Restoration: A Statistical Model Based Approach, Springer, London, 1988.

# Sampling and Quantization

# 5

Håkan Johansson

*Division of Electronics Systems, Department of Electrical Engineering Linköping University, Sweden*

## 1.05.1 Introduction

The shift from analog signal processing (ASP) to digital signal processing (DSP) lies behind the tremendous development of information processing systems that are used today in practically all areas one can think of. For example, DSP has enabled the fast growth of mobile communication systems and has opened up for new sophisticated medical aids, just to mention a couple of important applications. The foremost advantages of DSP over ASP are its robustness and that it can perform functions with arbitrary precision. This is because arithmetic operations in the digital domain can be done with as small numerical errors as desired by simply increasing the resolution, i.e., by allocating more bits for the operands. This can always be done as the resolution in DSP systems is independent of physical variations, like temperature changes and processing inaccuracies etc., and is in contrast to ASP systems which are highly dependent on such variations. The trend during the past decades has therefore been to use as little ASP as possible and to replace analog functions with the corresponding digital functions. This trend has been sped up by the fast development of integrated circuit techniques and progress in the implementation of digital systems. Specifically, the introduction of cost-efficient signal processors and so called application-specific integrated circuits (ASICs) has enabled low-cost implementation and manufacturing of digital systems. The real world is however analog by nature which means that the need for ASP cannot be eliminated completely. In particular, it will always be necessary to use analog-to-digital converters (abbreviated A/D converters or ADCs) and digital-to-analog converters (abbreviated D/A converters or DACs) for interfacing the two realms.

Evidently, ADCs and DACs are crucial components in virtually all practical applications today. The fundamental signal processing operations in these components are sampling and quantization (in ADCs), and signal reconstruction (in DACs). This chapter considers the basic principles of these operations as detailed below.

### 1.05.1.1 Scope and prerequisites

The theory of sampling has a long history [1–4], and the literature is abundant, see for example [5–7] and references therein. In this chapter, we are only able to cover some parts of this field. The emphasis is on basic principles and analysis methods for uniform sampling and quantization which are the most common sampling and quantization schemes used in conventional ADCs and DACs. Further, we will

only consider linear models of the sampling and quantization processes. Practical ADCs and DACs exhibit (small) nonlinearities which also need to be analyzed in practice, but it is beyond the scope of this chapter to cover also this part. In other words, this chapter concentrates on the underlying principles of uniform sampling and quantization rather than fine implementation details. For such details, we refer to books on ADCs and DACs [8]. Furthermore, we take an “engineering approach” in that we concentrate on explaining the concepts that are sufficient for understanding and analyzing practical signals and systems. Some of the fine mathematical details are therefore sometimes left out.

It is assumed that the reader has a basic knowledge of continuous-time and discrete-time signals and systems as well as the Fourier, Laplace, and  $z$ -transforms. Some of the basics regarding signals and systems are however briefly recapitulated in Section 1.05.2. Further, Section 1.05.4 assumes that the reader is familiar with stochastic processes. Most of the material in the other sections can however be understood without reading Section 1.05.4.

### 1.05.1.2 Chapter outline

After this introductory section, Section 1.05.2 gives some basic concepts. Then, Section 1.05.3 discusses the basics of uniform sampling and reconstruction of deterministic signals. The content of this section is typically also found in introductory books in signal processing [9,10]. Section 1.05.4 considers the extension to sampling and reconstruction of stochastic processes (also called random processes). In introductory books in signal processing, this is typically treated very briefly or not at all.

Section 1.05.5 considers an application referred to as time-interleaved ADCs, which is a scheme that has been introduced for high-speed A/D conversion [11]. The section discusses the principles and signal reconstruction methods which are required in this scheme due to analog channel mismatches. Without correction, this scheme corresponds to nonuniform sampling (or generalizations), but the overall goal, with correction, is to achieve uniform sampling.

Section 1.05.6, covers the principles of quantization, both in ADCs and internally in digital systems, as they are closely related. In Section 1.05.7, oversampled ADCs and DACs are briefly discussed. Oversampling techniques are used in practice to, e.g., increase the performance and relax the requirements of the analog components. Finally, Section 1.05.8 presents a method for discrete-time modeling of mixed-signal systems. This is useful in systems with feedback, like so called  $\Sigma\Delta$ -modulator-based ADCs. Such converters can reach a very high resolution using oversampling together with noise shaping [12,13].

Finally, it is pointed out that, throughout the chapter, real-valued signals are used in the examples. However, the basic principles to be presented are valid for complex signals as well.

### 1.05.1.3 Current trends

The research and development of data converters have been quite intensive for several decades, both in academia and industry. This interest is foreseen to continue for many years to come as the requirements on the signal processing systems and thus the converters continuously increase. To exemplify, it was envisaged in [14] that an ADC capable of bandwidths of more than 100 MS/s and 17 bits resolution is required in fourth-generation communication systems. This is beyond the state-of-the art ADCs which can manage only some 13–14 effective bits of resolution at such sampling rates [15–17]. At the same time, it is also desirable to decrease the size and power consumption of the converters. To enable the

high-performance applications of tomorrow, it is therefore vital to continue to conduct research on new principles of data conversions as well their practical implementations.

There are different paths one can take to increase the performance. One way is to make use of parallel converters like time-interleaved ADCs [11] (to be discussed in Section 1.05.5) and/or parallel  $\Sigma\Delta$  ADCs [18]. Such converters have great potentials but due to analog channel mismatches they also introduce errors that must be compensated for. A recent trend here is “digitally assisted analog circuits,” where digital circuitry is added to correct for analog errors [19]. With the increasing computational capacity of digital systems, one can expect that this trend will continue. The digital parts may however be computationally excessive, and it is therefore vital to devise new algorithms to reduce the complexity.

Another trend is that of sub-Nyquist sampling covering so called sparse sampling and compressed sampling (also called compressed sensing) [20–22]. This is different from the traditional Nyquist-sampled systems in that additional knowledge of the input signal is taken into account. Specifically, if the signal is sparse in some sense, one can reduce the sampling rate and thereby relax the requirements on the converters. One scenario is for applications where the signals are locally narrow-band (in time) but globally wide-band. Conventionally, high-speed converters are then needed. Utilizing the sparse or compressed sampling paradigm, the sampling rate can be reduced substantially. However, at this point, mainly the concepts have been presented and analyzed for some specific cases and classes of signals. It is an open question how well such principles will work in practical implementations.

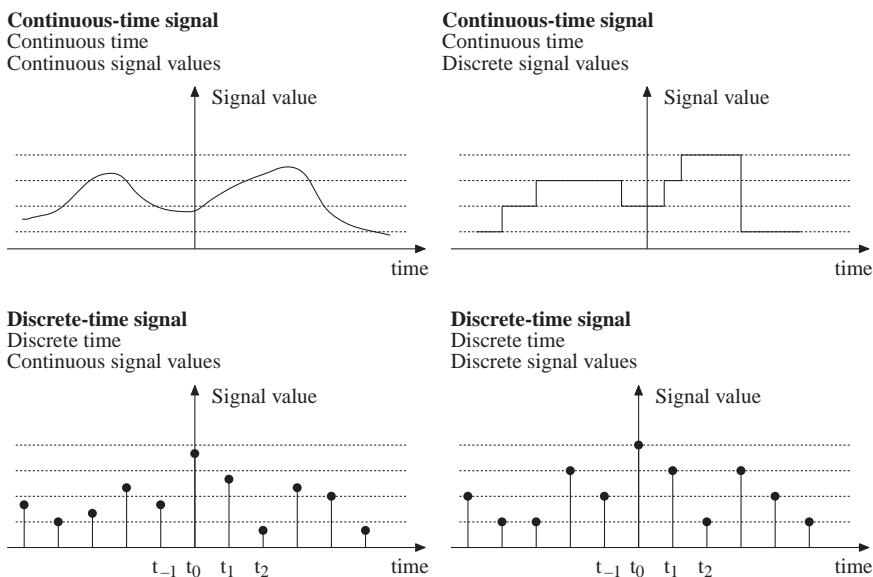
## 1.05.2 Preliminaries

### 1.05.2.1 Classification of signals

We communicate by conveying information (messages), like speech and images. In some contexts, one wishes to transmit information from one place to another, like in telephone systems. In other cases, it may be desired to store the information for later use, e.g., music stored on CDs.

Information can be transmitted using information bearing signals. A signal is a quantity that varies with time or other independent variables. Usually, the information must be converted to a desired signal type. For example, when we speak in a microphone, the generated sound is converted to an electrical voltage that varies with time. The sound can then be recovered by connecting the voltage to a loudspeaker. (In this context, however, we usually do not consider the sound as being recovered since it, in principal, is reconstructed at the same time as it is generated.) The same information can be represented in many different ways. Different types of signals can in other words contain exactly the same information. For example, speech and music can be represented in analog form on LPs and in digital form on CDs. The corresponding signals are then analog and digital, respectively.

In mathematical form, a signal is represented as a function (or sequence) of one or several independent variables. The signal is a one-dimensional signal if it is a function of one variable and a multi-dimensional signal if it is a function of several variables. This chapter considers one-dimensional signals only. An example of such signals is a speech signal. It is customary to let the independent variable represent time although it may represent whatever we wish, like temperature, pressure, etc. The time variable in the representation of a signal can be either continuous or discrete (quantized). The signal is a continuous-time signal if the variable is continuous and a discrete-time signal if the variable is discrete. A continuous-time signal is thus defined for all time instances (except possibly at discontinuities) whereas a

**FIGURE 5.1**

Different types of signals.

discrete-time signal is defined only at discrete time instances. We also differ between signals that can take on continuous signal values and those that only can take on discrete signal values. Consequently, signals can be divided into four different categories according to Figure 5.1.

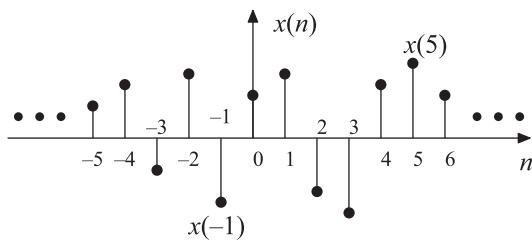
In the literature, a continuous-time signal with continuous signal values is often referred to as an analog signal, whereas a discrete-time signal with discrete signal values is called a digital signal. However, in some (earlier) literature, “analog” and “digital” are only related to the signal values, i.e., the signals to the left in Figure 5.1 are then analog whereas the signals to the right in the same figure are digital. In this book, we assume hereafter that analog signals are continuous-time signals whereas digital signals are discrete-time signals.

### 1.05.2.2 Discrete-time signals—sequences

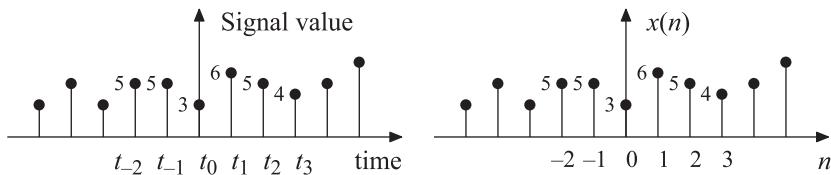
A continuous-time signal can mathematically be represented by a continuous function  $x(t)$  where  $t$  is a continuous variable. A discrete-time signal cannot be represented in this way since it is only defined at discrete instances. To handle discrete-time signals mathematically, one therefore makes use of sequences of numbers.

As one can see from Figure 5.1, the signal values of a discrete-time signal constitute a sequence of numbers. In mathematical form, a sequence of numbers can be written as

$$\{x(n)\}, \quad n = \dots, -2, -1, 0, 1, 2, \dots, \quad (5.1)$$

**FIGURE 5.2**

Graphical representation of a sequence  $x(n)$ .

**FIGURE 5.3**

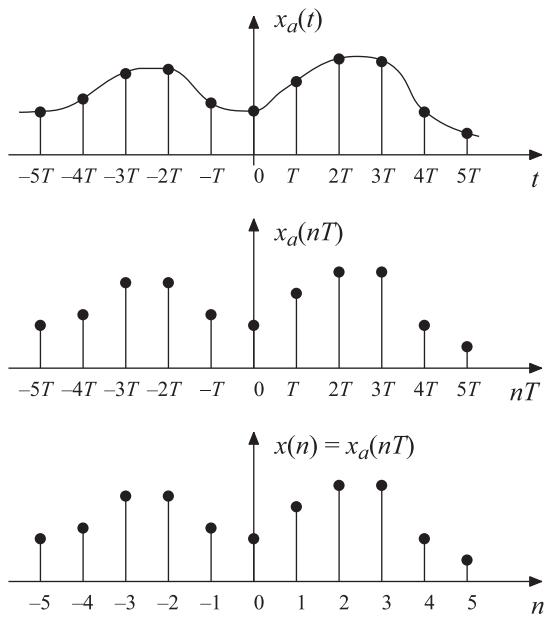
A discrete-time signal and its representation in the form of a sequence  $x(n)$ .

where  $\{x(n)\}$  denotes the sequence whereas  $x(n)$  denotes value number  $n$  in the sequence. The values  $x(n)$  are usually called samples because they are often generated through sampling of continuous-time signals, see the subsequent subsection. The notation of the sequence is thus  $\{x(n)\}$ , but when no misunderstandings can occur we write “the sequence  $x(n)$ ” for the sake of simplicity. A sequence can be represented graphically as illustrated in Figure 5.2. The horizontal axis is drawn as a continuous line but it is important to remember that the sequence is only defined for discrete values.

When a sequence represents a discrete-time signal, the samples  $x(n)$  equal the signal values of the signal at the time instances  $t = t_n$ , as illustrated in Figure 5.3. Since sequences often represent discrete-time signals, we frequently write “the discrete-time signal  $x(n)$ ” instead of “the sequence  $x(n)$ .” This is in accordance with the continuous-time case where one often writes “the continuous-time signal  $x(t)$ ” instead of “the function  $x(t)$ .”

### 1.05.2.3 Sampling of continuous-time signals

Discrete-time signals are often generated through sampling (measurement) of continuous-time signals. It is common to use the same time interval  $T$  between the sampling instances. The sampling then takes place at the time instances  $t = nT$ , as illustrated in Figure 5.4. The time interval is called the sampling period whereas  $f_s = 1/T$  is referred to as the sampling frequency, which thus denotes the number of sampling instances per second. This type of sampling is called uniform sampling. If  $x_d(t)$  denotes the

**FIGURE 5.4**

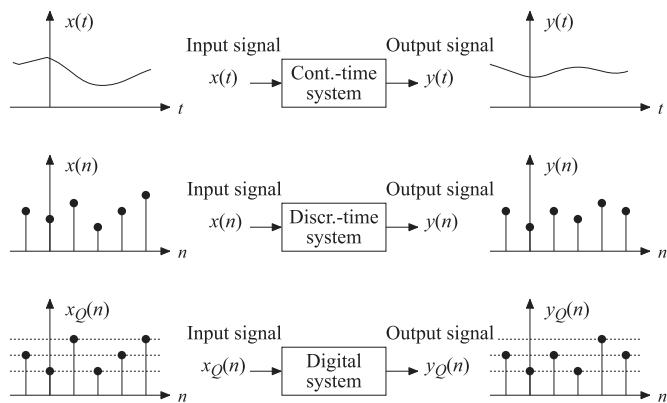
Sampling of a continuous-time signal.

continuous-time signal,<sup>1</sup> the corresponding discrete-time signal  $x(n)$  becomes  $x(n) = x_a(nT)$ . Instead of using the notation  $x(n)$  one may thus use  $x_a(nT)$  to stress that the signal values are obtained via uniform sampling of a continuous-time signal. Throughout this chapter, we mainly use  $x(n)$  for the sake of simplicity.

#### 1.05.2.4 Classification of systems

Systems can be classified in accordance with their input and output signal types. As we saw in the previous section, we can divide signals into four different categories. Hence, one may also divide systems into four basic classes as seen in Figure 5.5. A continuous-time system is a system whose input and output signals are continuous-time signals, whereas a discrete-time system is a system whose input and output signals are discrete-time signals. Mathematically, the role of a discrete-time system is thus to generate an output sequence given an input sequence. Therefore, instead of the terms “input signal” and “output signal,” we often use the terms “input sequence” and “output sequence.” A digital system is a discrete-time system in which the samples only can take on discrete values. Today, signal processing systems usually contain both continuous-time and discrete-time parts.

<sup>1</sup>The subscript “a” in  $x_a(t)$  stands for analog. We use the notation  $x_a(t)$  instead of  $x_c(t)$ , which would be natural when we deal with continuous-time signals. This is to avoid confusion with frequency edges  $\Omega_c$  etc., where “c” stands for cutoff.

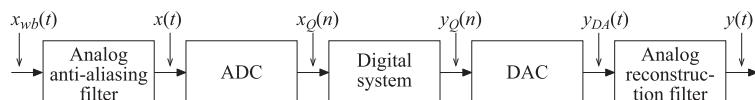
**FIGURE 5.5**

Three different types of systems.

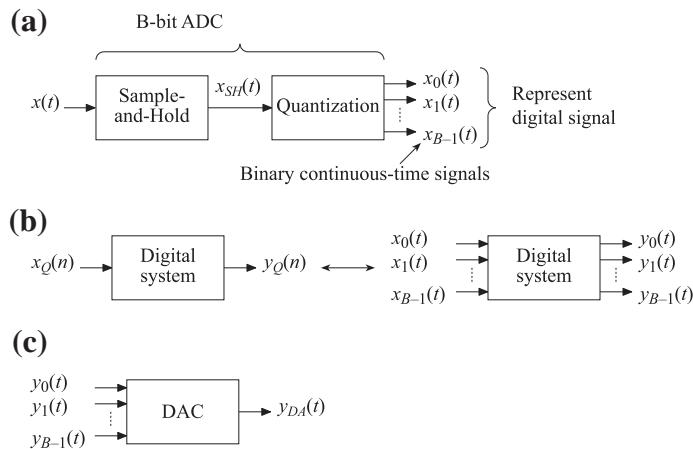
### 1.05.2.5 Digital signal processing of analog signals

The principle of digital signal processing of analog signals is illustrated in Figures 5.6–5.8. The first step is to convert the signal from analog into digital form. This is done by an ADC which takes care of both sampling and quantization. The so obtained digital signal can then be processed by a digital system which takes as input a sequence of samples,  $x_Q(n)$ , and generates an output sequence,  $y_Q(n)$  [ $Q$  indicates quantized signal values]. Finally, reconstruction takes place in order to go back to a continuous-time representation. This is performed by a DAC followed by an analog so called reconstruction filter.

To be able to perform A/D and D/A conversion without errors, the original signal must be bandlimited. Before the A/D conversion, the signal is therefore bandlimited with the aid of an analog so called anti-aliasing filter ( $wb$  in  $x_{wb}(t)$  in Figure 5.6 stands for wide-band). A/D conversion is done by first sampling and then quantizing the signal, see Figure 5.7a. In practice, the sampling is performed by a sample-and-hold-circuit (S/H circuit). An S/H circuit samples an analog signal at the time instances  $t = nT$  and holds this sample value until the next sampling instance. The output signal from an ideal S/H circuit is a new analog signal that ideally is piece-wise constant. The function of the S/H circuit can be seen in Figure 5.8a and b. The next step in the A/D conversion is to quantize the sample values using a quantizer. The input signal of the quantizer is thus the output signal from the S/H circuit. The input to

**FIGURE 5.6**

Digital signal processing of analog signals.

**FIGURE 5.7**

ADC, digital system, and DAC.

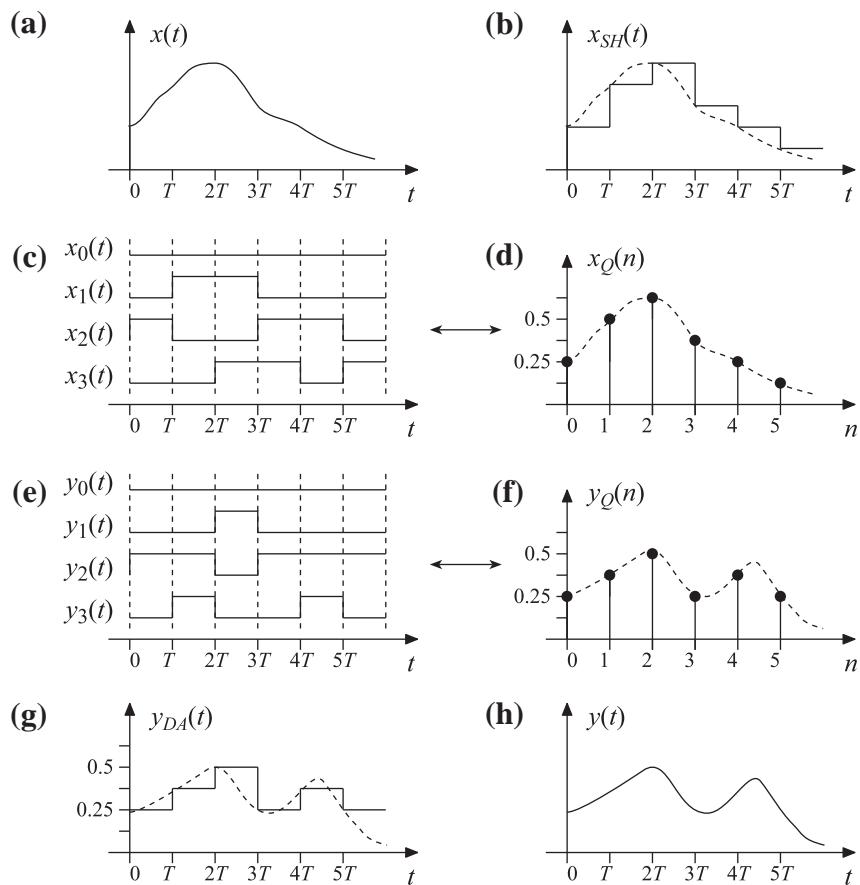
the quantizer needs to be kept constant long enough for the quantizer to produce a correct result which is why an S/H circuit is needed.

Quantization amounts to representing each sample in a binary form with a finite wordlength. There exist many different types of binary representations. One commonly used binary form is the two's-complement representation. A number  $x$ , which for the sake of simplicity is assumed here to lie between  $-1$  and  $1$ , is then represented as

$$x = -x_0 + \sum_{i=1}^{B-1} 2^{-i} x_i, \quad (5.2)$$

where  $x_i, i = 0, 1, \dots, B-1$ , can take on the values zero and one. The number of bits is thus  $B$ . The bits from a quantizer can be delivered in serial or parallel. In the latter case, the output signal from a  $B$ -bit quantizer consists in practice of  $B$  continuous-time signals which together represent the different sample values. In the ideal case, these continuous-time signals are binary. That is, they can only take on two different values, usually zero and one. The principle can be seen in Figure 5.8c and d. In practice, the signals must hold the values zero or one long enough so one can store them in registers, etc. Furthermore, it takes a certain time to perform the quantizations which means that a certain amount of delay will be introduced. The output signal from the quantizer is the desired digital signal that can be processed by a digital system. In a practical implementation, the sample values are represented in the same form as the quantizer output. The output signal from a digital system is thus represented in the same form as its input signal, see Figure 5.8e and f.

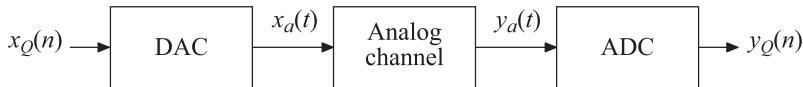
A digital signal can be converted to an analog signal using a DAC followed by an analog reconstruction filter. The input signal to the DAC is a digital signal represented in the same form as the output signal from the ADC, i.e., the output from the quantizer. The output signal from the DAC is a continuous-time signal that is piece-wise constant with a limited number of levels, see Figure 5.8(g). In other words, it

**FIGURE 5.8**

Typical signals in digital signal processing of analog signals.

is of the same type as the output signal from the S/H circuit with the difference that the DAC output signal takes on discrete signal values whereas the S/H circuit output signal takes on continuous signal values. Finally, to obtain the desired analog signal, the output signal from the DAC must be filtered. The role of this filter is to “smooth out” the piece-wise-constant signal from the DAC so as to obtain the desired signal.

As evident from above, sampling and reconstruction as well as quantization play fundamental roles in digital signal processing. The following sections treat the basic principles of these operations. We will only deal with relations between continuous-time signals and the discrete-time counterparts. Mathematically, this amounts to study relations between functions  $x_a(t)$  and the corresponding sequences  $x(n) = x_a(nT)$  as well as their transform-domain relations. We will not discuss the physical representation of the signals seen in Figures 5.7 and 5.8.

**FIGURE 5.9**

Simple model of a digital communication system.

### 1.05.2.6 Digital communication—analog signal processing of digital signals

In digital communication systems, the situation is opposite to that in the previous section in that analog signal processing is now used for processing digital signals. Figure 5.9 shows a simple model of a digital communication system. The original digital information is contained in  $x_Q(n)$ . This information is D/A converted before being transmitted over an analog channel. Finally, on the receiver side, the signal is A/D converted, generating  $y_Q(n)$ . It is desired to have  $y_Q(n) = x_Q(n)$  so as to receive the information sent. In practice, due to many different error sources in the communication channel, advanced digital signal processing methods are required to transmit information successfully. Regarding the sampling and reconstruction, one can make use of the same analysis methods here as for the system depicted earlier in Figure 5.6.

---

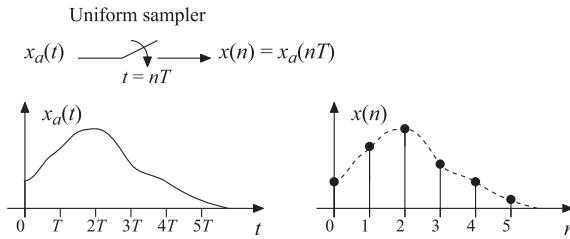
## 1.05.3 Sampling of deterministic signals

Discrete-time signals are often generated through sampling of continuous-time signals. It is therefore of interest to investigate the relations between the Fourier transform of a discrete-time signal and that of the underlying continuous-time signal. This relation is given by the so called Poisson's summation formula which leads us to the sampling theorem. This theorem states that a continuous-time signal can be sampled and perfectly reconstructed provided that the signal is bandlimited and the sampling frequency exceeds twice the bandwidth. If the theorem is fulfilled, the original signal can be retained via an ideal pulse amplitude modulator. This section treats basic concepts and theory on sampling and reconstruction. In practice, one can neither fulfill the sampling theorem nor perform the reconstruction perfectly because ideal pulse amplitude modulators cannot be implemented. Some of these practical aspects are included in the section. It is also noted that this section deals with deterministic signals. The extension to stochastic signals is treated in Section 1.05.4.

### 1.05.3.1 Uniform sampling

Discrete-time signals are typically generated through sampling (measurement) of continuous-time signals. Most signal processing applications are based on uniform sampling which means that the time interval between two consecutive sampling instances is constant. The sampling then takes place at the time instances  $t = nT$ ,  $n = \dots, -2, -1, 0, 1, 2, \dots$ , as illustrated in Figure 5.10. The time interval  $T$  is called the sampling period whereas  $f_s = 1/T$  is referred to as the sampling frequency, which thus denotes the number of sampling instances per second. If  $x_a(t)$  denotes the continuous-time signal, the corresponding discrete-time signal, represented by the sequence  $x(n)$ , becomes

$$x(n) = x_a(nT). \quad (5.3)$$

**FIGURE 5.10**

Uniform sampling.

### 1.05.3.2 Poisson's summation formula

This section derives a relation between the Fourier transforms of  $x(n)$  and  $x_a(t)$ . Assume that the continuous-time signal  $x_a(t)$  has the Fourier transform  $X_a(j\Omega)$ . We can then express  $x_a(t)$  in terms of its inverse Fourier transform according to

$$x_a(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_a(j\Omega) e^{j\Omega t} d\Omega. \quad (5.4)$$

In particular, we have for  $t = nT$ ,

$$x_a(nT) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_a(j\Omega) e^{j\Omega nT} d\Omega, \quad (5.5)$$

which can be rewritten as

$$\begin{aligned} x_a(nT) &= \dots + \frac{1}{2\pi} \int_{-3\pi/T}^{-\pi/T} X_a(j\Omega) e^{j\Omega nT} d\Omega + \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} X_a(j\Omega) e^{j\Omega nT} d\Omega \\ &\quad + \frac{1}{2\pi} \int_{\pi/T}^{3\pi/T} X_a(j\Omega) e^{j\Omega nT} d\Omega + \dots \end{aligned}$$

By making the variable substitutions  $\Omega \rightarrow \Omega - 2\pi k/T$ ,  $k$  integer, utilizing that  $e^{-j2\pi kn} = 1$ , and changing the order between summation and integration, we obtain

$$x_a(nT) = \sum_{k=-\infty}^{\infty} \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} X_a\left(j\Omega - j\frac{2\pi k}{T}\right) e^{j(\Omega - \frac{2\pi k}{T})nT} d\Omega, \quad (5.6)$$

$$= \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \sum_{k=-\infty}^{\infty} X_a\left(j\Omega - j\frac{2\pi k}{T}\right) e^{j\Omega T n} d\Omega. \quad (5.7)$$

With  $\Omega T$  as integration variable we get

$$x_a(nT) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a\left(j\Omega - j\frac{2\pi k}{T}\right) e^{j\Omega T n} d(\Omega T). \quad (5.8)$$

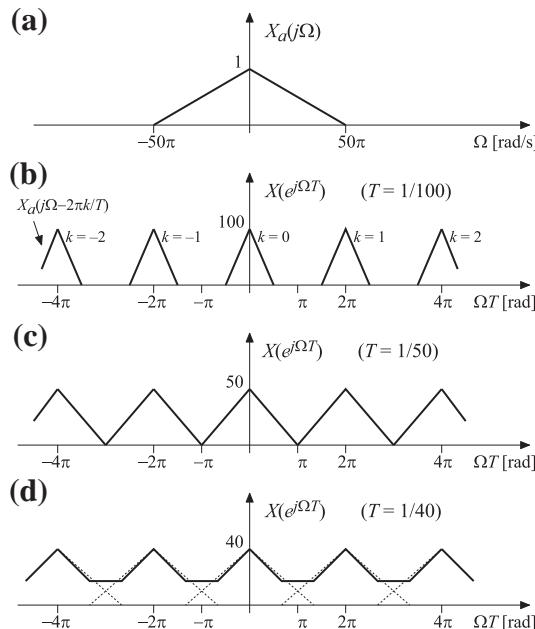
Further, we know that a discrete-time signal  $x(n)$  can be expressed in terms of its inverse Fourier transform according to

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\Omega T}) e^{j\Omega T n} d(\Omega T). \quad (5.9)$$

Thus, if we let  $x(n) = x_a(nT)$ , and utilize the uniqueness of the Fourier transform, we obtain the following relation from (5.8) and (5.9):

$$X(e^{j\Omega T}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a \left( j\Omega - j\frac{2\pi k}{T} \right). \quad (5.10)$$

Equation (5.10) is referred to as Poisson's summation formula<sup>2</sup> which gives the relation between the Fourier transform of a continuous-time signal and the Fourier transform of the discrete-time signal that is generated by sampling the continuous-time signal uniformly with the sampling period  $T$  (sampling frequency  $f_s = 1/T$ ) [23]. Poisson's summation formula states that the spectrum of the discrete-time signal is obtained by summing the spectrum of the continuous-time signal and its shifted versions, weighted with  $1/T$ . This is illustrated in Figure 5.11 in the following example.



**FIGURE 5.11**

Spectra for a continuous-time signal and the corresponding sequences generated via sampling with three different sampling frequencies.

<sup>2</sup>If the sampled signal is discontinuous at a certain time instant, and if the signal is sampled at this instant, then the sample value must be chosen as the average of the left and right limits in order to make (5.10) valid.

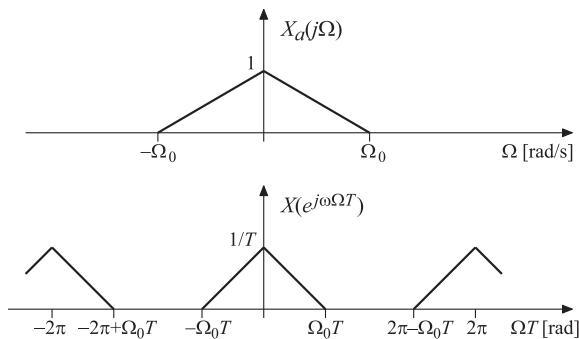
**FIGURE 5.12**

Illustration of the sampling theorem.

### 1.05.3.2.1 Example of Poisson's summations formula

Figure 5.11a shows the spectrum of a bandlimited continuous-time signal. Figure 5.11b–d show the spectra of three different discrete-time signals generated by sampling the continuous-time signal with the sampling frequencies 100 Hz, 50 Hz, and 40 Hz, respectively, i.e.,  $T = 1/100$  s,  $1/50$  s, and  $1/40$  s, respectively. Studying Figure 5.11, the following observations can be made. In the first two cases, the partial spectra do not overlap each other [Figure 5.11b, c]. This means that the shape of  $X_a(j\Omega)$  is preserved and it is thereby possible in principle to reconstruct  $x_a(t)$  from  $x(n)$ , as we shall see later. In the third case, the shape of  $X_a(j\Omega)$  is deteriorated since the partial spectra to some extent overlap each other [Figure 5.11d]. In this case, we can no longer perfectly reconstruct  $x_a(t)$  from  $x(n)$ . What distinguishes the three sequences is that they have been obtained using three different sampling frequencies. If the continuous-time signal is bandlimited, it can thus be sampled and reconstructed perfectly provided that the sampling frequency is sufficiently high. This is summarized below in the sampling theorem.

### 1.05.3.3 The sampling theorem

As illustrated above, a continuous-time signal can be sampled and reconstructed from the generated samples if the so called sampling theorem is fulfilled.

**Sampling theorem:** If a continuous-time signal  $x_a(t)$  is bandlimited to  $\Omega = \Omega_0$  ( $f = f_0$ ), i.e.,

$$X_a(j\Omega) = 0, \quad |\Omega| > \Omega_0 = 2\pi f_0, \quad (5.11)$$

then  $x_a(t)$  can be recovered from the samples  $x(n) = x_a(nT)$  provided that

$$f_s = \frac{1}{T} > 2f_0. \quad (5.12)$$

Thus, if (5.11) and (5.12) are satisfied, the continuous-time signal  $x_a(t)$  can be recovered from  $x(n) = x_a(nT)$ . One can understand this by studying Figure 5.12 (and also Figure 5.11). If the partial spectra in Poisson's summation formula do not overlap each other, the shape of  $X_a(j\Omega)$  is preserved

and it is thereby possible, in principle, to reconstruct  $x_a(t)$  from  $x(n)$ . From Figure 5.12, we see that overlap of spectra is avoided if

$$2\pi - \Omega_0 T > \Omega_0 T \Leftrightarrow \Omega_0 T < \pi. \quad (5.13)$$

Since  $\Omega = 2\pi f$  and  $f_s = 1/T$ , (5.13) is equivalent to

$$2\pi f_0 T < \pi \Leftrightarrow f_s = \frac{1}{T} > 2f_0, \quad (5.14)$$

which is the same as (5.12). The frequency  $f_s/2$  is called the Nyquist frequency whereas  $f_s$  is referred to as the Nyquist sampling frequency. Further, the frequency band  $\Omega \in [-\pi f_s, \pi f_s]$  (thus  $f \in [-f_s/2, f_s/2]$ ) is referred to as the first Nyquist band.

The sampling theorem says that the sampling frequency  $f_s$  must be strictly greater than  $2f_0$  in order to be able to recover the signal. It should be mentioned however that for some classes of signals, like energy signals, one may select  $f_s$  equal to  $2f_0$  and still reconstruct the signals (in a certain sense). For other classes of signals, like sinusoidal signals, it is necessary to fulfill  $f_s > 2f_0$ . This is because, for a sinusoidal signal with the frequency  $f_0$ , one may for example sample the zero-crossings. That is, if  $f_s$  equals  $2f_0$ , one may obtain a sequence with zero-valued samples which means that the continuous-time signal can not be reconstructed.

If  $f_s \leq 2f_0$ , the sampling theorem is not fulfilled in which case the signal is undersampled. This means that the partial spectra will overlap, as was discussed earlier and illustrated in Figure 5.11d. This in turn implies that the shape of the continuous-time signal's spectrum is not preserved, which means that the signal cannot be reconstructed exactly. The error that is then introduced is called aliasing distortion. The reason for this name is that frequency components above the Nyquist frequency are aliased to the baseband  $\Omega T \in [-\pi, \pi]$ , see Figure 5.11d. In practice, one often chooses a higher sampling frequency than that required to fulfill the sampling theorem, the reason being that one can thereby reduce the requirements on the analog components that are used in the A/D and D/A conversions. In such a case, the signal is oversampled. If  $f_s = M2f_0$ , one has  $M$  times oversampling. Oversampled A/D and D/A converters are discussed in Section 1.05.7.

#### 1.05.3.3.1 Example of aliasing

To further illustrate that the sampling theorem must be fulfilled in order to sample and reconstruct a signal, we consider stationary sinusoids. Let  $x_{a1}(t)$  and  $x_{a2}(t)$  be two stationary sinusoidal signals according to

$$x_{a1}(t) = \sin(\Omega_0 t) = \sin(2\pi f_0 t) \quad (5.15)$$

and

$$x_{a2}(t) = -\sin(3\Omega_0 t) = -\sin(6\pi f_0 t). \quad (5.16)$$

We now form the discrete-time signals  $x_1(n)$  and  $x_2(n)$  by sampling  $x_{a1}(t)$  and  $x_{a2}(t)$ , respectively, with a sampling frequency of  $f_s = 4f_0$ , i.e., a sampling period of  $T = 1/(4f_0)$ . This gives us

$$x_1(n) = x_{a1}(nT) = x_{a1}(0.25n/f_0) = \sin(0.5\pi n) \quad (5.17)$$

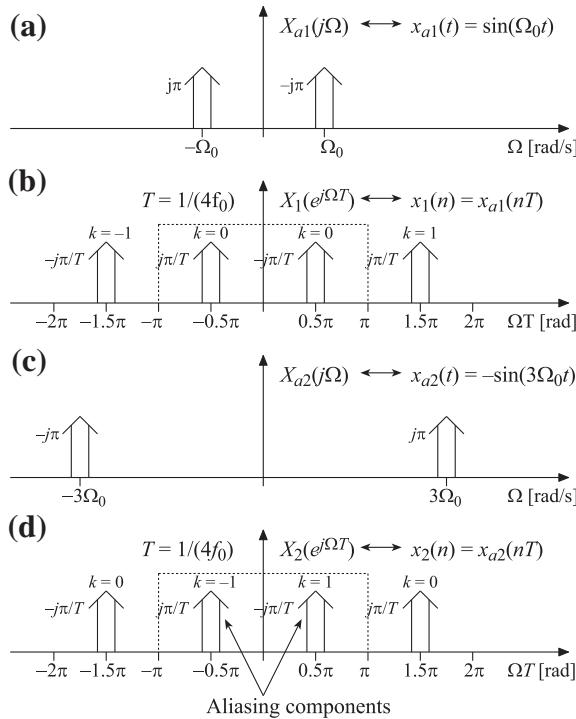
**FIGURE 5.13**

Illustration of aliasing in the example of Section 1.05.3.3.1.

and

$$x_2(n) = x_{a2}(nT) = x_{a2}(0.25n/f_0) = -\sin(1.5\pi n). \quad (5.18)$$

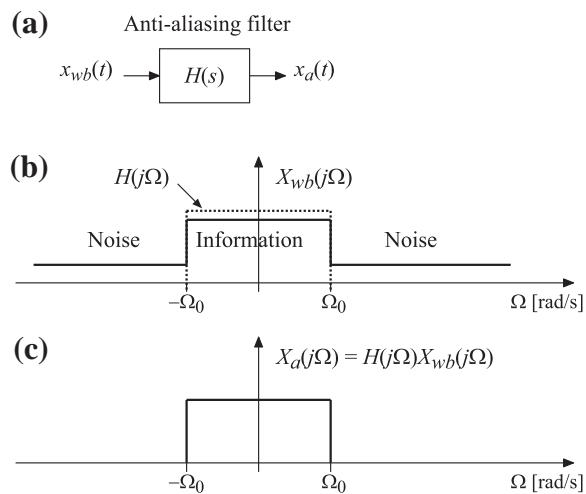
Since  $\sin(a) = \sin(a + 2\pi n)$  for all integers  $n$ , and  $\sin(a) = -\sin(-a)$ , we can rewrite  $x_2(n)$  as

$$\begin{aligned} x_2(n) &= -\sin(1.5\pi n) = -\sin(1.5\pi n - 2\pi n), \\ &= -\sin(-0.5\pi n) = \sin(0.5\pi n) = x_1(n). \end{aligned}$$

That is,  $x_2(n)$  equals  $x_1(n)$ . The reason for this is that the sampling theorem is not fulfilled for  $x_{a2}(t)$ . This means that the spectrum of  $x_{a2}(t)$  is aliased into the band  $\Omega T \in [-\pi, \pi]$ , as illustrated in Figure 5.13. If the sampling theorem is not met, we can thus obtain the same sequence through sampling of two different signals.

#### 1.05.3.4 Anti-aliasing filter

A condition for sampling and perfect reconstruction of a continuous-time signal is that it is strictly bandlimited. In practice, continuous-time signals are not strictly bandlimited which means that we will

**FIGURE 5.14**

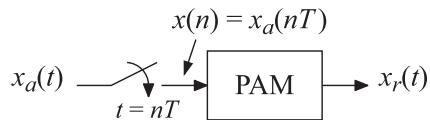
Band limitation using an anti-aliasing filter (wb stands for wide-band).

always have aliasing distortion since the partial spectra then overlap each other. This in turn implies that we can not reconstruct the signal perfectly, but instead we obtain a distorted signal. To obtain an acceptable level of distortion, one usually have to filter the signal before the sampling takes place, as seen in Figure 5.14a. In this context, the filter  $H(s)$  is called anti-aliasing filter because it is used to reduce the aliasing distortion.

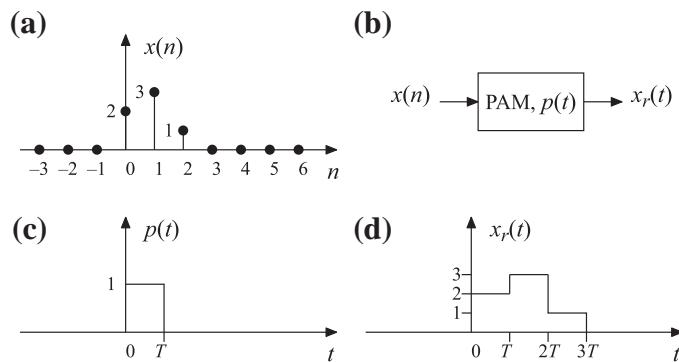
Apparently, the anti-aliasing filter affects the signal that is to be sampled. However, in many cases, the information in the signal lies in the low-frequency region whereas the high-frequency band only contains undesired frequency components like noise etc. If the anti-aliasing filter is an ideal lowpass filter, the information is unaffected whereas the undesired components are removed. This is illustrated in Figure 5.14b and c. In practice, one can not implement ideal filters which means that one always has to account for a certain amount of distortion. More about distortion is given in Section 1.05.3.6.

### 1.05.3.5 Reconstruction

Reconstruction refers to the process that forms a continuous-time signal from a discrete-time signal. The reconstruction can be done with the aid of pulse amplitude modulation. The system that performs the pulse amplitude modulation is called pulse amplitude modulator (PAM). The sampling and reconstruction can be represented as in Figure 5.15. We denote the reconstructed signal as  $x_r(t)$  to separate it from the original signal  $x_d(t)$ .

**FIGURE 5.15**

Sampling and reconstruction using a PAM.

**FIGURE 5.16**

Example of reconstruction (5.19).

The PAM forms the output signal  $x_r(t)$  through multiplication of the samples  $x(n)$  by the pulse (signal)  $p(t)$  and its shifted versions  $p(t - nT)$ . The output signal is given by

$$x_r(t) = \sum_{n=-\infty}^{\infty} x(n)p(t - nT). \quad (5.19)$$

Figure 5.16 shows an example of reconstruction according to (5.19).

#### 1.05.3.5.1 Ideal reconstruction

When the sampling theorem is fulfilled, it is in principle possible to obtain  $x_r(t) = x_a(t)$  by properly selecting  $p(t)$ . If the original signal  $x_a(t)$  is perfectly reconstructed from the samples  $x(n) = x_a(nT)$ , we have ideal reconstruction. As will be clear below, it is however not possible to perform ideal reconstruction in practice.

To see how to choose  $p(t)$  in order to obtain  $x_r(t) = x_a(t)$ , we proceed as follows. If we utilize the expression for the reconstructed signal  $x_r(t)$  in (5.19), its Fourier transform can be written as

$$\begin{aligned} X_r(j\Omega) &= \int_{-\infty}^{\infty} x_r(t) e^{-j\Omega t} dt = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(n) p(t - nT) e^{-j\Omega t} dt \\ &= \sum_{n=-\infty}^{\infty} x(n) \int_{-\infty}^{\infty} p(t - nT) e^{-j\Omega t} dt \end{aligned} \quad (5.20)$$

assuming that integration and summation can be interchanged. The variable substitution  $v = t - nT$  in (5.20) yields

$$\begin{aligned} X_r(j\Omega) &= \sum_{n=-\infty}^{\infty} x(n) \int_{-\infty}^{\infty} p(v) e^{-j\Omega(v+nT)} dv \\ &= \left( \sum_{n=-\infty}^{\infty} x(n) e^{-j\Omega T n} \right) \left( \int_{-\infty}^{\infty} p(v) e^{-j\Omega v} dv \right) \\ &= X(e^{j\Omega T}) P(j\Omega). \end{aligned}$$

We also know that  $X(e^{j\Omega T})$  is related to  $X_a(j\Omega)$  via Poisson's summation formula. The Fourier transform of the reconstructed signal can therefore be related to the Fourier transform of the original signal according to

$$X_r(j\Omega) = P(j\Omega) X(e^{j\Omega T}) = P(j\Omega) \frac{1}{T} \sum_{k=-\infty}^{\infty} X_a \left( j\Omega - j \frac{2\pi k}{T} \right). \quad (5.21)$$

If the sampling theorem is fulfilled, the partial spectra  $X_a(j\Omega - j2\pi k/T)$  do not overlap each other as illustrated in Figure 5.17. We can therefore obtain  $X_r(j\Omega) = X_a(j\Omega)$  by letting  $P(j\Omega)$  be an ideal lowpass filter according to

$$P(j\Omega) = \begin{cases} T, & |\Omega| \leq \Omega_c < \pi/T, \\ 0, & |\Omega| > \Omega_c \end{cases} \quad (5.22)$$

with  $\Omega_0 < \Omega_c \leq \pi/T$  since  $P(j\Omega)$  then leaves  $X_a(j\Omega)$  unaffected and removes the remaining partial spectra. This is also illustrated in Figure 5.17. We then obtain (utilizing the uniqueness of the Fourier transform)

$$X_r(j\Omega) = X_a(j\Omega) \Leftrightarrow x_r(t) = x_a(t). \quad (5.23)$$

That is, the original signal  $x_a(t)$  is reconstructed from the samples  $x(n)$ .

The impulse  $p(t)$  is obtained by taking the inverse Fourier transform of  $P(j\Omega)$  in (5.22). For example, with  $\Omega_c = \pi/T$ , we get

$$p(t) = \frac{\sin(\pi t/T)}{\pi t/T}, \quad (5.24)$$

which is a two-sided function and thus not realizable in practice by causal systems.

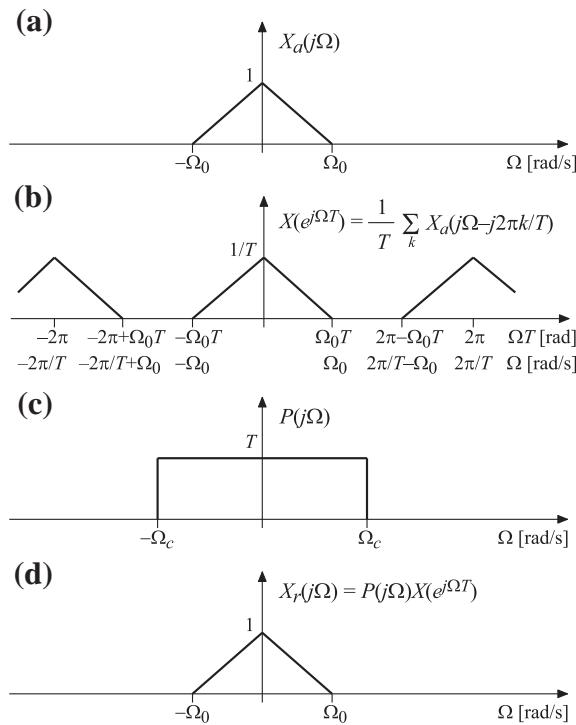
**FIGURE 5.17**

Illustration of ideal reconstruction.

### 1.05.3.5.2 Reconstruction using a D/A converter and an analog reconstruction filter

In practice, the reconstruction is performed using a D/A converter followed by an analog so called reconstruction filter as depicted in Figure 5.18. The D/A converter is in principle a PAM and can therefore be described with a pulse in accordance with the previous section. If we denote this pulse as  $q(t)$ , and let  $x_1(t)$  denote the output from the D/A converter, we get

$$X_1(j\Omega) = Q(j\Omega)X(e^{j\Omega T}). \quad (5.25)$$

Furthermore, we have

$$X_r(j\Omega) = H(j\Omega)X_1(j\Omega). \quad (5.26)$$

Combining (5.25) and (5.26) gives us

$$X_r(j\Omega) = Q(j\Omega)H(j\Omega)X(e^{j\Omega T}). \quad (5.27)$$

We can now write  $X_r(j\Omega)$  as

$$X_r(j\Omega) = P(j\Omega)X(e^{j\Omega T}), \quad (5.28)$$

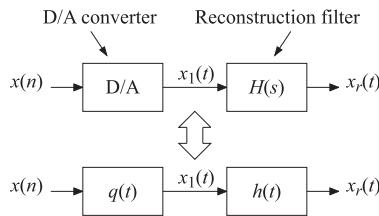
**FIGURE 5.18**

Illustration of ideal reconstruction.

where

$$P(j\Omega) = Q(j\Omega)H(j\Omega). \quad (5.29)$$

We know that, if  $P(j\Omega)$  is chosen as in (5.22), then  $x_r(t) = x_a(t)$ . Hence, also a scheme with a D/A converter followed by a reconstruction filter according to Figure 5.18 achieves ideal reconstruction provided that  $Q(j\Omega)H(j\Omega)$  satisfies

$$Q(j\Omega)H(j\Omega) = \begin{cases} T, & |\Omega| > \Omega_c < \pi/T, \\ 0, & |\Omega| > \Omega_c, \end{cases} \quad (5.30)$$

where  $\Omega_0 < \Omega_c \leq \pi/T$ . Dividing the reconstruction into two steps gives us more degrees of freedom. In particular, one can thereby let  $q(t)$  be a simple pulse since the only requirement is that the product  $Q(j\Omega)H(j\Omega)$  must meet the right-hand side of (5.30). In principle, the actual shape of the individual responses  $Q(j\Omega)$  and  $H(j\Omega)$  are therefore arbitrary. In practice, it is much easier to generate simple pulses, that subsequently are filtered through a conventional analog filter, instead of generating the more complicated pulses directly, like the sinc function in (5.24).

A common type of D/A converter is of the type zero-order hold. In this case,  $q(t)$  is a square pulse according to Figure 5.19d. This pulse can mathematically be expressed as

$$q(t) = \begin{cases} 1, & 0 < t < T, \\ 0, & t < 0, t > T. \end{cases} \quad (5.31)$$

Its Fourier transform is

$$Q(j\Omega) = e^{-j\Omega T/2} \frac{\sin(\Omega T/2)}{\Omega/2}. \quad (5.32)$$

Hence, the Fourier transform  $Q(j\Omega)$  is a sinc function with zero-crossings at  $\pm 2\pi k/T$ , for all integers  $k$ . We note that  $Q(j\Omega)H(j\Omega)$  satisfies (5.30) if the frequency response  $H(j\Omega)$  is selected as

$$H(j\Omega) = \begin{cases} T/Q(j\Omega), & |\Omega| \leq \Omega_c < \pi/T, \\ 0, & |\Omega| > \Omega_c. \end{cases} \quad (5.33)$$

For low frequencies,  $H(j\Omega)$  is thus

$$H(j\Omega) = e^{j\Omega T/2} \frac{\Omega/2}{\sin(\Omega T/2)}, \quad |\Omega| \leq \Omega_c < \pi/T. \quad (5.34)$$

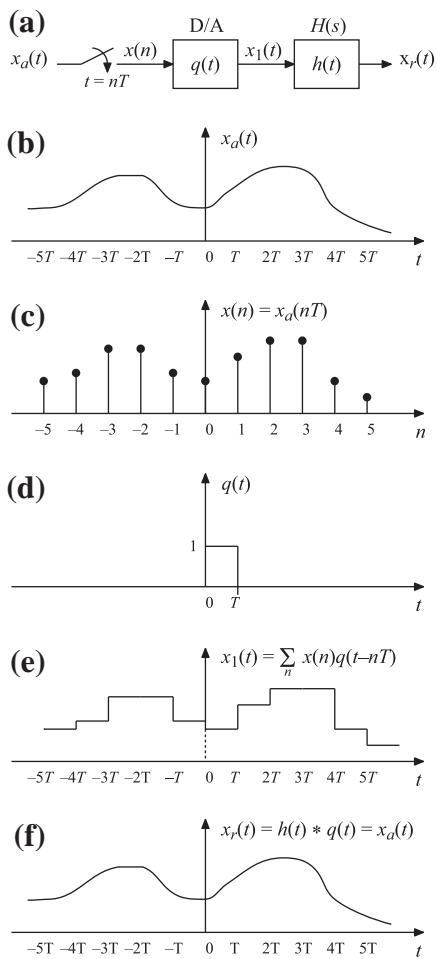
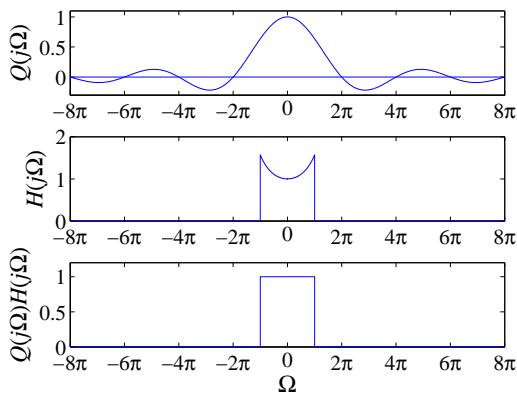
**FIGURE 5.19**

Illustration of reconstruction using a D/A converter and an analog filter.

When  $\Omega_c = \pi/T$ , the frequency response is as shown in Figure 5.20. The response of this filter is discontinuous at  $\Omega_c = \pi/T$  and it can therefore not be implemented in practice. Further, it would be very costly to approximate this function since a high filter order would be required in order to avoid a large distortion.

To get around these problems in practical implementations, one uses a certain amount of oversampling. This means that the sampling frequency is higher than the sampling theorem requires. One can thereby let the reconstruction filter have a transition band as shown in Figure 5.21. The higher is the sampling frequency, the wider can the transition band be. The filter requirements are thus relaxed when

**FIGURE 5.20**

Frequency responses in ideal reconstruction using a D/A converter followed by an analog reconstruction filter ( $T = 1$ ).

the sampling frequency is increased. Another advantage of using oversampling is that the quantization noise that is introduced in the A/D conversion thereby can be reduced. Oversampling will be discussed later in Section 1.05.7. An oversampled A/D converter generates more samples than is actually required to reconstruct the signal. To make sure that the subsequent digital system does not have to work at higher data rates than necessary, one reduces the data rate by using decimation [24]. This can be done without losing information since the signal has been oversampled. In other words, there is a redundancy in the corresponding discrete-time signal.

### 1.05.3.6 Distortion caused by undersampling

In practice, a signal is never strictly bandlimited which means that one will always undersample the signal and thereby obtain a certain amount of aliasing distortion. The sampling frequency must be high enough to ensure that the reconstructed signal does not differ too much from the original one. The following example illustrates how the choice of sampling frequency affects the reconstructed signal.

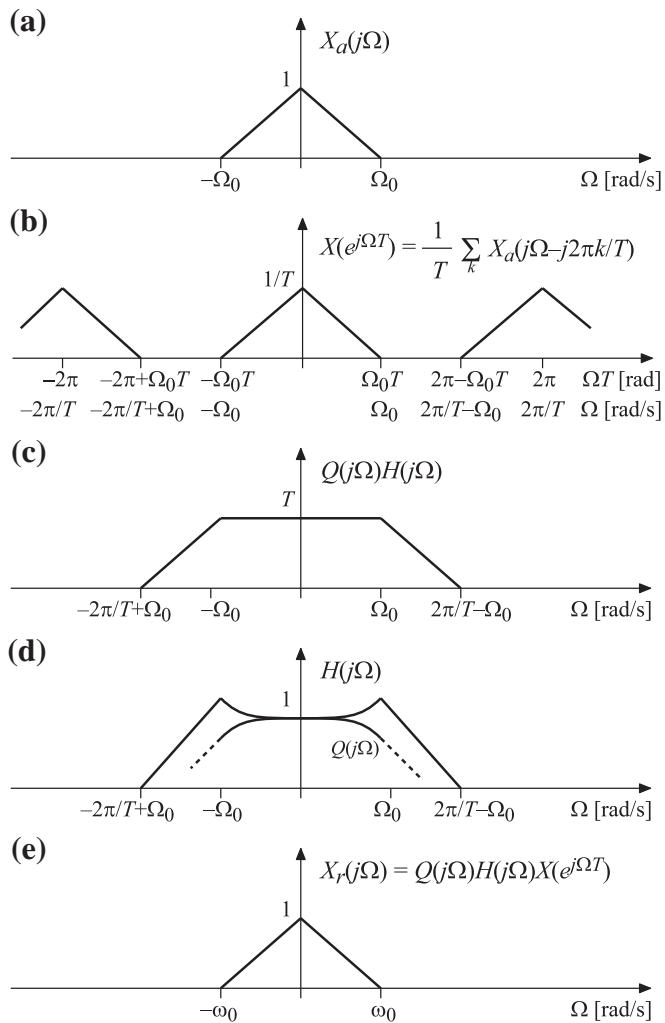
#### 1.05.3.6.1 Example of undersampling

Let  $x_a(t)$  be a continuous-time signal according to

$$x_a(t) = e^{-a|t|}, \quad a > 0. \quad (5.35)$$

This signal has a real Fourier transform given by

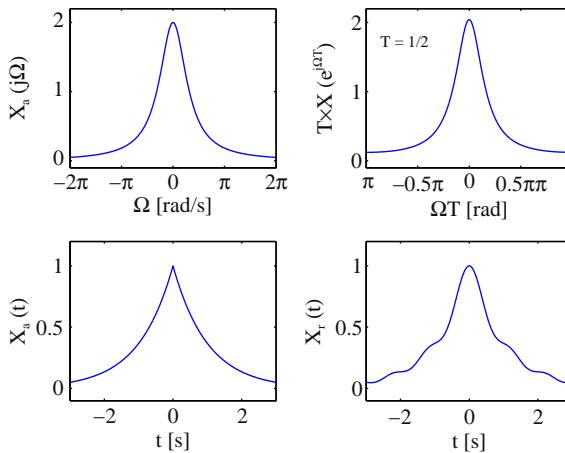
$$X_a(j\Omega) = \frac{2a}{\Omega^2 + a^2}. \quad (5.36)$$

**FIGURE 5.21**

Typical spectra using oversampling and reconstruction via a D/A converter followed by an analog reconstruction filter.

Apparently,  $X_a(j\Omega)$  is not bandlimited. We now sample  $x_a(t)$  with the sampling frequency  $f_s = 1/T$  which gives us the discrete-time signal  $x(n)$  according to

$$x(n) = x_a(nT) = e^{-a|nT|} = (e^{-aT})^{|n|}. \quad (5.37)$$

**FIGURE 5.22**

Spectra and signals in the example in Section 1.05.3.6.1 for  $T = 1/2$ .

The Fourier transform of  $x(n)$  is

$$X(e^{j\Omega T}) = \frac{1 - e^{-2aT}}{1 - 2e^{-aT} \cos(\Omega T) + e^{-2aT}}. \quad (5.38)$$

When  $X_a(j\Omega)$  is real, so is  $X(e^{j\Omega T})$  since the two transforms can be related to each other via Poisson's summation formula. Finally, we perform reconstruction using an ideal PAM where  $P(j\Omega)$  is given by

$$P(j\Omega) = \begin{cases} T, & |\Omega| \leq \pi/T, \\ 0, & |\Omega| > \pi/T. \end{cases} \quad (5.39)$$

This gives us the signal  $x_r(t)$  whose Fourier transform is

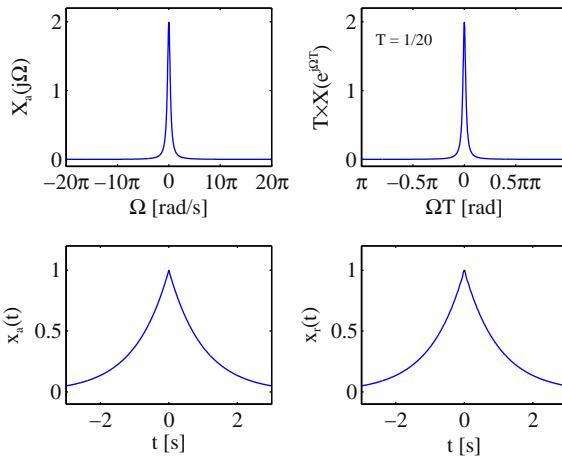
$$X_r(j\Omega) = \begin{cases} TX(e^{j\Omega T}), & |\Omega| \leq \pi/T, \\ 0, & |\Omega| > \pi/T, \end{cases} \quad (5.40)$$

where  $X(e^{j\Omega T})$  is given by (5.38). Figures 5.22 and 5.23 show the spectra  $X_a(j\Omega)$  and  $X(e^{j\Omega T})$  as well as the signals  $x_a(t)$  and  $x_r(t)$ . In Figure 5.22,  $T = 1/2$  whereas  $T = 1/20$  in Figure 5.23. In both cases,  $a = 1$ . For  $T = 1/2$ , the distortion is clearly visible, both in the time and frequency domains. If we instead choose  $T = 1/20$ , i.e., we increase the sampling frequency by a factor of ten, the distortion is no longer visually noticeable. An acceptable level of distortion can thus be obtained by selecting the sampling frequency high enough.

### 1.05.3.7 Distortion measure for energy signals

When the sampling theorem is not satisfied, the reconstructed signal  $x_r(t)$  will differ from the original signal  $x_a(t)$ . We thereby get an error  $e(t)$  according to

$$e(t) = x_r(t) - x_a(t). \quad (5.41)$$

**FIGURE 5.23**

Spectra and signals in the example in Section 1.05.3.6.1 for  $T = 1/20$ .

For energy signals, a common measure of the “size” of the distortion is the ratio between the error energy and the signal energy, whereby one computes  $E_e/E_x$  where  $E_x$  and  $E_e$  denote the energies of  $x_a(t)$  and  $e(t)$ , respectively. These energies are given by

$$E_x = \int_{-\infty}^{\infty} x_a^2(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X_a(j\Omega)|^2 d\Omega \quad (5.42)$$

and

$$E_e = \int_{-\infty}^{\infty} e^2(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |E(j\Omega)|^2 d\Omega, \quad (5.43)$$

where we have utilized Parseval’s formula. If we now make use of (5.41) we can rewrite (5.43) as

$$E_e = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X_r(j\Omega) - X_a(j\Omega)|^2 d\Omega. \quad (5.44)$$

When the reconstruction is done with an ideal PAM,  $X_r(j\Omega)$  is given by

$$X_r(j\Omega) = \begin{cases} TX(e^{j\Omega T}), & |\Omega| \leq \pi/T, \\ 0, & |\Omega| > \pi/T, \end{cases} \quad (5.45)$$

where

$$TX(e^{j\Omega T}) = \sum_{k=-\infty}^{\infty} X_a \left( j\Omega - j \frac{2\pi k}{T} \right). \quad (5.46)$$

The energy  $E_e$  can in this case be expressed as

$$E_e = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} |TX(e^{j\Omega T}) - X_a(j\Omega)|^2 d\Omega + \frac{1}{2\pi} \int_{|\Omega| \geq \pi/T} |X_a(j\Omega)|^2 d\Omega \quad (5.47)$$

or, alternatively,

$$E_e = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \left| \sum_{k \neq 0} X_a \left( j\Omega - j \frac{2\pi k}{T} \right) \right|^2 d\Omega + \frac{1}{2\pi} \int_{|\Omega| \geq \pi/T} |X_a(j\Omega)|^2 d\Omega. \quad (5.48)$$

The energies  $E_x$  and  $E_e$  are often difficult to compute analytically. In such cases, one has to use numerical computations.

#### 1.05.3.7.1 Example of distortion measure

We consider the same signals as in the example of Section 1.05.3.6.1 with  $a = 1$ . The energies  $E_x$  and  $E_e$  can in this case be computed as

$$E_x = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left( \frac{2}{\Omega^2 + 1} \right)^2 d\Omega \quad (5.49)$$

and

$$\begin{aligned} E_e &= \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \left( \frac{1 - e^{-2aT}}{1 - 2e^{-aT} \cos(\Omega T) + e^{-2aT}} - \frac{2}{\Omega^2 + 1} \right)^2 d\Omega \\ &\quad + \frac{1}{2\pi} \int_{|\Omega| \geq \pi/T} \left( \frac{2}{\Omega^2 + 1} \right)^2 d\Omega, \end{aligned} \quad (5.50)$$

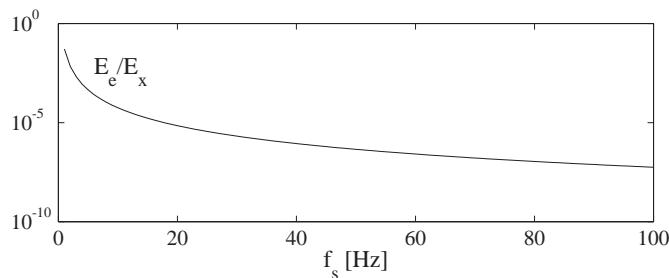
respectively. Since the integrands are symmetric here, we can alternatively express  $E_x$  and  $E_e$  according to

$$E_x = \frac{1}{\pi} \int_0^{\infty} \left( \frac{2}{\Omega^2 + 1} \right)^2 d\Omega \quad (5.51)$$

and

$$\begin{aligned} E_e &= \frac{1}{\pi} \int_0^{\pi/T} \left( \frac{1 - e^{-2aT}}{1 - 2e^{-aT} \cos(\Omega T) + e^{-2aT}} - \frac{2}{\Omega^2 + 1} \right)^2 d\Omega \\ &\quad + \frac{1}{\pi} \int_{\pi/T}^{\infty} \left( \frac{2}{\Omega^2 + 1} \right)^2 d\Omega \end{aligned} \quad (5.52)$$

respectively. The integral in (5.51), and the downmost one in (5.52), can easily be computed analytically whereby one obtains  $E_x = 1$ . For the uppermost integral in (5.52), it is more difficult to find a closed-form expression and we therefore compute it numerically. Figure 5.24 plots the ratio  $E_e/E_x$  for different sampling frequencies  $f_s = 1/T$ . (Here,  $E_e/E_x$  equals  $E_e$  since  $E_x = 1$ .) We see that the distortion as expected reduces when the sampling frequency is increased. The example in Section 1.05.3.6.1 considered the two cases  $f_s = 2$  and  $f_s = 20$ . In these cases, the distortion figures are about  $6.86 \times 10^{-3}$  and  $5.30 \times 10^{-6}$ , respectively.

**FIGURE 5.24**

Distortion as a function of the sampling frequency in the example of Section 1.05.3.7.1.

### 1.05.3.8 Bandpass sampling

We have so far considered lowpass signals meaning that the signal spectrum  $X_a(j\Omega)$  is contained in the low-frequency region  $\Omega \in [-\Omega_0, \Omega_0]$ . When the sampling frequency satisfy  $f_s > 2f_0 = \Omega_0/\pi$ , sampling and ideal reconstruction can theoretically be achieved. For a given sampling frequency,  $f_s = 1/T$ , the signal is in other words restricted to the frequency region  $\Omega \in [-\pi/T, \pi/T]$ . This frequency region is referred to as the first Nyquist band (or Nyquist zone). The theory can then be extended to bandpass signals for which the spectrum is contained in the band-pass frequency region  $\Omega \in [-N\pi/T, -(N-1)\pi/T] \cup [(N-1)\pi/T, N\pi/T]$  which is called the  $N$ th Nyquist band. Using again Poisson's summation formula (and following, e.g., the example in Section 1.05.3.2.1) it is then readily shown that the partial spectra do not overlap which means that the signal is not aliased and thus theoretically can be reconstructed without errors. In the reconstruction, the pulse  $p(t)$  must here correspond to a bandpass filter instead of a lowpass filter.

---

## 1.05.4 Sampling of stochastic processes

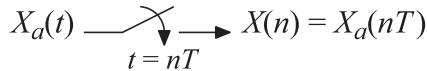
This section extends the sampling theory from deterministic signals to wide-sense stationary (WSS) stochastic processes.

### 1.05.4.1 Uniform sampling

Let  $X_a(t)$  be a continuous-time WSS stochastic process with mean  $m_{X_a}$ , autocorrelation function  $r_{X_a X_a}(\tau)$ , and power spectrum  $R_{X_a X_a}(j\Omega)$ . We now form  $X(n)$  by sampling  $X_a(t)$  uniformly according to

$$X(n) = X_a(nT). \quad (5.53)$$

Graphically, we represent uniform sampling of a stochastic process as in Figure 5.25.

**FIGURE 5.25**

Uniform sampling of a stochastic process.

As shown below,  $X(n)$  is also a WSS process whose mean  $m_X$ , autocorrelation sequence  $r_{XX}(k)$ , and power spectrum  $R_{XX}(e^{j\Omega T})$  are given by

$$m_X = m_{X_a}, \quad (5.54)$$

$$r_{XX}(k) = r_{X_a X_a}(kT), \quad (5.55)$$

and

$$R_{XX}(e^{j\Omega T}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} R_{X_a X_a} \left( j\Omega - j \frac{2\pi p}{T} \right). \quad (5.56)$$

Equation (5.56) is the same relation as in Poisson's summation formula for deterministic signals. One can therefore understand that the following sampling theorem can be formulated for WSS processes.

*Sampling theorem for WSS processes:* If a continuous-time WSS process  $X_a(t)$  is bandlimited to  $\Omega = \Omega_0$  ( $f = f_0$ ), i.e., if

$$R_{X_a X_a}(j\Omega) = 0, \quad |\Omega| > \Omega_0 = 2\pi f_0, \quad (5.57)$$

then,  $X_a(t)$  can be recovered from the samples  $X(n) = X_a(nT)$  if

$$f_s = \frac{1}{T} > 2f_0. \quad (5.58)$$

Here, recovered means that we can form a new process  $X_r(t)$  from  $X(n)$ , such that  $X_r(t)$  equals  $X_a(t)$  in the sense that  $E\{[X_r(t) - X_a(t)]^2\} = 0$ , where  $E$  denotes expectation.

Equations (5.54) and (5.55) follow immediately from their definitions according to

$$m_X = E\{X(n)\} = E\{X_a(nT)\} = m_{X_a} \quad (5.59)$$

and

$$\begin{aligned} r_{XX}(k) &= E\{X(n)X(n-k)\} \\ &= E\{X_a(nT)X_a(nT - kT)\} = r_{X_a X_a}(kT), \end{aligned} \quad (5.60)$$

respectively. This shows that  $X(n)$  is a WSS process since both the mean and autocorrelation sequence are independent of  $n$ . As for (5.56), it is first noted that the autocorrelation function can be written in terms of its inverse Fourier transform according to

$$r_{X_a X_a}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) e^{j\Omega\tau} d\Omega. \quad (5.61)$$

In particular, one obtains for  $\tau = kT$

$$r_{X_a X_a}(kT) = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) e^{j\Omega T k} d\Omega. \quad (5.62)$$

By dividing the indefinite integral into an infinite number of definite integrals, in the same way as we did in Section 1.05.3.2 when deriving Poisson's summation formula, we obtain

$$r_{X_a X_a}(kT) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{T} \sum_{p=-\infty}^{\infty} R_{X_a X_a}\left(j\Omega - j\frac{2\pi p}{T}\right) e^{j\Omega T k} d(\Omega T). \quad (5.63)$$

We also know that the autocorrelation sequence can be expressed in terms of its inverse Fourier transform according to

$$r_{XX}(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{XX}(e^{j\Omega T}) e^{j\Omega T k} d(\Omega T). \quad (5.64)$$

Since  $r_{XX}(k) = r_{X_a X_a}(kT)$ , and the Fourier transform is unique, the right-hand sides in the two equations above must be equal, which finally gives us (5.56).

### 1.05.4.2 Reconstruction of stochastic processes

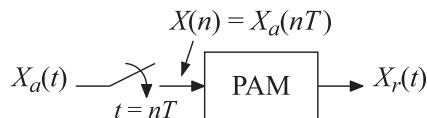
Reconstruction of stochastic processes can be performed with a PAM according to Figure 5.26. The reconstructed process  $X_r(t)$  is given by

$$X_r(t) = \sum_{n=-\infty}^{\infty} X(n) p(t - nT). \quad (5.65)$$

One can show that  $X_r(t)$  is equal to  $X_a(t)$  if the sampling theorem is fulfilled and  $p(t)$  is chosen as

$$p(t) = \frac{\sin(\pi t/T)}{\pi t/T} = \text{sinc}(t/T). \quad (5.66)$$

This is in analogy with ideal reconstruction of deterministic signals. Note again that equality holds in the sense that  $E[X_r(t) - X_a(t)]^2 = 0$ . However, in practice, one cannot have strictly bandlimited processes which introduces aliasing distortion in the sampling process. Further, errors are introduced in the reconstruction since only approximate sinc functions can be generated. The reconstructed process will therefore differ from the original process.



**FIGURE 5.26**

Sampling and reconstruction of a stochastic process using a PAM.

An additional problem here is that  $X_r(t)$  generally will not be a WSS process which complicates the analysis of distortion etc. To get around this dilemma, the sampling and reconstruction in (5.53) and (5.65) are replaced with

$$X(n) = X_a(nT + \Psi) \quad (5.67)$$

and

$$X_r(t) = \sum_{n=-\infty}^{\infty} X(n)p(t - nT - \Psi), \quad (5.68)$$

respectively, where  $\Psi$  is a stochastic variable that is uniformly distributed in the interval  $(0, T)$ . In practice, this means that each realization of the sampling and reconstruction process incorporates a stochastic delay  $\Psi$ . However, the sampling and reconstruction are still synchronized since they incorporate the same delay  $\Psi$ . The purpose of introducing the delay  $\Psi$  is that the reconstructed process  $X_r(t)$  then becomes WSS. The stochastic delay  $\Psi$  introduced in (5.67) does not have any implications as to stationarity and ensemble averages. That is,  $X(n)$  is still WSS and (5.54)–(5.56) still hold. Further, as shown below, the same stochastic delay  $\Psi$  introduced in (5.68) makes  $X_r(t)$  a WSS process with the following mean  $m_{X_r}$ , autocorrelation function  $r_{X_r X_r}(\tau)$ , and power spectrum  $R_{X_r X_r}(j\Omega)$ :

$$m_{X_r} = \frac{1}{T} m_X P(0), \quad (5.69)$$

$$r_{X_r X_r}(\tau) = \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) \int_{-\infty}^{\infty} p(u)p(u - \tau + kT)du, \quad (5.70)$$

and

$$R_{X_r X_r}(j\Omega) = \frac{1}{T} |P(j\Omega)|^2 R_{XX}(e^{j\Omega T}). \quad (5.71)$$

Moreover, as is also shown below, the reconstruction error  $E\{[X_r(t) - X_a(t)]^2\}$  is given by

$$\begin{aligned} E\{[X_r(t) - X_a(t)]^2\} &= \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} |P(j\Omega) - T|^2 R_{X_a X_a}(j\Omega) d\Omega \\ &\quad + \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} \sum_{p \neq 0} |P(j\Omega)|^2 R_{X_a X_a}\left(j\Omega - j\frac{2\pi p}{T}\right) d\Omega. \end{aligned} \quad (5.72)$$

From this equation, we see that, if  $p(t)$  is ideal according to (5.66), in which case  $P(j\Omega)$  is an ideal lowpass filter with passband up to  $\pi/T$  and gain  $T$ , then the reconstruction error becomes

$$\begin{aligned} E\{[X_r(t) - X_a(t)]^2\} &= \frac{1}{2\pi} \int_{|\Omega| \geq \pi/T} R_{X_a X_a}(j\Omega) d\Omega \\ &\quad + \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \sum_{p \neq 0} R_{X_a X_a}\left(j\Omega - j\frac{2\pi p}{T}\right) d\Omega, \end{aligned} \quad (5.73)$$

which can be rewritten as

$$E\{[X_r(t) - X_a(t)]^2\} = \frac{2}{\pi} \int_{\pi/T}^{\infty} R_{X_a X_a}(j\Omega) d\Omega. \quad (5.74)$$

If, in addition,  $X_a(t)$  is bandlimited to  $\pi/T$ , i.e.,  $R_{X_a X_a}(j\Omega) = 0$  for  $|\Omega| \geq \pi/T$ , it follows immediately from (5.74) that  $E[X_r(t) - X_a(t)]^2 = 0$  in which case perfect reconstruction is achieved.

We now show (5.69)–(5.71). The mean of  $X_r(t)$  becomes

$$\begin{aligned} m_{X_r} &= E\{X_r(t)\} = E\left\{\sum_{n=-\infty}^{\infty} X(n)p(t-nT-\Psi)\right\} \\ &= \sum_{n=-\infty}^{\infty} E\{X(n)p(t-nT-\Psi)\} \\ &= \sum_{n=-\infty}^{\infty} E\{X(n)\} E\{p(t-nT-\Psi)\}, \end{aligned} \quad (5.75)$$

where we have utilized that  $X(n)$  and  $p(t-nT-\Psi)$  are uncorrelated. Further, we have  $E\{X(n)\} = m_X$  and

$$E\{p(t-nT-\Psi)\} = \frac{1}{T} \int_0^T p(t-nT-\Psi)d\Psi = \frac{1}{T} \int_{t-nT-T}^{t-nT} p(v)dv \quad (5.76)$$

which gives us

$$m_{X_r} = \frac{1}{T} m_X \sum_{n=-\infty}^{\infty} \int_{t-nT-T}^{t-nT} p(v)dv = \frac{1}{T} m_X \int_{-\infty}^{\infty} p(v)dv = \frac{1}{T} m_X P(0). \quad (5.77)$$

We see that the mean is independent of  $t$ .

The autocorrelation function for  $X_r(t)$  becomes

$$\begin{aligned} r_{X_r X_r}(\tau) &= E\{X_r(t)X_r(t-\tau)\} \\ &= E\left\{\sum_{n=-\infty}^{\infty} X(n)p(t-nT-\Psi) \sum_{m=-\infty}^{\infty} X(m)p(t-\tau-mT-\Psi)\right\} \\ &= E\left\{\sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} X(n)X(m)p(t-nT-\Psi)p(t-\tau-mT-\Psi)\right\} \\ &= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} E\{X(n)X(m)\} \\ &\quad \times E\{p(t-nT-\Psi)p(t-\tau-mT-\Psi)\} \\ &= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} r_{XX}(n-m) \\ &\quad \times \frac{1}{T} \int_0^T p(t-nT-\Psi)p(t-\tau-mT-\Psi)d\Psi. \end{aligned} \quad (5.78)$$

The variable substitutions  $k = n - m$  and  $u = t - nT - \Psi$  yield

$$\begin{aligned} r_{X_r X_r}(\tau) &= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} r_{XX}(k) \frac{1}{T} \int_{t-nT-T}^{t-nT} p(u) p(u - \tau + kT) du \\ &= \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) \int_{-\infty}^{\infty} p(u) p(u - \tau + kT) du. \end{aligned} \quad (5.79)$$

We see that also the autocorrelation function is independent of  $t$ . Hence,  $X_r(t)$  is a WSS process. The power spectrum of  $X_r(t)$  becomes

$$\begin{aligned} R_{X_r X_r}(\tau) &= \int_{-\infty}^{\infty} r_{X_r X_r}(\tau) e^{-j\Omega\tau} d\tau \\ &= \int_{-\infty}^{\infty} \left( \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) \int_{-\infty}^{\infty} p(u) p(u - \tau + kT) du \right) e^{-j\Omega\tau} d\tau \\ &= \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) \int_{-\infty}^{\infty} p(u) \int_{-\infty}^{\infty} p(u - \tau + kT) e^{-j\Omega\tau} d\tau du. \end{aligned} \quad (5.80)$$

The variable substitution  $v = u - \tau + kT$  gives us

$$\begin{aligned} R_{X_r X_r}(\tau) &= \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) \int_{-\infty}^{\infty} p(u) \int_{-\infty}^{\infty} p(v) e^{-j\Omega(u-v+kT)} dv du \\ &= \frac{1}{T} \sum_{k=-\infty}^{\infty} r_{XX}(k) e^{-j\Omega Tk} \int_{-\infty}^{\infty} p(u) e^{-j\Omega u} du \int_{-\infty}^{\infty} p(v) e^{j\Omega v} dv du \\ &= \frac{1}{T} R_{XX}(e^{j\Omega T}) P(j\Omega) P^*(j\Omega) \\ &= \frac{1}{T} |P(j\Omega)|^2 R_{XX}(e^{j\Omega T}). \end{aligned} \quad (5.81)$$

The last two equalities hold when  $p(t)$  is a real function.

Next, (5.72) is shown. To this end, the reconstruction error is first expanded as

$$E \left\{ [X_r(t) - X_a(t)]^2 \right\} = E \left\{ X_r^2(t) \right\} + E \left\{ X_a^2(t) \right\} - 2E \{ X_r(t) X_a(t) \}. \quad (5.82)$$

The first term in (5.82) can be computed as

$$\begin{aligned} E \left\{ X_r^2(t) \right\} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_r X_r}(j\Omega) d\Omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{1}{T} |P(j\Omega)|^2 R_{XX}(e^{j\Omega T}) d\Omega \\ &= \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} |P(j\Omega)|^2 \sum_{p=-\infty}^{\infty} R_{X_a X_a} \left( j\Omega - j \frac{2\pi p}{T} \right) d\Omega. \end{aligned} \quad (5.83)$$

The second term in (5.82) is simply

$$E \left\{ X_a^2(t) \right\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) d\Omega. \quad (5.84)$$

In the third term of (5.82) we have

$$\begin{aligned} E \{ X_r(t) X_a(t) \} &= E \left\{ \sum_{n=-\infty}^{\infty} X_a(nT + \Psi) p(t - nT - \Psi) X_a(t) \right\} \\ &= E \left\{ \sum_{n=-\infty}^{\infty} r_{X_a X_a}(t - nT - \Psi) p(t - nT - \Psi) \right\} \end{aligned} \quad (5.85)$$

assuming that  $X_a(t)$  and  $\Psi$  are uncorrelated. Since it is assumed that  $\Psi$  is uniformly distributed on  $[0, T]$ , it follows from (5.85) that

$$E \{ X_r(t) X_a(t) \} = \sum_{n=-\infty}^{\infty} \frac{1}{T} \int_0^T r_{X_a X_a}(t - nT - \Psi) p(t - nT - \Psi) d\Psi. \quad (5.86)$$

The variable substitution  $u = t - nT - \Psi$  now yields

$$\begin{aligned} E \{ X_r(t) X_a(t) \} &= \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_{t-nT-T}^{t-nT} r_{X_a X_a}(u) p(u) du \\ &= \frac{1}{T} \int_{-\infty}^{\infty} r_{X_a X_a}(u) p(u) du. \end{aligned} \quad (5.87)$$

Using Parseval's relation

$$\int_{-\infty}^{\infty} x(t) y^*(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega) Y^*(j\Omega) d\Omega, \quad (5.88)$$

together with the fact that  $p(u)$  is real, i.e.,  $p(u) = p^*(u)$ , (5.87) can equivalently be written as

$$E \{ X_r(t) X_a(t) \} = \frac{1}{2\pi T} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) P^*(j\Omega) d\Omega. \quad (5.89)$$

Making use of the facts that  $R_{X_a X_a}(j\Omega) = R_{X_a X_a}(-j\Omega)$  and  $P^*(j\Omega) = P(-j\Omega)$ , (5.89) can equivalently be written as

$$E \{ X_r(t) X_a(t) \} = \frac{1}{2\pi T} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) P(j\Omega) d\Omega. \quad (5.90)$$

Inserting (5.83), (5.84), (5.89), and (5.90) into (5.82), we finally obtain

$$\begin{aligned}
 E[X_r(t) - X_a(t)]^2 &= \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} |P(j\Omega)|^2 \sum_{p=-\infty}^{\infty} R_{X_a X_a} \left( j\Omega - j\frac{2\pi p}{T} \right) d\Omega \\
 &\quad + \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) d\Omega \\
 &\quad - \frac{1}{2\pi T} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) P^*(j\Omega) d\Omega \\
 &\quad - \frac{1}{2\pi T} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) P(j\Omega) d\Omega \\
 &= \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} |P(j\Omega) - T|^2 R_{X_a X_a}(j\Omega) d\Omega \\
 &\quad + \frac{1}{2\pi T^2} \int_{-\infty}^{\infty} \sum_{p \neq 0} |P(j\Omega)|^2 R_{X_a X_a} \left( j\Omega - j\frac{2\pi p}{T} \right) d\Omega. \tag{5.91}
 \end{aligned}$$

#### 1.05.4.2.1 Example of reconstruction error power

Let  $X_a(t)$  be the output from an analog filter  $H(s)$  with the input  $X_{wb}(t)$ . Assume that  $X_{wb}(t)$  is white noise with zero mean and power spectrum  $\sigma_{X_{wb}}^2$ , and that  $H(s)$  is a first-order lowpass filter according to

$$H(s) = \frac{\Omega_0}{s + \Omega_0}. \tag{5.92}$$

We now form  $X(n)$  and  $X_r(t)$  through uniform sampling of  $X_a(t)$  and reconstruction with a PAM according to (5.67) and (5.68), respectively. We assume that  $P(j\Omega)$  is ideal and given by (5.66). We now wish to compute the average powers of  $X_a(t)$ ,  $X(n)$ , and  $X_r(t)$ . These quantities are obtained by integrating the corresponding power spectra.

The power spectrum of  $X_a(t)$  is

$$R_{X_a X_a}(j\Omega) = |H(j\Omega)|^2 R_{X_{wb} X_{wb}}(j\Omega), \tag{5.93}$$

where

$$R_{X_{wb} X_{wb}}(j\Omega) = \sigma_{X_{wb}}^2 \tag{5.94}$$

and

$$|H(j\Omega)|^2 = \frac{\Omega_0^2}{\Omega^2 + \Omega_0^2}. \tag{5.95}$$

The power of  $X_a(t)$  therefore becomes

$$\begin{aligned}
 E \left\{ X_a^2(t) \right\} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) d\Omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} |H(j\Omega)|^2 R_{X_{wb} X_{wb}}(j\Omega) d\Omega \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\Omega_0^2}{\Omega^2 + \Omega_0^2} \sigma_{X_{wb}}^2 d\Omega = \frac{\Omega_0 \sigma_{X_{wb}}^2}{2\pi} [\arctan(\Omega/\Omega_0)]_{-\infty}^{\infty} \\
 &= \frac{\Omega_0 \sigma_{X_{wb}}^2}{2}.
 \end{aligned} \tag{5.96}$$

The power of  $X(n)$  is the same as that of  $X_a(t)$  because

$$E \left\{ X^2(n) \right\} = r_{XX}(0) = r_{X_a X_a}(0) = E \left\{ X_a^2(t) \right\}. \tag{5.97}$$

The power spectrum of  $X_r(t)$  is

$$R_{X_r X_r}(j\Omega) = \frac{1}{T} |P(j\Omega)|^2 R_{XX}(e^{j\Omega T}), \tag{5.98}$$

where

$$R_{XX}(e^{j\Omega T}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} R_{X_a X_a} \left( j\Omega - j\frac{2\pi p}{T} \right). \tag{5.99}$$

When  $p(t)$  is given by (5.66), one obtains

$$P(j\Omega) = \begin{cases} T, & |\Omega| \leq \pi/T, \\ 0, & |\Omega| > \pi/T. \end{cases} \tag{5.100}$$

The power of  $X_r(t)$  thus becomes

$$\begin{aligned}
 E \left\{ X_r^2(t) \right\} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_r X_r}(j\Omega) d\Omega = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} T R_{XX}(e^{j\Omega T}) d\Omega \\
 &= \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \sum_{p=-\infty}^{\infty} R_{X_a X_a} \left( j\Omega - j\frac{2\pi p}{T} \right) d\Omega \\
 &= \frac{1}{2\pi} \sum_{p=-\infty}^{\infty} \int_{-\pi/T}^{\pi/T} R_{X_a X_a} \left( j\Omega - j\frac{2\pi p}{T} \right) d\Omega.
 \end{aligned} \tag{5.101}$$

The variable substitutions  $j\Omega - j2\pi p/T \rightarrow j\Omega$  finally give us

$$\begin{aligned}
 E \left\{ X_r^2(t) \right\} &= \frac{1}{2\pi} \sum_{p=-\infty}^{\infty} \int_{-\pi/T-j2\pi p/T}^{\pi/T-j2\pi p/T} R_{X_a X_a}(j\Omega) d\Omega \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{X_a X_a}(j\Omega) d\Omega = E \left\{ X_a^2(t) \right\}.
 \end{aligned} \tag{5.102}$$

That is, also the power of  $X_r(t)$  is the same as that of  $X_a(t)$ . We thus have

$$E \{ X_a^2(t) \} = E \{ X^2(n) \} = E \{ X_r^2(t) \}. \quad (5.103)$$

Finally, it is observed that the reconstruction error is

$$\begin{aligned} E \{ [X_r(t) - X_a(t)]^2 \} &= \frac{2}{\pi} \int_{\pi/T}^{\infty} R_{X_a X_a}(j\Omega) d\Omega \\ &= \frac{2}{\pi} \int_{\pi/T}^{\infty} |H(j\Omega)|^2 R_{X_{wb} X_{wb}}(j\Omega) d\Omega \\ &= \frac{2}{\pi} \int_{\pi/T}^{\infty} \frac{\Omega_0^2}{\Omega^2 + \Omega_0^2} \sigma_{X_{wb}}^2 d\Omega \\ &= \frac{2\Omega_0 \sigma_{X_{wb}}^2}{\pi} [\arctan(\Omega/\Omega_0)]_{\pi/T}^{\infty} \\ &= \frac{2\Omega_0 \sigma_{X_{wb}}^2}{\pi} (\pi/2 - \arctan[\pi/(\Omega_0 T)]). \end{aligned} \quad (5.104)$$

We see that, with a fixed  $\Omega_0$ ,  $E \{ [X_r(t) - X_a(t)]^2 \} \rightarrow 0$  when  $T \rightarrow 0$ , i.e., we can make the reconstruction error as small as desired by a proper selection of the sampling period  $T$  (sampling frequency  $f_s = 1/T$ ).

## 1.05.5 Nonuniform sampling and generalizations

This section considers the extension to nonuniform sampling [25] and generalizations. This appears in many different forms and contexts and it is beyond the scope of this section to present a comprehensive survey. Instead, we will concentrate on one particular application, namely  $M$ -channel time-interleaved A/D converters which in the simplest case corresponds to an  $M$ -periodic nonuniform sampling grid [26–28]. In this application, the sampling grid is close to a uniform grid and, thus, the obtained nonuniform-sequence is close to the desired uniform-sampling sequence. Nevertheless, reconstruction is required to recover the uniform-sampling sequence from the nonuniform-sampling sequence. The generalized case is an extension in the sense that the  $M$  channels experience different and general frequency responses.

### 1.05.5.1 Time-interleaved ADCs

Time interleaving of multiple parallel ADCs is a technique to increase the effective sampling rate of the overall converter. Using an  $M$ -channel time-interleaved ADC, the effective sampling rate is increased by a factor of  $M$ . Unfortunately, the effective resolution of the individual channel converters is not maintained in the overall converter because of channel mismatch errors. It is therefore necessary to compensate for these errors in order to restore the resolution [29,30]. The errors can broadly be divided into linear and nonlinear mismatch errors [30,31]. This section only deals with linear mismatch errors

in which case the channels are modeled as linear systems, thus having specific frequency responses. There are also static offset mismatch errors present but they are signal independent and straightforward to compensate for, and therefore not explicitly included in the formulas to be presented in this section. However, as noted later in Section 1.05.5.1, one can make use of the same frequency domain expression as that used for relating the input and output signals.

Up to a certain resolution, one can assume that the channel frequency responses have frequency-independent magnitude and phase delay responses, which corresponds to static gain and linear-phase (time-skew) mismatch errors. Without gain errors, this case corresponds to nonuniform sampling and the problem is then to recover the uniform-sampling sequence from the nonuniform-sampling sequence. Numerous papers have addressed this problem over the last decades, see for example [29, 32–37] and references therein. However, to reach a very high resolution for high-speed conversion, one needs to extend the channel model to general frequency responses, thus with frequency dependent magnitude and phase delay responses [30, 38]. In this case, one has to compensate for these frequency-response mismatch errors, not only static gain and linear-phase parts which correspond to approximations of the true responses. Recent papers have considered the more general problem [38–48] which corresponds to the “generalizations” mentioned above.

Before proceeding, it is also noted that calibration of time-interleaved ADCs requires estimation of and compensation for the channel frequency response mismatches. This section only discusses the compensation which requires that accurate models of the channel frequency responses are available. However, the estimation benefits from efficient compensation techniques, for example when the calibration is done through simultaneous estimation and compensation by minimizing an appropriate cost measure [42]. The compensation techniques to be discussed here can thus be used either after the channel frequency responses have been estimated or as a part of a calibration technique.

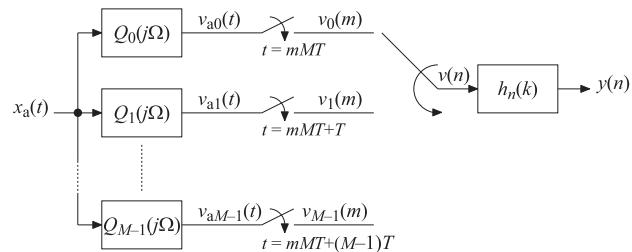
### 1.05.5.2 Problem formulation

The point of departure is that we have a continuous-time signal  $x_a(t)$  bandlimited to  $\Omega_0 < \pi/T$  which means that the Nyquist criterion for uniform sampling with a sampling frequency of  $1/T$  without aliasing is fulfilled. That is, sampling according to  $x(n) = x_a(nT)$  does not introduce aliasing and we have in the frequency domain

$$X(e^{j\Omega T}) = \frac{1}{T} X_a(j\Omega), \quad \Omega T \in [-\pi, \pi], \quad (5.105)$$

where  $X(e^{j\Omega T})$  and  $X_a(j\Omega)$  denote the Fourier transforms of the sequence  $x(n)$  and continuous-time signal  $x_a(t)$ , respectively. This means that  $x_a(t)$  can be recovered from  $x(n)$ . It is generally advisable to make  $\Omega_0$  correspond to some 80–90% of the overall Nyquist band as the computational complexity of the compensation system becomes prohibitive when it approaches 100%, as exemplified later in Section 1.05.5.6. However, normally this does not impose additional band limitations as a certain amount of oversampling is required anyhow to get reasonable requirements on the analog anti-aliasing filter that precedes the ADC, in accordance with Figure 5.39 seen later in Section 1.05.7.1, which depicts a typical system for conversion of an analog signal into its digital representation.

In an  $M$ -channel time-interleaved ADC, without correction, we do not obtain the desired uniform sequence  $x(n)$  but instead another sequence, say  $v(n)$ , through interleaving of  $M$  subsequences, as seen

**FIGURE 5.27**

$M$ -channel time-interleaved ADC with different channel frequency responses  $Q_n(j\Omega)$ ,  $n = 0, 1, \dots, M - 1$ , and an  $M$ -periodic time-varying compensation system with impulse response  $h_n(k) = h_{n+M}(k)$ .

in Figure 5.27. This can equivalently be viewed as if  $v(n)$  is obtained by applying  $x_a(t)$  to a time-varying continuous-time system and then sampling the output at  $t = nT$ .

Utilizing (5.105), the sequence  $v(n)$  can then be expressed via the inverse Fourier transform as

$$v(n) = \frac{1}{2\pi} \int_{-\Omega_0 T}^{\Omega_0 T} Q_n(j\Omega) X(e^{j\Omega T}) e^{j\Omega T n} d(\Omega T), \quad (5.106)$$

where  $Q_n(j\Omega) = Q_{n+M}(j\Omega)$  is an  $M$ -periodic time-varying system frequency response and  $Q_n(j\Omega)$ ,  $n = 0, 1, \dots, M - 1$ , constitute the  $M$  ADC channel frequency responses. For example, with static gain constants,  $g_n$ , and static time skews,  $d_n$  (given in percent of the sampling period  $T = 1/f_s$ ), the channel frequency responses are modeled as  $Q_n(j\Omega) = g_n e^{j\Omega T d_n}$ . With  $g_n = 1$ , this corresponds to  $v(n) = x_a(nT + d_n T)$  and thus a nonuniform sampling and reconstruction problem. It is also noted that the model above holds also in the more general case when  $M \rightarrow \infty$ , thus also for single-channel ADCs with a time-varying frequency response.

The problem to be solved can now be formulated as follows: Given the sequence  $v(n)$  in (5.106), form a new sequence  $y(n)$  that is to approximate the sequence  $x(n) = x_a(nT)$  as closely as desired. This problem is considered in Section 1.05.5.4.

### 1.05.5.2.1 Relaxed problem

The overall output  $y(n)$  should ideally approximate the sequence  $x(n) = x_a(nT)$ . This requires that all channel frequency responses  $Q_n(j\Omega)$  are known. In practice, it is customary to determine the ratio between the channel frequency responses and then match the channels. Typically, one chooses the first channel as a reference and then matches all other channels to this reference channel. In the reconstruction, this means that we set  $Q_0(j\Omega) = 1$  and divide the remaining  $Q_n(j\Omega)$  by the actual  $Q_0(j\Omega)$ . In the estimation, the frequency response ratios are obtained directly so we will never have available  $Q_n(j\Omega)$  but only an estimation of  $Q_n(j\Omega)/Q_0(j\Omega)$ . After the compensation, the overall ADC will then experience a linear distortion,  $Q_0(j\Omega)$ , because we then have  $Y(e^{j\Omega T}) = Q_0(j\Omega)X(e^{j\Omega T})$ . In other words, the linear distortion after compensation is determined by the frequency response of the reference ADC. In the compensation, this also means that the samples from the reference channel are taken as they are and only the other channels' samples are corrected. Finally, it is noted that, in the

overall system where the A/D converter is to be used, the linear distortion needs to be equalized but this problem exists for all types of A/D converters. In a larger composite system, like a communication system, a single equalizer can be used to equalize all linear distortion at the same time, emanating from the filters, A/D converters, communication channel, etc.

### 1.05.5.3 Reconstruction

Regardless whether the sequence  $x(n)$  has been obtained through uniform sampling of  $x_a(t)$  or through nonuniform sampling and/or its generalization (like in time-interleaved ADCs considered here), it is often desired to reconstruct  $x_a(t)$  from the generated sequence of samples  $x(n)$ . Thus, in the generalized case, it is desired to recover  $x_a(t)$  from the sequence  $v(n)$ . This can, in principle, be done in two different ways. The first way is to reconstruct  $x_a(t)$  directly from  $v(n)$  through analog reconstruction functions. Although it is known how to do this in principle (see e.g., [5, 25, 49, 50]), problems arise when it comes to practical implementations. In particular, it is very difficult to practically implement analog functions with high precision. It is therefore desired to use the second way which is to perform the reconstruction in the digital domain, i.e., to first recover  $x(n) = x_a(nT)$ . One then needs only one conventional digital-to-analog converter (DAC) and analog filter to obtain  $x_a(t)$ . Such components are much easier to implement than components that are to approximate complicated analog functions. Recovery of  $x(n)$  is also of interest even if  $x_a(t)$  is not to be reconstructed. For example, in receivers in digital communication systems,  $x(n)$  is the desired result. The subsequent two sections discusses two reconstruction methods for recovering  $x(n)$  from  $v(n)$  in time-interleaved ADCs.

### 1.05.5.4 Reconstruction using a time-varying FIR system

A time-interleaved ADC corresponds to a time-varying system, and the compensation thus corresponds to inverting such a system. For an  $M$ -channel system, this amounts to determining an  $M$ -periodic time-varying discrete-time system characterized by an  $M$ -periodic impulse response, say  $h_n(k) = h_{n+M}(k)$ , or, equivalently,  $M$  time-invariant impulse responses  $h_n(k)$ ,  $n = 0, 1, \dots, M - 1$ .

Using an  $N$ th-order  $M$ -periodic time-varying finite-length impulse response (FIR) system, the reconstructed output sequence,  $y(n)$ , is given by the convolution sum as

$$y(n) = \sum_{k=-N/2}^{N/2} v(n-k)h_n(k), \quad (5.107)$$

where  $h_n(k) = h_{n+M}(k)$  denotes the  $M$ -periodic impulse response. Thus, to correct each output sample, one needs to compute  $N + 1$  multiplications and  $N$  additions. It is convenient here to make use of a noncausal system, which means that the impulse response is assumed to be centered around  $k = 0$  which corresponds to zero delay. A causal system is then obtained by introducing a delay of  $N/2$  samples into the system. It is noted that we have assumed here that the order  $N$  of the system is even, to keep the notation simple. Odd-order systems can be used as well after some minor appropriate modifications.

To determine the impulse response coefficients, it is convenient to express the output  $y(n)$  in the time-frequency domain [27], instead of the time-domain given in (5.107). This will be done here in terms of an  $M$ -periodic time-frequency function  $A_n(j\Omega) = A_{n+M}(j\Omega)$ , or, equivalently,  $M$  frequency

functions  $A_n(j\Omega)$ ,  $n = 0, 1, \dots, M - 1$ . To this end, we insert (5.106) into (5.107), and interchange the summation and integration, and obtain

$$y(n) = \frac{1}{2\pi} \int_{-\Omega_0 T}^{\Omega_0 T} A_n(j\Omega) X(e^{j\Omega T}) e^{j\Omega T n} d(\Omega T), \quad (5.108)$$

where

$$A_n(j\Omega) = \sum_{k=-N/2}^{N/2} h_n(k) Q_{n-k}(j\Omega) e^{-j\Omega T k}. \quad (5.109)$$

Further, we can write the desired sequence  $x(n)$  in terms of the inverse Fourier transform according to

$$x(n) = \frac{1}{2\pi} \int_{-\Omega_0 T}^{\Omega_0 T} X(e^{j\Omega T}) e^{j\Omega T n} d(\Omega T). \quad (5.110)$$

Comparing (5.108) and (5.110), it is seen that perfect reconstruction (PR) is obtained if

$$A_n(j\Omega) = 1, \quad \Omega \in [-\Omega_0, \Omega_0] \quad (5.111)$$

for  $n = 0, 1, \dots, M - 1$ , because, then,  $y(n) = x(n)$  for all  $n$  as  $A_n(j\Omega)$  is  $M$ -periodic. Thus, if all  $A_n(j\Omega) = 1$ , we get the desired result  $x(n) = x_a(nT)$ . The problem is thus to determine the  $M$  impulse responses  $h_n(k)$ ,  $n = 0, 1, \dots, M - 1$ , so that  $A_n(j\Omega)$  approximate unity. If the channel frequency responses  $Q_n(j\Omega)$  are known, this can be done optimally in e.g., least-squares or minimax senses. In practice, one does not explicitly estimate  $Q_n(j\Omega)$  but instead  $Q_n(j\Omega)/Q_0(j\Omega)$  if  $Q_0(j\Omega)$  is the reference channel, in accordance with the discussion earlier in Section 1.05.5.2.1. The ratios  $Q_n(j\Omega)/Q_0(j\Omega)$  are then used in the design which corresponds to selecting  $h_0(k) = \delta(k)$  (unit impulse). It thus suffices to determine the  $M - 1$  impulse responses  $h_n(k)$  for  $n = 1, 2, \dots, M - 1$ , so that the corresponding  $A_n(j\Omega)$  approximate unity. After compensation with these filters, the overall ADC will experience the actual linear distortion  $Q_0(j\Omega)$ .

### 1.05.5.5 Error metrics

A common metric of ADCs is the signal-to-noise ratio (SNR). In this case, the aim is to minimize  $A_n(j\Omega) - 1$  in the least-squares sense which means that the quantities  $P_n$  given by

$$P_n = \frac{1}{2\pi} \int_{-\Omega_0 T}^{\Omega_0 T} |A_n(j\Omega) - 1|^2 d(\Omega T) \quad (5.112)$$

are minimized, either separately or simultaneously (in which case e.g., the mean of  $P_n$  is minimized). The minimization of  $P_n$  corresponds to the minimization of the expected error  $E\{[y(n) - x(n)]^2\}$  when the input signal is a stochastic process with a constant power spectrum in the region  $[-\Omega_0, \Omega_0]$  [27].

Another common metric of ADCs is the spurious-free dynamic range (SFDR). In this case, it is appropriate to write the input-output relation in the frequency domain in terms of a distortion function  $V_0(e^{j\Omega T})$  and  $M - 1$  aliasing functions  $V_m(e^{j\Omega T})$ ,  $m = 1, 2, \dots, M - 1$ . This is similar to frequency-domain input-output relations for multirate filter banks [24]. One can then write the output Fourier transform as

$$Y(e^{j\Omega T}) = V_0(e^{j\Omega T})X(e^{j\Omega T}) + \sum_{m=1}^{M-1} V_m(e^{j\Omega T})X(e^{j(\Omega T - 2\pi m/M)}), \quad (5.113)$$

where

$$V_0(e^{j\Omega T}) = \frac{1}{M} \sum_{n=0}^{M-1} B_n(e^{j\Omega T}) \quad (5.114)$$

and

$$V_m(e^{j\Omega T}) = \frac{1}{M} \sum_{n=0}^{M-1} e^{-j2\pi mn/M} B_n(e^{j(\Omega T - 2\pi m/M)}), \quad (5.115)$$

with  $B_n(e^{j\Omega T})$  being the  $2\pi$ -periodic extensions of  $A_n(j\Omega)$  in (5.109). That is,  $B_n(e^{j\Omega T}) = A_n(j\Omega)$  for  $-\pi \leq \Omega T \leq \pi$ . The distortion function corresponds to a regular frequency response seen from the input to the output, whereas the aliasing functions give the size of the aliased (frequency shifted) versions of the input. Specifically, a frequency component at  $\Omega = \Omega_0$  is affected by the modulus and phase of the distortion frequency response  $V_0(e^{j\Omega_0 T})$ , and we also get aliased versions at the “digital frequencies”  $\Omega_m T = \Omega_0 T + 2\pi m/M$ ,  $m = 1, 2, \dots, M - 1$ , with amplitudes determined by  $V_m(e^{j(\Omega_0 T + 2\pi m/M)})$ . Note also that, for real signals which always have frequency components at  $\Omega_0 T$  and  $-\Omega_0 T$  simultaneously, aliased versions also appear at  $-\Omega_0 T + 2\pi m/M$ ,  $m = 1, 2, \dots, M - 1$ .

Ideally, the distortion function should equal unity whereas the aliasing functions should be zero, in which case we have PR, because then we obtain  $Y(e^{j\Omega T}) = X(e^{j\Omega T})$  and thus  $y(n) = x(n)$ . In practice, we can only approximate PR. Typically, one then aims at minimizing two tolerances, say  $\delta_d$  and  $\delta_a$ , between unity and zero, respectively. One can then solve the problem in several slightly different ways. One typical way is to minimize  $\delta$  subject to the constraints

$$|V_0(e^{j\Omega T}) - 1| \leq \delta \quad \Omega T \in [-\Omega_0 T, \Omega_0 T] \quad (5.116)$$

and

$$|V_m(e^{j\Omega T})| \leq \delta (\delta_a / \delta_d), \quad \Omega T \in \Omega_m T, \quad (5.117)$$

for  $m = 1, 2, \dots, M - 1$ , where

$$\Omega_m T = \left[ -\Omega_0 T + \frac{2\pi m}{M}, \Omega_0 T + \frac{2\pi m}{M} \right]. \quad (5.118)$$

Clearly,  $|V_0(e^{j\Omega T})| \leq \delta_d$  and  $|V_m(e^{j\Omega T})| \leq \delta_a$  if  $\delta \leq \delta_d$ . It is noted that  $\delta_d$  and  $\delta_a$  are related through

$$|\delta_d| \leq (M - 1)|\delta_a| + |A_n(j\Omega) - 1|_{\min}, \quad (5.119)$$

which means that it is usually sufficient to ensure that the aliasing is suppressed as this implies a small distortion as well. Specifically, for the relaxed problem, (5.119) reduces to  $|\delta_d| \leq (M - 1)|\delta_a|$ , provided  $|V_0(e^{j\Omega T}) - 1|$  in (5.116) is replaced with  $|V_0(e^{j\Omega T}) - B_q(e^{j\Omega T})|$ ,  $q$  denoting the reference channel. The relation above is a consequence of the following equation which can be derived from (5.115):

$$\sum_{m=0}^{M-1} e^{j2\pi mq/M} V_m(e^{j(\Omega T + 2\pi m/M)}) = B_q(e^{j\Omega T}). \quad (5.120)$$

In a practical system, we cannot minimize  $\delta$  directly though as we do not have access to the distortion and aliasing functions. Instead,  $\delta$  is reduced indirectly to an acceptable level through some estimation and correction scheme.

#### 1.05.5.1 DC offset

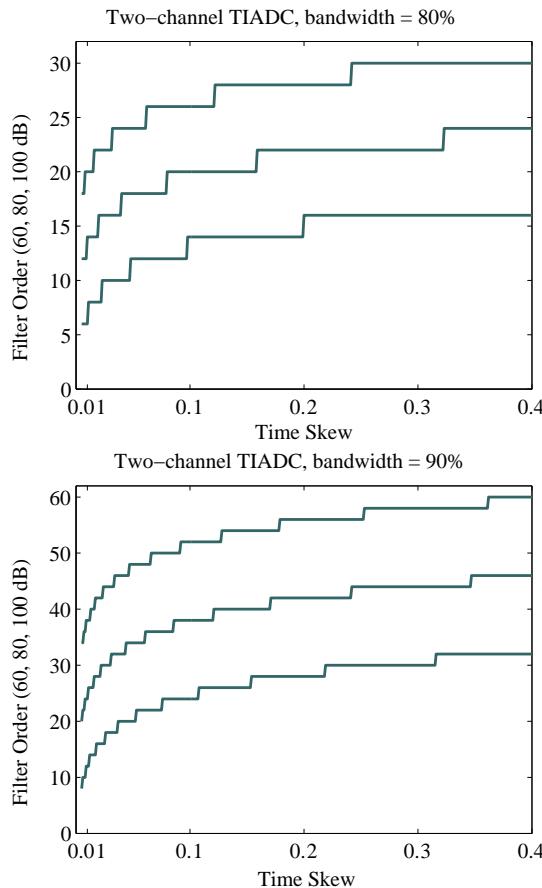
It is finally noted in this section that the representation in (5.113) can be used for DC offset mismatch as well, if we let  $X$  represent a DC signal, because it can be seen as a DC input that is time-interleaved and amplified differently in the different channels. In accordance with the discussion on aliasing above, it is seen that offset mismatches give rise to tones at  $\Omega_m T = 2\pi m/M, m = 0, 1, \dots, M - 1$ , with amplitudes determined by frequency independent  $V_m$ , as  $B_n$  in this case are frequency independent constants equaling the channel offset values. Specifically, with all  $B_n$  being equal, say,  $B_n = c$ , we obtain from (5.115) that  $V_0 = c$  and  $V_m = 0$  for  $m = 1, 2, \dots, M - 1$ , as we in this case have a regular DC offset.

#### 1.05.5.6 Oversampling

A good approximation can generally only be achieved if the input signal is bandlimited in accordance with the Nyquist theorem. It is generally advisable to use at least 10% oversampling as the complexity of the reconstructor otherwise may become prohibitive. However, if low complexity is not crucial, the amount of oversampling can be reduced. The increase of complexity with decreasing oversampling is illustrated in Figures 5.28 and 5.29 which plot the system (filter) order versus time skew in a two-channel time-interleaved ADCs for an reconstruction error [ $P_1$  in (5.112)] of  $-60$ ,  $-80$ , and  $-100$  dB. It is seen that the order is roughly doubled when the don't-care band between the upper band edge and the Nyquist frequency  $f_s/2$  is halved, (which occurs when we go from 80% to 90%, etc.). However, it is also seen that for small time skews and moderate reconstruction errors, one may increase the bandwidth as the order in such cases is relatively low. Recall that, for a reconstruction system of order  $N$ , in accordance with the convolution sum in (5.107), one needs to compute  $N + 1$  multiplications and  $N$  additions to correct each output sample.

#### 1.05.5.7 Reconstruction based on least-squares design

From the previous section, we see that the overall compensation system can be designed by minimizing the quantities  $P_n, n = 0, 1, \dots, M - 1$ , in (5.112). The quantities  $P_n$  can be minimized separately or

**FIGURE 5.28**

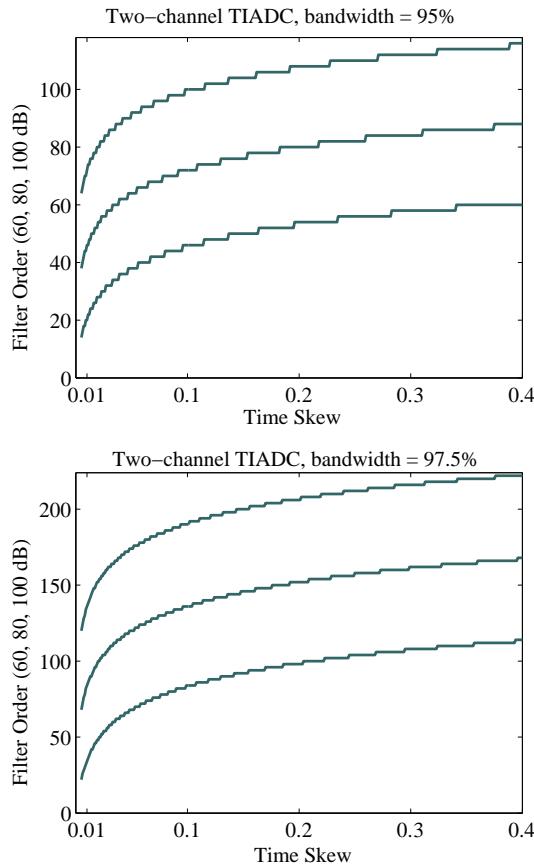
System (filter) order versus time skew in percent of the sampling period  $T = 1/f_s$  in a two-channel time-interleaved ADC for 80% and 90% of the Nyquist band.

simultaneously. In the former case, the corresponding  $M$  impulse responses  $h_n(k)$ ,  $n = 0, 1, \dots, M-1$ , can be computed separately as [45]

$$\mathbf{h}_n = -0.5\mathbf{S}_n^{-1}\mathbf{c}_n, \quad (5.121)$$

with  $c_{n,k}$ ,  $k = -N, -N+1, \dots, N$ , being

$$\begin{aligned} c_{n,k} &= -\frac{1}{\pi} \int_{-\Omega_0 T}^{\Omega_0 T} |Q_{n-k}(j\Omega)| \\ &\quad \times \cos(\Omega T k - \arg\{Q_{n-k}(j\Omega)\}) d(\Omega T), \end{aligned} \quad (5.122)$$

**FIGURE 5.29**

System (filter) order versus time skew in percent of the sampling period  $T = 1/f_s$  in a two-channel time-interleaved ADC for 95% and 97.5% of the Nyquist band.

$S_n$  being  $(N + 1) \times (N + 1)$  symmetric and positive definite matrices with entries  $s_{n,kp}$ ,  $k, p = -N/2, -N/2 + 1, \dots, N/2$  given by

$$s_{n,kp} = \frac{1}{2\pi} \int_{-\Omega_0 T}^{\Omega_0 T} |Q_{n-k}(j\Omega)| |Q_{n-p}(j\Omega)| \times \cos(\Omega T(p - k) + \arg\{Q_{n-k}(j\Omega)\} - \arg\{Q_{n-p}(j\Omega)\}) d(\Omega T) \quad (5.123)$$

and the constant  $C$  being  $C = \Omega_0 T / \pi$ . In order to compute  $h_n$  in (5.121), it is necessary to compute the integrals in (5.122) and (5.123), which generally have to be computed numerically.

Above, the overall compensation system is designed by determining the  $M$  filter impulse responses  $h_n(k)$  through  $M$  separate matrix inversions ( $M - 1$  in the practical relaxed case), where the size of

the matrices is  $(N + 1) \times (N + 1)$ , where  $N + 1$  is the filter impulse response length [45,47]. In the alternative case, corresponding to simultaneous design, the impulse responses  $h_n(k)$  are designed simultaneously. This was done in [39], which poses the design problem in terms of  $M$  synthesis filters that are designed simultaneously by inverting one matrix of size  $M(N + 1) \times M(N + 1)$ . Whereas the separate-design technique yields optimum filters with respect to the reconstruction error in the band of interest, the technique in [39] may give somewhat better results as to distortion and aliasing, because it includes those quantities in the overall error cost measure. The main advantage of the separate-design approach is that the design problem comprises  $M$  smaller subproblems instead of one large problem. It may therefore be more numerically robust and simpler to design and implement.

In addition to the above mentioned design techniques, [39,45,47], there are a few other related techniques [38,40,41,47]. In [38], separately designed synthesis filters were obtained through windowing techniques, which, however, are known to result in suboptimum filters. A somewhat different compensation technique, that also utilizes separately designed filters, was proposed in [40,41], but that technique needs additional cosine and sine modulators which increases the implementation cost of the compensation system. Finally, it is noted that one may alternatively use numerical optimization in accordance with [27], but this is costly and therefore less suitable for on-line design applications.

For all methods described above, new values of the impulse responses  $h_n(k)$  must be computed whenever the frequency responses  $Q_n(j\Omega)$  are changed, which occur in a practical time-interleaved ADC due to temperature variations etc. This requires recomputation of the filter coefficients using windowing techniques (including inverse Fourier transforms), matrix inversions, or numerical optimization. This may be acceptable for off-line design which can be adopted in, e.g., spectrum analysis applications. In real-time low-power applications, on the other hand, these on-line redesign procedures can become too costly and time consuming to implement. The next section discusses a compensation structure for which on-line design is not needed. This structure was introduced in [46]. A similar structure has been reported in [48].

### 1.05.5.8 Reconstruction using a polynomial based approach

As discussed in the previous section, the reconstruction can in principle be done through the time varying system  $h_n(k)$ . For an  $N$ th-order reconstructor, each system  $h_n(k)$  has then  $N + 1$  free parameters (multipliers) that must be estimated and implemented, and thus  $(M - 1)(N + 1)$  in total for a practical  $M$ -channel system (in the relaxed case). This becomes difficult from the estimation point of view and expensive to implement except for small values of  $M$ .

Depending on the behavior of the channel responses  $Q_n(j\Omega)$ , the number of free parameters may be reduced by modeling them as  $P$ th-order polynomials in  $j\Omega$  according to

$$Q_n(j\Omega) = \varepsilon_n^{(0)} \left[ 1 + \sum_{p=1}^P \varepsilon_n^{(p)} (j\Omega T)^p \right], \quad (5.124)$$

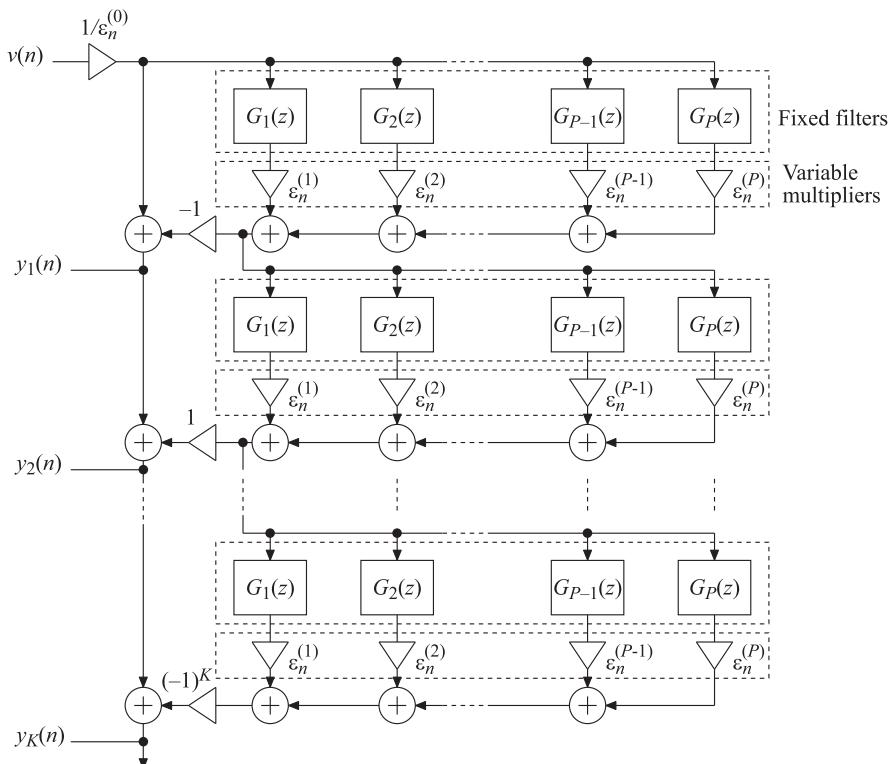
where  $\varepsilon_n^{(0)}$  is a gain constant (ideally equal to unity) and the remaining  $\varepsilon_n^{(p)}$  are constants (ideally equal to zero) used for modeling the frequency dependency. This does not impose a fundamental restriction as it is known that any frequency response can be approximated as closely as desired by a properly chosen polynomial and polynomial order. For example, when the system has frequency independent

time skews,  $d_n T$ ,  $Q_n(j\Omega)$  can be written as

$$Q_n(j\Omega) = \varepsilon_n^{(0)} e^{j\Omega T d_n} \approx \varepsilon_n^{(0)} [1 + \sum_{p=1}^P d_n^p (j\Omega T)^p / p!], \quad (5.125)$$

from which we obtain  $\varepsilon_n^{(p)} = d_n^p / p!$ . In this case it suffices to estimate two parameters per channel (three, including offset errors).

A general structure for the reconstruction based on the model in (5.124) is seen in Figure 5.30. This structure makes use of  $K$  sets of fixed subfilters  $G_p(z)$ ,  $p = 1, 2, \dots, P$ , that approximate the  $p$ th-order differentiators in (5.124). Figure 5.30 shows a noncausal filter structure for convenience. The corresponding causal structure is obtained by propagating  $D$  delay elements into each vertical branch and replacing  $\varepsilon_n^{(p)}$  with  $\varepsilon_{n-D}^{(p)}$ , where  $D$  is the (possibly approximate) delay of the subfilters, which can be either linear-phase finite-length impulse response (FIR) subfilters, or approximately linear-phase FIR or infinite-length impulse response (IIR) subfilters. Using  $N$ th-order linear-phase FIR subfilters,



**FIGURE 5.30**

Reconstruction based on the model in (5.124).

$D = N/2$ . The remaining errors in the output  $y_K(n)$  of the  $K$ -stage structure in Figure 5.30 are of orders  $(\Omega_0 T \varepsilon_n^{(p)})^{K+1}$ . This means that one can reach any desired error level by increasing the number of stages, provided  $|\Omega_0 T \varepsilon_n^{(p)}|$  is small which is the case in practical time-interleaved ADCs. The error will in practice decrease with  $K$  to a level that is determined by the channel model accuracy, i.e., by the distances between  $Q_n(j\Omega)$  and the actual frequency responses.

### 1.05.5.9 Performance discussion

The performance of the overall compensated ADC is naturally bounded by the performance of the individual converters. In this section, we only consider the compensation of the linear mismatch errors. Ultimately, the overall ADC's performance is therefore determined by the nonlinear distortion of the individual converters. In practice, the compensation structure presented above has virtually no effect on the nonlinear distortion. That is, the undesired frequency components caused by nonlinear distortion pass the compensation structure more or less unaffected. This is because the nonlinear distortion components, which are small themselves, passes a time-varying system that approximates a pure delay, provided  $\varepsilon_n^{(p)}$ ,  $p = 1, 2, \dots, P$ , are small which is the case in practice. This will be illustrated below in an example. It should be noted though that the time interleaving introduces nonlinear-distortion frequency components that are not present in the individual converters, but the total nonlinear-distortion power is virtually the same [30,31].

#### 1.05.5.9.1 Example of frequency response mismatch correction

To illustrate the theory above, an example is provided. It is assumed here that the number of channels is  $M = 4$ , the bandwidth is  $\Omega_0 T = 0.9\pi$ , and the four channel frequency responses  $Q_n(j\Omega)$ ,  $n = 0, 1, 2, 3$ , are modeled as

$$Q_n(j\Omega) = \frac{e^{j\Omega T d_n}}{1 + j\frac{\Omega}{\Omega_c}(1 + \Delta_n)} = \frac{e^{j\Omega T d_n}}{1 + j\Omega T \frac{f_s}{2\pi f_c}(1 + \Delta_n)}, \quad (5.126)$$

where  $f_s = 1/T$  denotes the sampling frequency whereas  $f_c$  and  $\Omega_c$  denote the 3-dB cutoff frequency and angular frequency, respectively, and  $d_n$  and  $\Delta_n$  correspond to analog matching errors. Here, we have chosen  $f_s/f_c = 1/2$ , which corresponds to a bandwidth of about two times the overall Nyquist band or, equivalently, eight times the individual channel ADC Nyquist band. Further, we have set  $d_0 = -0.02$ ,  $d_1 = 0.02$ ,  $d_2 = -0.01$ ,  $d_3 = 0.01$ ,  $\Delta_0 = -0.005$ ,  $\Delta_1 = 0.005$ ,  $\Delta_2 = -0.004$ , and  $\Delta_3 = 0.004$ . The term in the numerator in (5.126) models static aperture delay mismatches whereas the denominator models the sample-and-hold circuit [30,31,38]. The values for these parameters correspond to time and bandwidth mismatch spurs of some  $-30$  dB and  $-60$  dB, respectively, which is realistic at least for well-matched ADCs operating in their lower Nyquist bands. Hence, for a 10-bit resolution, one may ignore the bandwidth mismatch.

We assume that the zeroth channel is the reference channel. Consequently, we need to find the polynomial models of the rational functions  $Q_n(j\Omega)/Q_0(j\Omega)$ ,  $n = 0, 1, 2, 3$ . Including terms up to third order, we obtain from (5.126)

$$\frac{Q_n(j\Omega)}{Q_0(j\Omega)} = 1 + j\Omega T \varepsilon_n^{(1)} + (j\Omega T)^2 \varepsilon_n^{(2)} + \dots + (j\Omega T)^3 \varepsilon_n^{(3)}, \quad (5.127)$$

where

$$\varepsilon_n^{(1)} = d_n - d_0 + \Delta_0 - \Delta_n, \quad (5.128)$$

$$\varepsilon_n^{(2)} = \frac{(d_n - d_0)^2}{2} - \Delta_n(\Delta_0 - \Delta_n) - (d_n - d_0)(\Delta_0 - \Delta_n) \quad (5.129)$$

and

$$\varepsilon_n^{(3)} = \frac{(d_n - d_0)^3}{6} + \Delta_n^2(\Delta_0 - \Delta_n) - (d_n - d_0)\Delta_n(\Delta_0 - \Delta_n) + \frac{(d_n - d_0)^2}{2}(\Delta_0 - \Delta_n). \quad (5.130)$$

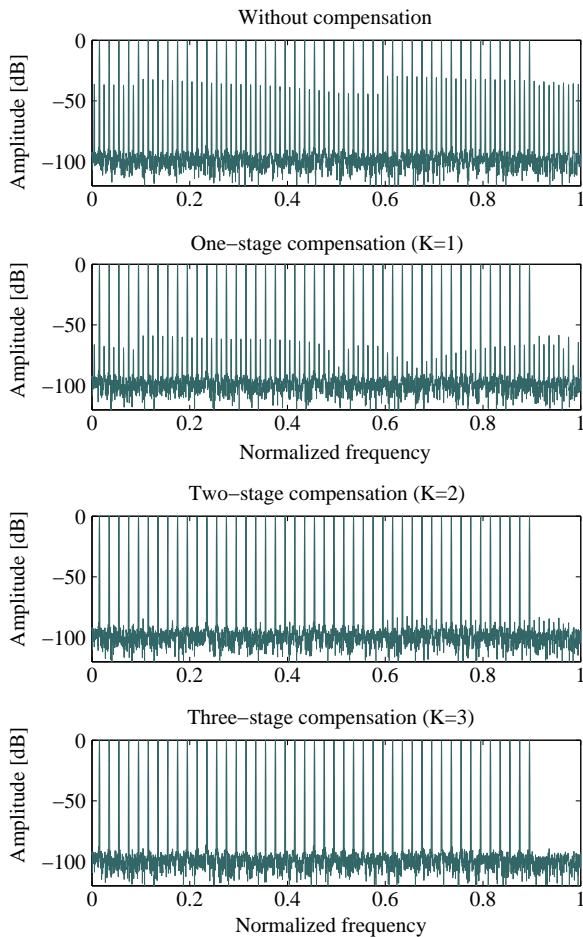
Here, we can compute the different parameters  $\varepsilon_n^{(p)}$  needed for the compensation, as we have assumed that we know the channel frequency responses. In practice, these parameters are obtained through some estimation technique for this purpose. It is noted that  $\varepsilon_0^{(1)} = \varepsilon_0^{(2)} = \varepsilon_0^{(3)} = 0$  as we have assumed that the zeroth channel is the reference, i.e., the samples from this channel are taken directly without compensation.

We have applied a 16-bit multi-sine input to this four-channel system. Without compensation, the output spectrum becomes as seen in Figure 5.31 which reveals large spurs. Figure 5.31 also plots the output spectra after compensation using one, two, and three stages, respectively. It is seen that, after three stages, the errors have been reduced to a level that corresponds to some 16 bits. This reveals that any desired error level can be reached in the frequency band  $\Omega \in [-\Omega_0, \Omega_0]$  by increasing the number of stages, provided the model order is high enough. In this example, the third-order model in (5.127) is required to reach 16 bits. Using instead a second-order model, we end up with spurs of some  $-80$  dB for  $K \geq 2$ , which corresponds to 13–14 bits. Hence, the approximation of (5.126) by a numerator polynomial of a certain order imposes a performance bound that cannot be surpassed by increasing  $K$  alone. In other words, to reach the desired result, one must select both the model order and number of stages appropriately. It is also noted that one stage suffices to reach some  $-60$  dB, thus about 10 bits.

Finally, to illustrate that the compensation has virtually no effect on nonlinear distortion, as discussed previously, we have also applied a two-tone input including the distortion term  $0.0001x^2(n)$  and plotted the output spectra in Figure 5.32 for the uncompensated system and compensated system with  $K = 3$ . From the figure, it is seen that the two large unwanted spurs (of some  $-40$  dB) caused by the interleaving have been suppressed whereas the smaller nonlinear-distortion spurs (of some  $-90$  dB to  $-85$  dB) are practically unaffected by the compensation.

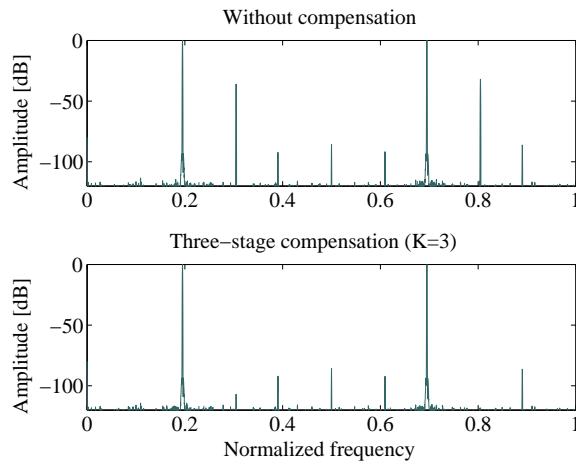
## 1.05.6 Quantization

In ideal linear systems, all signal values are represented with real numbers and all computations are done with infinite precision. In a practical implementation, one can however only use finite precision both for the input and output signal values and the internal signal values. Hence, instead of the ideal system in Figure 5.33a, we will in practice implement the one shown in Figure 5.33b. The boxes  $Q$  (with additional indices) represent quantizers. The function of a quantizer  $Q$  depends on the

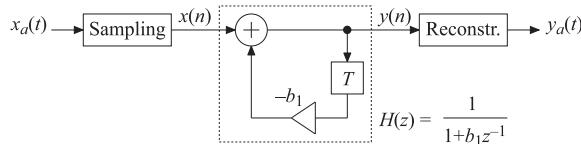
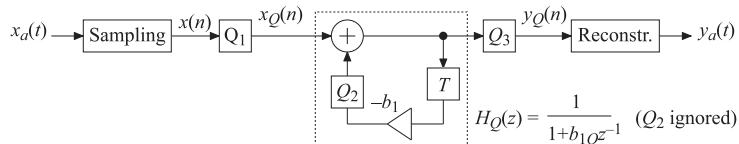
**FIGURE 5.31**

Spectra in the example of frequency response mismatch correction.

type of arithmetic that is to be used in the implementation. In general-purpose computers, one uses floating-point arithmetic. However, for many fixed functions (like filters), the large number range provided by floating-point arithmetic is not needed and thus becomes unnecessarily expensive to implement. In very-large-scale integration (VLSI) implementations, one therefore usually employs fixed-point arithmetic in order to minimize the implementation cost. We henceforth understand that fixed-point arithmetic is used. One may further choose between several different formats like sign and magnitude, one's complement, and two's complement, etc. Here, we consider two's complement format which is commonly used in VLSI implementations.

**FIGURE 5.32**

Spectra in the example of frequency response mismatch correction with the presence of a nonlinear distortion term  $0.0001x^2(n)$ .

**(a) Ideal linear system****(b) Actual nonlinear system****FIGURE 5.33**

(a) Ideal linear system. (b) Actual nonlinear system in a practical implementation.

Using two's complement arithmetic, a quantized number  $x_Q$ , within the number range  $-X_m \leq x_Q \leq X_m(1 - Q)$ , is represented with the  $B$ -bit fractional binary number  $x_B$  given by

$$x_B = -x_0 + \sum_{i=1}^{B-1} 2^{-i} x_i, \quad (5.131)$$

where  $x_i, i = 0, 1, \dots, B - 1$ , are either zero or one and where  $x_B$  is related to  $x_Q$  according to

$$x_Q = X_m x_B. \quad (5.132)$$

The quantity  $x_0$  is referred to as the sign bit which is zero for positive numbers (including zero) and one for negative numbers. Further, it is seen that  $-1 \leq x_B \leq 1 - Q$ . Hence, the quantity  $X_m$  is a scaling constant that is used for relating numbers within the range  $-1 \leq x_B \leq 1 - Q$  to numbers within the range  $-X_m \leq x_Q \leq X_m(1 - Q)$ . For example, in the context of A/D conversion, we can view  $X_m$  as the full-scale amplitude of the A/D converter. Another role of  $X_m$  is to enable representation of numbers larger than one in magnitude. In this case,  $X_m = 2^P$ , for some integer  $P$ , which can be interpreted as if the binary point has been moved to the right. Often, this scaling constant is not explicitly implemented but instead implicit.

The quantizers that inevitably are present introduce a number of errors that need to be analyzed when designing and implementing digital systems. The errors can be divided into four broad categories as discussed below. It is beyond the scope of this section to discuss all of these issues in detail. The subsequent subsections concentrate on quantization errors in the A/D conversion process and the related round-off errors in digital systems. For a comprehensive treatment of quantization effects, we refer in particular to the digital filter literature [51–53].

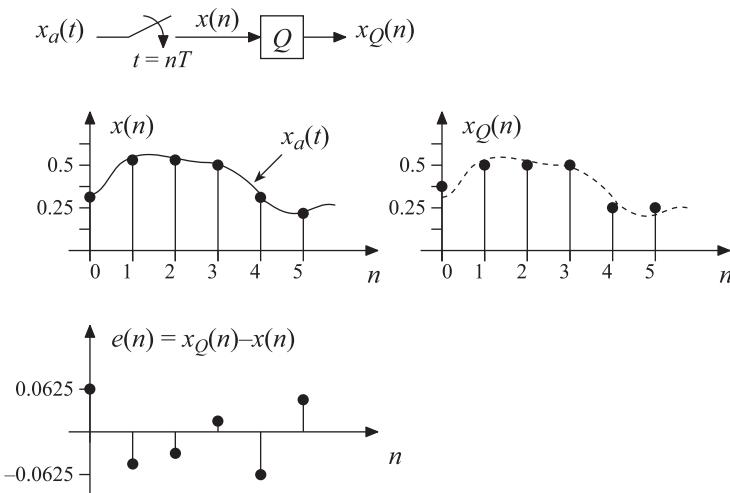
- Quantization errors in the A/D conversion—A/D conversion consists of sampling followed by quantization of the sample values. Thereby, a quantization error is introduced in the sampled signal.
- Overflow errors—Overflow errors occur when the available number range is exceeded. This may give rise to large sustaining errors referred to as parasitic large-scale oscillations. Digital system must be designed so that such errors are suppressed once the disturbance that incurred them vanishes. This is done by scaling the signal levels appropriately, not only at inputs and outputs but also inside the systems.
- Round-off errors—in digital systems, the results of arithmetic operations normally need be rounded (truncated). This gives rise to round-off errors at the output of the systems.
- Coefficient errors—the coefficients of a digital system must be represented with finite precision. This give rise to a static error in the corresponding transfer function, frequency response, etc.

### 1.05.6.1 Quantization errors in A/D conversion

A/D conversion consists of sampling followed by quantization according to Figure 5.34. In the sampling process, the samples  $x(n)$  are obtained as  $x(n) = x_a(nT)$ . Quantization means that each sample is represented in binary form with finite wordlength. This gives rise to a quantization error, i.e., an error in the representation of  $x_a(nT)$ , according to

$$e(n) = x_Q(n) - x(n). \quad (5.133)$$

This is also illustrated in Figure 5.34. The size of the quantization error depends on the types of binary representation and quantizer that are used. It is common to use fixed-point two's complement representation and uniform quantization in which case  $x(n)$  is rounded to the nearest number in the number representation.

**FIGURE 5.34**

A/D conversion—sampling and quantization.

If we assume that two's complement representation is used, and that the number range for the quantizer is  $-X_m \leq x_Q \leq X_m(1 - Q)$ , where  $Q$  is the quantization step,  $x_Q(n)$  can for a  $B$ -bit quantizer be written as

$$x_Q(n) = X_m x_B(n) \quad (5.134)$$

with

$$x_B(n) = -x_0(n) + \sum_{i=1}^{B-1} 2^{-i} x_i(n), \quad (5.135)$$

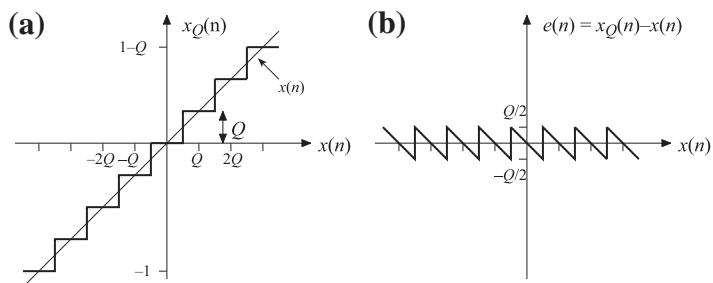
where  $x_i(n)$ ,  $i = 0, 1, \dots, B-1$ , are either zero or one. When the samples  $x_Q(n)$  are represented with  $B$  bits, they can take on  $2^B$  different values, as illustrated in Figure 5.35 for a uniform quantizer, assuming for simplicity that  $X_m = 1$ . The quantization step  $Q$  is given by

$$Q = X_m 2^{-(B-1)}. \quad (5.136)$$

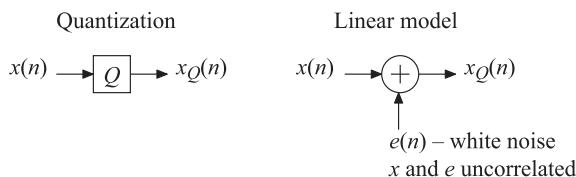
Quantization is a nonlinear operation which in general is much more difficult to analyze than a linear operation. To analyze the effects of the quantization errors, one therefore makes use of a linear model of the quantization according to Figure 5.36. In the linear model,  $x_Q(n)$  is obtained by adding  $x(n)$  and an error sequence  $e(n)$  according to

$$x_Q(n) = x(n) + e(n), \quad (5.137)$$

where it is assumed that  $x(n)$  and  $e(n)$  are uncorrelated. Further,  $e(n)$  is assumed to be uniformly distributed white noise. That is, one understands that the samples  $e(n)$  are uncorrelated and the errors that  $e(n)$  can take on occur with equal probability. This model is appropriate when  $x_a(t)$  varies irregularly since the quantization errors then vary more or less randomly and independently of previous errors.

**FIGURE 5.35**

(a) Uniform quantization with \$X\_m = 1\$ and \$B = 3\$ bits. (b) Quantization error.

**FIGURE 5.36**

Linear model of quantization.

Using uniform quantization, we have

$$|e(n)| \leq Q/2, \quad (5.138)$$

where \$Q\$ is the quantization step according to (5.136). Each sample \$e(n)\$ is therefore assumed to have the probability density function

$$f(e) = \begin{cases} 1/Q, & |e| \leq Q/2, \\ 0, & |e| > Q/2. \end{cases} \quad (5.139)$$

The mean value and variance of \$e(n)\$ become

$$m_e = 0 \quad (5.140)$$

and

$$\sigma_e^2 = \frac{Q^2}{12} = X_m^2 \frac{2^{-2(B-1)}}{12}, \quad (5.141)$$

respectively.

A measure that is commonly used for comparing the (useful) signal power and noise power is the SNR which is defined as

$$\text{SNR} = 10 \times \log_{10} \left( \frac{\text{Signal power}}{\text{Noise power}} \right). \quad (5.142)$$

Since the noise power is here the variance of  $e(n)$  [due to  $m_e = 0$ ], i.e.,  $\sigma_e^2$  in (5.141), we can write the SNR as

$$\text{SNR} = 10 \times \log_{10} (\text{Signal power}) - 10 \times \log_{10} \left( X_m^2 \frac{2^{-2(B-1)}}{12} \right), \quad (5.143)$$

which alternatively can be written as

$$\text{SNR} = 10 \times \log_{10} (\text{Signal power}) + 4.77 + 6.02B - 20 \times \log_{10} (X_m). \quad (5.144)$$

We note that the SNR increases by 6 dB for each bit that we add. For a full-scale sinusoidal signal with the amplitude  $X_m$ , the signal power is  $X_m^2/2$  which corresponds to  $20 \times \log_{10} (X_m) - 3.01$  dB. In this case, the SNR is

$$\text{SNR} = 1.76 + 6.02B. \quad (5.145)$$

In for example a CD player,  $B = 16$  bits. The SNR becomes in this case 98.1 dB.

### 1.05.6.2 Round-off errors

This section considers round-off errors (or round-off noise) in digital systems. The round-off noise is computed under the assumption that the quantization errors can be modeled as white noise, which is appropriate in many practical situations. It is further assumed that the digital system operate under normal conditions which means that overflows or other abnormal disturbances do not occur. In other words, we assume a linear model of the system incorporating one input signal and one or several noise sources. The problem is then to determine the round-off noise power and SNR at the output of the system. To this end, one makes use of the theory of linear systems excited with white stochastic processes. We will explain the principle by means of examples, where we assume  $X_m = 1$  for simplicity, but without loss of generality.

#### 1.05.6.2.1 Quantization and round-off noise example 1

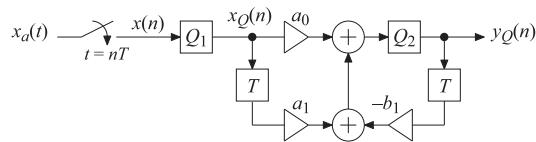
Consider a system represented by the structure in Figure 5.37a. First, sampling and quantization take place, with the quantization step  $Q_1$ , i.e., A/D conversion with  $B_1$  bits where  $Q_1 = 2^{-(B_1-1)}$ . The output from the quantization is  $x_Q(n)$  which subsequently is filtered. The filter is realized by a first-order direct form recursive IIR filter structure. In an implementation of this filter, the result of the arithmetic operations must at some point(s) inside the filter be rounded or truncated. Otherwise, the wordlength would increase in each iteration of the filter algorithm since it is recursive. Here, only one quantizer with the quantization step  $Q_2 = 2^{-(B_2-1)}$  is used and it is placed after the upper-most addition<sup>3</sup>.

We now wish to analyze the effects at the output of the filter caused by quantization errors. To be able to separate the origins of the errors we distinguish between the errors emanating from the A/D conversion, here referred to as quantization noise, and the errors coming from the quantizations of the

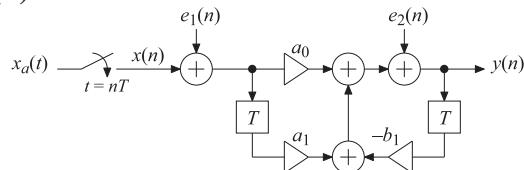
---

<sup>3</sup>The results of the remaining operations must thereby in this case be represented with more bits but this is only one option. For example, one may instead introduce quantizers after each operation. This will increase the noise at the output of the filter but, on the other hand, one may then use a shorter wordlength inside the filter. It is generally not obvious which alternative one should use in order to arrive at the cheapest implementation. It has to be investigated in each specific case.

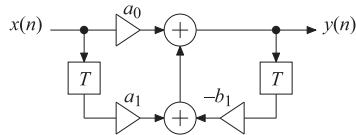
(a) Sampling, quantization, and filtering



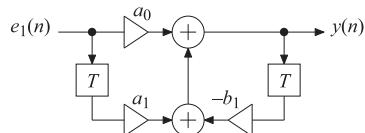
(b) Linear model



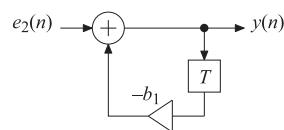
(c) Contribution from the input



(d) Contribution from noise source #1



(e) Contribution from noise source #2

**FIGURE 5.37**

Signal-flow graphs used for computation of the round-off noise.

arithmetic operations inside the filter, usually referred to as round-off noise. To be able to analyze the different errors, we make use of a linear model of the quantizations which results in the structure shown in Figure 5.37b. The error sequences  $e_1(n)$  and  $e_2(n)$  are assumed to be white noise with zero mean and variances  $\sigma_{e1}^2$  and  $\sigma_{e2}^2$ , respectively<sup>4</sup>. These error sequences are often called noise sources.

<sup>4</sup>When quantizing already quantized numbers, as is the case at the output of  $Q_2$ , one should actually use a discrete rectangular probability function. The difference is however negligible except when only a few bits are discarded.

Since we use linear models, we can regard the input and noise sources one at a time according to Figure 5.37c–e. The total output round-off noise variance is then computed as the sum

$$\sigma_y^2 = \sigma_{y1}^2 + \sigma_{y2}^2, \quad (5.146)$$

where

$$\sigma_{y1}^2 = \sigma_{e1}^2 \sum_{n=-\infty}^{\infty} h_1^2(n) \quad (5.147)$$

is the output quantization noise variance and

$$\sigma_{y2}^2 = \sigma_{e2}^2 \sum_{n=-\infty}^{\infty} h_2^2(n) \quad (5.148)$$

is the output round-off noise variance. Here,  $h_1(n)$  and  $h_2(n)$  are the impulse responses as seen from the respective noise source to the output. They can be obtained through inverse transformation of the corresponding transfer functions  $H_1(z)$  and  $H_2(z)$ .

Here, the transfer function from  $e_1(n)$  to  $y(n)$  is the same as that from  $x(n)$  to  $y(n)$  and given by

$$H_1(z) = \frac{a_0 + a_1 z^{-1}}{1 + b_1 z^{-1}}, \quad (5.149)$$

whereas the transfer function from  $e_2(n)$  to  $y(n)$  is

$$H_2(z) = \frac{1}{1 + b_1 z^{-1}}. \quad (5.150)$$

In the causal-filter case, the region of convergence is  $|z| > b_1$  for both of these transfer functions. We then get

$$h_1(n) = a_0(-b_1)^n u(n) + a_1(-b_1)^{n-1} u(n-1) \quad (5.151)$$

and

$$h_2(n) = (-b_1)^n u(n), \quad (5.152)$$

respectively, where  $u(n)$  denotes the unit-step sequence. This gives us the corresponding noise gains

$$\sum_{n=-\infty}^{\infty} h_1^2(n) = a_0^2 + \left(a_0 - \frac{a_1}{b_1}\right)^2 \sum_{n=1}^{\infty} b_1^{2n} = \frac{a_0^2 + a_1^2 - 2a_0a_1b_1}{1 - b_1^2} \quad (5.153)$$

and

$$\sum_{n=-\infty}^{\infty} h_2^2(n) = \sum_{n=0}^{\infty} b_1^{2n} = \frac{1}{1 - b_1^2}, \quad (5.154)$$

respectively.

If the quantizations  $Q_1$  and  $Q_2$  correspond to  $B_1$  and  $B_2$  bits, respectively, the variances  $\sigma_{e1}^2$  and  $\sigma_{e2}^2$  become

$$\sigma_{e1}^2 = \frac{Q_1^2}{12}, \quad \sigma_{e2}^2 = \frac{Q_2^2}{12}. \quad (5.155)$$

This gives us the variances  $\sigma_{y1}^2$  and  $\sigma_{y2}^2$  according to

$$\sigma_{y1}^2 = \sigma_{e1}^2 \sum_{n=-\infty}^{\infty} h_1^2(n) = \frac{Q_1^2}{12} \frac{a_0^2 + a_1^2 - 2a_0a_1b_1}{1 - b_1^2} \quad (5.156)$$

and

$$\sigma_{y2}^2 = \sigma_{e2}^2 \sum_{n=-\infty}^{\infty} h_2^2(n) = \frac{Q_2^2}{12} \frac{1}{1 - b_1^2}, \quad (5.157)$$

respectively. Finally, we obtain

$$\sigma_y^2 = \sigma_{y1}^2 + \sigma_{y2}^2 = \frac{Q_1^2}{12} \frac{a_0^2 + a_1^2 - 2a_0a_1b_1}{1 - b_1^2} + \frac{Q_2^2}{12} \frac{1}{1 - b_1^2}, \quad (5.158)$$

which can be rewritten as

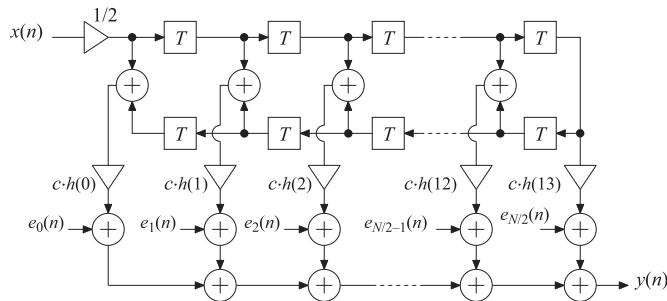
$$\sigma_y^2 = \frac{Q_1^2}{12} \frac{1}{1 - b_1^2} \left( a_0^2 + a_1^2 - 2a_0a_1b_1 + \frac{Q_2^2}{Q_1^2} \right). \quad (5.159)$$

From the equation above, we see that the contribution from the quantizer  $Q_2$  can be neglected if we choose  $Q_2$  sufficiently small as compared to  $Q_1$ . That is, by increasing the number of bits within the filter, the round-off noise can be made negligible compared to the quantization noise. This shows that digital systems can be implemented with as high resolution as desired. However, an increased internal wordlength also results in a more expensive implementation. One should therefore not use longer wordlengths than necessary to meet the specification at hand, typically given in terms of an SNR required. It should also be noted that the digital signal that in the end is to be D/A converted must be quantized to  $B_1$  bits if we want to use the same number of bits as that used in the A/D converter. This will give rise to additional round-off noise, equally large as that in the A/D conversion, which in turn degrades the resolution by half a bit. One can alleviate this problem by using oversampling techniques.

### 1.05.6.2.2 Quantization and round-off noise example 2

In this example, we compute the round-off noise at the output of a linear-phase direct-form FIR filter structure and the increase in SNR that is achieved by using so called  $L_2$ -norm scaling instead of safe scaling.

Assuming that quantization takes place after each multiplication, the linear model of the whole filter is as shown in Figure 5.38. Apparently, the errors pass directly to the output, i.e., the transfer function from each noise source to the filter's output is equal to unity. Assuming that the errors are uncorrelated white noise sources, the total output noise variance equals the sum of the individual variances. As

**FIGURE 5.38**

Linear model of a linear-phase FIR filter with quantizers.

a consequence, if all noise sources have the same variance, say  $\sigma_e^2$ , the total output variance for an  $N$ th-order filter,  $N$  being even, becomes

$$\sigma_y^2 = (N/2 + 1)\sigma_e^2. \quad (5.160)$$

With  $N = 26$ , as in this example, we thus have

$$\sigma_y^2 = 14\sigma_e^2. \quad (5.161)$$

This corresponds to a degradation of almost two bits as compared to the case where one quantizer is placed at the output. That is, the outputs of quantizers placed inside the filter need to be two bits longer than the output of a single quantizer located at the output, in order to achieve (roughly) the same round-off noise. On the other hand, this means that we can use adders with shorter wordlengths which reduces the implementation cost. The alternative of using one quantizer at the output requires the use of full-length adders inside the filter which increases the implementation cost.

Next, we consider the increase in SNR that is achieved by using so called  $L_2$ -norm scaling instead of safe scaling in the filter. Scaling of signal levels is used to avoid or reduce the probability of overflows, but it may also be used to increase the SNR. In the FIR filter in Figure 5.38, the only node that needs to be scaled (assuming a scaled input) is the output which is scaled by multiplying all impulse response values  $h(n)$  by the scaling coefficient  $c$ . In this filter, the output SNR is affected by the scaling because the output round-off noise variance is independent of the multiplier coefficient values whereas the output signal power is scaled with  $c^2$ . With safe scaling,  $c$  is chosen as

$$c = \frac{1}{\sum_{n=-\infty}^{\infty} |h(n)|}. \quad (5.162)$$

Using the convolution sum and the triangle inequality, one can show that  $c$  according to (5.162) guarantees no overflow. On the other hand, it also results in a lower SNR. The SNR can be increased by using a larger value of  $c$ , but this also comes with overflow with a certain probability. With  $L_2$ -norm scaling,  $c$  is chosen as

$$c = \frac{1}{\sqrt{\sum_{n=-\infty}^{\infty} |h(n)|^2}}. \quad (5.163)$$

Using  $L_2$ -norm scaling for white-noise Gaussian input signals, the probability of overflow at the output will be the same as the probability of overflow at the input.

There is thus a trade-off between high SNR and probability of overflow. In this example, we obtain  $c = 1.238477$  using safe scaling and  $c = 3.330192$  using  $L_2$ -norm scaling. The increase in SNR that is achieved by using  $L_2$ -norm scaling instead of safe scaling is therefore

$$20 \times \log_{10} (3.330192) - 20 \times \log_{10} (1.238477) \approx 8.59 \text{ [dB]} \quad (5.164)$$

which roughly corresponds to some 1.5 bits. That is, using  $L_2$ -norm scaling instead of safe scaling, we can implement the filter using 1.5 bits shorter data wordlength, which in practice means 1 bit or 2 bits shorter wordlength.

## 1.05.7 Oversampling techniques

This section discusses oversampling techniques which are used in A/D and D/A conversions for relaxing the requirements on the analog filters and quantizers.

### 1.05.7.1 Oversampled A/D converters

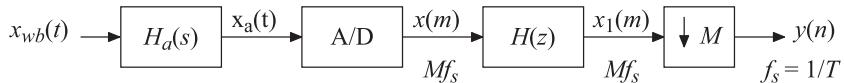
The principle of an oversampled A/D converter is shown in Figure 5.39. To elaborate on this principle, assume the typical spectra shown in Figure 5.40. Say that we have a signal  $x_{wb}(t)$  containing information in the baseband up to  $\pi/T$  and undesired frequency components above  $\pi/T$ , like wideband noise [Figure 5.40a]. The information can, in principle, be sampled and reconstructed without errors if the sampling frequency is  $f_s = 1/T$ . Due to the wideband noise, the sampling will however also introduce aliasing distortion. Theoretically, aliasing distortion can be avoided by band limiting the input signal using an ideal analog lowpass filter with passband up to  $\pi/T$ , but such a filter cannot be realized in practice. Hence, we must let the filter have a transition band. The wider this band is, the simpler it becomes to implement the filter. On the other hand, we must then increase the sampling frequency so as to avoid aliasing distortion.

Assume that we use  $M$  times oversampling, i.e., that we choose a sampling frequency  $f_{s1}$  according to

$$f_{s1} = Mf_s. \quad (5.165)$$

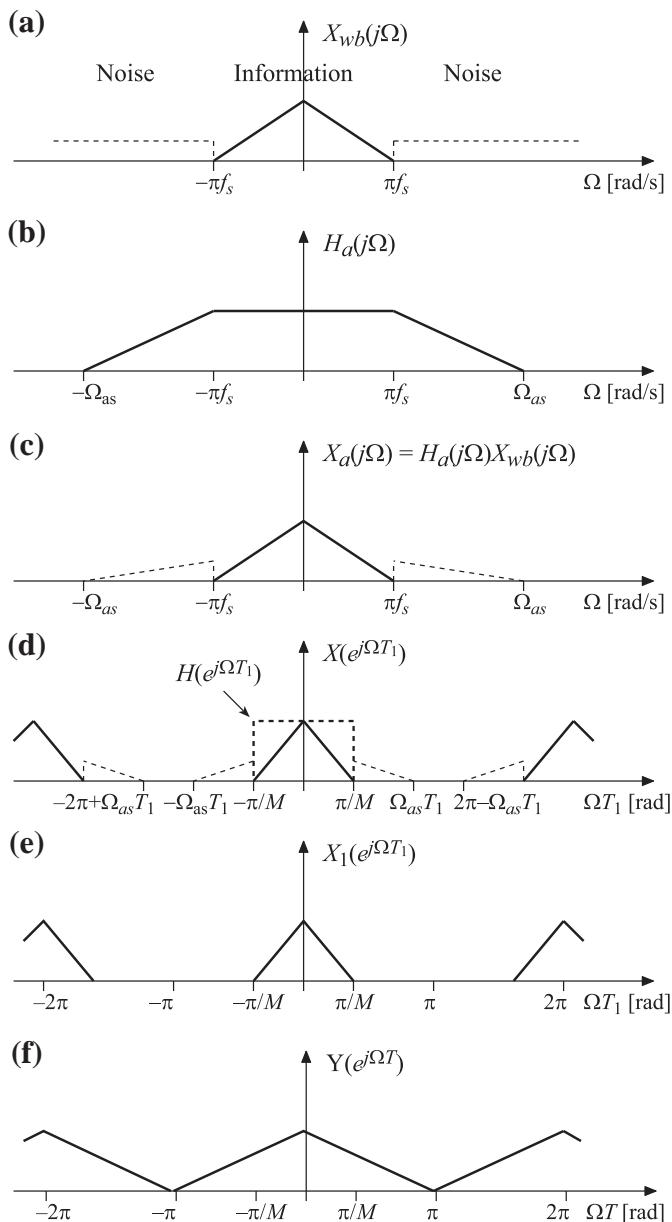
This corresponds to a sampling period  $T_1$  according to

$$T_1 = T/M. \quad (5.166)$$



**FIGURE 5.39**

Oversampled A/D converter.

**FIGURE 5.40**

Principle spectra in an oversampled A/D converter.

We can now avoid aliasing distortion by selecting the stopband edge of the anti-aliasing filter  $H_a(s)$  appropriately. It suffices to ensure that aliasing into the baseband is avoided since the information is then unaffected, see Figure 5.40c. Frequency components that are aliased into the remaining frequency region are allowed since they can be eliminated by the digital filter  $H(z)$  that succeeds the A/D converter, see Figure 5.40d. We see that the stopband edge of the analog filter must satisfy

$$2\pi - \Omega_{as} T_1 > \pi/M \iff \Omega_{as} < 2\pi M f_s - \pi f_s \quad (5.167)$$

in order to avoid aliasing distortion. In practice, the distortion is suppressed to a level determined by the attenuation of the analog anti-aliasing filter. To make sure that one can reduce the distortion to any desired level, (5.167) must be satisfied as the distortion then is determined by the filter's stopband attenuation.

In addition, one has to take into account that the digital filter has a transition band as well, which means that the overall design is somewhat more complicated than indicated above. Specifically, to avoid aliasing into the baseband, the information must therefore be limited to an angle frequency below  $\pi/T$ . If the information is limited to, say  $\Omega_c$ , then the transition band of the digital filter ranges from  $\Omega_c T_1$  to  $\pi/M$ .

We see that  $x_1(m)$  corresponds to the information oversampled by a factor of  $M$ . The A/D converter thus generates  $M$  times more samples per time unit than actually needed for reconstruction of the information. To make sure that the succeeding digital system does not have to work at a higher data rate than necessary, one therefore reduces the sampling rate through decimation (downsampling) by a factor of  $M$ . Since we have  $M$  times oversampling after  $H(z)$ , we can do this without loosing information. When  $M$  is an integer, the decimation is done by simply extracting only every  $M$ th sample in the output sequence of  $H(z)$  [24, 54, 55]. This is done by a downsample, represented by  $\downarrow M$ . (For a noninteger  $M$ , more elaborate schemes must be employed.) The result is the sequence  $y(n)$  corresponding to the information sampled with the frequency  $f_s = 1/T$ , see Figure 5.40f.

One motivation for using oversampling is, as mention earlier, that one in this way can relax the requirements on the analog anti-aliasing filter. Another reason is that oversampling enables the use of an A/D converter with lower resolution than the desired one, i.e., with fewer bits than the desired number of bits. The reason is that the digital filter removes a large part of the noise power whereas it, virtually, leaves the information unaffected. This means that the signal-to-noise ratio (SNR) of  $y(n)$  is higher than that of  $x(m)$ . To elaborate on this point, say that  $x(m)$  is quantized to  $B_1$  bits (and again assuming a maximum signal value of  $X_m = 1$  for simplicity). We then know that the noise variance is

$$\sigma_e^2 = \frac{2^{-2(B_1-1)}}{12}. \quad (5.168)$$

Further, we know that the noise variance at the output of the filter  $H(z)$ , say  $\sigma_h^2$ , is given by

$$\sigma_h^2 = \sigma_e^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\Omega T})|^2 d(\Omega T). \quad (5.169)$$

If we now for the sake of simplicity assume that  $H(z)$  is an ideal unity-gain lowpass filter with passband up to  $\pi/M$ , we get

$$\sigma_h^2 = \sigma_e^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\Omega T})|^2 d(\Omega T) = \frac{1}{2\pi} \int_{-\pi/M}^{\pi/M} |H(e^{j\Omega T})|^2 d(\Omega T) = \frac{\sigma_e^2}{M}. \quad (5.170)$$

Since  $H(z)$  leaves the information unaffected, the SNR is  $M$  times higher at the output of the filter than at the input of the filter, i.e., the output of the A/D converter. We can therefore obtain a desired resolution of  $B$  bits by using oversampling and an A/D converter with a lower resolution of  $B_1$  bits. From the equations above, we obtain

$$\frac{2^{-2(B_1-1)}}{12M} = \frac{2^{-2(B-1)}}{12}, \quad (5.171)$$

which gives us

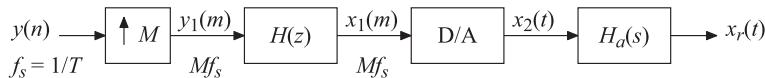
$$B_1 = B - 0.5 \log_2(M). \quad (5.172)$$

We see that for each doubling of the sampling frequency, we gain half a bit in resolution. By using oversampling techniques, one can thus use A/D converters with lower resolution than the desired one. In the extreme case, so called one-bit converters are employed. For such an A/D converter, one must however use a very high oversampling factor if a high resolution is desired. This causes problems in wideband applications since the sampling frequency then becomes too high to handle. By utilizing so called  $\Sigma\Delta$ -modulators, the oversampling factor required to achieve a certain resolution can be reduced [12]. The basic principle is to shape the noise so that most of its energy is located in the high-frequency region (for a lowpass converter). In this way, one gains more than a factor of  $M$  in noise reduction when using a lowpass filter with a passband width of  $\pi/M$ . It is beyond the scope of this chapter to discuss  $\Sigma\Delta$ -converters in detail. However, Section 1.05.8 deals with modeling of mixed-signal systems, which is useful in the analysis of such converters. That section will also provide an example of second-order  $\Sigma\Delta$ -modulators.

### 1.05.7.2 Oversampled D/A converters

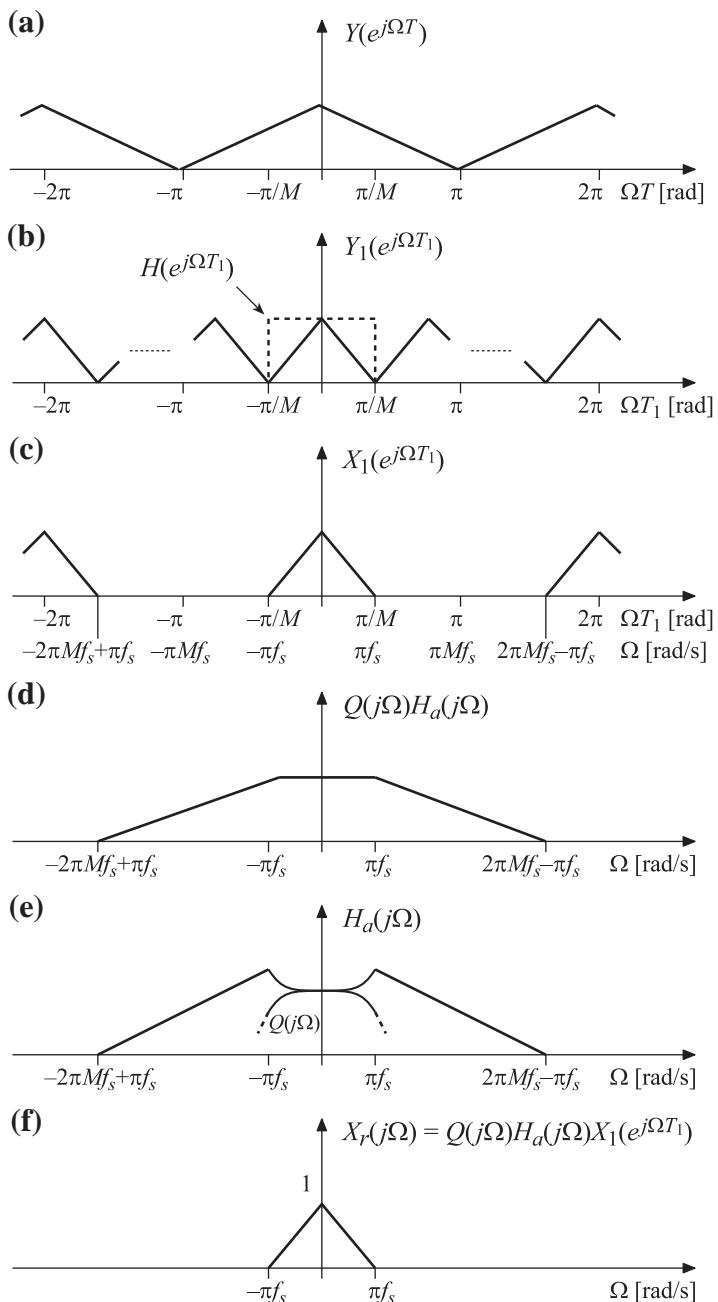
As discussed in the previous section, the requirements on the analog anti-aliasing filters can be relaxed by using oversampled A/D converters. The requirements on the analog reconstruction filter can likewise be relaxed by using oversampled D/A converters according to Figure 5.41.

We start with the sequence  $y(n)$  generated by the oversampled A/D converter in Figure 5.39. To illustrate the principle we use again the spectra in Figure 5.40. The spectrum of  $y(n)$  is then as shown in Figure 5.42a. The first step in an oversampled D/A converter is digital interpolation by a factor of  $M$ . This is done through an upsampler, represented by  $\uparrow M$ , followed by a filter  $H(z)$  [24,54,55]. The result is the sequence  $x_1(m)$  whose spectrum is shown in Figure 5.42c. We see that this sequence has the same spectrum as  $x_1(m)$  in the oversampled A/D converter, see Figures 5.39 and 5.40. In other words, it corresponds to the information oversampled by a factor of  $M$ . In the next step, reconstruction is performed through a D/A converter followed by a reconstruction filter. The D/A converter works at the higher sampling frequency  $Mf_s$ . As in Section 1.05.3.5.2, the D/A converter is here described by



**FIGURE 5.41**

Oversampled D/A converter.

**FIGURE 5.42**

Principle spectra in an oversampled D/A converter.

the pulse  $q(t)$  according to Figure 5.19d. Ideally,  $Q(j\Omega)H_a(j\Omega)$  and  $H_a(j\Omega)$  are then as shown in Figures 5.42d and e, respectively. We see that the stopband edge of the analog reconstruction filter must satisfy

$$\Omega_{as} < 2\pi M f_s - \pi f_s, \quad (5.173)$$

which is the same requirement as that of the anti-aliasing filter in an oversampled A/D converter. By increasing  $M$ , we thus relax the filter requirements since a wider transition band is then allowed.

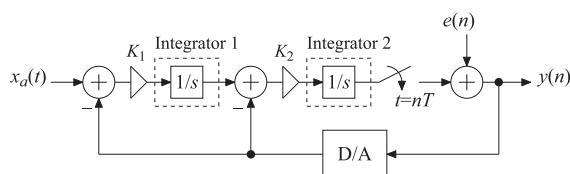
In practice, one can let the digital filter equalize the deviation of  $Q(j\Omega)$ , which is needed since  $Q(j\Omega)$  is not constant in the baseband. That is,  $H(z)$  is designed so that

$$|H(e^{j\Omega T_1})Q(j\Omega)| \approx \text{constant}, \quad \text{for } |\Omega| < \pi f_s. \quad (5.174)$$

As analog reconstruction filter, one can then use a conventional lowpass filter with (approximately) constant gain in both the passband and stopband. In that way, the requirements on this filter are further relaxed. Instead, the requirements on the digital filter has become more severe, but it is much easier to implement digital filters with stringent requirements than analog filters.

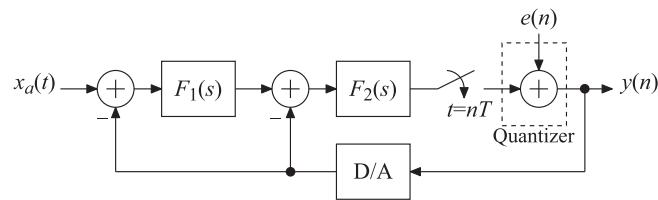
### 1.05.8 Discrete-time modeling of mixed-signal systems

Many systems today are mixed-signal systems conducting both continuous-time and discrete-time signal processing. From the analysis point of view, it is often advantageous to use a discrete-time model of the mixed-signal system. This section presents an analysis method for linear time-invariant systems containing sampling and reconstruction inside a feedback loop [56]. Using this method, the input/output stability as well as the frequency responses of such mixed systems can easily be analyzed. One application is ADCs realized with sigma-delta modulators ( $\Sigma\Delta$ -modulators) with continuous-time input signals. An example is seen in Figure 5.43 which is a linear model of a double feedback loop continuous-time-input  $\Sigma\Delta$ -modulator. Figure 5.44 shows a generalization where  $F_0(s)$  and  $F_1(s)$  are transfer functions of general loop filters. However, the analysis method that will be presented here is applicable to any linear and time-invariant system that can be represented by the general structure shown in Figure 5.45, where  $N$  is a two-input/one-output system and where the single output is uniformly sampled and reconstructed through pulse amplitude modulation before being fed back again to  $N$ . Apparently, the structures of Figures 5.43 and 5.44 belong to the class of all structures that can be represented by the general structure of Figure 5.45.

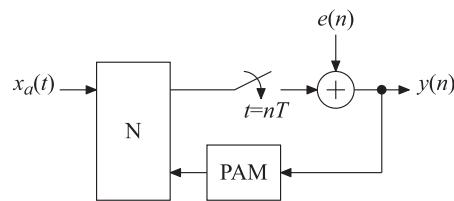


**FIGURE 5.43**

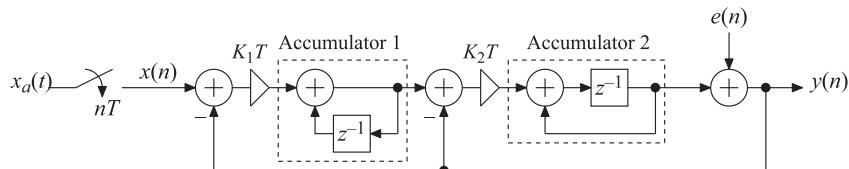
Linear model of a second-order double feedback loop continuous-time-input  $\Sigma\Delta$ -modulator.


**FIGURE 5.44**

Generalization of the scheme in Figure 5.44.


**FIGURE 5.45**

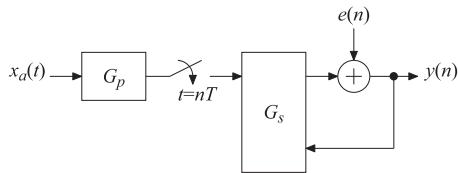
General hybrid continuous-time/discrete-time system setup with a feedback loop.


**FIGURE 5.46**

 Linear model of a second-order discrete-time-input  $\Sigma\Delta$ -modulator.

It is of practical interest to analyze systems such as that of Figure 5.43 with respect to stability as well as frequency responses as seen from  $x_a(t)$  and  $e(n)$  to the output  $y(n)$ . Here,  $e(n)$  models quantization errors that occur in the quantizer that follows the sampler. For continuous-time-input  $\Sigma\Delta$ -modulators, stability and frequency response analyses are often performed by using the so called “sampled-data equivalent” discrete-time-input  $\Sigma\Delta$ -modulator seen in Figure 5.46 [12,13,57], where integrators are replaced with accumulators. However, these “sampled-data equivalents” are only approximations of the actual behavior of the continuous-time-input  $\Sigma\Delta$ -modulator, as will be exemplified later in this section. That is, they do not correctly describe the properties of the continuous-time-input circuit.

To get a correct description, the system at hand can instead be represented by the system in Figure 5.47, where the continuous-time input signal  $x_a(t)$  is prefiltered through a continuous-time system  $G_p$  before being sampled uniformly. The resulting sampled signal and the system output signal

**FIGURE 5.47**

State-space representation of the system in Figure 5.48.

are then used as input signals to a linear discrete-time system  $G_s$ , usually represented in state-space form [12,58]. It is then required to determine a corresponding input-output difference equation using state variable representation. To that end, a state matrix,  $A$ , of  $G_s$  is needed, requiring the computation of the power series

$$A = e^{A_c} = \sum_{k=0}^{\infty} \frac{1}{k!} A_c^k, \quad (5.175)$$

where  $A_c$  is the state matrix of the system  $N$  in Figure 5.45, as well as the inversion of two particular matrices. This method gives a correct description of the circuit at hand but requires the computation of (5.175), which can be a non-trivial task, and the above mentioned matrices often are singular. In fact, this is always the case when  $F_0(s)$  and  $F_1(s)$  represent ideal integrators  $1/s$ .

In this section, we discuss an alternative method for the analysis of the external stability as well as the frequency responses from the inputs  $x_a(t)$  and  $e(n)$  to the output  $y(n)$  of the general system in Figure 5.47 [56]. This is done by using the models shown in Figure 5.48. It is readily shown that all linear time-invariant systems belonging to the class of systems that can be represented by the system in Figure 5.45, can be represented as in Figure 5.48. It should however be noted that the systems in Figure 5.48 are only used for input/output analysis purposes and do not capture the internal properties of the underlying system of interest. From the input/output stability and frequency response points of view, the systems in Figure 5.48 are equivalent to their underlying system in Figure 5.45.

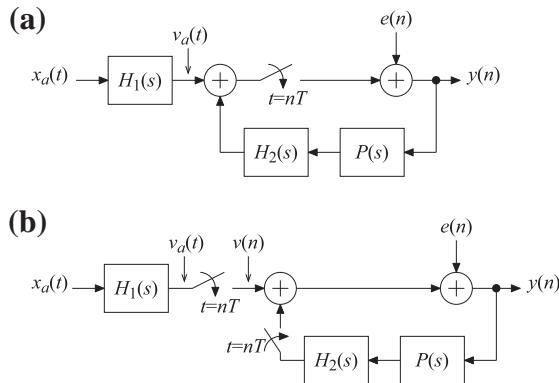
### 1.05.8.1 Input/output relations

For mixed-signal systems, direct use of transfer functions is less appropriate as they depend on two variables ( $s$  and  $z$ ). Therefore, the analysis becomes somewhat more involved than for pure discrete-time or continuous-time systems. The mixed-signal systems are analyzed here by using equivalent discrete-time systems that produce the same output samples as the actual mixed-signal system.

To be precise, stability is analyzed by first determining  $G(z)$  in the  $z$ -transform relation between  $y(n)$ ,  $v(n)$ , and  $e(n)$  given as

$$Y(z) = G(z)V(z) + G(z)E(z). \quad (5.176)$$

Stability is then investigated by considering the poles of  $G(z)$  and the boundedness of the resulting output signal. As to the frequency responses, as seen from  $x_a(t)$  and  $e(n)$  to the output  $y(n)$ , it is


**FIGURE 5.48**

Two equivalent models for the system in Figure 5.45.

appropriate to express the output Fourier transform as

$$Y(e^{j\Omega T}) = H(j\Omega) \frac{1}{T} X_a(j\Omega) + G(e^{j\Omega T}) E(e^{j\Omega T}), \quad (5.177)$$

where  $H(j\Omega)$  represents the signal frequency response whereas  $G(e^{j\Omega T})$  represents the noise frequency response. The latter is immediately obtained from the noise transfer function  $G(z)$  by evaluating it for  $z = e^{j\Omega T}$ .

In order to arrive at (5.176) and (5.177), the models of Figure 5.48 are used to represent the system in Figure 5.45. In Figure 5.48,  $P(s)$  represents pulse amplitude modulation.

### 1.05.8.2 Determining $G(z)$

We will make use of the generalized Poisson's summation formula which relates the Laplace transform of a signal  $x_a(t)$  with the corresponding  $z$ -transform of its uniformly sampled version  $x(n) = x_a(nT)$  according to

$$X(e^{sT}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} X_a \left( s - j \frac{2\pi p}{T} \right). \quad (5.178)$$

with the  $z$ -transform evaluated for  $z = e^{sT}$ . Further, we will make use of the following equation that relates the output  $x_r(t)$  of the pulse amplitude modulator  $p(t)$  with its input  $x(n)$ :

$$X_r(s) = P(s) X(e^{sT}). \quad (5.179)$$

Using the above relations, the  $z$ -transform of the output signal  $y(n)$  in Figure 5.48a is easily derived as

$$\begin{aligned} Y(e^{sT}) &= E(e^{sT}) \\ &+ \frac{1}{T} \sum_{p=-\infty}^{\infty} H_1 \left( s - j \frac{2\pi p}{T} \right) X_a \left( s - j \frac{2\pi p}{T} \right) \\ &- Y(e^{sT}) \frac{1}{T} \sum_{p=-\infty}^{\infty} H_2 \left( s - j \frac{2\pi p}{T} \right) P \left( s - j \frac{2\pi p}{T} \right), \end{aligned} \quad (5.180)$$

where  $E(z)$  is the  $z$ -transform of  $e(n)$  and  $P(s)$  is defined by the PAM used. Here, a rectangular pulse  $p(t)$  is assumed according to

$$p(t) = \begin{cases} 1, & 0 \leq t \leq T, \\ 0, & \text{otherwise,} \end{cases} \quad (5.181)$$

which is commonly used in D/A conversion. This gives

$$P(s) = \frac{1 - e^{-sT}}{s}. \quad (5.182)$$

Equation (5.180) can be rearranged according to (5.176) with

$$V(e^{sT}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} H_1 \left( s - j \frac{2\pi p}{T} \right) X_a \left( s - j \frac{2\pi p}{T} \right), \quad (5.183)$$

which corresponds to the  $z$ -transform of  $v(n)$  in Figure 5.48b, i.e., a uniformly sampled version of  $v_a(t)$ , and

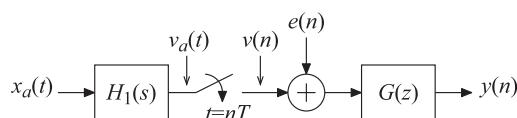
$$G(z) = \frac{1}{1 + L(z)} = \frac{N(z)}{D(z)} \quad (5.184)$$

with

$$L(e^{sT}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} H_2 \left( s - j \frac{2\pi p}{T} \right) P \left( s - j \frac{2\pi p}{T} \right). \quad (5.185)$$

Equation (5.176) can, further, be interpreted as shown in Figure 5.49.

In Figure 5.48,  $H_1(s)$  models the output signal dependency of the input signal whereas  $H_2(s)$  describes the behavior of the feedback signal. Eqs. (5.176) and (5.183)–(5.185) do not give us suitable



**FIGURE 5.49**

Representation of the system in Figure 5.48 in view of (5.176).

expressions for the stability analysis. In order to express  $L(z)$  in a more suitable form for this purpose, we make use of the observation that  $L(z)$  is the  $z$ -transform of the sequence  $l(n) = l_a(nT)$ , i.e., a uniformly sampled version of the signal  $l_a(t)$ . Further, from (5.185), it is seen that  $l_a(t)$  is the inverse Laplace transform of  $H_2(s)P(s)$ . Hence, a more convenient representation of  $L(z)$  and thereby  $G(z)$  can be found using the following procedure.

Starting with  $H_2(s)P(s)$ , its inverse Laplace transform gives  $l_a(t)$  from which  $l(n) = l_a(nT)$  follows directly. A closed form expression for  $L(z)$  is then obtained through the computation of the  $z$ -transform of  $l(n)$ . The approach is applicable for general  $H_2(s)$  and  $P(s)$ , and is straightforward when  $H_2(s)$  and  $P(s)$  are rational functions in  $s$ .

### 1.05.8.3 Stability

In order to investigate the stability, it is convenient to first consider the system in Figure 5.50a, where the numerator and denominator polynomials of  $G(z)$  in Figure 5.49, i.e.,  $N(z)$  and  $D(z)$ , respectively, have been shown explicitly. The system of Figure 5.45 will be bounded-input/bounded-output (BIBO) stable if the following two conditions are satisfied:

1. The poles of  $G(z)$ , i.e., the roots of  $D(z)$ , lie inside the unit circle,
2. The signal  $w(n)$  in Figure 5.50 is bounded.

The first condition above is easily checked once  $G(z)$  is available. The second condition requires an analysis of  $H_1(s)$  and  $N(z)$ . We consider the following two different cases which are of practical interest.

**Case 1:**  $H_1(s)$  is stable. This gives directly that the second condition above is satisfied and BIBO stability is then ensured by also fulfilling the first condition.

**Case 2:**  $H_1(s)$  can be factored according to

$$H_1(s) = A(s) \frac{1}{s^{Q_1}} \prod_{q=1}^{Q_2} \frac{1}{s + p_q}, \quad (5.186)$$

where  $A(s)$  is stable and  $p_q$  are imaginary roots. Then,  $H_1(s)$  contains  $Q_1$  poles at  $s = 0$  and  $Q_2$  poles at  $s = -p_q$ . In order to investigate stability in this case, it is convenient to consider

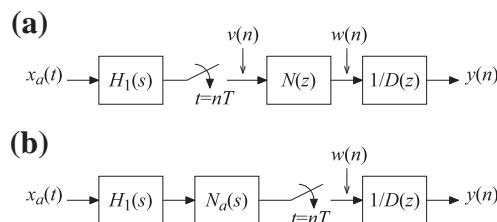


FIGURE 5.50

Representations of the system in Figure 5.49. They are useful in the stability analysis.

the system shown in Figure 5.50b, where

$$N_a(s) = N(e^{sT}). \quad (5.187)$$

Since  $N(z)$  is of the form

$$N(z) = \sum_{k=0}^K a_k z^{-k}, \quad (5.188)$$

the signal  $w(n)$  of Figure 5.50(a) is identical to that in Figure 5.50b. This follows from Poisson's summation formula together with the fact that  $N_a(s)$  above is periodic. It then remains to investigate if  $H_1(s)N_a(s)$  is stable, i.e., if it exists for all  $\Re\{s\} \geq 0$ , assuming a causal system, and if  $G(z)$  [or, equivalently,  $1/D(z)$ ] is stable.

#### 1.05.8.4 Frequency responses $H(j\Omega)$ and $G(e^{j\Omega T})$

The frequency response from  $e(n)$  to the output  $y(n)$  i.e.,  $G(e^{j\Omega T})$ , is directly obtained from  $G(z)$  in (5.176) if it exists for  $z = e^{j\Omega T}$ . It is then given by

$$G(e^{j\Omega T}) = \frac{N(e^{j\Omega T})}{D(e^{j\Omega T})} = \frac{1}{1 + \frac{1}{T} \sum_{p=-\infty}^{\infty} H_2\left(j\Omega - j\frac{2\pi p}{T}\right) P\left(j\Omega - j\frac{2\pi p}{T}\right)}. \quad (5.189)$$

As to the signal frequency response,  $H(j\Omega)$ , it is first noted from (5.176) and (5.177) that  $H(j\Omega)\frac{1}{T}X_a(j\Omega) = G(e^{j\Omega T})V(e^{j\Omega T})$ , where

$$V(e^{j\Omega T}) = \frac{1}{T} \sum_{p=-\infty}^{\infty} H_1\left(j\Omega - j\frac{2\pi p}{T}\right) X_a\left(j\Omega - j\frac{2\pi p}{T}\right). \quad (5.190)$$

Assume now that the input signal  $x_a(t)$  is bandlimited according to the Nyquist criterion for sampling with the sampling period  $T$ , i.e.,

$$X_a(j\Omega) = 0, \quad |\Omega| \geq \pi/T. \quad (5.191)$$

In this case only the term for  $p = 0$  in (5.190) is non-zero in the interval  $-\pi \leq \Omega T \leq \pi$ . Hence,  $V(e^{j\Omega T})$  can be written as in (5.177) with

$$H(j\Omega) = H_1(j\Omega)G(e^{j\Omega T}). \quad (5.192)$$

#### 1.05.8.5 Example 1

Consider the second-order continuous-time-input  $\Sigma\Delta$  modulator shown earlier in Figure 5.43. This is a special case of the system of Figure 5.44 with  $F_1(s)$  and  $F_2(s)$  being ideal integrators, i.e.,

$$F_1(s) = \frac{K_1}{s}, \quad F_2(s) = \frac{K_2}{s}. \quad (5.193)$$

In this case, one obtains that  $H_1(s)$  and  $H_2(s)$  are given by

$$H_1(s) = F_1(s)F_2(s), \quad H_2(s) = [1 + F_1(s)]F_2(s). \quad (5.194)$$

From this equation, we obtain

$$L_a(s) = H_2(s)P(s) = \left( \frac{K_2}{s^2} + \frac{K_1 K_2}{s^3} \right) (1 - e^{-sT}). \quad (5.195)$$

Taking the inverse Laplace transform, we get

$$l_a(t) = K_2[tu(t) - (t - T)u(t - T)] + \frac{K_1 K_2}{2}[t^2 u(t) - (t - T)^2 u(t - T)]. \quad (5.196)$$

Sampling  $l_a(t)$  at  $t = nT$  then gives us

$$\begin{aligned} l(n) = l_a(nT) &= K_2 T[n \times u(n) - (n - 1) \times u(n - 1)] \\ &\quad + \frac{K_1 K_2 T^2}{2}[n^2 \times u(n) - (n - 1)^2 \times u(n - 1)], \end{aligned} \quad (5.197)$$

where  $u(n)$  is the unit-step sequence. Then, taking the  $z$ -transform of  $l(n)$  gives us

$$L(z) = \frac{K_2 T z^{-1}}{1 - z^{-1}} \left[ 1 + \frac{K_1 T}{2} \frac{1 + z^{-1}}{1 - z^{-1}} \right], \quad (5.198)$$

whereby

$$G(z) = \frac{N(z)}{D(z)} = \frac{1}{1 + L(z)} = \frac{(1 - z^{-1})^2}{1 + az^{-1} + bz^{-2}}, \quad (5.199)$$

where

$$a = K_2 T + 0.5 K_1 K_2 T^2 - 2 \quad (5.200)$$

and

$$b = 1 - K_2 T + 0.5 K_1 K_2 T^2. \quad (5.201)$$

Further, with  $F_0(s)$  and  $F_1(s)$  as in (5.193), we get

$$H_1(s)N_a(s) = \frac{K_1 K_2}{s^2} (1 - e^{-sT})^2, \quad (5.202)$$

which is a stable system because the corresponding impulse response is finite. Hence, stability is in this case solely determined by the stability of  $G(z)$  given by (5.199). For example with  $K_1 = K_2 = K$ , it is straightforward to show that stability in this case is ensured by constraining  $K$  according to  $0 < K < 2/T$ . Using instead the often employed and so called “sampled-data equivalents,” where integrators  $1/s$  are replaced by the accumulators  $1/(1 - z^{-1})$ , and  $z^{-1}/(1 - z^{-1})$ , one obtains  $0 < K < 1.236/T$ . Hence, the accumulator-based structures are not appropriate models.

Furthermore, from (5.192), (5.194), and (5.199), we obtain the frequency response

$$H(j\Omega) = \frac{K_0 K_1}{(j\Omega)^2} \frac{(1 - e^{-j\Omega T})^2}{1 + ae^{-j\Omega T} + be^{-j2\Omega T}} \quad (5.203)$$

This expression differs from that obtained via the accumulator-based structures. Also the noise transfer functions are different. Again, this means that such structures are not appropriate. Figures 5.51 and 5.52 illustrate these differences by plotting the modulus of the signal and noise transfer functions in the two cases.

### 1.05.8.6 Example 2

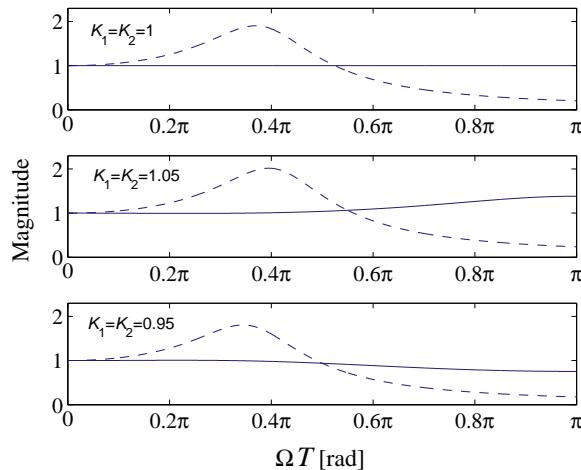
Consider the scheme in Figure 5.53 with  $F(s)$  being a second order transfer function according to

$$F(s) = \frac{K}{s(s + p)} \quad (5.204)$$

with a real-valued pole at  $s = -p$ , with  $p > 0$  assumed. For the equivalent representation of Figure 5.48,  $H_1(s) = H_2(s) = F(s)$ . Thus,  $H_1(s)$  follows (5.186) as  $H_1(s) = A(s)/s$  with  $A(s) = K/(s + p)$ .

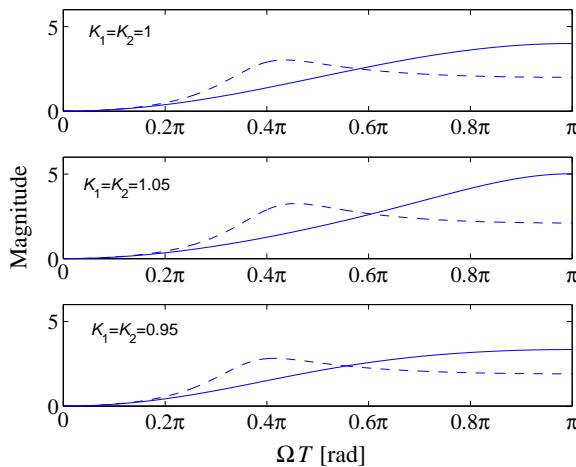
We first investigate the stability. We note that Case 2 applies here since  $H_1(s)$  has a pole at  $s = 0$ . To derive  $G(z)$ , we first note that  $H_1(s)P(s)$ , using partial fraction expansion, becomes

$$H_1(s)P(s) = \left( \frac{K/p}{s^2} - \frac{K/p^2}{s} + \frac{K/p^2}{s + p} \right) (1 - e^{-sT}). \quad (5.205)$$

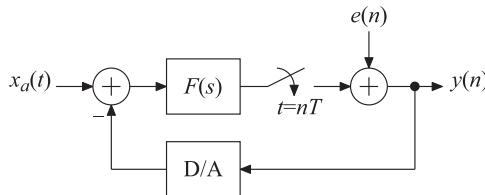


**FIGURE 5.51**

Modulus of the signal frequency responses for second-order continuous-time-input (dashed) and discrete-time-input  $\Sigma\Delta$ -modulators (solid).

**FIGURE 5.52**

Modulus of the noise frequency responses for second-order continuous-time-input (dashed) and discrete-time-input  $\Sigma\Delta$ -modulators (solid).

**FIGURE 5.53**

System with a single feedback loop.

Hence,  $l(n) = l_a(nT)$  is given by

$$l(n) = l_{a0}(nT) - l_{a0}(nT - T) \quad (5.206)$$

with  $l_{a0}(t)$  being

$$l_{a0}(t) = \left( \frac{K}{p}t - \frac{K}{p^2} + \frac{K}{p^2}e^{-pt} \right) u(t). \quad (5.207)$$

Hence, taking the  $z$ -transform of  $l(n)$ , we obtain

$$L(z) = \frac{KT}{p(z-1)} - \frac{K}{p^2} + \frac{K}{p^2} \frac{z-1}{z-e^{-pT}}. \quad (5.208)$$

Finally, this gives us

$$G(z) = \frac{1}{1 + L(z)} = \frac{(1 - z^{-1})(1 - z^{-1}e^{-pT})}{1 + az^{-1} + bz^{-2}}, \quad (5.209)$$

where

$$a = \frac{KT}{p} - (1 + e^{-pT}) - \frac{K}{p^2}(1 - e^{-pT}) \quad (5.210)$$

and

$$b = \frac{K}{p^2}(1 - e^{-pT}) + \left(1 - \frac{KT}{p}\right)e^{-pT}. \quad (5.211)$$

We note that

$$H_1(s)N_a(s) = \frac{A(s)(1 - e^{sT})(1 - e^{-(s+p)T})}{s} \quad (5.212)$$

which corresponds to a stable system, because it consists of (readily shown) stable subsystems. Hence, the stability is given by the poles of  $G(z)$ , which are most easily found numerically.

Further, we get the frequency responses

$$G(e^{j\Omega T}) = \frac{(1 - e^{-j\Omega T})(1 - e^{-j\Omega T}e^{-pT})}{1 + ae^{-j\Omega T} + be^{-j2\Omega T}} \quad (5.213)$$

and

$$\begin{aligned} H(j\Omega) &= H_1(j\Omega)G(e^{j\Omega T}) \\ &= \frac{K}{j\Omega(j\Omega + p)} \frac{(1 - e^{-j\Omega T})(1 - e^{-j\Omega T}e^{-pT})}{1 + ae^{-j\Omega T} + be^{-j2\Omega T}}. \end{aligned} \quad (5.214)$$

### Relevant Theory: Signal Processing Theory

See this Volume, [Chapter 2](#) Continuous-Time Signals and Systems

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 4](#) Random Signals and Stochastic Processes

See this Volume, [Chapter 7](#) Multirate Signal Processing

## References

- [1] E.T. Whittaker, On the functions which are represented by the expansion of interpolating theory, Proc. Roy. Soc. Edinburgh 35 (1915) 181–194.
- [2] H. Nyquist, Certain topics in telegraph transmission theory, Trans. Am. Inst. Elect. Eng. 47 1928.
- [3] V.A. Kotelnikov, On the transmission capacity of ether and wire in electrocommunications, Izd. Red. Upr. Svyazi RKKA, Moscow, 1933.
- [4] C.E. Shannon, Communication in the presence of noise, Proc. IRE 37 (1949) 10–21.
- [5] A.J. Jerri, The shannon sampling theorem—its various extensions and applications: A tutorial review, IEEE Proc. 65 (11) (1977) 1565–1596.
- [6] M. Unser, Sampling – 50 years after Shannon, IEEE Proc. 88 (4) (2000) 569–587.

- [7] P.P. Vaidyanathan, Generalizations of the sampling theorem: Seven decades after Nyquist, *IEEE Trans. Circ. Syst. I* 48 (9) (2001) 1094–1109.
- [8] R.J. van de Plassche, CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters, Kluwer Academic Publ., 2003.
- [9] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, Prentice Hall, 1989.
- [10] S.K. Mitra, Digital Signal Processing: A Computer-Based Approach, McGraw-Hill, 2006.
- [11] W.C. Black, D.A. Hodges, Time interleaved converter arrays, *IEEE J. Solid-State Circ.* SC-15 (6) (1980) 1022–1029.
- [12] S.R. Norsworthy, R. Schreier, G.C. Temes, Delta-Sigma Data Converters: Theory, Design, and Simulation, IEEE Press, 1997.
- [13] R. Schreier, G.C. Temes, Understanding Delta-Sigma Data Converters, John Wiley and Sons, NJ, 2005.
- [14] S.Y. Hui, K.H. Yeung, Challenges in the migration to 4G mobile systems, *Comm. Mag.* 41 (12) (2003) 54–59.
- [15] B. Le, T.W. Rondeau, J.H. Reed, C.W. Bostian, Analog-to-digital converters: a review of the past, present, and future, *Signal Process. Mag.* 22 (6) (2005) 69–77.
- [16] Analog Devices Inc., <<http://www.analog.com>>.
- [17] Texas Instruments Inc., <<http://www.ti.com>>.
- [18] R. Khioni-Poorfard, L.B. Lim, D.A. Johns, Time-interleaved oversampling A/D converters: Theory and practice, *IEEE Trans. Circ. Syst. II: Anal. Digital Signal Process.* 44 (8) (1997) 634–645.
- [19] B. Murmann, Trends in low-power, digitally assisted A/D conversion, *IEICE Trans. Electron.* E93-C (6) (2010) 718–729.
- [20] E.J. Candes, M.B. Wakin, An introduction to compressive sampling, *IEEE Signal Process. Mag.* 25 (2) (2008) 21–30.
- [21] J.A. Tropp, J.N. Laska, M.F. Duarte, J.K. Romberg, R.G. Baraniuk, Beyond nyquist: efficient sampling of sparse bandlimited signals, *IEEE Trans. Inf. Theory* 56 (1) (2010) 520–544.
- [22] M. Mishali, Y.C. Eldar, Sub-Nyquist sampling: bridging theory and practice, *IEEE Signal Process. Mag.* 28 (6) (2011) 98–124.
- [23] A. Papoulis, The Fourier Integral and Its Applications, McGraw-Hill, 1962.
- [24] P.P. Vaidyanathan, Multirate Systems and Filter Banks, Prentice Hall, 1993.
- [25] F. Marvasti, Nonuniform Sampling: Theory and Practice, Kluwer, NY, 2001.
- [26] Y.C. Eldar, A.V. Oppenheim, Filterbank reconstruction of bandlimited signals from nonuniform and generalized samples, *IEEE Trans. Signal Process.* 48 (10) (2000) 2864.
- [27] H. Johansson, P. Löwenborg, Reconstruction of nonuniformly sampled bandlimited signals by means of time-varying discrete-time FIR filters, Hindawi Publishing Corporation EURASIP J. Appl. Signal Process., Vol. 2006, Article ID 64185, 2006, pp. 1–18, <<http://dx.doi.org/10.1155/ASP/2006/64185>>.
- [28] H. Johansson, P. Löwenborg, K. Vengattaramane, Reconstruction of M-periodic nonuniformly sampled signals using multivariate polynomial impulse response time-varying FIR filters, in: Proc. XII European Signal Processing Conf., Florence, Italy, September 2006.
- [29] C. Vogel, The impact of combined channel mismatch effects in time-interleaved ADCs, *IEEE Trans. Instrum. Meas.* 55 (1) (2005) 415–427.
- [30] C. Vogel, Modeling, Identification, and Compensation of Channel Mismatch Errors in Time-Interleaved Analog-to-Digital Converters, Diss., Graz Univ., 2005.
- [31] C. Vogel, G. Kubin, Modeling of time-interleaved ADCs with nonlinear hybrid filter banks, *AEU - Int. J. Electronics Comm.* 59 (5) (2005) 288–296.
- [32] J. Elbornsson, F. Gustafsson, J.E. Eklund, Blind equalization of time errors in a time-interleaved ADC system, *IEEE Trans. Signal Process.* 53 (4) (2005) 1413.
- [33] V. Divi, G. Wornell, Scalable blind calibration of timing skew in high-resolution time-interleaved ADCs, in: Proc. IEEE Int. Symp. Circ. Syst., Kos, Greece, May 2006, pp. 21–24.

- [34] S. Huang, B.C. Levy, Adaptive blind calibration of timing offset and gain mismatch for two-channel time-interleaved ADCs, *IEEE Trans. Circ. Syst. I* 53 (6) (2006) 1278–1288.
- [35] S. Huang, B.C. Levy, Blind calibration of timing offsets for four-channel time-interleaved ADCs, *IEEE Trans. Circ. Syst. I* 54 (4) (2007) 863–876.
- [36] A. Haftbaradaran, K.W. Martin, A background sample-time error calibration technique using random data for wide-band high-resolution time-interleaved ADCs, *IEEE Trans. Circ. Syst. II* 55 (3) (2008) 234–238.
- [37] M. El-Chammas, B. Murmann, General analysis on the impact of phase-skew mismatch in time-interleaved ADCs, in: Proc. IEEE Int. Symp. Circ. Syst., Seattle, USA, May 2008, pp. 18–21.
- [38] T. Tsai, P.J. Hurst, S.H. Lewis, Bandwidth mismatch and its correction in time-interleaved analog-to-digital converters, *IEEE Trans. Circ. Syst. II* 53 (10) (2006) 1133–1137.
- [39] M. Seo, M.J.W. Rodwell, U. Madhow, Comprehensive digital correction of mismatch errors for a 400-msamples/s 80-dB SFDR time-interleaved analog-to-digital converter, *IEEE Trans. Microwave Theory Techn. MTT-53* (3) (2005) 1072–1082.
- [40] S. Mendel and C. Vogel, A compensation method for magnitude response mismatches in two-channel time-interleaved analog-to-digital converters, in: Proc. IEEE Int. Conf. Electronics, Circ. Syst., Nice, France, December 2006.
- [41] S. Mendel and C. Vogel, On the compensation of magnitude response mismatches in M-channel time-interleaved ADCs, in: Proc. IEEE Int. Symp. Circ. Syst., New Orleans, USA, May 2007, pp. 3375–3378.
- [42] M. Seo, M.J.W. Rodwell, U. Madhow, Generalized blind mismatch correction for two-channel time-interleaved A-to-D converters, in: Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing, Hawaii, USA, April 2007.
- [43] M. Seo, M.J.W. Rodwell, U. Madhow, Generalized blind mismatch correction for a two-channel time-interleaved ADC: Analytic approach, in: Proc. IEEE Int. Symp. Circ. Syst., New Orleans, USA, May 2007, pp. 27–30.
- [44] P. Satarzadeh, B.C. Levy, P.J. Hurst, Bandwidth mismatch correction for a two-channel time-interleaved A/D converter, in: Proc. IEEE Int. Symp. Circ. Syst., New Orleans, USA, May 2007.
- [45] H. Johansson, P. Löwenborg, A least-squares filter design technique for the compensation of frequency-response mismatch errors in time-interleaved A/D converters, *IEEE Trans. Circ. Syst. II: Express Briefs* 55 (11) (2008) 1154–1158.
- [46] H. Johansson, A polynomial-based time-varying filter structure for the compensation of frequency-response mismatch errors in time-interleaved ADCs: special issue on DSP techniques for RF/analog circuit impairments, *IEEE J. Selected Topics Signal Process.* 3 (3) (2009) 384–396.
- [47] Y.C. Lim, Y.X. Zou, J.W. Lee, S.C. Chan, Time-interleaved analog-to-digital converter compensation using multichannel filters, *IEEE Trans., Circ. Syst. I: Regular Papers* 56 (10) (2009) 2234–2247.
- [48] C. Vogel, S. Mendel, A flexible and scalable structure to compensate frequency response mismatches in time-interleaved ADCs, *IEEE Trans. Circ. Syst. I: Regular papers* 56 (11) (2009) 2463–2475.
- [49] J.L. Yen, On nonuniform sampling of bandlimited signals, *IRE Trans. Circuit Theory CT-3* (1956).
- [50] A. Papoulis, Generalized sampling expansion, *IEEE Trans. Circ. Syst. CAS-24* (11) (1977) 652–654.
- [51] A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, second ed., McGraw-Hill, 1993.
- [52] L.B. Jackson, *Digital Filters and Signal Processing*, third ed., Kluwer Academic Publishers, 1996.
- [53] L. Wanhammar and H. Johansson, *Digital Filters using Matlab*, Linköping University, 2011.
- [54] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice Hall, NJ, 1983.
- [55] N.J. Fliege, *Multirate Digital Signal Processing*, Wiley, New York, 1994.
- [56] P. Löwenborg, H. Johansson, Analysis of continuous-time-input  $\Sigma\Delta$  A/D modulators and their generalizations, in: Proc. European Conf. Circuit Theory Design, Krakow, Poland, September 1–4 2003.
- [57] S.H. Ardalan and J.J. Paulos, An analysis of nonlinear behavior in delta-sigma modulators, *IEEE Trans. Circ. Syst. CAS-34* (6) (1987) 593–603.
- [58] H. Freeman, *Discrete-Time Systems*, John Wiley, 1965.

# Digital Filter Structures and Their Implementation

# 6

Lars Wanhammar<sup>\*</sup> and Ya Jun Yu<sup>†</sup>

<sup>\*</sup>Department of Electrical Engineering, Linköping University, Sweden

<sup>†</sup>School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore

## 1.06.1 Introduction

There has been a clear trend over the past decades towards an increasing use of digital processing rather than analog processing of signals. This is in large part due to the inherent flexibility and reliability of digital processing. Other important factors, that support this trend, are the rapid development in integrated circuit technology and the associated tools, i.e., that simplifies and reduces the cost of implementing digital signal processing systems, and digital filters in particular. Of interest here is frequency-selective filters, for example, lowpass and bandpass filters.

### 1.06.1.1 Properties of digital filters

The following properties are characteristic of digital filters.

#### 1.06.1.1.1 Flexibility

The frequency response can easily and quickly be changed. This means that a digital filter implementation can be time-shared between a number of input signals and act as several filters. Furthermore, a digital filter can be multiplexed in such a way that it simultaneously acts as several filters with one input signal and realize, for example, a filter bank. The high flexibility is exploited in adaptive filters.

#### 1.06.1.1.2 Special transfer functions

Discrete-time and digital filters can realize special transfer functions that are not realizable with continuous-time, lumped element filters, for example filters with exact linear phase.

#### 1.06.1.1.3 Coefficient sensitivity

Digital filters are not affected by temperature variations, variations in the power supply voltage, stray capacitances, etc., which are major problems in analog filters. The reason for this is that the properties of a digital filter is determined by the numerical operations on numbers and not dependent on any tolerances of electrical components. For the same reason there is no aging or drift in digital filters. The independence of element sensitivity leads to high flexibility, miniaturization, and high signal quality.

#### **1.06.1.1.4 Reproducibility**

Exact reproducible filters can be manufactured. This is also a consequence of the fact that the properties of the filter are only dependent on the numerical operation and not on the tolerances of the electrical components. Thus, tolerance problems as such do not exist in digital filters. However, we will see that sensitivity of a filter structure to errors in the coefficient has a direct influence on the required signal wordlength and implementation cost. Furthermore, no trimming is necessary, which leads to low cost. Precision digital filters can be manufactured with, in principle, arbitrary precision, linearity, and dynamic range, however, at an increasing cost.

#### **1.06.1.1.5 Frequency range**

Digital filters can be manufactured to operate over a wide range of frequencies, from zero up to several hundred MHz, depending on the complexity of the filter and on the technology used for the implementation. However, the cost in terms of power consumption and the number of gates in the integrated circuits increases rapidly with the order of the filter and the sample frequency.

#### **1.06.1.1.6 Power consumption**

Digital filters have in general rather high power consumption, but filters with low power consumption can be implemented for applications with low sample frequencies. However, as the geometries of the CMOS process used for the implementation are reduced, the power consumption is also reduced. The power consumption compared with a corresponding analog filter counterpart is therefore often lower for a digital filter.

#### **1.06.1.1.7 Implementation techniques**

Digital filters can be implemented by using several techniques. At low sample rates and arithmetic workloads, typically below a few MHz, standard signal processors can be used while application-specific or algorithm-specific digital signal processors can be used up to a few tenth of MHz. For high sample rates specialized implementation techniques are required [1].

### **1.06.1.2 Design target**

The design should meet the requirements with minimal cost. The cost to be minimized, is typically composed of several different parameters, e.g., power consumption, chip area, design effort. To complicate the issue even further, the designer has many alternative design options available. Hence, filter design is a challenging task, and it typically involves several conflicting optimization problems.

#### **1.06.1.2.1 Power consumption**

Until recently throughput and chip area was the main metrics for evaluation of DSP algorithms and their implementations. However with the advancement of CMOS technology it is no longer a major problem to achieve high enough throughput and the cost of the chip area can usually be neglected, except in very high-volume products. Today, the focus should be placed on design efficiency and low power consumption since many implementations are battery-powered and a too high power dissipation may require more expensive packaging and cooling. Further, there is also a strong trend to use a more sophisticated and complex digital signal processing which tend to increase the power dissipation.

CMOS technology is currently and will continue to be the dominating implementation technology. The feature size is rapidly shrinking towards 10 nm and below. Circuits implemented using modern CMOS processes are no longer characterized by the number of transistors on the chip. Instead the overall throughput and power consumption are more appropriate metrics. Here we assume the digital filter is operated continuously and therefore the leakage and short-circuit currents can be neglected [2]. The power dissipation associated with switching in CMOS circuits is approximately given by

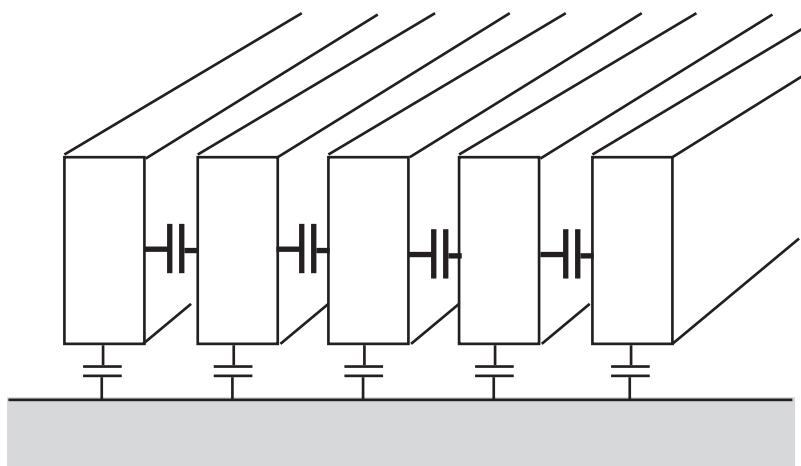
$$P = \alpha f_{CL} C_L V_{DD} \Delta V_{DD}, \quad (6.1)$$

where  $\alpha$  is the activity factor, i.e., the average number of transitions from 0 to 1 for the equivalent electrical node per clock cycle,  $f_{CL}$  is the clock frequency,  $C_L$  is the equivalent switched capacitance of the whole circuit,  $V_{DD}$  is the power supply voltage, and  $\Delta V_{DD}$  is the voltage swing. In most cases we use CMOS circuits with full swing, i.e.,  $\Delta V_{DD} = V_{DD}$ . Reducing any of the factors in (6.1) can reduce the power consumption. Below we briefly review various techniques that can be applied to reduce the power consumption in digital filters and are useful in many other DSP algorithms.

The activity factor  $\alpha$  can be reduced in several ways. Most techniques exploit known statistical properties of the signals. For example, operations to be executed on an adder or multiplier can be scheduled with respect to the correlation between successive operands so that the carry propagation and glitches are minimized. For example, in the first moment only the first full-adder (LSB) in a bit-parallel ripple-carry adder has correct inputs and performs a valid addition. The other full-adders have carry-ins that are not yet determined, hence, they may perform invalid additions that consume energy. In the next moment, the second full-adder has correct inputs and performs a valid addition while the first full-adder is idle. In each instance, the remaining full-adders may change their states and contribute with an increase in the power dissipation. Note that the bit-parallel ripple-carry adder performs a sequential bit-by-bit addition while in bit-serial arithmetic a bit-by-bit addition is performed sequentially on a single full-adder. The execution time for a bit-parallel addition using a ripple-carry adder is the same as for a bit-serial addition if the safety margins needed for the flip-flops that separate two successive bit additions are neglected.

The use of a redundant number representation [3] is another technique to avoid long carry propagation and glitches, which are costly in terms of power consumption. Glitches in logic circuits occur when the difference in arrival times of two logic signals to a common gate is so large that false transitions occur and change the states of some electrical nodes. Slowing down the signal paths that are too fast so that the signals arrive simultaneously can minimize this effect and may reduce the power consumption by up to 40% in a bit-parallel multiplier.

In older CMOS processes the equivalent switched capacitance is due to devices and wires. The dominant wire capacitance is between the wire and substrate. In deep submicron CMOS technologies the capacitance associated with the devices are relatively small and the dominant wire capacitance is instead dominated by interwire capacitance as illustrated in Figure 6.1. For long buses it is therefore important to order the transmission of successive data on the bus so that the charging/discharging of the interwire capacitors is minimized. A useful measure to minimize is the sum of the Hamming distances of the transmitted words. In speech signals, for example, it may be advantageous to use sign-magnitude representation on long buses, since most samples have small magnitudes, which causes only the least significant bits to vary while in two's-complement representation almost all bits change when a sample value changes sign.

**FIGURE 6.1**

Interwire and wire-to-substrate capacitance model.

The clock frequency,  $f_{CL}$ , of the circuitry should, of course, not be reduced since this would reduce the computational work performed by the circuitry and the inherent processing capacity of the circuitry would not be fully utilized. Generally, it is efficient to use minimum size transistors throughout the whole integrated circuit, except for a few critical paths in which slightly larger and thereby faster transistors are used. This strategy leads to a significant increase in the maximal clock frequency at the expense of a small increase in the switched capacitance.

If a long logic path requires gates with extra drive capacity in order to meet the throughput requirement, it may often be more efficient to break the long path into smaller paths in terms,  $n$  order to reduce the chip area and power consumption.

The required clock frequency for the algorithm may, however, be reduced at the expense of increased chip area, by exploiting computational parallelism of the algorithm by mapping the arithmetic operations to several processing elements (PEs) or by using multirate techniques in which parts of the digital filter operate at lower sample rates. Parts of the algorithm that operate at a lower rate can often be multiplexed onto a single hardware structure in order to save chip area.

The equivalent switched capacitance,  $C_L$ , can be made small by using a combination of different techniques. On the algorithm level most techniques focus on using filter structures with a small number of arithmetic operations per sample. Multirate filter structures are typical examples where a significant reduction in arithmetic workload can be obtained. Low-sensitivity structures require shorter data wordlengths and shorter and more favorable coefficients that allow fixed multipliers to be simplified [1]. Canonic signed-digit code (CSDC) [1,3], and multiple-constant techniques are efficient techniques to simplify the realization of fixed multipliers [4–11]. Trade-off between the two extremes, bit-parallel and bit-serial arithmetic, i.e., digit-serial arithmetic is a promising route to lower power consumption as well as smaller chip area [1]. As mentioned above, transmission of data over long buses is costly in terms of time and power consumption. Architectures with local computations are therefore preferred.

Digit-serial processing elements yield smaller chip area, and, hence, shorter communication distances and is a good compromise between the large number of glitches in bit-parallel arithmetic and many clock cycles per operation of the flip-flops in bit-serial arithmetic. At the logic level the use of efficient, low-power CMOS circuit and clocking techniques is important [2].

The power consumption can be significantly reduced by using a lower power supply voltage. Unfortunately, this has the effect that the maximal clock frequency of the CMOS circuit is reduced [2] according to

$$f_{CL} = \frac{k(V_{DD} - V_T)^\beta}{V_{DD}}, \quad (6.2)$$

where  $k$  is process-dependent constant,  $\beta$  is a constant slightly larger than one in modern CMOS processes, and  $V_T$  is the threshold voltage of an MOSFET. However, the reduction in throughput can often be compensated in parallel algorithms by exploiting more of the computational parallelism [1]. From (6.2) it is evident that from a speed point of view, a CMOS process with a low threshold voltage is to be preferred. In practice, many circuits are therefore manufactured with a process with two threshold voltages. Transistors with low threshold voltages are used for high-speed circuits while transistor with high threshold voltages are used for low speed circuits since the latter have smaller leakage currents.

### 1.06.1.2.2 Chip area

The required chip area is important since is directly affect the cost of manufacturing the filter. Moreover, the leakage current will be proportional to the chip area. A designer can minimize the required chip area by using filter structures with few arithmetic operations per sample, simplified arithmetic operations, and small memory requirement. That is, we are interested in structures that have low element sensitivity, i.e., a structure where a small error in the coefficients causes only a small error in the frequency response. Hence, it is possible to select favorable coefficient values that deviate slightly from their ideal values and still meet the requirements on the frequency response. We will later discuss filter structures with very low coefficient sensitivities that allows us to reduce the required chip area.

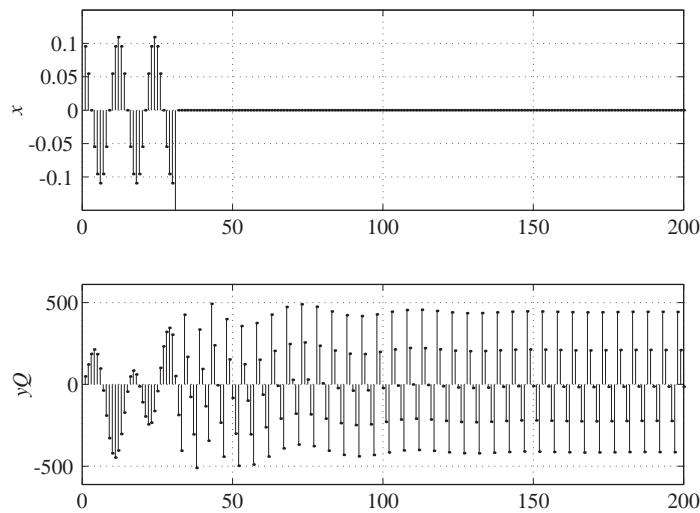
The data wordlength generally affects the execution time and the required chip area for the arithmetic operations. It has been shown by Jackson [12] that the roundoff noise is bounded from below by the coefficient sensitivities, i.e.,

$$\sigma^2 \geq \sum_i \sigma_{0i}^2 \left\| \frac{\partial |H(e^{j\omega T})|}{\partial c_i} \right\|_p^2, \quad (6.3)$$

where  $p = 1$  or  $p = 2$ ,  $\sigma_0^2$  is the variance of the roundoff noise sources, and the sum is over all noise sources due to rounding at the outputs of the coefficients,  $c_i$ . Hence, the right hand side of (6.3) is small for a low-sensitive structure and the roundoff noise at the output of the structure may be low. In fact, it will be low, even though there is no proof for it. A structure with high sensitivity will generate a large roundoff noise and it will therefore require a large data wordlength. To summarize, the element sensitivity affects both the required data and coefficient wordlengths.

### 1.06.1.2.3 Throughput

The throughput of a non-recursive algorithm is in principle unlimited. In practice, however, the amount of circuitry as well as the inherent speed of the used CMOS technology becomes a limit. Recursive algorithms have an inherent limit on the throughput.

**FIGURE 6.2**

Persistent parasitic overflow oscillation.

#### 1.06.1.2.4 Robustness

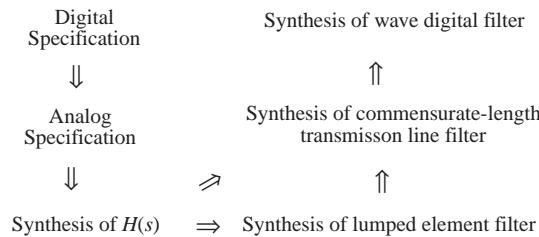
Recursive algorithms require that the data wordlength is rounded or truncated at least once in every recursive loop that contains at least one multiplication with a non-integer coefficient. Otherwise the data wordlength would increase to infinity. Rounding/truncation and overflow of the available signal range are non-linear operations that will in most cases cause parasitic oscillations. That is, the output of the filter will contain a periodic or even a chaotic signal component that is not due to the input signal [13, 14]. In fact, this unwanted distortion can be very large. Of course, this is highly undesirable and should be avoided. Figure 6.2 shows an example of the input and output of a recursive loop when the input signal overflows the signal range just before it becomes zero. Obviously this is not an acceptable behavior. We will later discuss a class of filter structures, i.e., wave digital filters, that have the property to suppress such undesirable oscillations.

#### 1.06.1.3 Design of digital filters

The design of digital filters can be done along several different routes.

##### 1.06.1.3.1 Design and realization of FIR filters

If the filter has a finite-length impulse response (FIR), then the frequency response is typically designed directly in the  $z$ -domain using computer-based optimization methods. In Section 1.06.2 we will briefly discuss the design of classical FIR filters and associated structures, which are suitable for cases when the transition band is not too small. Unfortunately, for requirements with narrow transition band, the filter

**FIGURE 6.3**

Design process for wave digital filters.

order and arithmetic workload becomes very large for FIR filters. In such cases, we instead recommend recursive structures (IIR), frequency-masking techniques, which are discussed in Section 1.06.8, or multirate techniques [14–16].

#### 1.06.1.3.2 Design and realization of IIR filters

In the case of infinite-length impulse response (IIR) filters there are many options available. For complex and non-standard requirements the frequency response is often determined directly in the  $z$ -domain using computer-based optimization methods [14, 17].

For frequency selective filters with standard requirements, i.e., filters with a passband that approximate a constant gain in the passband, it is favorable to employ the vast available knowledge of design of analog filters. There are a number of advantages of this approach, which is illustrated in Figure 6.3:

- Accurate numerical algorithms have been developed for analog filters [18–21].
- Classical analog attenuation approximations can be converted using the bilinear transformation to their digital counterparts. The bilinear transformation [14, 22–24] between the  $s$ - and  $z$ -domains is

$$s = \frac{2}{T} \frac{z - 1}{z + 1}. \quad (6.4)$$

Optimal highpass filters can be designed by frequency transformation of an optimal lowpass filter. Near optimal bandpass and bandstop filters can also be designed by frequency transformations. However, we will take an indirect route that will yield a low-sensitive digital filter algorithm:

- Once the analog transfer function,  $H(s)$ , (frequency response) is determined, it is realized using a lumped element ( $LC$ ) filter structure, e.g., a ladder structure. We will later show that a properly designed  $LC$  filter will have optimal element sensitivity in the passband. Unfortunately, this structure is not possible to simulate in the  $z$ -domain, because delay-free loops occur.
- To circumvent this problem, we use an analogy between a lumped element network and commensurate-length transmission line network. The latter contain sources, resistors, and lossless transmission lines of equal length. The transmission line network has a periodic frequency response and there is a one-to-one mapping between the continuous-time transmission line network and its digital counterpart. We will later discuss this in more detail.

- The digital algorithm is obtained by using a wave description, i.e., wave digital filters [25–27], consisting of a linear combination of voltages and current in the analog network. This allows us to simulate power in the analog filter, which we later will show is a necessity to obtain a low-sensitive algorithm.
- Moreover, the concept of pseudo-power in the digital domain allows us to suppress parasitic oscillations. That is, there exists a Lyapunov function for wave digital filters.

We will therefore describe various steps in the design process, illustrated in Figure 6.3, and options the designer has to minimize the implementation cost. First, we discuss in Section 1.06.3, the analog approximation problem and show that classical design recommendations are not always the best. In Section 1.06.4, we discuss the realization of the analog transfer function using a doubly resistively terminated network. We particularly discuss what the designer can do in order to minimize the element sensitivity. We also discuss the realization of commensurate-length transmission line filters. In Sections 1.06.5 and 1.06.6 we discuss two of the most common analog filter networks and their properties. In Section 1.06.7, we present the basic principles of wave digital filters and their building blocks. In addition we compare two types of wave digital filter realizations using an example. In Section 1.06.8 we discuss techniques that are particularly efficient for filters with very narrow transition bands. These techniques may employ both FIR and IIR structures, but first we discuss in the next section classical FIR filters. Finally, in Sections 1.06.9–1.06.12, we will discuss some computational properties of algorithms and some techniques for their implementation.

## 1.06.2 Digital FIR filters

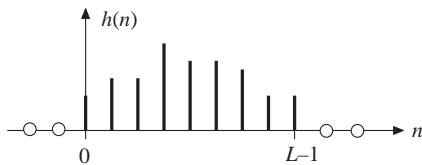
Digital filters can be categorized into FIR (Finite-length Impulse Response) and IIR (Infinite-length Impulse Response) filters. Advantages of FIR filters over IIR filters are that they can be guaranteed to be stable and may have a linear-phase response. Further, they require generally shorter data wordlength than the corresponding IIR filters. However, FIR filters usually require much higher orders than IIR filters for the same magnitude specification and they often introduce a large group delay, that make them unsuitable in many applications. FIR filters are well suited for many multirate filter realizations. An advantage is that FIR filters are always stable, except when they are used inside a recursive loop. High-order FIR filters can be realized efficiently using the FFT (Fast Fourier Transform) [14,28–30] and in some cases using multirate techniques [15,16,31–33].

The most common method to design a linear-phase FIR filter, that satisfies a specification of the magnitude response, is to use a numeric optimization procedure to determine the impulse response  $h(n)$  in (6.5). A widely used algorithm for designing linear-phase FIR filters with multiple passbands and stopbands is that of McClellan, Parks, and Rabiner [17,30,34,35].

The impulse response  $h(n)$  of a causal FIR filter of order  $N$  (length  $L = N + 1$ ) is nonzero for  $0 \leq n \leq L - 1$  and zero for all other values of  $n$  as illustrated in Figure 6.4.

The transfer function and frequency response of an  $N$ th order FIR filter is

$$H(z) = \sum_{n=0}^N h(n)z^{-n}. \quad (6.5)$$

**FIGURE 6.4**

Impulse response of a causal FIR filter of order  $N$  (length  $L = N + 1$ ).

The poles of the transfer function can only be placed at the origin of the  $z$ -plane for nonrecursive FIR filters. The zeros can be placed anywhere in the  $z$ -plane, but they are usually located on the unit circle or as pairs that are mirrored in the unit circle.

Of main interest among FIR filters are filters with linear-phase for which the impulse response exhibit symmetry or antisymmetry. Linear-phase FIR filters are widely used in digital communication systems, speech and image processing systems, spectral analysis and particularly in applications where nonlinear phase distortion cannot be tolerated. Most filter design software packages support the design of linear-phasePlease note that the Reference number seem to be missing. FIR filters [17,30,34,35].

A linear-phase FIR filter is obtained by letting the impulse response exhibit symmetry around  $n = N/2$ , i.e.,

$$h(n) = h(N - n) \quad (6.6)$$

or antisymmetry around  $n = N/2$ , i.e.,

$$h(n) = -h(N - n), \quad (6.7)$$

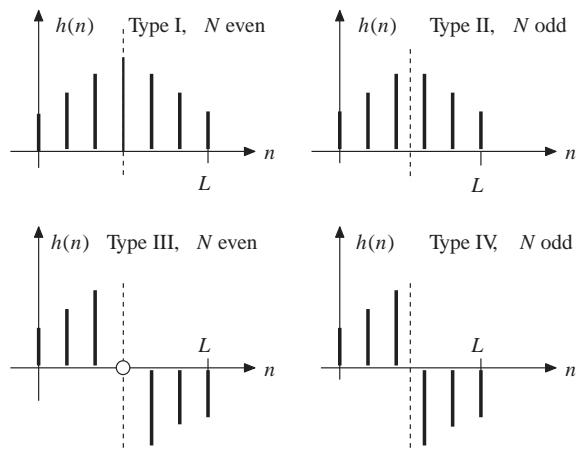
for  $n = 0, 1, \dots, N$ . Since the order  $N$  is either even or odd, there are four different types of FIR filters with linear phase. These are denoted as Type I, II, III, IV linear-phase FIR filters, respectively, according to the following:

Type I	$h(n) = h(N - n)$	$N$ even,
Type II	$h(n) = h(N - n)$	$N$ odd,
Type III	$h(n) = -h(N - n)$	$N$ even,
Type IV	$h(n) = -h(N - n)$	$N$ odd.

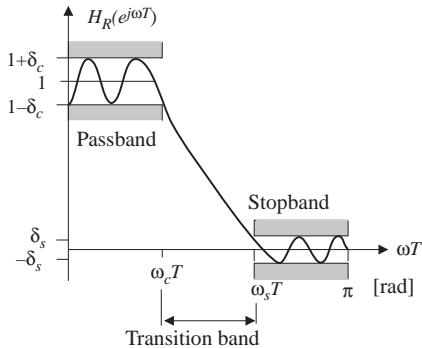
Figure 6.5 shows typical impulse responses for the different cases. Note that the centre value  $h(N/2)$  is always equal to zero for Type III filters. Also, when  $N$  is even, the point of symmetry is an integer corresponding to one of the sample values. When  $N$  is odd, the point of symmetry lies between two sample values.

A special case, is so-called half-band filters, where about half of the impulse response values are zero and, hence, they do not require any arithmetic operations. A half-band filter has a frequency response that is antisymmetric around  $\omega T = \pi/2$ . Hence,  $\omega_c T + \omega_s T = \pi$ .

Figure 6.6 show a typical specification of the zero-phase response for a lowpass linear-phase FIR filter. Several approximate expressions for the required filter order for equiripple, linear-phase FIR filters have been determined empirically.

**FIGURE 6.5**

Impulse responses for the four types of linear-phase FIR filters.

**FIGURE 6.6**

Specification of FIR filters.

One such simple estimate of the required filter order for a linear-phase lowpass (or highpass) filter that meets the specification shown in Figure 6.6, is

$$N \approx \frac{-4\pi}{3} \frac{\log(13\delta_c\delta_s)}{\Delta\omega T}, \quad (6.8)$$

where  $\Delta\omega T = \omega_s T - \omega_c T$  is the transition band. Note that the required filter order will be high for filters with narrow transition bands.

Here we are concerned with linear-phase FIR filters, that are frequency selective, i.e., lowpass, highpass, bandpass, and bandstop filters, etc. When designing such filters it is convenient to consider

the zero-phase frequency response  $H_R(e^{j\omega T})$ , which is defined by

$$H(e^{j\omega T}) = H_R(e^{j\omega T})e^{j\phi(\omega T)}. \quad (6.9)$$

Note that  $|H(e^{j\omega T})| = |H_R(e^{j\omega T})|$ .

The specification of a lowpass filter is commonly given as

$$\begin{aligned} 1 - \delta_c &\leq |H(e^{j\omega T})| \leq 1 + \delta_c & \omega T \in [0, \omega_c T] \\ |H(e^{j\omega T})| &\leq \delta_s & \omega T \in [\omega_s T, \pi]. \end{aligned} \quad (6.10)$$

For linear-phase FIR filters, the specification of (6.10) can be restated with the aid of  $H_R(e^{j\omega T})$  according to

$$\begin{aligned} 1 - \delta_c &\leq H_R(e^{j\omega T}) \leq 1 + \delta_c & \omega T \in [0, \omega_c T] \\ -\delta_s &\leq H_R(e^{j\omega T}) \leq \delta_s & \omega T \in [\omega_s T, \pi] \end{aligned} \quad (6.11)$$

The McClellan-Parks-Rabiners program will find a unique set of filter coefficients that minimizes a weighted error function. The algorithm solves the approximation problem

$$\min(E_\infty) = \max |E(e^{j\omega T})|, \quad \omega T \in \Omega, \quad (6.12)$$

where  $E(e^{j\omega T})$  is the weighted (Chebyshev) error function given by

$$E(e^{j\omega T}) = W(e^{j\omega T}) \left[ H_R(e^{j\omega T}) - D(e^{j\omega T}) \right], \quad \omega T \in \Omega \quad (6.13)$$

with  $\Omega$  being the union of the passband and stopband regions. In practice,  $\Omega$  is a dense set of frequency samples taken from the passbands and stopbands, including the bandedges. It should be stressed that all functions involved here,  $H_R(e^{j\omega T})$ ,  $E(e^{j\omega T})$ ,  $D(e^{j\omega T})$ , and  $W(e^{j\omega T})$  are real functions of  $\omega T$ . The desired function  $D(e^{j\omega T})$  is the one to be approximated by  $H_R(e^{j\omega T})$ . For example,  $D(e^{j\omega T})$ , is for the standard lowpass filter

$$D(e^{j\omega T}) = \begin{cases} 1 & \omega T \in [0, \omega_c T], \\ 0 & \omega T \in [\omega_s T, \pi]. \end{cases} \quad (6.14)$$

That is,  $D(e^{j\omega T})$  is for conventional filters equal to one in the passbands and zero in the stopbands. The weighting function  $W(e^{j\omega T})$  specifies the cost of the deviation from the desired function. The use of the weighting function makes it possible to obtain different ripples in different frequency bands. The larger the weighting function is, the smaller is the ripple. For example, for a standard lowpass filter,  $W(e^{j\omega T})$  is usually given as

$$W(e^{j\omega T}) = \begin{cases} 1 & \omega T \in [0, \omega_c T], \\ \frac{\delta_c}{\delta_s} & \omega T \in [\omega_s T, \pi]. \end{cases} \quad (6.15)$$

The specification of (6.11) is then satisfied if  $E_\infty \leq \delta_c$ . The resulting filter is optimized in the Cheby-shev (minimax) sense. MATLAB programs for design of FIR are found in [17, 36, 37]. Design of special cases of linear-phase FIR filters, as well as non-linear phase FIR filters, is discussed in detail in [14].

### 1.06.2.1 FIR structures

An FIR filter can be realized using both recursive or nonrecursive algorithms. The former, however, suffer from a number of drawbacks and should rarely be used in practice. An exception is so-called running-sum filters, which often are used in decimation filters for  $\sum \Delta$  converters.

Nonrecursive filters are always stable and cannot sustain parasitic oscillations, except when the filters are a part of a recursive loop. They generate little roundoff noise and can therefore be operated with short data wordlengths. However, they usually require a large number of arithmetic operations and a large amount of memory. Therefore large efforts have been directed towards developing methods to reduce the arithmetic workload.

This can be done by reducing the number of multiplications per sample, or by simplifying the coefficient values so that the multiplications becomes simpler to implement. The latter approaches results in what is referred to as multiplier-free structures, since all multiplications have been replaced with addition/subtraction and shift operations. That is, the coefficients are selected among a restricted set of values, i.e., as a sum-of-powers-of-two, SPOT,  $c = \pm 2^{\pm n_1} \pm 2^{\pm n_2} \pm \dots \pm 2^{\pm n_m}$ . Most FIR filters can be realized using no more than four powers of two. Today the arithmetic workload for a properly designed FIR structure is still relatively large, but the costs, i.e., chip area and power consumption, associated with the memory is becoming more and more significant.

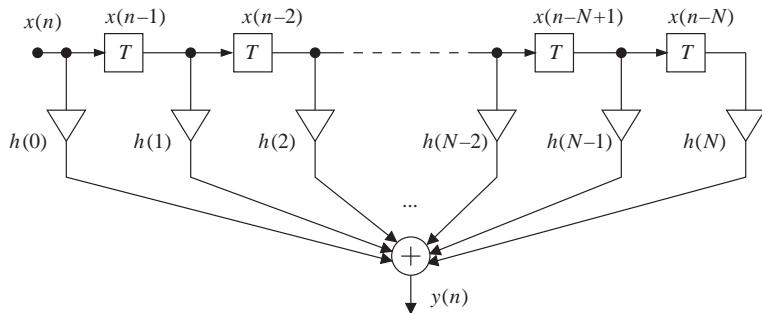
#### 1.06.2.1.1 Direct form FIR structures

There exist several structures that are of interest for the realization of FIR filters, particularly for filters with several sample frequencies so-called multirate filters. One simple structure is the direct form FIR, which has relatively small element sensitivity, i.e., the deviation in the frequency response due to a small change in any of the coefficients, is relatively small. This structure has many arithmetic operations and is therefore only suitable for relatively short FIR filters. The roundoff noise variance is  $(N + 1) \frac{Q^2}{12}$  if all products are rounded off. For filters with very narrow transition bands we recommend frequency-masking structures, which are discussed later.

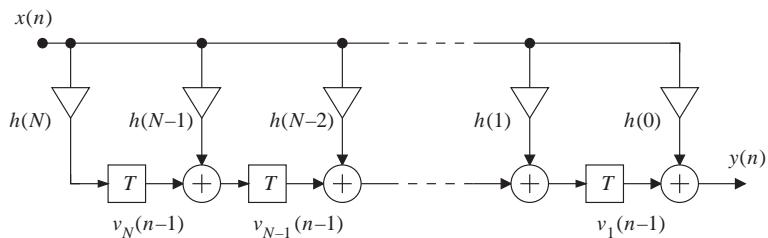
#### 1.06.2.1.2 Transposed direct form

The transposition theorem [33] can be used to generate new structures that realizes the same transfer function as the original filter structure. The transposed direct form FIR structure, shown in Figure 6.8, is derived from the direct form structure. The throughput is higher for the transposed form because the longest critical path is through only one multiplier and an adder, but the capacitive load at the input node is larger, since all multipliers load this node. The transposed structure is often preferred for an FPGA implementation.

The direct form structures requires in general  $N + 1$  multiplications, which may become a too large workload for high order FIR filters. However, the required amount of hardware can be significantly reduced, since many multiplications are performed with the same input value, or the transposed case where many signals are multiplied and then added. The internal signal level increases from left to right as more and more internal signals are added. Also in this structure the roundoff noise is  $(N + 1) \frac{Q^2}{12}$  if all products are rounded off.

**FIGURE 6.7**

Direct form FIR structure (Transversal filter).

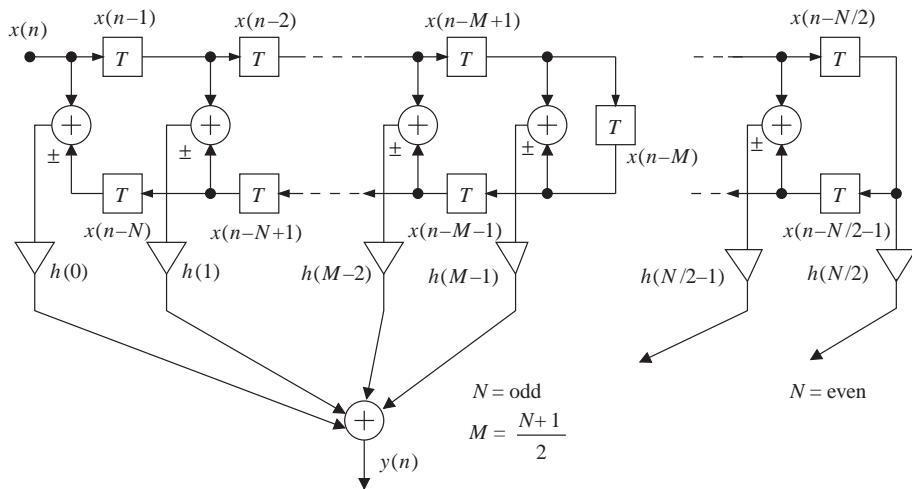
**FIGURE 6.8**

Transposed direct form FIR structure.

### 1.06.2.1.3 Linear-phase FIR structures

A major reason why FIR filters are used in practice is that they can realize an exact linear-phase response. Linear-phase implies that the impulse response is either symmetric or antisymmetric. An advantage of linear-phase FIR filters is that the number of multiplications can be reduced by exploiting the symmetry (or antisymmetry) in the impulse response, as illustrated in Figure 6.9. This structure is called a direct form linear-phase FIR structure since the phase response is independent of the coefficient values.

The number of multiplications for the direct form linear-phase structure is  $(N + 1)/2$  for  $N = \text{odd}$  and  $N/2 + 1$  for  $N = \text{even}$ . The number of multiplications is thus significantly smaller than for the more general structures in Figures 6.7 and 6.8, whereas the number of additions is the same. Subtractors are used instead of adders if the impulse response is antisymmetric. The signal levels at the inputs of the multipliers are twice as large at the input signal. Hence, the input signal should be divided by two and the filter coefficients should be scaled so that a proper out signal level is obtained. The roundoff noise is only  $(N + 1)Q^2/24$  and  $(N + 2)Q^2/24$  for  $N = \text{odd}$  and even, respectively, independently of the filter coefficients.

**FIGURE 6.9**

Direct form linear-phase FIR structure.

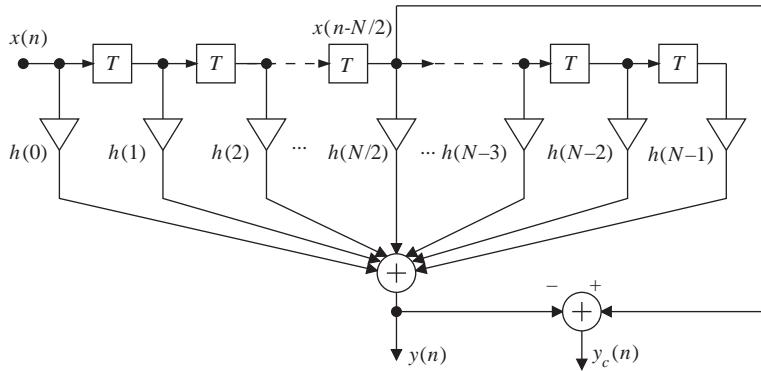
A major drawback is that the group delay for linear-phase FIR filters is often too large to be useful in many applications. Even-order FIR filters are often preferred since the group delay is an integer multiple of the sample period.

The number of arithmetic operations is further reduced in linear-phase, half-band filters. Each zero-valued coefficient makes one multiplication and one addition redundant. The number of actual multiplications in a linear-phase, half-band filter is only  $(N + 6)/4$ . The order is always  $N = 4m + 2$  for some integer  $m$ . A half-band filter is a special case of the Nyquist filters.

#### 1.06.2.1.4 Delay-complementary FIR structure

Even more dramatic reductions of the required amount of arithmetic operations is possible when a pair of delay-complementary FIR filters are needed. We illustrate the basic ideas by an example. The delay-complementary filter  $H_c(z)$  to an even-order linear-phase filter of type I or II can be obtained from the direct form linear-phase structure by subtracting the ordinary filter output from the delayed input value, as shown in Figure 6.10.

Odd order delay-complementary FIR filters are not feasible, since the symmetry axis, as shown in Figure 6.5, do not coincide with a sample in the impulse response. The cost of realizing the delay-complementary filter seems to be only one subtraction. However, the passband requirement on the FIR filter realizing  $H(z)$  must usually be increased significantly in order to meet the stopband requirements on the complementary filter,  $H_c(z)$ . Fortunately, a reduction of the passband ripple requires only a slight increase in the filter order, see (6.8). Furthermore, only even-order filters are useful. Hence, the arithmetic workload is somewhat larger than that for one single FIR filter, but is still significantly smaller than that for two separate filters. This technique can, of course, also be applied to linear-phase FIR filter structures, with either symmetric or antisymmetric impulse response, and their transposes.

**FIGURE 6.10**

Delay-complementary FIR filter with  $N = \text{even}$ .

Complementary half-band FIR filters are particularly useful, since the saving in arithmetic workload is substantial.

In many applications the need arises to split the input signal into two or more frequency bands. For example, in certain transmission systems for speech, the speech signal is partitioned into several frequency bands using a filter bank. A filter bank is a set of bandpass filters with staggered center frequencies so that the whole frequency range is covered. The first and the last filter are lowpass and highpass filters, respectively. The filtered signals are then processed individually to reduce the number of bits that has to be transmitted to the receiver, where the frequency components are added into an intelligible speech signal.

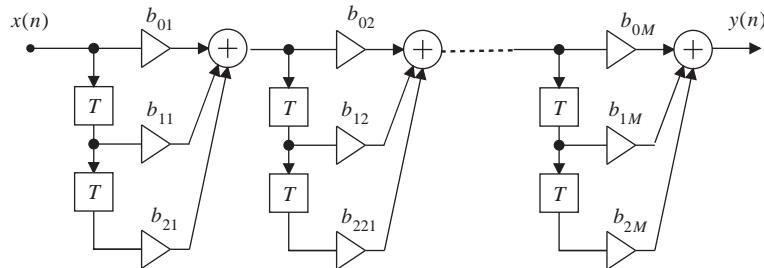
A special case of band splitting filters, a lowpass and highpass filter pair, is obtained by imposing the following symmetry constraints. Let  $H(z)$  be an even order FIR filter ( $N$  even). The delay-complementary transfer function  $H_c(z)$  is defined by

$$|H(e^{j\omega T}) + H_c(e^{j\omega T})| = 1. \quad (6.16)$$

This requirement can be satisfied by two FIR filters that are related according to

$$H(z) + H_c(z) = z^{-N/2}. \quad (6.17)$$

Hence, the two transfer functions are complementary. A signal is split into two parts so that if the two parts are added they combine into the original signal except for a delay corresponding to the group delay of the filters. Note that the attenuation is 6.02 dB at the cross-over. The output of the complementary filter  $H_c(z)$  can be obtained, as shown in Figure 6.10, by subtracting the ordinary filter output from the central value  $x(n - N/2)$ , which reduces the arithmetic workload significantly. However, the complementary filter is not obtained for free, because if the stopband attenuation is large, then the passband ripple of the ordinary filter must be very small. Hence, the filter order must be increased. The complementary FIR filters considered here cannot be of odd order since the center value of the impulse response in that case is not available.

**FIGURE 6.11**

FIR filter in cascade form.

#### 1.06.2.1.5 Cascade form FIR structures

High-order FIR and IIR filters may be realized in cascade form, i.e., as a cascade of several lower-order subfilters, as shown in Figure 6.11 for an FIR filter. The overall transfer function is

$$H(z) = \prod_{k=1}^M H_k(z), \quad (6.18)$$

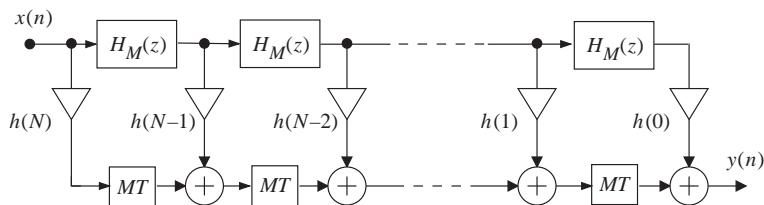
where the subfilters  $H_k(z)$  usually have order one, two, or four. Cascading of two minimum-phase filters yields an overall minimum-phase filter. Moreover cascading linear-phase filter also results in an overall linear-phase filter.

The benefit of cascading several filters is that the stopband attenuations of the filters adds. Hence, each subfilter need only to contribute with a small fraction of the overall stopband attenuation and they can therefore be simplified. The coefficients in the subfilters need not to be as accurate, i.e., the coefficient sensitivity in the stopband is small. The benefit obtained in the stopband sensitivity is, however, offset by an increased sensitivity in the passband since errors in the lower-order filters add. Furthermore, the cascade form suffers from a decrease in the dynamic signal range. Hence, the internal data wordlength must be increased. The ordering of the sections is important, since it affects dynamic signal range significantly. There is  $M!$  possible ways to order the sections for an FIR filter realized with  $M$  sections.

A minimum-phase filter can be partitioned into a linear-phase filter and a filter with nonlinear phase in cascade form [14].

#### 1.06.2.1.6 Lattice FIR structures

Lattice FIR structures are used in a wide range of applications, for example, speech and image processing, adaptive filters, and filter banks [31]. The lattice structure can be of FIR or IIR type. Note, however, the lattice structures discussed in this section are a different type of structure compared to the lattice wave digital filter structures that will be discussed in Section 1.06.7.9, although confusingly they have the name lattice in common.

**FIGURE 6.12**

FIR filter with identical subnetworks.

#### 1.06.2.1.7 FIR filters with identical subnetworks

Saramäki has developed a general theory for the design of structural frequency transformation of linear-phase FIR filters. The resulting structures consist of a tapped cascaded interconnection of identical FIR subfilters [37–39]. Figure 6.12 shows an example of these structures where the subfilters are identical, even-order linear-phase filters.

#### 1.06.2.1.8 Recursive frequency-sampling structures

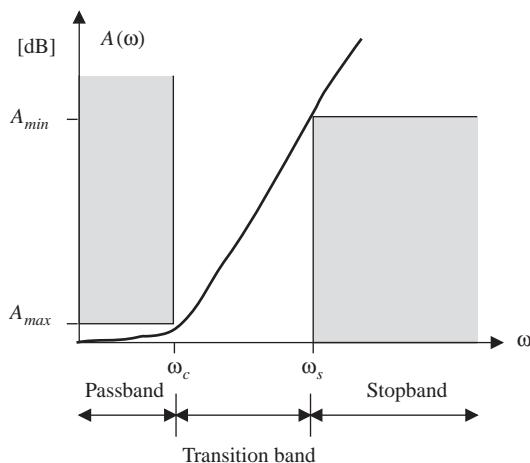
Traditionally, most textbooks [24,29,40] contain a discussion of so-called frequency-sampling FIR filter structures. These realizations are recursive algorithms that rely on pole-zero canceling techniques. Although they may seem to be interesting they are not recommended due to their high coefficient sensitivity, low dynamic range, and severe stability problems [14].

#### 1.06.2.1.9 Shifted permuted difference coefficient structure

The PDC approach was proposed by Nakayama [41]. This method was proposed for implementation of FIR filters, but it can be used to compute any sum-of-products. The basic idea is to exploit the fact, that the adjacent values in the impulse response have similar magnitudes, to simplify the multiplier network into a network consisting of only add/subtractors and shift operations. The shifted permuted difference method (SPDC) [14,42] is a generalization of the PDC method that yields fewer arithmetic operations. These approaches are simple alternatives to using more sophisticated multiple-constant multiplier techniques, which often yields better results. Multiple-constant multiplier techniques are discussed further in Section 1.06.12.1.

#### 1.06.2.1.10 Multirate FIR filters

Multirate filters are efficient in terms of arithmetic workload since a large part of the arithmetic operations can be performed at a lower rate [15,16]. Polyphase structures are efficient for decimation and interpolation since redundant arithmetic operations are avoided. FIR and IIR filters with narrow transition bands can be realized by methods based on decimation-filtering-interpolation schemes [14]. Common to these techniques is that the effective number of arithmetic operations per sample is reduced.

**FIGURE 6.13**

Standard specification of a digital lowpass filter.

### 1.06.3 The analog approximation problem

Frequency selective filters are specified in the frequency domain in terms of an acceptable deviation from the desired behavior of the magnitude or attenuation function. Figure 6.13 shows a typical attenuation specification for an analog lowpass filter. The variation (ripple) in the attenuation (loss) function in the passband may not be larger than  $A_{max}$  and the attenuation in the stopband may not be smaller than  $A_{min}$ . Usually the acceptable tolerances are piecewise constant. It is convenient during the early stages of the filter design process to use a normalized filter with unity gain, i.e., the minimum attenuation is normalized to 0 dB. The filter is provided with the proper gain in the later stages of the design process.

#### 1.06.3.1 Typical requirements

*Passband:* The frequency band occupied by the wanted signal is referred to as the passband. In this band the ideal requirement for the filter is to provide constant loss and constant group delay so that the wanted signal will be transmitted with no distortion. The passband frequency and the acceptable tolerance  $A_{max}$  for an analog lowpass filter is from 0 to  $\omega_c$ .

*Stopband:* The frequency bands occupied by the unwanted signals are referred to as the stopbands. In these bands the common form of specification merely requires the attenuation relative to the lower limit set for the passband, to be equal to or greater than some minimum amount. The stopband begins at  $\omega_s$  for an analog lowpass filter and extends towards infinity. For a digital lowpass filter the stopband begins at  $\omega_s T$  and extends to  $\pi$  rad. It is normally of no interest whether the actual filter loss in the stopband exceeds the specified amount by 1 dB or even 20 dB, and in this sense

stopband attenuations have only lower limits, in contrast to passbands where there are both upper and lower limits.

*Transition band:* The transition band is from  $\omega_c$  to  $\omega_s$ . There are no requirements on the attenuation in the transition band. The loss function of a filter is a continuous function of frequency and is thus unable to have jump discontinuities. For this reason there must always be some interval in the frequency spectrum, separating the edge of the passband from the edge of the stopband, in which the loss can raise from the low value in the passband to that required in the stopband. The bandwidth allocated to the transition band is one of the main factors determining the order of the filter needed to meet the specification. Moreover, as the width of the transition bands are decreased, not only does the complexity of the filter increase, but it becomes also more difficult to meet the specification for the passband. Narrower transition bands mean that the attenuation has to change more rapidly near the passband edges, and this causes the passband response to be more sensitive to both component losses and component tolerances.

*Phase response:* Linear-phase or near linear-phase response is often desirable or even required in many filter applications. However, this is usually specified in terms of the group delay.

*Group delay:* For the same reason that one cannot get exactly constant loss over a finite band, it is also impossible to get exactly constant group delay over a finite band with lumped element filters. However it is possible to get exactly linear-phase with filters built with distributed elements and with digital FIR filters.

*Impulse response:* The impulse response is mainly used as a theoretical concept and very rarely the impulse response is specified.

*Step response:* In most applications frequency domain characteristics such as phase and group delay requirements must be met, but in some cases additional requirements in the time domain are used. For example, step response and intersymbol interference requirements.

### 1.06.3.2 Standard lowpass approximations

Many filter approximations, have been developed to meet different requirements, particularly for analog filters. The main work has focused on approximations of lowpass filters, since highpass, bandpass, and bandstop filters can be obtained from lowpass filters through frequency transformations [18,20,43]. It is also possible to use these results to design digital filters. The classical lowpass filter approximations (standard approximations), which can be designed by using most standard filter design programs [21] are:

*Butterworth:* The magnitude function is maximally flat at the origin and monotonically decreasing in both the passband and the stopband. The group delay and variation within the passband is large. This approximation requires a larger filter order to meet a given specification than the filter approximations discussed below.

*Chebyshev I:* The magnitude function has equal ripple in the passband and decreases monotonically in the stopband. The group delay and the variation of the group delay within the passband is somewhat less than for a Butterworth approximation that meets the same attenuation requirement. A lower filter order is required compared to the Butterworth approximation.

*Chebyshev II (Inverse Chebyshev):* The magnitude function is maximally flat at the origin, decreases monotonically in the passband, and has equal ripple in the stopband. The group delay is smallest

of the four approximations and it has the smallest variation within the passband. The same filter order is required as for the Chebyshev I approximation.

*Cauer:* The magnitude function has equal ripple in both the passband and the stopband. The group delay and its variation is almost as low as the for Chebyshev II approximation. The Cauer filter, also called elliptic filter, requires the smallest order to meet a given magnitude specification.

#### 1.06.3.2.1 Comparison of the standard approximations

In comparing the standard approximations Butterworth, Chebyshev I, Chebyshev II, and Cauer filters, we find that the two latter has lower variations in the group delay. In literature it is often stated that Cauer filters have larger variation in the group delay than, i.e., Butterworth filters and that this is a valid reason for using the Butterworth approximation. The mistake in this argument is that the two filter approximations are compared using the same filter order. This is obviously not correct, which is evident of the following example, since Cauer filters can handle a considerably stricter requirement on the magnitude function. Even the step response for a Cauer filter is better. The difference between Chebyshev II and Cauer filters is however relatively small, the former has a somewhat smaller group delay, but the order is on the other hand larger.

#### 1.06.3.2.2 Example 1

Compare the Butterworth, Chebyshev I, Chebyshev II, and Cauer approximations, which meet the same standard LP specification:  $A_{\max} = 0.00695 \text{ dB}$  ( $\rho = 4\%$ ),  $A_{\min} = 45 \text{ dB}$ ,  $\omega_c = 1 \text{ rad/s}$  and  $\omega_s = 2 \text{ rad/s}$ .

As will be further discussed in Section 1.06.4.5,  $A_{\max}$  has been chosen very small in order to achieve low passband sensitivity and we may use components with large tolerances  $\varepsilon$  at the expense of a slightly higher filter order.

We get the following filter orders with the four standard approximations:

*Butterworth:*  $N_B = 12.12 \implies N_B = 13$ ,

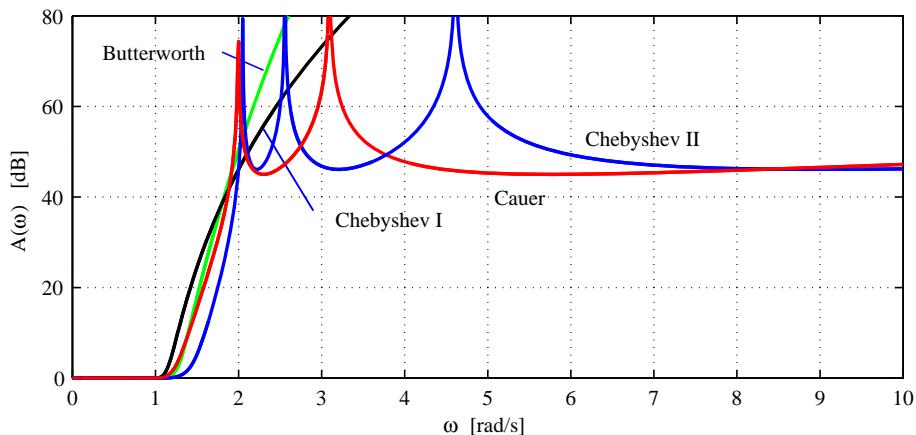
*Chebyshev I and Chebyshev II:*  $N_C = 6.904 \implies N_C = 7$ ,

*Cauer:*  $N_{Ca} = 4.870 \implies N_{Ca} = 5$ .

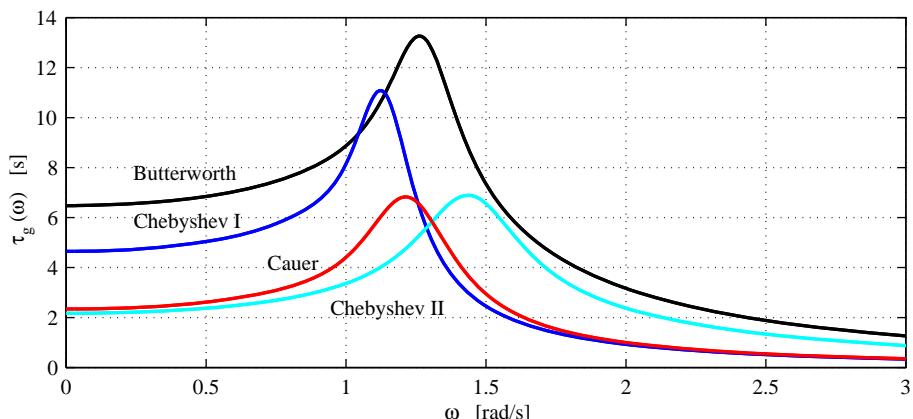
Note the large difference between the required orders for the standard approximations that meet the same requirement. The difference tends to increase if the transition band is reduced.

Figure 6.14 shows the attenuation for the four approximations. The allowed passband ripple is very small and we are only interested in that the requirement is met, and not in detailed variation inside the passband.

The attenuation in the transition band and the stopband varies between the different filters. The Chebyshev II filter has a more gradual transition between the passband and stopband compared with the Chebyshev I filter in spite of the fact that they have the same order. Note that the Cauer filter has a smaller transition band than required. The attenuation approaches, in this case, i.e., odd order filters, infinity for all of the filters. Figure 6.15 shows the corresponding group delays. The difference in the group delay is large between the different approximations and the group delays maximum lies above the passband edge  $\omega_c = 1 \text{ rad/s}$ . The Butterworth and Chebyshev I filters have larger group delay in the passband while the Chebyshev II and Cauer filters have considerably smaller group delay and

**FIGURE 6.14**

Attenuation for the four standard approximations.

**FIGURE 6.15**

Group delays for the four approximations.

the difference between the latter two is relatively small. In literature, it is commonly stated that the Butterworth filter has the best group delay properties. This is obviously not correct and is based on an unfair comparison between the standard approximations of the same order. According to Figure 6.15, Chebyshev II and Cauer filters have considerably lower group delays.

The element sensitivity for an *LC* filter is proportional to the group delay. The group delays at the passband edge and the passband variations are shown in Table 6.1. Notice that the Chebyshev II

**Table 6.1** Group Delays [s]

	$\tau_g(\omega_c)$	$\tau_g(\omega_c) - \tau_g(0)$
Butterworth	8.87115	2.39415
Chebyshev I	8.15069	3.4990
Chebyshev II	3.36427	1.1934
Cauer	4.41338	2.0718

approximation have the smallest variation. We will in Section 1.06.4.5 discuss the affect of the group delay on the sensitivity of a class of filter structures.

### 1.06.3.3 Frequency transformations

Optimal highpass filters can be derived from an optimal lowpass filter using frequency transformations [18, 20, 43]. However, the frequency transformations yield suboptimal bandpass and bandstop filters. Optimal bandpass and stopband filters require more advanced methods [18, 20, 21, 43].

---

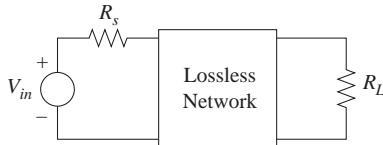
## 1.06.4 Doubly resistively terminated lossless networks

Passive *LC* filters belongs to the oldest implementation technologies, but they still plays an important role since they are being used as prototypes for the design of advanced frequency selective filters. A drawback with *LC* filters is that it is difficult to integrate resistors and coils with sufficiently high quality in integrated circuit technology. *LC* filters are for this reason not well suited for systems, that are implemented in an integrated circuit.

A more important aspect is, that *LC* filters are used as basis for realizing high-performance frequency selective filters. This is the case for mechanical, active, discrete-time, and *SC* filters as well as for digital filters. Examples of such methods are: imittance simulation using generalized imittance converters and inverters, Gorski-Popiel's, Bruton's, wave active filters, and topological simulations methods like leapfrog structures [20]. The main reason is, that the magnitude function for a well-designed *LC* filter has low sensitivity in the passband for variations in the element values. It is important that the reference filter is designed for low sensitivity, since the active and digital filters inherits its sensitivity properties. In fact, the sensitivity properties of the reference filter become a “lower” bound for the active and digital counterparts.

### 1.06.4.1 Maximal power transfer

To explain the good sensitivity properties of correctly designed *LC* filters, we first consider the power transferred from the source to the load in the circuit shown in Figure 6.16. A lossless reciprocal network can be realized by using only lossless circuit elements, e.g., inductors, capacitors, transformers, gyrators, and lossless transmission lines. Although, these filters are often referred to as *LC* filters. The maximal power that can be transferred to the load, which occur when the input impedance to the lossless network

**FIGURE 6.16**

Doubly resistively terminated lossless network.

equals  $R_s$ , is

$$P_{L\max} = \frac{|V_{in}|^2}{4R_s}. \quad (6.19)$$

The ratio of the output power and the maximal output power is

$$\frac{P_{out}}{P_{out\max}} = \frac{4R_s}{R_L} \left| \frac{V_{out}(j\omega)}{V_{in}(j\omega)} \right|^2 \leq 1, \quad (6.20)$$

where  $V_{in}(j\omega)$  is a sinusoidal input signal. An important observation is that the power, that the signal source can deliver to the load, is limited. The upper bound for the maximal power transfer is the base for the design of filter structures with low element sensitivity.

We define the frequency response as the ratio between input and output voltages, i.e., the relation between signal quantities and corresponding physical signal carrier, according to

$$H(j\omega) = \sqrt{\frac{4R_s}{R_L}} \frac{V_{out}(j\omega)}{V_{in}(j\omega)}. \quad (6.21)$$

It is convenient to normalize the magnitude response to  $|H(j\omega)| \leq 1$ .

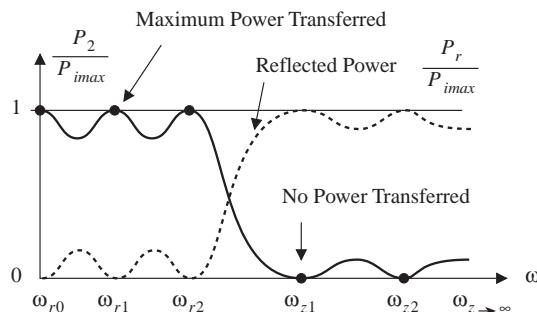
### 1.06.4.2 Reflection function

Note that the signal source, shown in Figure 6.16, does not deliver maximal power for all frequencies, since the input impedance to the reactance network is in general not equal to  $R_s$ . This can be interpreted as a fraction of the maximally available power is reflected back to the source. The relationship between the power that is absorbed in  $R_L$  and the power, which is reflected back to the source, is illustrated in Figure 6.17. Feldtkeller's equation, which is based on the assumption that the source delivers constant power, relate the power delivered to the load and the power reflected back to the source using the reflection function,  $\rho(j\omega)$ . We have

$$\frac{4R_s}{R_L} |H(j\omega)|^2 + |\rho(j\omega)|^2 = 1. \quad (6.22)$$

The reflection function is defined

$$\rho(j\omega) = \frac{Z(j\omega) - R_s}{Z(j\omega) + R_s}, \quad (6.23)$$

**FIGURE 6.17**

Transferred and reflected power.

where  $Z(j\omega)$  is the input impedance to the lossless network in Figure 6.16. The magnitude of the reflection function will be small in the passband. We will later show that the reflection function is a key to design of low sensitive filter structures.

#### 1.06.4.3 Element sensitivity

A measure of sensitivity is relative sensitivity of the magnitude function

$$\frac{|H(j\omega)|}{S_x} = \frac{\frac{\partial |H(j\omega)|}{\partial x}}{\frac{|H(j\omega)|}{x}}. \quad (6.24)$$

It is difficult to find a simple and good measure of how the attenuation changes, when several circuit element varies at the same time. The reason for this is that the influence of errors in different element values interacts. In fact, for a doubly resistively terminated reactance network we will demonstrate, that they tend to cancel. We shall therefore use and interpret sensitivity measures according to (6.24) with care. It is very difficult to compare different filter structures in a fair way.

#### 1.06.4.4 Passband sensitivity

The sensitivity of, for example, the magnitude function with respect to a circuit element,  $x$ , is a function of the angular frequency. The sensitivity in the passband can be determined from the derivative of the Feldtkeller's equation with respect to an arbitrary circuit element,  $x$ . We get

$$\frac{|H(j\omega)|}{S_x} = -\frac{R_L}{4R_s} \left| \frac{\rho(j\omega)}{H(j\omega)} \right| \frac{|\rho(j\omega)|}{S_x}. \quad (6.25)$$

For a doubly resistively terminated  $LC$  filter we have

$$|\rho_1(j\omega)| = |\rho_2(j\omega)| = \sqrt{10^{0.1A_{\max}} - 1}, \quad (6.26)$$

where  $A_{\max}$  is the acceptable ripple in the passband. Hence, the magnitude of the reflection function will be small in the passband if  $A_{\max}$  is small.

Fettweis showed (1960) that the sensitivity becomes minimal if the filter is designed for maximal power transfer at a number of angular frequencies in the passband. At these angular frequencies, the reflection function  $\rho(j\omega)$  is zero, since input impedance to the network in Figure 6.16 is  $Z(j\omega) = R_s$ . The sensitivity at these frequencies is therefore, according to (6.25), zero. If  $A_{\max}$  is small, both the reflection coefficient, according to (6.23) and the magnitude of the reflection function  $|\rho(j\omega)|$ , according to (6.26), will be small throughout the passband. Hence, the sensitivity will be small. If the ripple is decreased in the passband, the sensitivity is also decreased.

The fact that a doubly resistively terminated  $LC$  filter has low element sensitivity can also be realized through the following reasoning. Irrespective of if the element value is increased or decreased from its nominal value,  $P_{\text{out}}$  will decrease, since  $P_{\text{out}} = P_{\text{outmax}}$  for the nominal element value. Since the derivative is zero where the function has a maximum, i.e., for  $\omega = \omega_k$  with nominal element values. If there are many angular frequencies,  $\omega_k$ , with maximal power transfer, the sensitivity will be low throughout the passband. This line of reasoning is referred to as Fettweis-Orchard's argument [20, 44].

#### 1.06.4.5 Errors in the elements in doubly terminated filters

Figure 6.18 shows the typical deviation in the attenuation for a doubly resistively terminated  $LC$  filters due to errors in the reactive elements.

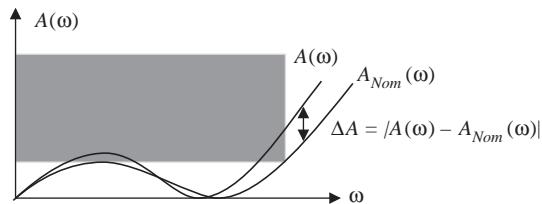
It can be shown that the deviation in the passband attenuation, as shown in Figure 6.18, for a doubly resistively terminated filter is [45].

$$\Delta A(\omega) \leq 8.69 \varepsilon \frac{|\rho(j\omega)|}{|H(j\omega)|^2} \omega \tau_g(\omega), \quad (6.27)$$

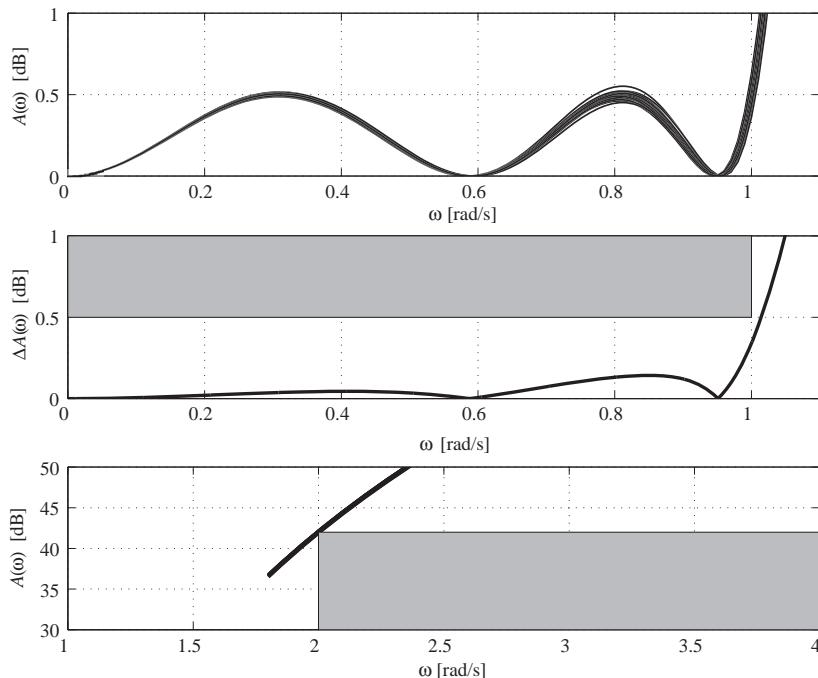
where  $e = |\Delta L/L| = |\Delta C/C|$  represent the uniformly distributed errors in the inductors and the capacitors, i.e.,  $(1 - \varepsilon)L \leq L \leq (1 + \varepsilon)L$ , etc. It can be shown that  $\Delta A(\omega)$  is proportional to the electric and magnetic energy stored in the capacitors and inductors and that (6.27) also holds for commensurate-length transmission line filters as well. The deviation will, according to (6.27), be largest for frequencies where  $\omega \tau_g(\omega)$  is largest, since the reflection function,  $|\rho(j\omega)|$ , is small and  $|H(j\omega)| \approx 1$  in the passband. Hence, a doubly resistively terminated filter with 3 dB ripple in the passband is significantly more sensitive for element errors, than a filter with smaller passband ripple, e.g., 0.01 dB. Moreover, the  $Q$  factors of the poles will be small if the passband ripple is small. Thus, it is often better to design a filter with a small ripple at the expense of a slightly higher filter order. Note that (6.27) is not valid for singly resistively terminated filters.

#### 1.06.4.6 Errors in the terminating resistors

The sensitivities with respect to  $R_s$  and  $R_L$  are proportional to the reflection function. Hence, the sensitivities are small in the passband, since  $|\rho(j\omega)| \ll 1$ , and equals zero for the frequencies at maximal power transfer. In addition, the errors will essentially appear as a small change in the gain of the filter and not affect the frequency selectivity.

**FIGURE 6.18**

Deviation in the attenuation due to element errors.

**FIGURE 6.19**

Deviation in the passband attenuation (top), bound on the attenuation (middle), and deviation in the stopband attenuation (bottom).

#### 1.06.4.7 Effects of lossy elements

The effect on the attenuation of lossy reactive elements can be estimated in terms of their  $Q$  factors, where we assume that all inductor have the same  $Q$  factor and the same holds for the capacitors [43,45].

$$\Delta A(\omega) = \frac{8.69}{2} \left( \frac{1}{Q_L} + \frac{1}{Q_C} \right) \omega \tau_g(\omega) + \frac{8.69}{2} \left( \frac{1}{Q_L} - \frac{1}{Q_C} \right) |\rho|_{\max}. \quad (6.28)$$

Also in this case it is favorable to select a small passband ripple. The deviation will be largest at the passband edge, i.e., where  $\omega\tau_g(\omega)$  is largest. However, in the case of wave digital filters, which will be discussed later, the components do not have any losses.

#### 1.06.4.7.1 Example 2

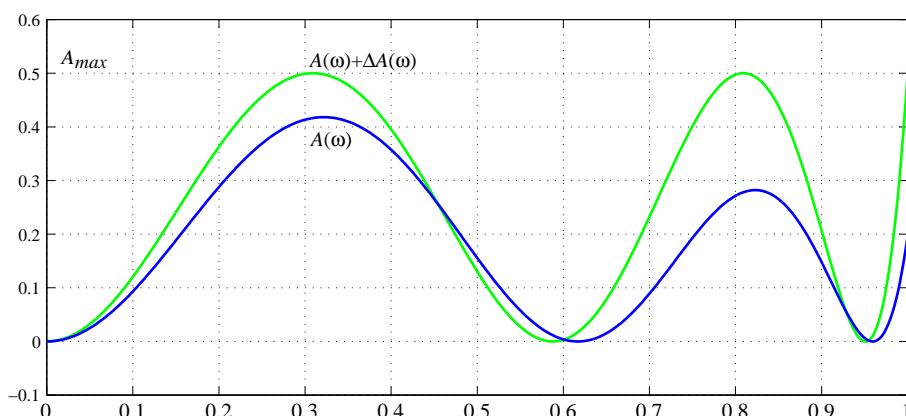
Consider a fifth-order Chebyshev I filter with  $A_{\max} = 0.5$  dB,  $A_{\min} = 42$  dB,  $\omega_c = 1$  rad/s, and  $\omega_s = 2$  rad/s and we assume that the components errors are uniformly distributed with  $\varepsilon = \pm 1\%$ .

Figure 6.19 shows the deviations in the attenuation in the passband and stopband as well as the bound on the deviation according to (6.27).

A significant number of filters do not meet the specification, since the design margin is very small. In practice, a design margin should be allocated to the two bands as well as to the bandedges [46]. Obviously, the deviation increases towards the passband edge while it is insignificant at low frequencies. Moreover, the sensitivity at the bandedge is large, and the cutoff frequency is sensitive to component errors.

#### 1.06.4.8 Filters with diminishing ripple

Due to deviations in the attenuation, caused by errors in the element values, a part of the allowed ripple in the passband,  $A_{\max}$  must be reserved to allow for errors in the component values. The filter must therefore be synthesized with a design margin, i.e., with a ripple, which is less than required by the application. According to (6.27), the deviation is smaller for low frequencies and increases towards the passband edge. In practice, however, in order to simplify the synthesis, the design margin is for the standard approximations distributed evenly in the passband, even though the margin will not be exploited for lower frequencies. In order to exploit the allowed passband ripple better, we may let the reflection function  $|\rho(j\omega)|$  of the synthesized filter decrease at the same rate as the other factors in  $\Delta A(\omega)$  increases so that  $A(\omega) + \Delta A(\omega) \leq A_{\max}$ , as shown in Figure 6.20. The ripple will decay



**FIGURE 6.20**

Passband attenuation for a fifth-order Chebyshev I filter with diminishing ripple and equiripple  $A(\omega) + \Delta A(\omega) \leq A_{\max}$  (green) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.).

towards the passband edge and the corresponding *LC* filter can be implemented with components with larger tolerances, i.e., the filter can be implemented with a lower overall cost. The group delay of a filter with diminishing ripples is slightly smaller than for the original filter. An additional advantage is that this will reduce the thermal noise as well [47].

#### 1.06.4.8.1 Example 3

The deviation due to losses and component errors, for the filters in Example 1, with  $Q_L > 200$  and  $Q_C > 50$ , are shown in Table 6.2.

Using a Chebyshev II or Cauer filter, instead of a Butterworth filter, reduces the group delay with a factor of 2.64 and 2, respectively. Hence, the component tolerances can be increased with the same factor. An additional improvement of a factor 2 to 3 can be obtained by using a diminishing ripple approximation, that allocate a larger design margin and reduces the sensitivity at the upper part of the passband. Components, with large tolerances, are considerably cheaper than those with small tolerances. In addition, the number of components is fewer; 9 and 7 compared to 13 for Butterworth. Therefore it is important to use an approximation with small group delay. Cauer is often the preferred approximation, since the required order is significantly lower than for Chebyshev II and the group delay is almost a low.

In addition, Chebyshev II or Cauer filters have lower  $Q$  factors as well. The  $Q$  factors for the four filters are shown in Table 6.3.

**Table 6.2** Deviation in the Attenuation

	$\Delta A(\omega)$ according to (6.28)	$\Delta A(\omega)$ according to (6.27)
Butterworth	0.4080 $\epsilon$ dB	3.0880 $\epsilon$ dB
Chebyshev I	0.2479 $\epsilon$ dB	2.8380 $\epsilon$ dB
Chebyshev II	0.1023 $\epsilon$ dB	1.1713 $\epsilon$ dB
Cauer	0.1342 $\epsilon$ dB	1.5365 $\epsilon$ dB

**Table 6.3**  $Q$  Factors for the Four Filters

Butterworth	Chebyshev I	Chebyshev II	Cauer
$N = 13$	$N = 7$	$N = 7$	$N = 5$
0.50000	0.50000	0.50000	0.50000
0.51494	0.68955	0.60835	0.83127
0.56468	1.33370	1.03162	3.12162
0.66799	4.34888	3.19218	
0.88018			
1.41002			
4.14812			

The conclusion is that Cauer is the best approximation in most cases, i.e., when we have requirements on both the attenuation and group delay. In addition, the Cauer approximation yields *LC* filter with fewer components and with almost as low sensitivity to errors in the element values as Chebyshev II filters.

#### 1.06.4.9 Design of doubly resistively terminated analog filters

Instead of using expensive components with low tolerances and large  $Q$  factor, we can compensate for an increase in  $\varepsilon$ , i.e., using components with larger tolerances, using some or all of the following possible trade-offs:

- Use a doubly resistively terminated reactance network that is designed for maximal power transfer, i.e., (6.27) is valid.
- Reduce  $|\rho(j\omega)|$  by reducing the passband ripple,  $A_{\max}$ , of the filter more than required by the application. However, this requires that the filter order is increased. That is, we can use a few more, but cheaper components to reduce the overall cost of the implementation.
- Use an approximation that have low group delay, i.e., Chebyshev II and Cauer filters are preferred over Butterworth and Chebyshev I filters.
- Use an approximation with diminishing ripple.

### 1.06.5 Ladder structures

*LC* ladder structures are often used to realize filters with transfer functions with zeros on the  $j\omega$ -axis or in the left-hand side of the  $s$ -plane. Zeros in the right-hand side of the  $s$ -plane cannot be realized by ladder structures, i.e., only minimum-phase transfer functions can be realized. Doubly terminated ladder structures have low element sensitivity in the passband, as discussed above, and relatively small sensitivity in the stopband.

#### 1.06.5.1 Structures for lowpass filters

Figures 6.21 and 6.22 show ladder structures of  $T$  and  $\pi$  type for realization of lowpass filters without finite zeros, respectively.  $T$  and  $\pi$  type refer to the leftmost stage of the ladder. These ladder structures can realize Butterworth and Chebyshev I lowpass filters.

Figures 6.23 and 6.24 show ladder structures that can realize transfer functions with finite zeros, e.g., Chebyshev II and Cauer filters.

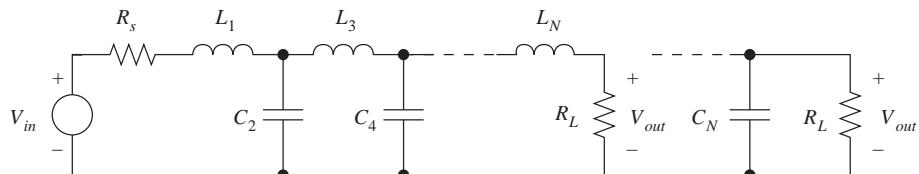


FIGURE 6.21

$N$ th-order  $T$  ladder structure for lowpass filters without finite zeros.

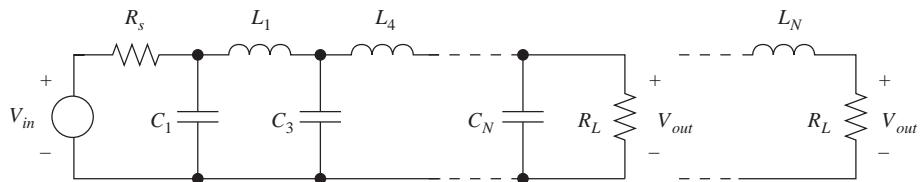


FIGURE 6.22

$N$ th-order  $\pi$  ladder structure for lowpass filters without finite zeros.

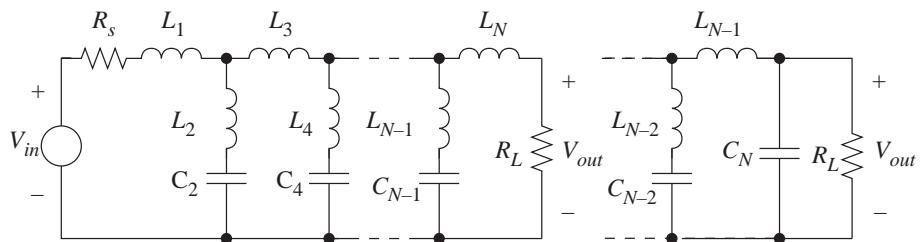


FIGURE 6.23

$N$ th-order  $T$  ladder structure with finite zeros.

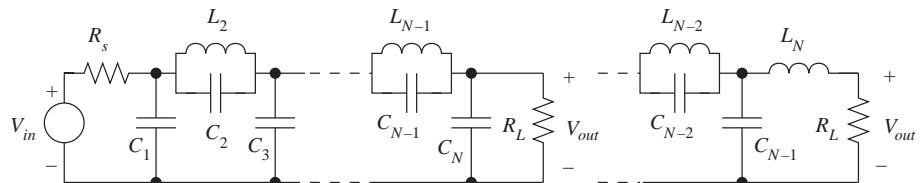


FIGURE 6.24

$N$ th-order  $\pi$  ladder structure with finite zeros.

There are two main methods for creating transmission zeros. The best method is to use series or parallel resonance circuits in the shunt and series arms in a ladder structure, respectively. The transmission zeros are created by reflecting the signal at the resonance circuits back to the source. The stopband sensitivity will be small if the transmission zeros are created by reflection. In fact, a zero pair on the  $j\omega$ -axis is determined by only two elements. The passband sensitivity and the ratio of the largest and smallest element value, depends on the ordering of the transmission zeros. For lowpass filters, are minimal sensitivity obtained if the transmission zeros closest to the passband edge is positioned in the center of the ladder structure. The positioning of the zeros is very important in narrow-band bandpass filters.

In lattice filters, which will be discussed in Section 1.06.7.9, the transmission zeros are instead created by cancellation. In fact, lattice filters have low passband sensitivities, but very poor stopband

sensitivities. The lattice structure is nevertheless useful for piezoelectric and ceramic resonator filters and as reference filter for digital filters, since the element errors in these technologies are small.

To summarize, we conclude that creating transmission zeros by means of power reflection at the shunt or series arms is a superior method compared to methods based on signal cancellation or dissipation.

A singly resistively terminated reactive network do not adhere to Feldtkeller's equation and the power transferred is either zero or upwards unbounded. It is well known that singly terminated filters are much more sensitive than doubly terminated filters. Hence, singly terminated filters are not recommended as reference filters.

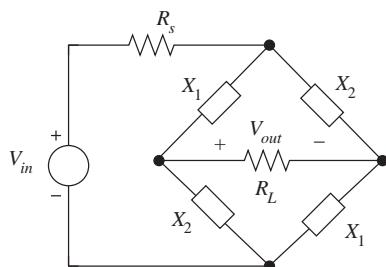
#### 1.06.5.1.1 Design of ladder structures

Doubly terminated ladder structures are generally designed by using the insertion loss method. This method involves long, complicated, and numerically ill-conditioned calculations, which must be done by a computer. Two exceptions are the Butterworth and Chebyshev I lowpass filters for which analytical solutions have been found. See [20] for details on the synthesis of ladder structures as well as design of highpass, bandpass, and bandstop  $LC$  filters. Efficient MATLAB functions for synthesis of ladder filters have been implemented in the toolbox [21].

## 1.06.6 Lattice structures

Lattice structures can be used to derive digital filter structures with low roundoff noise, high degree of parallelism, and modular circuitry that are suitable for implementation. The problems associated with unstable and inaccurate element values are not present in digital filters, since the coefficients are fixed binary values.

Figure 6.25 shows a symmetric analog lattice filter with the lossless reactances  $X_1$  and  $X_2$  and  $R_s = R_L$ . An analog lattice filter is in practice unusable, because of its high sensitivity in the stopband, except if the reactances are realized by highly stable and accurate components, for example quartz and ceramic resonators. In fact, due to the high stopband sensitivity, the lattice structure is often used as a measuring device. Lattice filters, which are designed for maximal power transfer, have, however, low coefficient sensitivity in the passband.



**FIGURE 6.25**

Analog lattice filter.

Lattice structures can be used to derive digital filter structures with low roundoff noise, high degree of parallelism, and modular circuits that are suitable for implementation. The problems associated with unstable and inaccurate element values are not present in digital filters, since the coefficients are fixed binary values.

### 1.06.6.1 Wave description of two-ports

Instead of using voltages and currents to describe electrical networks, it is convenient to use a linear combination thereof. One such linear combination is voltage waves. An impedance  $Z$  is normally described by the ratio  $Z = V_1/I_1$ , but it can also be uniquely described using voltage waves, i.e.,

$$\begin{cases} A_1 = V_1 + RI_1, \\ B_1 = V_1 - RI_1, \end{cases} \quad (6.29)$$

where  $R$  is a positive constant. Other possibilities are, for example, current waves and power waves [14,48]. The latter is often used to describe distributed networks.

#### 1.06.6.1.1 Analog lattice structure

Consider the lattice structure shown in Figure 6.26 where we have added a second optional input signal source.

The lattice structure can be described by the incident ( $A_1$  and  $A_2$ ) and reflected ( $B_1$  and  $B_2$ ) voltage waves

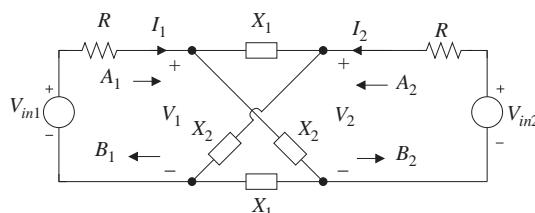
$$\begin{cases} A_1 = V_1 + RI_1 = V_{in1}, \\ B_1 = V_1 - RI_1, \end{cases} \quad (6.30)$$

and

$$\begin{cases} A_2 = V_2 + RI_2 = V_{in2}, \\ B_2 = V_2 - RI_2. \end{cases} \quad (6.31)$$

We define the (voltage wave) scattering matrix by

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = S \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}. \quad (6.32)$$



**FIGURE 6.26**

Symmetric analog lattice filter.

We get after elimination of voltages and currents in (6.30) through (6.32)

$$\begin{cases} 2B_1 = S_1(A_1 - A_2) + S_2(A_1 + A_2), \\ 2B_2 = S_1(A_2 - A_1) + S_2(A_1 + A_2), \end{cases} \quad (6.33)$$

where

$$S_1 = \frac{X_1 - R}{X_1 + R}, \quad (6.34)$$

$$S_2 = \frac{X_2 - R}{X_2 + R} \quad (6.35)$$

are the reflectance functions for  $X_1$  and  $X_2$ . Note that  $S_1$  and  $S_2$  are allpass functions if  $X_1$  and  $X_2$  are reactances. We can rewrite (6.33) as

$$\begin{cases} B_1 = \frac{S_1 + S_2}{2} A_1 + \frac{S_2 - S_1}{2} A_2, \\ B_2 = \frac{S_2 - S_1}{2} A_1 + \frac{S_1 + S_2}{2} A_2 \end{cases} \quad (6.36)$$

from which it follows that

$$s_{11} = \frac{S_1 + S_2}{2}, \quad (6.37)$$

$$s_{21} = \frac{S_2 - S_1}{2}. \quad (6.38)$$

The scattering parameter  $s_{21}(s)$  correspond to the transfer function and  $s_{11}(s)$  to the reflection function. Feldtkeller's equation can be expressed in terms of the scattering parameters  $s_{11}$  and  $s_{21}$  as

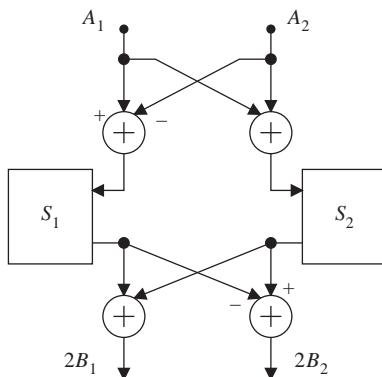
$$|s_{11}|^2 + |s_{21}|^2 = 1 \quad (6.39)$$

Hence,  $s_{11}$  and  $s_{21}$  are power-complementary transfer functions. Figure 6.27 shows the wave-flow diagram for the lattice filter. Note that the filter consists of two allpass filters in parallel and that the two outputs are obtained by adding and subtracting the outputs of the allpass filters. The normal output is  $B_2$ . Only odd-order lowpass filters can be realized using a lattice structure. Hence,  $S_1$  and  $S_2$  are odd (even) and even (odd), respectively. See [14,49,50] how to select  $S_1$  and  $S_2$  for more general cases.

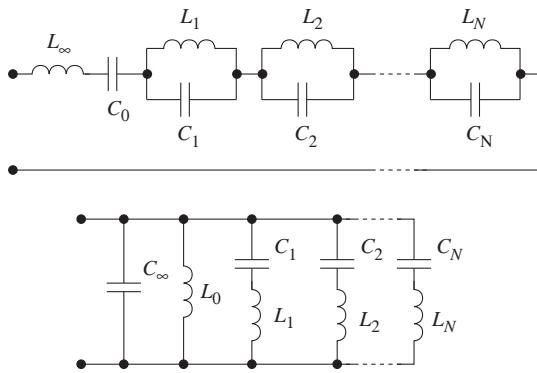
### 1.06.6.2 Realization of reactances

The reflectance functions  $S_1$  and  $S_2$  correspond according to (6.34) and (6.35) to the reactances  $X_1$  and  $X_2$ . A reactance,  $X(s)$  can be realized in several canonic ways, see [20] for more design details and MATLAB programs. For example, Foster I and II realizations are shown in Figure 6.28. Alternative realizations are Cauer I and II ladders, which are shown in Figure 6.29.

An arbitrary  $N$ th-order reactance  $X(s)$  can be realized by  $N$  lossless commensurate-length transmission lines (unit elements) using a Richards' structure, which is shown in Figure 6.30. The structure is terminated with either a short-circuit or open-circuit.

**FIGURE 6.27**

Wave-flow diagram for a lattice filter.

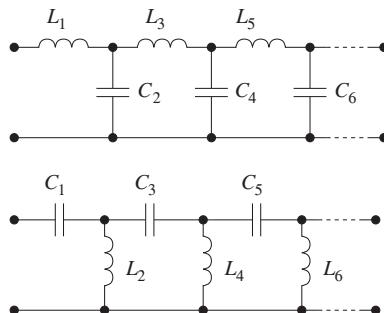
**FIGURE 6.28**

Foster I (upper) and II (lower) structures.

### 1.06.6.3 Design of analog lattice structures

Note that both  $S_1$  and  $S_2$  are allpass functions. The scheme of splitting an odd-order lowpass transfer function into two allpass sections, is as follows:

- Assign the real pole to  $S_1$  and remove it from the set of poles. Note that only odd-order lowpass and highpass filters are feasible.
- Next, assign to  $S_2$  the pole pair with the smallest angle between the negative real axis and the vector from the origin to the positive pole and remove the pole pair from the set of pole.
- Continue in the same way by alternating assigning the pole pair with the smallest angle to  $S_1$  and  $S_2$  until all poles have been assigned.

**FIGURE 6.29**

Cauer I (upper) and II (lower) structures.

**FIGURE 6.30**

Realization of an  $N$ th-order reactance.

- The two allpass filters are obtained by adding the zeros according to  $s_z \leftarrow -s_p$ . That is, each pole has a corresponding zero, but in the right-hand side half-plane.
- Determine the reflectance functions from the allpass functions.
- Finally, select suitable realizations of the two reactances.

#### 1.06.6.4 Commensurate-length transmission line networks

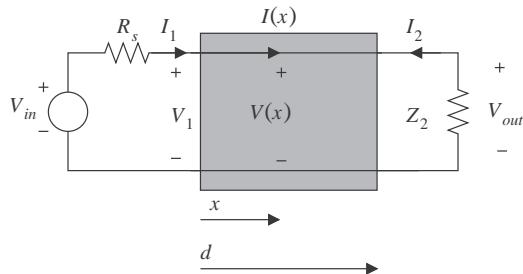
In a commensurate-length transmission line networks, all transmission lines have a common electrical propagation time. Commensurate-length transmission line networks constitutes a special case of distributed element networks that can easily be designed by using a mapping to a lumped element structure, which can be designed using common analytical or computer-aided tools. This mapping is based on a one-to-one mapping between the lumped element network and the transmission line network expressed in terms of a complex frequency variable, Richards' variable,  $\Psi$ . Lossless commensurate-length transmission lines can be used to build low sensitivity filters.

A commensurate-length transmission line with length  $d$  is illustrated in Figure 6.31. The voltage and current at various positions along the line can be described by an incident and a reflected wave. Under stationary conditions we have

$$\begin{cases} V(x) = Ae^{-\gamma x} + Be^{\gamma x}, \\ I(x) = \frac{1}{Z_0}(Ae^{-\gamma x} - Be^{\gamma x}), \end{cases} \quad (6.40)$$

where  $A$  and  $B$  are constants, which can be determined by the boundary conditions. For example,  $V_1 = V(0)$  and  $I_1 = I(0)$ . Further we have

$$\gamma^2 = (r + j\omega l)(g + j\omega c) \quad (6.41)$$

**FIGURE 6.31**

Transmission line.

and

$$Z_0^2 = \frac{r + j\omega l}{g + j\omega c}, \quad (6.42)$$

where  $r$ ,  $l$ ,  $g$ , and  $c$  are the primary constants for the line: resistance, inductance, and conductance per unit length, respectively.  $\gamma$  is the propagation constant and  $Z_0$  is the characteristic impedance.  $Z_0$  is real-valued positive constant,  $Z_0 = R = \sqrt{\frac{l}{c}}$ , for lossless transmission lines and is therefore sometimes called characteristic resistance.

#### 1.06.6.4.1 Richards' variable

Commensurate-length transmission line filters constitute a special case of distributed element networks that can easily be designed by mapping them to a lumped element structure. This mapping involves Richards' variable, which is defined

$$\Psi = \frac{s^{sT} - 1}{s^{sT} + 1} = \tanh\left(\frac{sT}{2}\right), \quad (6.43)$$

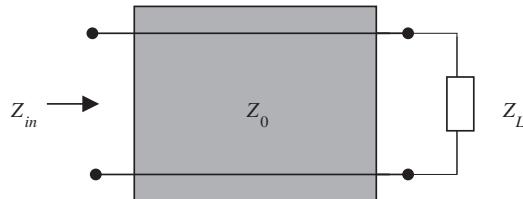
where  $\Psi = \Sigma + j\Omega$  and  $T$  is the propagation time for a wave through the transmission line and back. Richards' variable is a dimensionless complex-valued variable. The real frequencies in the  $s$ - and  $\Psi$ -domains are related by

$$\Omega = \tan\left(\frac{\omega T}{2}\right). \quad (6.44)$$

Note the similarity between the bilinear transformation and Richards' variable. In fact, commensurate-length transmission line filters have the same periodic frequency responses as digital filters. Substituting Richards' variable into the chain matrix for a transmission line we get [14]

$$K = \frac{1}{\sqrt{1 - \Psi^2}} \begin{bmatrix} 1 & Z_0\Psi \\ \Psi & 1 \end{bmatrix}. \quad (6.45)$$

The matrix in (6.45) has element values that are rational functions in Richards' variable, except for the square-root factor. However, this factor can be handled separately during the synthesis. The synthesis procedures (programs) used for lumped element design can therefore be used with small modifications for synthesis of commensurate-length transmission line filters.

**FIGURE 6.32**

Terminated transmission line (UE–Unit Element).

#### 1.06.6.4.2 Unit elements

The transmission line filters of interest are with a few exceptions built using only one-ports. At this stage it is therefore interesting to study the input impedance of the one-port shown in Figure 6.32. A lossless transmission line described by (6.45) is called a unit element (UE). The input impedance to a unit element, which is terminated by the impedance  $Z_L$ , can be derived from (6.45).

We get the input impedance to a transmission line, with characteristic impedance  $Z_0$  and loaded with an impedance  $Z_L$ , as

$$Z_{in}(\Psi) = \frac{Z_L + Z_0\Psi}{Z_0 + Z_L\Psi}. \quad (6.46)$$

We are interested in the input impedance of a lossless transmission line with characteristic impedance  $Z_0 = R$  that is terminated by an impedance in the following three cases.

#### 1.06.6.4.3 $Z_L = Z_0$ (**matched termination**)

For case,  $Z_L = Z_0$ , we have according to (6.46)

$$Z_{in} = R. \quad (6.47)$$

Hence, we have a matching between the unit element and the load, and an incident wave to the load, will not be reflected.

#### 1.06.6.4.4 $Z_L = \infty$ (**open-ended**)

We get

$$Z_{in} = \frac{R}{\Psi}. \quad (6.48)$$

Hence, an open-ended unit element can be interpreted as a new kind of capacitor, i.e., a  $\Psi$ -plane capacitor with the value  $1/R$ .

#### 1.06.6.4.5 $Z_L = 0$ (**short-circuited**)

A short-circuited unit element has the input impedance

$$Z_{in} = R\Psi \quad (6.49)$$

which can be interpreted as a  $\Psi$ -plane inductor with the value  $R$ .

## 1.06.7 Wave digital filters

In this section we discuss the design of wave digital filters based on different classes of doubly resistively terminated networks consisting of lossless commensurate-length transmission lines. Wave digital filters inherit the low element sensitivity properties from their analog counterparts, which should be designed for maximal power transfer in the passband [14, 20]. Hence, simple and short coefficients can be used in the wave digital filters. In addition, the dynamic signal range is large and the roundoff noise is low in these low-sensitivity structures.

Wave digital filters constitute a wide class of digital IIR filters and in some special cases FIR filters. Wave digital filters have many advantageous properties, and they are therefore the recommended type of IIR filters. A wave digital filter is derived from an analog filter, which is referred to as its reference filter. Due to this relationship between the wave digital filter and the corresponding analog reference filter, wave digital filter structures inherit many fundamental properties from their analog counterparts. Of foremost interest are favorable stability properties and low sensitivity with respect to variations in element values. Furthermore, the approximation problem for the wave digital filter can be carried out in the analog domain using well-known design programs. Wave digital filters comprise a whole variety of subclasses, derived from the corresponding analog filter structures.

In order to retain the low element sensitivity and robustness of the analog reference filter it is necessary to adhere to the maximal power transfer principle, which was discussed in Section 1.06.4.1. For the representation of power, or energy, it is necessary to use two variables, e.g., voltage and current [51]. Thus the signal-flow graphs discussed in Section 1.06.2.1 can not represent power.

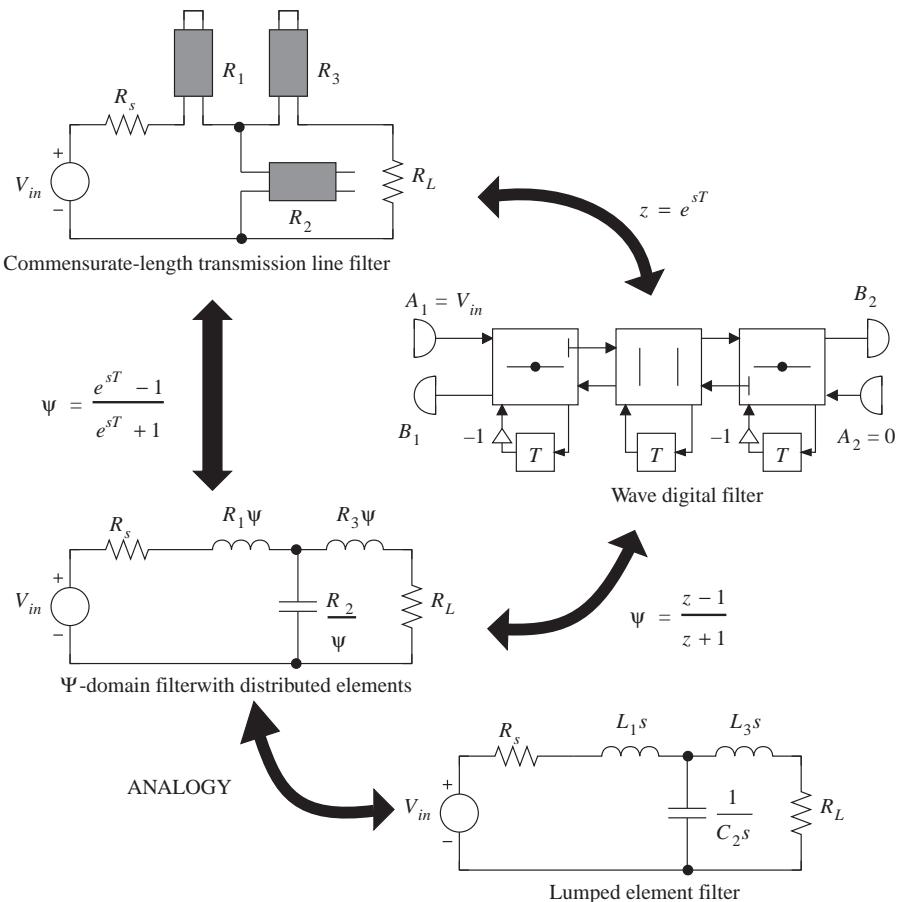
Alfred Fettweis has developed a comprehensive theory, so-called wave digital filter theory, which solves these problems [27, 48, 52]. In fact, this theory is not only a digital filter design theory, it is a general theory describing the relations between distributed element and discrete-time networks. Furthermore, it inherently contains an energy concept, which is important for stability analysis of nonlinear digital networks as well as for retaining the low-sensitivity properties of the reference filter.

An important property of wave digital filters is the guaranteed stability, which is inherited from the reference filter. In practice, the inductors in an *LC* ladder filter are nonlinear. Such nonlinearities may cause and sustain parasitic oscillations. However, in the passive *LC* filter such oscillations are attenuated, since the filter dissipates signal power. Hence, any oscillation will eventually vanish. This property is retained in the wave digital filter.

Wave digital filters, particularly wave digital lattice and circulator-tree filters, are suitable for high-speed applications. They are modular and possess a high degree of parallelism, which makes them easy to implement in hardware.

### 1.06.7.1 Design of wave digital filters

A wave digital filter is derived from an analog reference filter. The basic relations between the wave digital filter, the corresponding reference filter, i.e., the transmission line filter, and the lumped element filter, are summarized in Figure 6.33. The purpose of the lumped element filter is only to simplify the design process by allowing the use of conventional, lumped element filter theory and design tools. In principle, all commensurate-length transmission line networks can be used as reference filters for wave

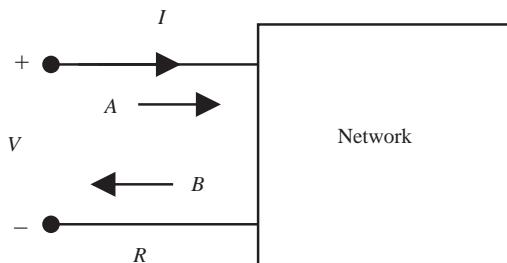
**FIGURE 6.33**

Summary of the design process for wave digital filters.

digital filters. Furthermore, wave digital filters representing classical filter structures are also viable since all lumped element networks can be uniquely mapped onto commensurate-length transmission line networks using Richards' variable.

However, certain reference structures result in wave digital filter algorithms that are not sequentially computable, because the wave-flow graph contains delay-free loops. Therefore, one of the main issues is to avoid these delay-free loops. There are three main approaches to avoid delay-free loops in wave digital filters [20, 27, 48]. Naturally, combinations of these methods can also be used. The main approaches to avoid delay-free loops are:

1. By using cascaded transmission lines, so called Richards' structures, and certain types of circulator filters.

**FIGURE 6.34**

Incident and reflected waves into a port with a port resistance  $R$ .

2. By introducing transmission lines between cascaded two-ports.
3. By using reflection-free ports.

### 1.06.7.2 Wave descriptions

It is necessary to use two variables to represent power [25,51]. In the analog domain we usually use voltages and currents, but we may also use any linear combination of voltage and current to describe a network [27,48]. In fact, a network can be described using incident and reflected waves instead of voltages and currents. Scattering parameter formalism has been used in microwave theory for a long time for describing networks with distributed circuit elements.

#### 1.06.7.2.1 Voltage waves

The one-port network shown in Figure 6.34 can be described by the incident and reflected waves instead of voltages and currents.

The voltage waves are defined as

$$\begin{cases} A = V + RI, \\ B = V - RI, \end{cases} \quad (6.50)$$

where  $A$  is the incident wave,  $B$  is the reflected wave, and  $R$  is a positive real constant, called port resistance. The port resistance corresponds to the characteristic impedance in a transmission line.

#### 1.06.7.2.2 Current waves

In a similar way, current waves can be defined, but they results in the same digital filter structure as voltage waves if we instead used the dual reference filter. Since the current wave description do not provide any additional structures we will not discuss current waves any further. There are also simple relations between wave digital filters derived using voltage, current, and power waves [27].

### 1.06.7.2.3 Power waves

In microwave theory, so-called power waves are used

$$\begin{cases} A = \frac{V}{\sqrt{R}} + \sqrt{RI}, \\ B = \frac{V}{\sqrt{R}} - \sqrt{RI}. \end{cases} \quad (6.51)$$

The name power waves comes from the fact that their squared values have the dimension of power. We will mainly use voltage waves, because they provide simpler digital realizations compared to using a power wave description.

### 1.06.7.2.4 Reflectance function

A one-port can be described by the reflectance function which is defined by

$$S = \frac{B}{A}. \quad (6.52)$$

The reflectance function serves a similar purpose as impedances when voltages and currents are used to describe a network. For example the reflectance for an impedance  $Z$  is

$$S = \frac{Z - R}{Z + R}. \quad (6.53)$$

The reflectance is an allpass function for a reactance.

## 1.06.7.3 Wave-flow building blocks

The basic building blocks for the reference filter are unit elements, that are either open- or short-circuited at the far end. The frequency response of such a unit element filter is periodic with a period of  $2\pi/\tau$ , i.e., the same as for a digital filter. The signals and components of the unit element filter can be mapped to a digital filter by sampling with the sample period,  $T = \tau$ , where  $\tau$  is the round-trip for a wave through the unit element. In the following sections, we will derive the wave-flow equivalents to some common circuit elements and interconnection networks.

### 1.06.7.3.1 Circuit elements

In this section we derive the wave-flow representation for some basic circuit elements.

### 1.06.7.3.2 Open-ended unit element

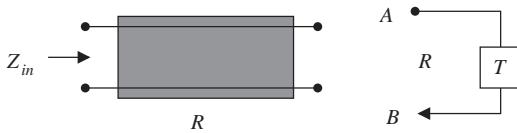
The input impedance to an open-circuited lossless unit element with  $Z_0 = R$  (a  $\Psi$ -plane capacitor) is given in (6.48). The reflectance is

$$S(\Psi) = \frac{Z_{in} - R}{Z_{in} + R} = \frac{1 - \Psi}{1 + \Psi} = e^{-sT} \quad (6.54)$$

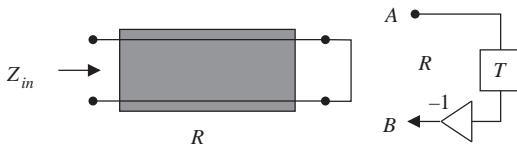
and

$$S(z) = z^{-1}. \quad (6.55)$$

Figure 6.35 show the wave-flow of an open-ended, lossless transmission line and its wave-flow equivalent.

**FIGURE 6.35**

Open-ended lossless transmission line and its wave-flow equivalent.

**FIGURE 6.36**

Short-circuited lossless transmission line and its wave-flow equivalent.

#### 1.06.7.3.3 Short-Circuited Unit element

The input impedance to a short-circuited, lossless unit element with  $Z_0 = R$  (a  $\Psi$ -plane inductor) is given by (6.49). The reflectance is

$$S(\Psi) = \frac{Z_{in} - R}{Z_{in} + R} = \frac{\Psi - 1}{\Psi + 1} = -e^{-s\tau} \quad (6.56)$$

and

$$S(z) = -z^{-1}. \quad (6.57)$$

Figure 6.36 shows the short-circuited lossless transmission line and its wave-flow equivalent.

For sake of simplicity we will not always distinguish between wave-flow graphs in the time and frequency domains. Hence, the wave variables in the figures are denoted  $A$  and  $B$ , but we will when convenient use the notation  $a(n)$  and  $b(n)$ .

An open-ended unit element corresponds to a pure delay, while a short-circuited unit element corresponds to a delay and a  $180^\circ$  phase shift.

#### 1.06.7.3.4 Matched termination

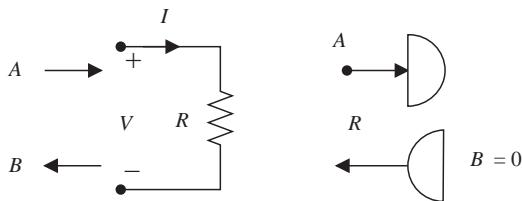
The reflectance for a lossless unit element terminated at the far end with a resistor with  $Z_0 = R$  (matched termination) is

$$S(\Psi) = 0. \quad (6.58)$$

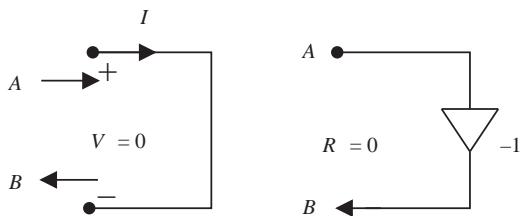
Hence, an input signal to the unit element is not reflected back to the input.

#### 1.06.7.3.5 Resistive load

The reference filters of interest are, for sensitivity reasons, doubly resistively terminated. The load resistor and its corresponding wave-flow equivalent, a wave sink, are shown to the right in Figure 6.37.

**FIGURE 6.37**

Wave-flow equivalent for a resistor,  $R$ .

**FIGURE 6.38**

Wave-flow equivalent for a short-circuit.

### 1.06.7.3.6 Short-circuit

For a short-circuit, we have  $V = 0$ , which yields

$$S(\Psi) = -1 \quad (6.59)$$

and  $B = -A$ . Note that this holds independently of the port resistance. The corresponding wave-flow graph is shown to the right in Figure 6.38.

### 1.06.7.3.7 Open-circuit

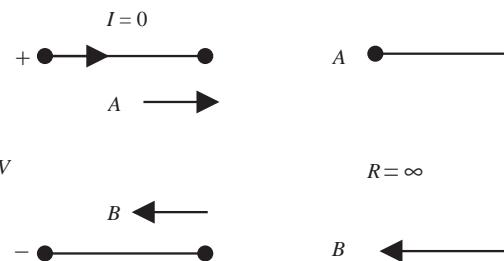
For an open-circuit, we have  $I = 0$ , which yields  $B = A$ . This result holds independently of the port resistance  $R$ . An open-circuit has the reflectance

$$S(\Psi) = 1. \quad (6.60)$$

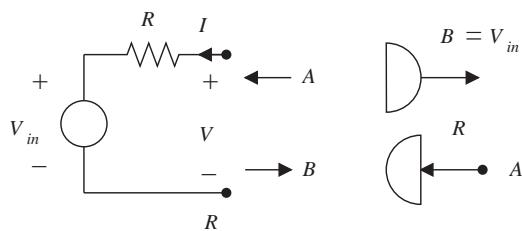
The wave-flow graph for an open-circuit is shown to the right in Figure 6.39.

### 1.06.7.3.8 Voltage signal source

For a signal source with a source resistance we have  $V_{in} = V - RI$ , which yields  $B = V_{in}$ . The incident wave is not reflected by the source since the source resistance equals the port resistance. The corresponding wave-flow graph is shown to the right in Figure 6.40.

**FIGURE 6.39**

Wave-flow equivalent for an open-circuit.

**FIGURE 6.40**

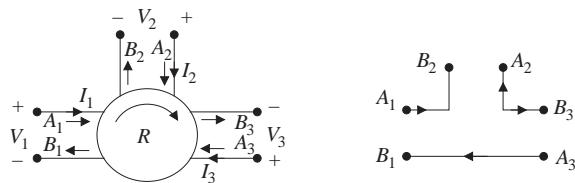
Wave-flow equivalent for a voltage source with source resistance.

### 1.06.7.3.9 Circulator

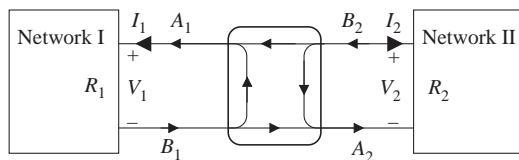
The symbol for a three-port called circulator and its corresponding wave-flow graph is shown in Figure 6.41. The name comes from the fact that an incident wave is “circulated to the next port as shown in Figure 6.41. Note that an incident wave to port 1 is reflected to port 2, and an incident wave to port 2 is reflected to port 3, and so on. Hence, the circulator “circulates the incident waves to the next port in order. Circulators can be realized in the microwave range by using ferrites. Circulators are used, for example, to direct the signals from the transmitter to the antenna and from the antenna to the receiver in a radar. The circulator is a useful component that is inexpensive to implement in wave digital filter.

### 1.06.7.4 Interconnection networks

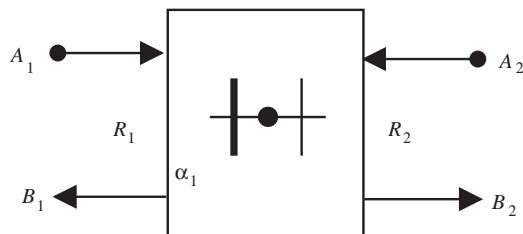
In order to interconnect different wave-flow graphs it is necessary to obey the Kirchhoff's laws at the interconnection. Generally, at the point of connection, the incident waves are partially transmitted and reflected as illustrated in Figure 6.42. The transmission and reflection at the connection port is described by a wave-flow graph called an adaptor. There are several types of adaptors corresponding to different types of interconnection networks [53]. It can be shown that any interconnection network can be decomposed into a set of interconnected two-port and three-port adaptors. Since the interconnection network is lossless, therefore the corresponding adaptor network will also be lossless (see Figure 6.43).

**FIGURE 6.41**

Three-port circulator and the corresponding wave-flow graph.

**FIGURE 6.42**

Connection of two ports.

**FIGURE 6.43**

Symmetric two-port adaptor.

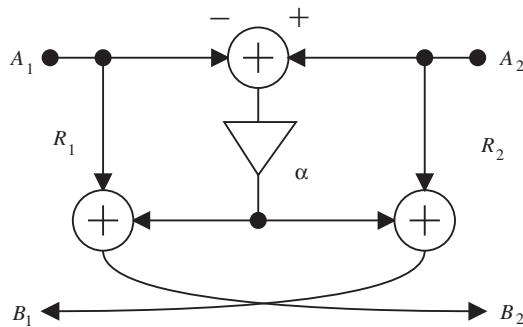
#### 1.06.7.4.1 Symmetric two-port adaptor

Figure 6.44 shows the symbol for the symmetric two-port adaptor. The incident and reflected waves for the two-port are

$$\begin{cases} A_1 = V_1 + RI_1, \\ B_1 = V_1 - RI_1 \end{cases} \quad (6.61)$$

and

$$\begin{cases} A_2 = V_2 + RI_2, \\ B_2 = V_2 - RI_2. \end{cases} \quad (6.62)$$

**FIGURE 6.44**

Symmetric two-port adaptor.

At the point of interconnection we have according to Kirchhoff's current and voltage laws

$$\begin{cases} I_1 = -I_2, \\ V_1 = V_2. \end{cases} \quad (6.63)$$

By elimination of the voltages and currents, we get the following relations between incident and reflected waves for the symmetric two-port adaptor

$$\begin{cases} B_1 = A_2 + \alpha(A_2 - A_1), \\ B_2 = A_1 + \alpha(A_2 - A_1) \end{cases} \quad (6.64)$$

and

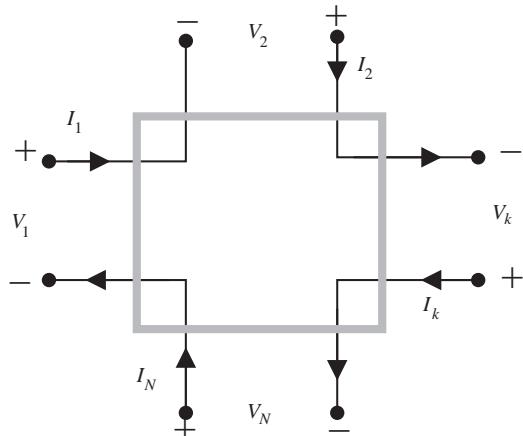
$$\alpha = \frac{R_1 - R_2}{R_1 + R_2}, \quad (6.65)$$

which are illustrated by the wave-flow graph in Figure 6.44.

Note that  $|\alpha| \leq 1$ . The adaptor coefficient  $\alpha$  is usually written on the side corresponding to port 1 and/or one of the parallel lines are thicker. As can be seen, the wave-flow graph is almost symmetric.

Note that  $\alpha = 0$  for  $R_1 = R_2$  and that the adaptor degenerates into a direct connection of the two ports and the incident waves are not reflected at the point of interconnection. For  $R_2 = 0$  we get  $\alpha = 1$  and the incident wave at port 1 is reflected and multiplied by  $-1$  while for  $R_2 = \infty$  we get  $\alpha = -1$  and the incident wave at port 1 is reflected without a change of sign.

Several different adaptor types exist that correspond to interconnections between circuit elements [53]. Symmetric adaptors are commonly used in the first step in the design of wave digital filters. In subsequent design steps these adaptors can be transformed into other types in order to optimize the dynamic signal range.

**FIGURE 6.45**

Series connected ports.

#### 1.06.7.4.2 Series adaptors

Consider the network topology shown in Figure 6.45 with  $N$  ports where the currents that flow through the ports are the same. Hence, the ports are connected in series. We have

$$\begin{cases} I_1 = I_2 = \cdots = I_N, \\ V_1 + V_2 + \cdots + V_N = 0. \end{cases} \quad (6.66)$$

After elimination of voltages and currents we get

$$\begin{cases} B_k = A_k - \alpha_k A_0, \\ A_0 = \sum_{k=1}^N A_k, \end{cases} \quad (6.67)$$

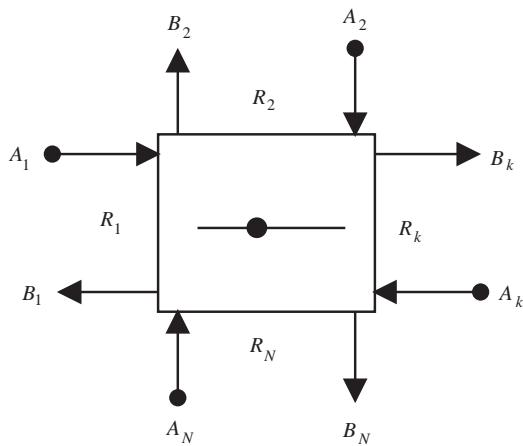
where

$$\alpha_k = \frac{2R_k}{\sum_{n=1}^N R_n}.$$

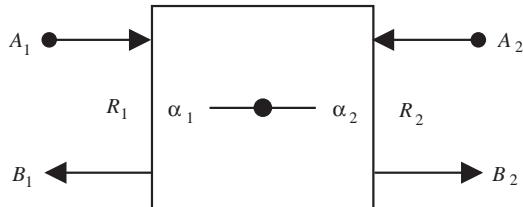
The symbol used for an  $N$ -port series adaptor is shown in Figure 6.46. It is straightforward to show that the sum of the adaptor coefficients is

$$\sum_{k=1}^N \alpha_k = 2. \quad (6.68)$$

Hence, the adaptor coefficients are linearly dependent and one of the coefficients can therefore be eliminated, i.e., expressed in terms of the others. A port for which the adaptor coefficient has been eliminated is called dependent port. It is favorable to select the port with the largest adaptor coefficient as dependent port. The number of different adaptor coefficients can be reduced further if some of the port resistances are the same.

**FIGURE 6.46**

Series adaptor.

**FIGURE 6.47**

Two-port series adaptor.

#### 1.06.7.4.3 Two-port series adaptor

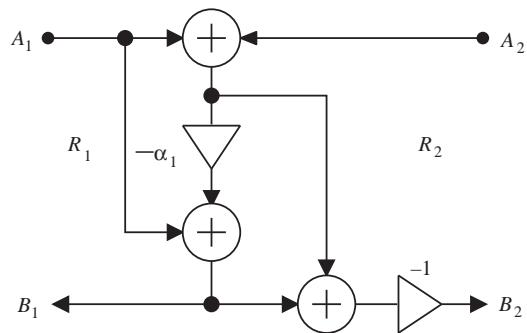
Figure 6.47 shows the symbol for the two-port series adaptor. In this case, we have only two adaptor coefficients and one of them can be eliminated since they are linearly dependent. The resulting wave-flow graph, where  $\alpha_2$  has been eliminated, is shown in Figure 6.48. The remaining adaptor coefficient is

$$\alpha_1 = \frac{2R_1}{R_1 + R_2} \quad (6.69)$$

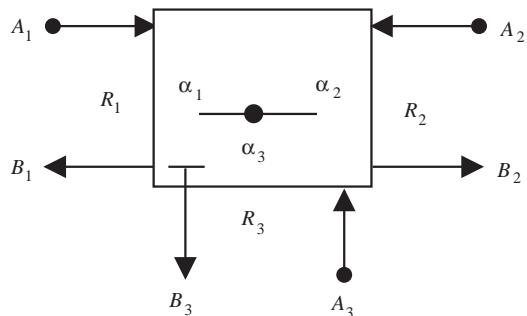
and  $0 \leq \alpha_1 \leq 2$ .

#### 1.06.7.4.4 Three-port series adaptor

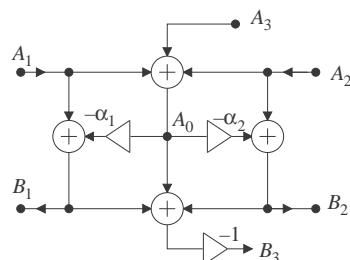
A symbol for the three-port series adaptor is shown in Figure 6.49 and in Figure 6.50 the corresponding realization of the three-port series adaptor, where adaptor coefficient  $\alpha_3$  has been eliminated. Hence, port 3 depends on the adaptor coefficients for the two other ports.

**FIGURE 6.48**

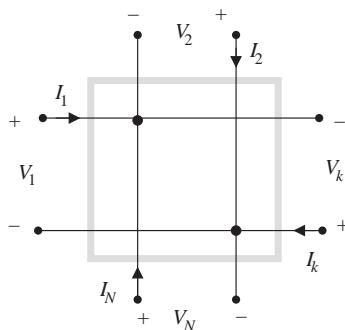
Wave-flow graphs for a two-port series adaptor.

**FIGURE 6.49**

Three-port series adaptor.

**FIGURE 6.50**

Three-port series adaptor with port 3 as the dependent port.

**FIGURE 6.51**

Parallel connection of ports.

This three-port series adaptor requires two multiplications and six additions. It is often favorable, since it required multiplications with shorter coefficient wordlength, to eliminate the adaptor coefficient with the largest magnitude. Hence, the corresponding port is select as the dependent port.

#### 1.06.7.4.5 Parallel adaptors

If all ports have the same voltage, as illustrated in Figure 6.51, then the ports are connected in parallel. We get by using the definition of parallel connection of the ports, i.e.,

$$\begin{cases} V_1 = V_2 = \dots = V_N, \\ I_1 + I_2 + \dots + I_N = 0. \end{cases} \quad (6.70)$$

By elimination of voltages and currents we get

$$\begin{cases} B_k = A_0 - A_k, \\ A_0 = \sum_{k=1}^N \alpha_k A_k, \end{cases} \quad (6.71)$$

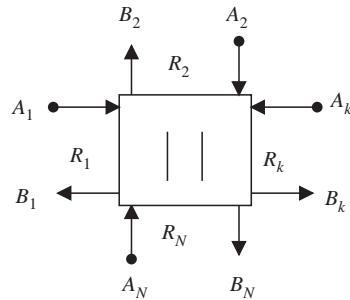
where

$$\alpha_k = \frac{2G_k}{\sum_{n=1}^N G_n}$$

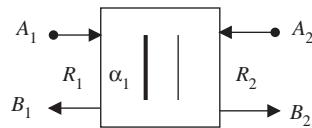
and  $G_k = 1/R_k$ . The symbol for an  $N$ -port parallel adaptor is shown in Figure 6.52. Also for the parallel adaptor we have

$$\sum_{k=1}^N \alpha_k = 2. \quad (6.72)$$

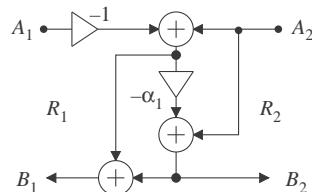
Hence, one of the adaptor coefficients can be expressed in terms of the others. The number of multiplications can be reduced further if some of the port conductances are equal.

**FIGURE 6.52**

*N*-port parallel adaptor.

**FIGURE 6.53**

Two-port parallel adaptor.

**FIGURE 6.54**

Wave-flow graph for two-port parallel adaptor.

#### 1.06.7.4.6 Two-port parallel adaptor

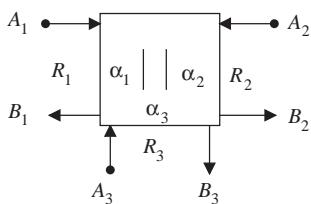
Figure 6.53 shows the symbol for a two-port parallel adaptor and the corresponding wave-flow graph is shown in Figure 6.54.

The sum of the adaptor coefficients in an adaptor is always equal to 2 with the exception of the symmetric two-port adaptor. Hence, one of the coefficients can be expressed in terms of the others and can therefore be eliminated.

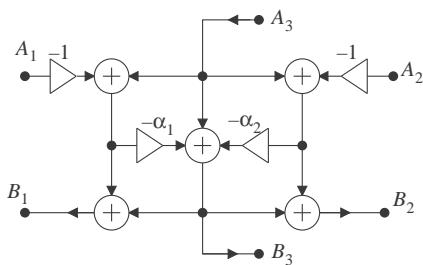
The adaptor coefficient in Figure 6.54 is

$$\alpha_1 = \frac{2G_1}{G_1 + G_2} \quad (6.73)$$

and  $0 \leq \alpha_1 \leq 2$ .

**FIGURE 6.55**

Three-port parallel adaptor.

**FIGURE 6.56**

Three-ports parallel adaptor with port 3 as dependent port.

#### 1.06.7.4.7 Three-port parallel adaptor

A general three-port parallel adaptor requires two multiplications and six additions. The symbol for the three-port parallel adaptor is shown in Figure 6.55.

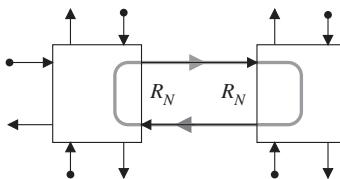
Figure 6.56 shows the wave-flow graph for the corresponding three-port parallel adaptor with port 3 as the dependent port.

#### 1.06.7.4.8 Direct interconnection of adaptors

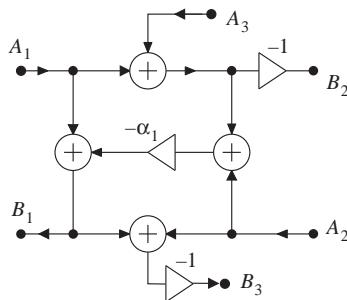
Arbitrary interconnection networks can be built using only two- and three-port series and parallel adaptors. However, in some cases also four-port adaptors are used. Unfortunately, delay-free loops will in general occur if two adaptors are connected, as illustrated in Figure 6.57. The port resistances of the two connected ports must, of course, be equal. This problem can, however, be solved by a judicious choice of the port resistances in one of the connected adaptors.

By selecting the port resistance so that one of the adaptor coefficients, corresponding to the connected ports, becomes equal to unity. We select, for example,  $\alpha_N = 1$ , which for the series adaptor is equivalent to

$$R_N = \sum_{k=1}^{N-1} R_k \quad (6.74)$$

**FIGURE 6.57**

Potentially delay-free loop.

**FIGURE 6.58**

Series adaptor with port 3 and 2 as dependent and reflection-free port, respectively.

and

$$B_N = - \sum_{k=1}^{N-1} A_k. \quad (6.75)$$

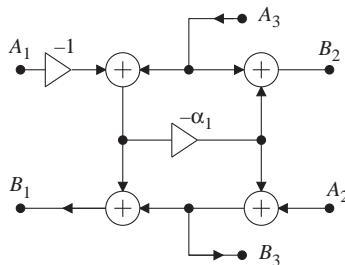
Hence, the reflected wave  $B_N$  is independent of the incident wave  $A_N$  and the delay-free loop is broken. For a parallel adaptor we have

$$G_N = \sum_{k=1}^{N-1} G_k \quad (6.76)$$

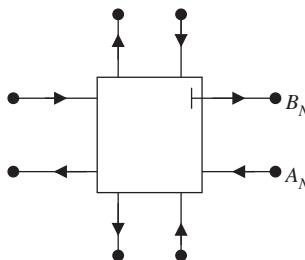
and

$$B_N = \sum_{k=1}^{N-1} \alpha_k A_k. \quad (6.77)$$

In both cases, the expression for  $B_N$  is independent of  $A_N$ , i.e., a direct path from  $A_N$  to  $B_N$  in the wave-flow graph does not exist. Hence, the two ports can be connected without that a delay-free loop occurs. There are no restrictions on the adjacent adaptor. The port with the coefficient equal to unity is called a reflection-free port. Since  $\alpha_N = 1$ , one of the remaining coefficients can be eliminated. Figure 6.58 shows a three-port series adaptor with  $\alpha_3 = 1$  and port 2 as the dependent port.

**FIGURE 6.59**

Parallel adaptor with port 3 as dependent port and port 2 is reflection-free, respectively.

**FIGURE 6.60**

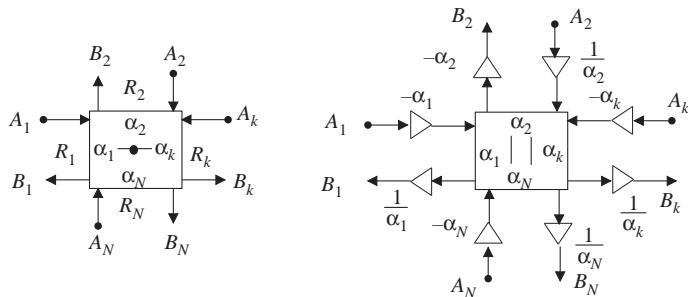
Adaptor with reflection-free port.

Note the non-existent path between A<sub>3</sub> and B<sub>3</sub>. A three-port parallel adaptor with  $\alpha_2 = 1$  and port 3 as dependent port is shown in Figure 6.59. We denote a reflection-free port with the symbol as shown in Figure 6.60.

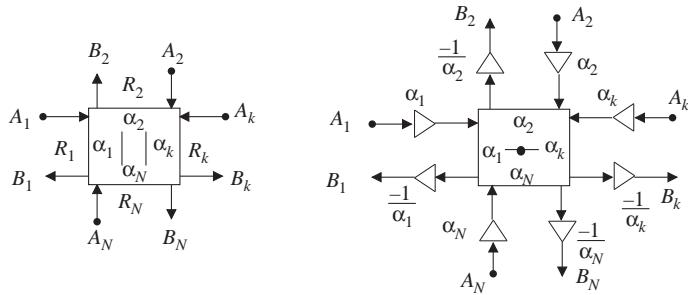
### 1.06.7.5 Adaptor transformations

Adaptor transformations, i.e., replacing series adaptors with parallel adaptors and vice versa, yet with the same adaptor coefficients, can be used to improve the dynamic signal range, lower roundoff noise, and improve the computational properties of the algorithm [53]. Replacing a series adaptor with a parallel adaptor corresponds in the reference filter to replacing an impedance in a series and with a shunt impedance embedded between two gyrators [20]. Thus, a series adaptor is replaced by a parallel adaptor with a pair of inverse multipliers at each port and an inverter at each output terminal as shown in Figure 6.61.

Figure 6.62 shows an N-port parallel adaptor and its equivalent series adaptor. A pair of inverse multipliers can be eliminated, or replaced with any other pair of inverse multipliers, and thereby changing the signal levels in the parallel adaptor network. Note that a pair of inverse multipliers correspond to a transformer in the reference filter. These equivalences transformation changes neither the adaptor coefficients nor the transfer functions, except for a possible change in the gain. Note however, inserting

**FIGURE 6.61**

$N$ -port series adaptor and its equivalent parallel adaptor.

**FIGURE 6.62**

$N$ -port parallel adaptor and its equivalent series adaptor.

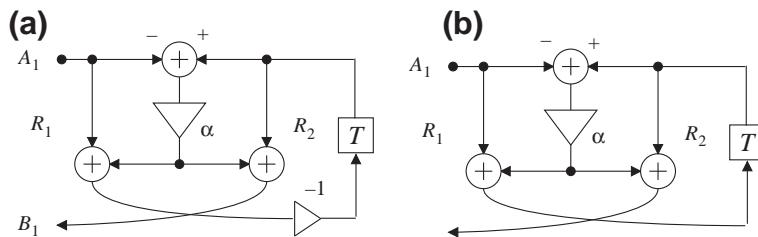
a pair of inverse multipliers requires that the coefficients are of the form  $c = 2^{\pm n}$ , where  $n$  is an integer, since the product  $c(1/c)$ , were both factors are in binary representation, must be equal to 1. A network with  $n$  adaptors, have  $2n$  equivalent realizations, which, however differ with respect to roundoff noise and overflow probability and dynamic signal range.

### 1.06.7.6 Resonance circuits

First- and second-order resonance circuits play an important role as building blocks for frequency selective filters. We are in practice only interested in low order resonance circuits, i.e., first- and second-order circuits. In this section we discuss the design and properties of some resonance circuits and their wave digital counterparts.

#### 1.06.7.6.1 First-order circuits

A first-order wave digital circuit can be derived from a  $\Psi$ -plane capacitor or inductor using series, parallel, or symmetric adaptors. Note that a  $\Psi$ -plane capacitor represents an open-ended unit element

**FIGURE 6.63**

First-order section with (a) inductor, (b) capacitor.

while a  $\Psi$ -plane inductor represents a short-circuited unit element. Figure 6.63 show a first-order circuit corresponding to (a) an inductor and (b) a capacitor realized using the symmetric adaptor.

The reflectance function for the inductor,  $R_2\Psi$ , is

$$S_L(z) = -\frac{\alpha z + 1}{z + \alpha}. \quad (6.78)$$

The reflectance function for the capacitor,  $R_2/\Psi$ , is

$$S_C(z) = \frac{1 - \alpha z}{z - \alpha}, \quad (6.79)$$

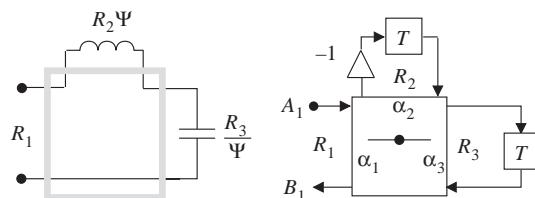
where  $\alpha$  is given by (6.69). Alternatively, first-order circuits corresponding to an inductor or a capacitor can be derived using series or parallel adaptors [14, 27, 53].

#### 1.06.7.6.2 Second-order series resonance circuit

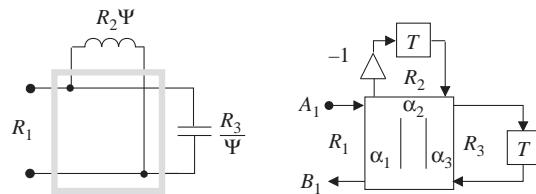
Figure 6.64 shows a second-order series resonance circuit and its wave digital realization using a three-port series adaptor. The reflectance function is

$$S = \frac{B_1}{A_1} = \frac{(1 - \alpha_1)z^2 - (2\alpha_2 + \alpha_1 - 2)z + 1}{z^2 - (2\alpha_2 + \alpha_1 - 2)z + 1 - \alpha_1}. \quad (6.80)$$

Note that the reflection function is an allpass function.

**FIGURE 6.64**

Realization of a series resonance circuit.

**FIGURE 6.65**

Parallel resonance circuit and its WDF counterpart.

### 1.06.7.6.3 Second-order parallel resonance circuit

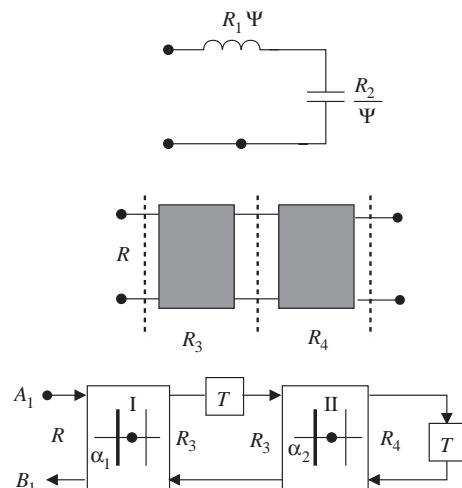
Figure 6.65 shows a parallel resonance circuit and its wave digital realization using a three-port parallel adaptor. The reflectance function is

$$S = -\frac{(1 - \alpha_1)z^2 + (2\alpha_3 + \alpha_1 - 2)z + 1}{z^2 + (2\alpha_3 + \alpha_1 - 2)z + 1 - \alpha_1}. \quad (6.81)$$

The pole density is the same as for the series resonance circuit [14].

### 1.06.7.6.4 Second-order Richards' structures

As discussed in Section 1.06.6.2 a Richards' structure that is either open-circuited or short-circuited at the far end can realize an arbitrary reactance. Figure 6.66 shows second-order Richards' structure

**FIGURE 6.66**

Richards' structure equivalent to second-order series resonance circuit with corresponding wave-flow graph.

and its wave digital counterpart. It can be shown by using Kuroda-Levy identities that this Richards' structure corresponds to a series resonance circuit in the  $\Psi$ -domain [14,27].

The reflectance function is

$$S(z) = \frac{-\alpha_1 z^2 + (\alpha_1 - 1)\alpha_2 z + 1}{z^2 + (\alpha_1 - 1)\alpha_2 z - \alpha_1}. \quad (6.82)$$

A Richard's parallel resonance circuit has the same reflectance function, except that  $\alpha_1$  has changed sign. Hence, the two structures have the same pole density.

### 1.06.7.7 Parasitic oscillations

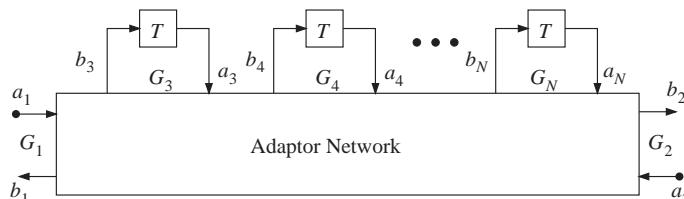
Parasitic oscillations can occur only in recursive structures. Nonrecursive digital filters can have parasitic oscillations only if they are used inside a closed loop. Nonrecursive FIR filters are therefore robust filters structures that do not support any kind of parasitic oscillation. Among the IIR filter structures, wave digital filters, and certain related state-space structures, are of major interest as they can be designed to suppress parasitic oscillations. A. Fettweis has shown that overflow oscillations can be suppressed completely in wave digital filters by placing appropriate restrictions on the overflow characteristic [27,54–57]. To show that a properly designed wave digital filter suppresses any parasitic oscillation we use the concept of pseudo-power, which corresponds to power in analog networks. Note that the pseudo-power concept is defined only for wave digital filters and not for arbitrary filter structures.

The instantaneous pseudo-power entering the adaptor network, shown in Figure 6.67, is defined

$$p(n) = \sum_{k=1}^N G_k (a_k(n)^2 - b_k(n)^2). \quad (6.83)$$

Ideally,  $p(n) = 0$ , since the adaptors are lossless and an arbitrary network of adaptors is also lossless.

Now, in order to suppress a disturbance (error signal component) caused by a nonlinearity, it is sufficient to introduce losses in each port of the network such that the nonlinearities corresponds to pure losses. This can be accomplished by making each term  $a_k(n)^2 - b_k(n)^2 > 0$ , i.e., each recursive loop will be lossy. Note that all recursive loops must contain at least one delay element for a sequentially computable algorithm. Any parasitic oscillation will therefore by necessity appear at one or several of the ports with delay elements. Hence, a parasitic oscillation, which, of course, has finite pseudo-energy



**FIGURE 6.67**

Wave digital filter.

and is not supported from an external signal source will decay to zero since it will dissipate energy in the lossy ports of the adaptor network. Lossy ports can be obtained by quantizing the reflected waves such that their magnitudes are always decreased. Hence, for each nonzero reflected wave we required that

$$\begin{cases} |b(n)_Q| < b(n)_{\text{exact}}, \\ b(n)_Q b(n)_{\text{exact}} \geq 0, \end{cases} \quad (6.84)$$

where the second constraint is to assure that the signals have the same sign.

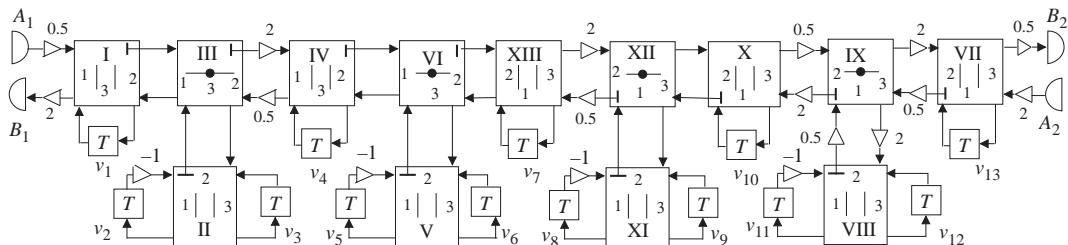
In order to suppress parasitic oscillations, all of the output signals from the adaptors should be sign-magnitude truncated. It can be shown that the sign-magnitude truncation can be substituted by truncation of the signals after the multiplications inside the adaptors and by adding a correcting term at the outputs of the adaptors [14]. This implies considerable hardware simplifications. In practice, it seems to be sufficient that the correcting terms are added only at those ports to which delays are connected

### 1.06.7.8 Ladder wave digital filters

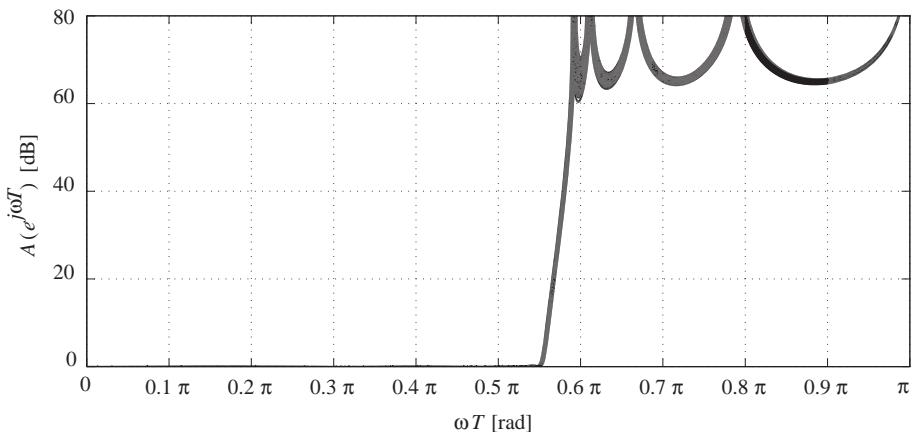
Doubly resistively terminated ladder structures, which are designed for maximal power transfer in the passband, are characterized by their low element sensitivity. A corresponding wave digital filter inherits the sensitivity properties as well as the stability properties of the analog reference filter. The positions of the transmission zeros are also relative insensitive since they are realized by reflections in the resonance circuits. This is true for filters that approximate a constant passband gain. However, it is not true for, for example, a filter that approximate a differentiator.

There are several alternative ladder design techniques:

- Unit elements can be inserted between the branches in the ladder structure in order to avoid delay-free loops [14, 25, 27, 52, 58]. These filters are of less importance.
- Reference filters of Richards' type, i.e., cascaded transmission lines, may be used to realize lowpass and highpass filters and a restricted set of bandpass and stopband filters [14]. Also these filters are of less importance.
- A commonly used class of wave digital filters are derived from a singly resistively terminated Richards' structure [14, 59–62]. The corresponding wave digital filter is used to realize the denominator. The numerator is realized by either injecting the input signal into several nodes in the recursive wave digital filter structure or using linear combination of the signal in several nodes, or a combination thereof. Singly resistively terminated filter has very high element sensitivity [20] and they are not recommended for use as frequency-selective filters. Moreover, these structures suffer from large variations in the internal signal levels and they also generate large roundoff noise. The design of the above wave digital filters is discussed in detail in [14, 17].
- Here we recommend wave digital filters derived from a doubly resistively terminated ladder and we use the technique with reflection-free ports to obtain a wave digital filter without delay-free loop.
- Alternatively we may select a wave digital filter derived from a doubly resistively terminated lattice filter, and use reflection-free ports or a cascade of allpass sections, to obtain a wave digital filter without delay-free loop.

**FIGURE 6.68**

Ninth-order ladder wave digital filter.

**FIGURE 6.69**

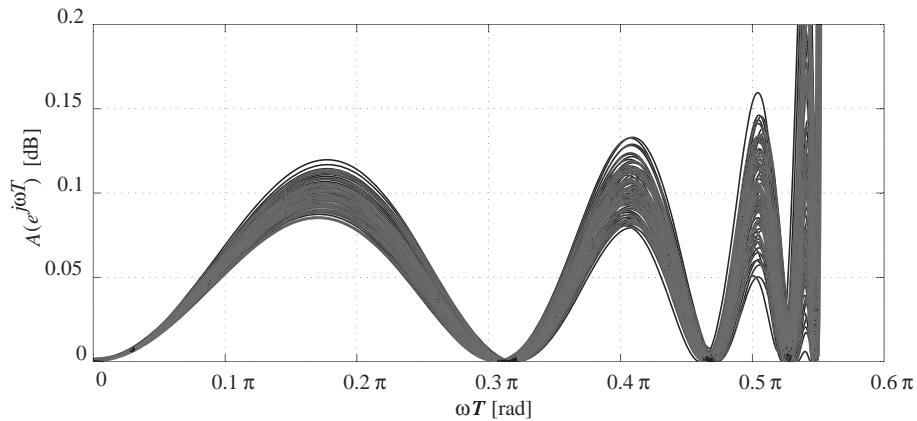
Variation in the attenuation for the ninth-order ladder filter.

#### 1.06.7.8.1 Example 4

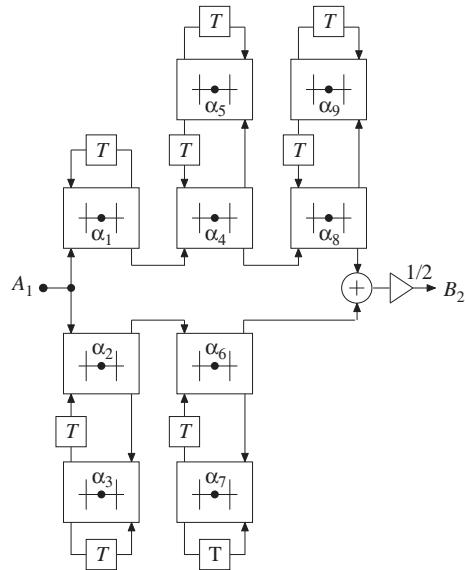
Consider a lowpass filter with the following specification  $A_{\max} = 0.1$  dB,  $A_{\min} = 65$  dB,  $\omega_c T = 0.55\phi$  rad, and  $\omega_s T = 0.6\phi$  rad. Minimum order is  $N_{\min} = 8.499$ , which we increase to  $N = 9$ .

The ladder structure can be scaled by inserting transformers between adjacent adaptors. Each transformer allows one node in an adaptor to be scaled. The adaptors with a reflection-free port have only one critical node, i.e., the input to the multiplier. Hence, all critical nodes can be scaled, except one node in adaptor XIII, which have two critical nodes. The  $L_2$ -norm scaled structure is shown in Figure 6.68. Note that filters should be scaled in order to fairly compare the sensitivity properties (see Figure 6.69).

In order to illustrate and make a crude comparison of the coefficient sensitivity of a  $\pi$ -ladder and lattice filter with cascaded allpass sections, we generate 100 frequency responses where we randomly vary the coefficients in the range  $-2^{-8}$  to  $2^{-8}$  around their nominal values. The overall variation in the attenuation is shown in Figure 6.72.

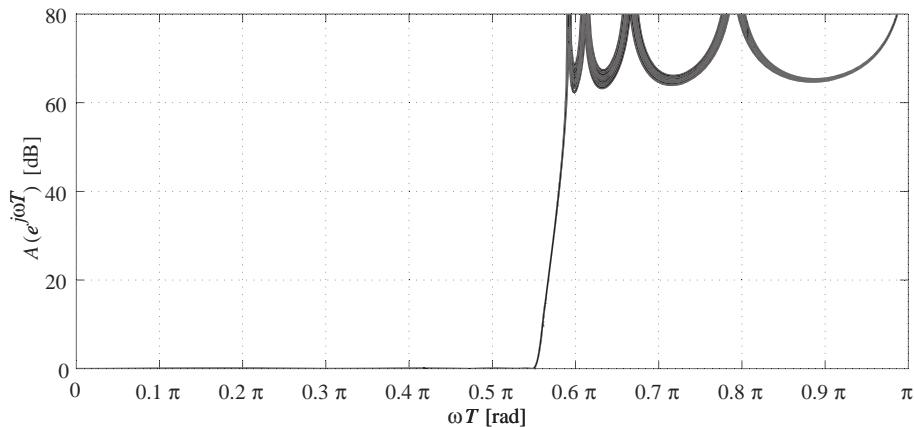
**FIGURE 6.70**

Passband variation in the attenuation for the ninth-order ladder filter.

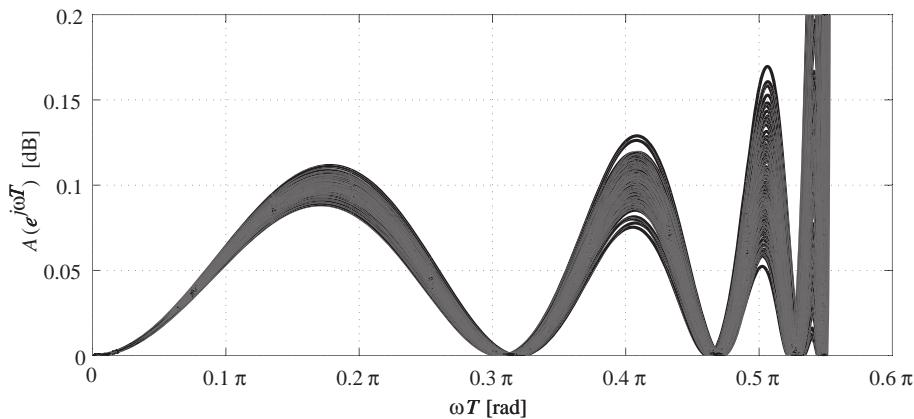
**FIGURE 6.71**

Ninth-order lattice WDF.

The variation is most significant close to the stopband edge. In fact, we will later see that the variation in the positions of the zeros is smaller for the ladder structure compared to a corresponding lattice structure. The variations in the zeros are relatively small, because a zero is determined by only one coefficient. For example, consider the structure, shown in Figure 6.68. Port 2 of adaptor II is

**FIGURE 6.72**

Variation in the attenuation for the ninth-order lattice filter.

**FIGURE 6.73**

Passband variation in the attenuation for the ninth-order lattice filter.

reflection-free and hence  $\alpha_2 = 1$  and therefore we have  $\alpha_1 + \alpha_3 = 1$ . Thus, the transmission zero, which is restricted to the unit circle, is determined by only a single coefficient and the accuracy is determined by the coefficient wordlength. The variation in the attenuation in the passband is shown in Figure 6.73.

The variation increases towards the passband edge, which is in agreement with (6.27). A realization, possibly with somewhat shorter coefficient wordlength, may have been found if the filter instead was synthesized with a diminishing passband ripple. Note that it is most likely that a set of coefficients can be found with shorter wordlength than 9 bit. Typically, the coefficients can be represented using only two or three nonzero bits, i.e., requiring only one or two adders. 14 multipliers are required for the ladder

structure. Wave digital ladder structures appear to yield complicated wave-flow graphs that may make them difficult to implement efficiently. However, a detailed analysis shows that this is not the case [14].

### 1.06.7.9 Lattice wave digital filters

Lattice wave digital filters are derived from an analog bridge structure and have therefore very high stopband sensitivity. Different wave digital lattice structures are obtained depending on how the lattice branches are realized. We may use Forster and Cauer networks or high-order Richards' structures. A realization consisting of a set of interconnected three-port circulators that are loaded with first- and second-order reactances have become popular, mainly due to the fact that they are easy to design and that they consist of simple allpass sections [14, 63–65].

#### 1.06.7.9.1 Example 5

Figure 6.71 shows a ninth-order lattice wave digital filter where the two allpass branches has been realized by a circulator structures [20] that is loaded with first- and second-order sections of the types shown in Figure 6.66 (see Figure 6.70).

In order to get a similar deviation in the stopband attenuation as in Figures 6.72 and 6.73, we restrict the error range of  $-2^{-15}$  to  $2^{-15}$ . That is, to about  $2^7 = 128$  times smaller range. The resulting variation in the attenuation is shown in Figure 6.72. It is evident that the stopband sensitivity is much larger compared to the sensitivity for the corresponding ladder structure and that the precision in the adaptor coefficients is not enough. The passband variation, when the error range is decreased to  $-2^{-8}$  to  $2^{-8}$ , is shown in Figure 6.73.

---

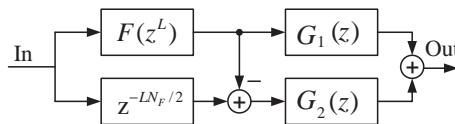
## 1.06.8 Frequency response masking (FRM) structure

For given passband ripple and stopband attenuation, the filter order, as well as the number of non-trivial coefficients, of a direct form FIR filter, is according to (6.8) inversely proportional to the transition width. Therefore, sharp filters suffer from high computational complexity due to the high filter orders. Lim [66] proposed a frequency response masking (FRM) FIR filter structure in 1986 for the design of sharp FIR filters with low computational complexity. The structure was thereafter modified and improved by many researchers to achieve even further computation savings, and has been extended to the design of various types of filters, such as half-band filters, multirate filters, recursive filters, and two-dimensional filters.

### 1.06.8.1 Basic FRM structure

Just as the name implies, FRM uses masking filters to obtain the desired frequency responses. The overall structure of an FRM is a subfilter network, consisting of three subfilters, as shown in Figure 6.74. The three subfilters are a bandedge shaping prototype filter with  $z$ -transform transfer function  $F(z)$  and two masking filters with  $z$ -transform transfer functions  $G_1(z)$  and  $G_2(z)$ , respectively; their corresponding frequency responses are denoted as  $F(e^{j\omega T})$ ,  $G_1(e^{j\omega T})$ , and  $G_2(e^{j\omega T})$ , respectively.

In FRM structure, every delay of the bandedge shaping prototype filter  $F(z)$  is replaced by  $L$  delays. The frequency response of the resulting filter,  $F(z^L)$ , is a frequency compressed (by a factor of  $L$ ) and periodic version of  $F(z)$ , as shown in Figures 6.75 a and b; the transition band of  $F(z)$  is mapped to

**FIGURE 6.74**

Structure of the frequency response masking filter.

$L$  transition bands with transition width shrunk by a factor of  $L$ . The complementary filter of  $F(z^L)$  is obtained by subtracting  $F(z^L)$  from a pure delay term,  $z^{-LN_F/2}$ , where  $N_F$  is the order of  $F(z)$ . (To avoid a half delay,  $N_F$  is chosen to be even.) Thus, the entire frequency region from DC to Nyquist frequency is decomposed into  $L$  bands;  $F(z^L)$  and its complement hold alternate band respectively, as shown in Figure 6.75b. Using two masking filters  $G_1(z)$  and  $G_2(z)$ , whose frequency responses are given in Figure 6.75c, to filter the outputs of  $F(z^L)$  and its complement, the desired frequency bands are kept and then combined to obtain the final frequency response, shown in Figure 6.75d.

Throughout this chapter, for simplicity, the number of delays replacing one delay in the impulse response of the bandedge shaping prototype filter,  $L$ , is referred to as the compression factor of the FRM filter.

The overall  $z$ -transform transfer function of the FRM structure,  $H(z)$ , is given by,

$$H(z) = F(z^L)G_1(z) + \left( z^{-LN_F/2} - F(z^L) \right) G_2(z) \quad (6.85)$$

In a synthesis problem, the passband and stopband edges of the overall filter  $H(z)$ , denoted as  $\omega_c T$  and  $\omega_s T$ , respectively, are given, whereas the passband and stopband edges of the prototype filter  $F(z)$ , denoted as  $\theta$  and  $\varphi$ , have to be determined. Depending on, if the transition band is generated from  $F(z^L)$  or its complement, as shown in Figures 6.75d and f, respectively,  $\omega_c T$  and  $\omega_s T$  are given by

$$\omega_c T = \frac{2l\pi + \theta}{L}, \quad \omega_s T = \frac{2l\pi + \varphi}{L} \quad (6.86)$$

and

$$\omega_c T = \frac{2l\pi - \varphi}{L}, \quad \omega_s T = \frac{2l\pi - \theta}{L} \quad (6.87)$$

respectively. Denoting the two cases as Case A and Case B, for a given  $L$ , it is shown that only one or none of the above two cases results in an  $F(e^{j\omega T})$ , so that  $0 \leq \theta \leq \varphi \leq \pi$  is satisfied.

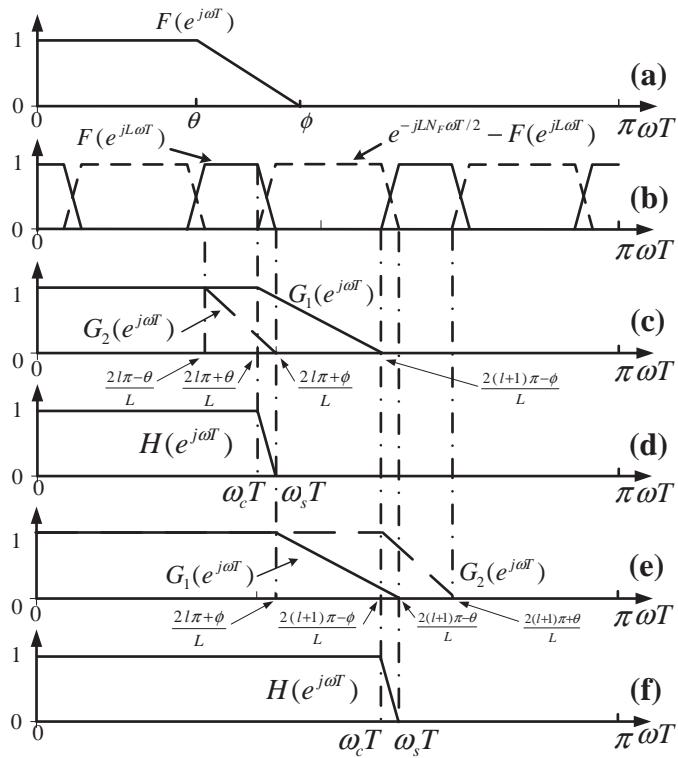
Case A,

$$\theta = \omega_c TL - 2l\pi, \quad \varphi = \omega_s TL - 2l\pi \quad \text{for } l = \lfloor \omega_c TL / 2\pi \rfloor.$$

Case B,

$$\theta = 2l\pi - \omega_s TL, \quad \varphi = 2l\pi - \omega_c TL \quad \text{for } l = \lceil \omega_s TL / 2\pi \rceil,$$

where  $\lfloor x \rfloor$  is the largest integer less than  $x$ , and  $\lceil x \rceil$  is the smallest integer larger than  $x$ . Figures 6.75c and d show a Case A example, whereas Figures 6.75e and f show a Case B example.


**FIGURE 6.75**

Frequency response of FRM subfilters.

Since the transition width of  $F(z)$  is  $L(\omega_s T - \omega_c T)$  for given  $\omega_c T$  and  $\omega_s T$ , and the sum of the transition width of  $G_1(z)$  and  $G_2(z)$  is  $2\phi/L$ , the complexity of  $F(z)$  decreases with increasing  $L$ , while the complexity of  $G_1(z)$  and  $G_2(z)$  increases with increasing  $L$ . The best  $L$  minimizing the arithmetic complexity in terms of the number of multipliers is achieved by

$$L = \frac{1}{\sqrt{2\beta(\omega_s T - \omega_c T)/\pi}}, \quad (6.88)$$

where  $\beta = 1$  if the subfilters are optimized separately [67], and  $\beta = 0.6$  if the subfilters are optimized jointly [68].

The basic FRM structure, utilizing the complementary periodic filter pairs and masking filters, synthesizes sharp filter with low arithmetic complexity. When [67]

$$\omega_s T - \omega_c T < 0.126\pi \text{ rad}$$

the arithmetic complexity saving is achieved by using FRM structure, compared with the direct form FIR structures. In a typical design with  $\omega_s T - \omega_c T < 0.002\pi \text{ rad}$ , the FRM structure reduces the

number of multipliers by a factor of 10, and the price paid for the arithmetic saving is a less than 5% increase in the overall filter order [68].

A special case of FRM filter, named as interpolated FIR (IFIR) filter [69], was earlier developed by Neuvo et al. IFIR consists of the upper branch of the FRM structure, and can therefore only be used to design narrow and wide band sharp FIR filters.

Some modified FRM structures were proposed in [70–72] for further complexity reductions mainly of the masking filters.

### 1.06.8.2 Multistage FRM structure

In the above basic FRM structure, if the order of  $F(z)$  is too high due to a sharp transition width of  $F(z)$ , the FRM technique can be used recursively to form a multistage FRM structure to reduce the complexity.

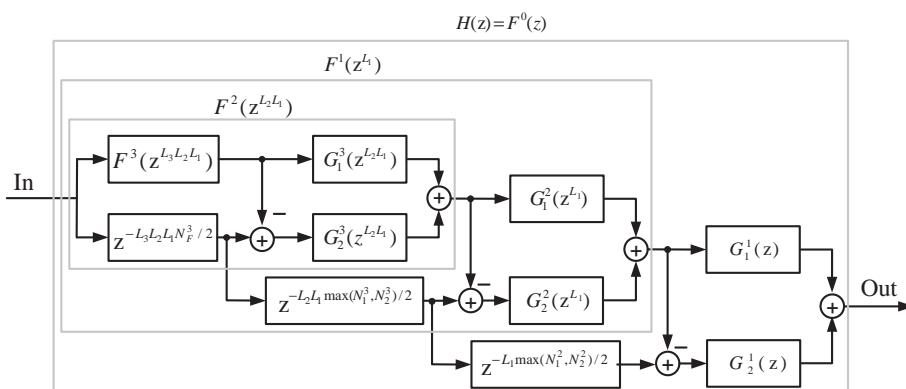
In a  $K$ -stage FRM structure, the  $z$ -transform transfer function of the overall system  $H(z)$  is recursively constructed as

$$H(z) = F^0(z),$$

$$F^{k-1}(z) = F^k(z^{L_k})G_1^k(z) + \left[ z^{-\frac{L_k N_F^k}{2}} - F^k(z^{L_k}) \right] G_2^k(z),$$

for  $k = 1, 2, \dots, K$ , where  $N_F^k$  is the filter order of the  $k$ th stage bandedge shaping prototype filter  $F^k(z)$ ,  $G_1^k(z)$  and  $G_2^k(z)$  are the  $k$ th stage masking filters, with orders of  $N_1^k$  and  $N_2^k$ , respectively. An example of a 3-stage FRM structure is shown in Figure 6.76.

It was discovered theoretically in [67] that the optimum number of stages to achieve the minimum arithmetic complexity is independent of the band ripple requirements, but solely determined by the



**FIGURE 6.76**

A 3-stage FRM structure.

transition width. The optimum value of the number of stage  $K$  satisfies the constraint

$$\beta_b(K) \leq \omega_s T - \omega_c T \leq \beta_b(K - 1), \quad (6.89)$$

where  $\beta_b(K) \cong \frac{1}{4} \left( \frac{K+1}{K+2} \right)^{(K+1)(K+2)}$ . When a filter is constructed in FRM structure with the optimum number of stage  $K_{\text{opt}}$  satisfying the constraint in (6.89), the compression factors of  $L_k$  for  $k = 1, 2, \dots, K$ , tends to be equal, and are given by

$$\left( \frac{K_{\text{opt}} + 1}{K_{\text{opt}}} \right)^{K_{\text{opt}}} \leq L_k \leq \left( \frac{K_{\text{opt}} + 2}{K_{\text{opt}} + 1} \right)^{K_{\text{opt}} + 2}. \quad (6.90)$$

It is interesting to note that both the lower bound and upper bound in (6.90) approaches  $e(\cong 2.7$ , the base of the nature logarithm) when  $K$  is large.

Practical design experience showed that for a given filter transition width and a given stage  $K$ , the minimum arithmetic complexity is achieved when [68]

$$L_k = \left\lceil \frac{1}{\sqrt[(K+1)]{2\beta(\omega_s T - \omega_c T)/\pi}} \right\rceil, \quad (6.91)$$

for  $k = 1, 2, \dots, K$ , where  $\beta = 1$  if the subfilters are optimized separately [67], and  $\beta = 0.6$  if the subfilters are optimized jointly [68].

Considering both the theoretic analysis in [67] and the practical design experience in [68], the overall arithmetic complexity of a given filter specification is in proximity to the minimum if  $K$  in (6.91) is chosen such that  $L_k$  is 2 or 3.

### 1.06.8.3 FRM structure for half-band filter and Hilbert transformer

The synthesis of half-band filters using FRM has to ensure, that the alternate coefficient values are trivial except, that the centre coefficient has a value of 0.5 [73].

Consider a zero phase half-band bandedge shaping prototype filter of order  $N_F = 4N - 2$ , and transfer function  $F(z)$  given by

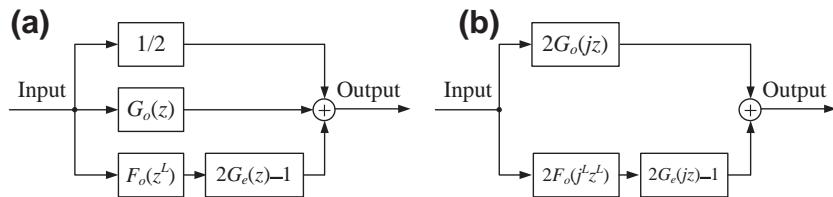
$$F(z) = 1 + F_0(z), \quad (6.92)$$

where  $F_0(z)$  consists of only odd powers of  $z$ . A half-band FRM filter  $H(z)$  can be constructed from this prototype filter by

$$H(z) = \left[ 1 + \frac{1}{2} F_0(z^L) \right] G_1(z) + \left[ 1 - \frac{1}{2} F_0(z^L) \right] [1 - G_1(-z)] \quad (6.93)$$

for odd  $L$ , where  $G_1(z)$  is an even order lowpass masking filter. Let  $G_o(z)$  consist of odd power terms of  $z$ , and  $G_e(z)$  consist of even power terms of  $z$ , of the masking filter  $G_1(z)$ , respectively. Hence  $G_1(z) = G_o(z) + G_e(z)$ , and we have

$$H(z) = \frac{1}{2} + G_o(z) + F_0(z^L)[2G_e(z) - 1]. \quad (6.94)$$

**FIGURE 6.77**

Structure for synthesizing (a) half-band filter and (b) Hilbert transformer.

Thus,  $H(z)$  consisting of a constant term and terms with odd power of  $z$ , is a half-band filter. The filter structure is shown in Figure 6.77a.

A Hilbert transformer may be derived from a unity half-band filter by subtracting the constant  $1/2$  from its transfer function, modulating the remaining coefficients by  $e^{j\pi/2}$ , and then scaling the response by 2. Thus, the transfer function of the Hilbert transformer  $H_H(z)$  can be written from (6.94) to [74]

$$H_H(z) = 2G_o(jz) + 2F_o(j^L z^L)[2G_e(jz) - 1] \quad (6.95)$$

The structure of Hilbert transformer is shown in Figure 6.77 (b).

#### 1.06.8.4 FRM structure for multirate filter and filter bank

##### 1.06.8.4.1 Decimator and interpolator

In polyphase implementation of multirate filters, such as interpolators and decimators with a sampling rate change factor of  $M$ , that are realized as a cascade of two filters, generally only one of the two filters may achieve the “factor of  $M$ ” reduction in computational complexity, unless the output of the first filter is nonzero only at intervals of  $M$ . In the original FRM structure shown in Figure 6.74, if  $L$  is an integer multiple of  $M$ , the outputs of  $F(z^L)$  and its complementary filter can be nonzero only at intervals of  $M$ . However, to synthesize a filter with transition band centered at  $\pi/M$  as in the interpolator and decimator, the original FRM has constraints on the selection of the compression factor  $L$  that neither  $L\omega_c T/\pi$  nor  $L\omega_s T/\pi$  is an integer according to (6.86) and (6.87), and  $\lfloor L\omega_c T/\pi \rfloor = \lfloor L\omega_s T/\pi \rfloor$ . Therefore, due to these two constraints,  $L$  cannot be selected as an integer multiple of  $M$  to achieve a “factor of  $M$  reduction” in computational complexity in multirate filters.

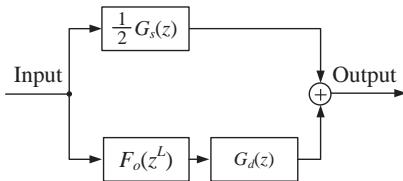
A variant of the FRM structure proposed in [74] achieves the “factor of  $M$  reduction” in a multirate filter with sampling rate change factor  $M$ . The following is an illustration of an interpolator with an upsampling rate  $M$ . The same technique applies to decimators as well. The variant of the FRM is constructed from

$$H(z) = \frac{1}{2}G_s(z) + F_o(z^L)G_d(z), \quad (6.96)$$

where

$$G_s(z) = G_1(z) + G_2(z),$$

$$G_d(z) = G_1(z) - G_2(z),$$


**FIGURE 6.78**

FRM structure for the synthesis of decimation and interpolation filters.

and  $F_o(z)$  is given in the condition of (6.92) for a half-band prototype bandedge shaping filter  $F(z)$ ;  $G_1(z)$  and  $G_2(z)$  are the masking filters as that in Figure 6.74. The factor  $M$  interpolation or decimation filter, constructed in such a manner, has a transition band centered at  $\pi/M$ . The filter structure is shown in Figure 6.78.

For  $M$  even,  $2L$  has to be an odd multiple of  $M$ , while for  $M$  odd,  $2L$  has to be an even multiple of  $M$ . In addition, when  $M$  is odd,  $F(z)$  is replaced by  $F(ze^{-j\frac{\pi}{2L}})$ , and  $G_1(z)$  and  $G_2(z)$  are replaced by  $G_1(ze^{-j\frac{2\theta}{L}})$  and  $G_2(ze^{j\frac{2\theta}{L}})$ , respectively, where  $G(z)$  is a prototype masking filter specified by passband and stopband edges of  $\theta_G = \frac{(2L-M)\pi}{2LM}$  and  $\varphi_G = \frac{(2L+M)\pi}{2LM}$ , respectively.

In implementation, both  $G_s(z)$  and  $F_o(z^L)$  can be decomposed into  $M$  polyphase components efficiently using polyphase structures since the inputs are nonzero only at intervals of  $M$ . For  $G_d(z)$ , besides the fact that the output of  $F_o(z^L)$  is nonzero at intervals of  $M$  samples, only the first polyphase component of  $F_o(z^L)$  has nonzero coefficients, because it is a function of  $z^{-2L}$  and  $2L$  is an integer multiple of  $M$ . Therefore,  $G_d(z)$  may be decomposed into  $M$  polyphase components efficiently as well.

In [75], two alternative FRM structures are proposed for the design of interpolation and decimation by a factor of  $M$ , where the transition band is not restricted to include  $\pi/M$ . In this approach, additional constraints are imposed to the masking filters, such that

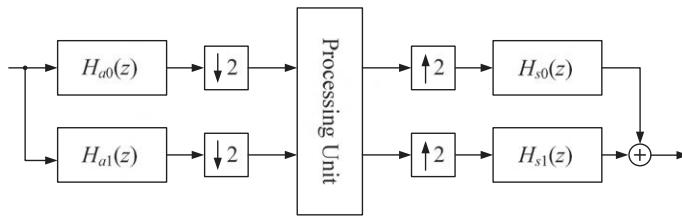
$$G_2(z) = G_1(z) - \frac{M}{M-1}G_{10}(z^M) + \frac{1}{M-1} \quad (6.97)$$

or

$$G_1(z) = G_2(z) - \frac{M}{M-1}G_{20}(z^M) + \frac{1}{M-1}, \quad (6.98)$$

where  $G_{10}(z)$  and  $G_{20}(z)$  are the 0-th polyphase components in the polyphase representations of  $G_1(z)$  and  $G_2(z)$ , respectively. The relations in (6.97) and (6.98) specify the class I and class II FRM structures. These relations impose further constraints on the selection of  $l$  and  $L$  in (6.86) and (6.87). However, the available choices are sufficient to obtain arbitrary frequency selectivity in the filter design and meanwhile achieve complexity reduction.

When the masking filters are related to each other as those shown in (6.97) and (6.98), both the bandedge shaping filter and the masking filters can be decomposed into polyphase structure, such that all filters are working at the lower sampling rate involved in the interpolation or decimation.

**FIGURE 6.79**

Two-channel maximally decimated filter bank.

It is interesting to note that both (6.97) and (6.98) are reduced to  $G_2(z) = 1 - G_1(z)$  when  $M = 2$ ; this relation is the same as that in the half-band FRM structure given in (6.93). In fact, when the transition band is restricted to include  $\pi/M$ , the proposed approach is a generalization of that in [73] and an  $M$ th band filters are obtained.

Besides the above two FRM filter structures, a factor of two interpolation and decimation filters are developed in [76], based on a recursive filter structure introduced in next section.

#### 1.06.8.4.2 Filter banks

A two-channel maximally decimated filter bank utilizing the FRM structure is proposed in [77]. In the diagram of a two-channel filter bank shown in Figure 6.79, each analysis and synthesis filter of the proposed technique is realized through a modified FRM structure with the same bandedge shaping prototype filter. The transfer functions of the analysis and synthesis filters are given by

$$\begin{aligned} H_{ak}(z) &= F(z^L)G_{k1}(z) + F_c(z^L)G_{k2}(z), \\ H_{sk}(z) &= F(z^L)G_{k1}(z) - F_c(z^L)G_{k2}(z), \end{aligned}$$

for  $k = 0, 1$ .

In this FRM structure, the original magnitude complementary filter of the bandedge shaping filter,  $1 - F(z)$ , is modified to an approximately power complementary filter, i.e.,  $F_c(z)$  is related to  $F(z)$  in a form of

$$F_c(z) = F(-z)$$

and the filter order is constrained to odd order. In addition, the compression factor  $L$  is selected to be odd. By relating the masking filters in the lower branch,  $G_{11}(z)$  and  $G_{12}(z)$ , to that in the upper branch,  $G_{01}(z)$  and  $G_{02}(z)$ , respectively as

$$G_{11}(z) = -G_{02}(-z)$$

and

$$G_{12}(z) = G_{01}(-z)$$

an aliasing free filter bank is constructed.

This technique is further extended to a modulated  $M$ -channel filter banks by using the analysis and synthesis filters in the upper branch of the two-channel filter bank as the prototype filters for

modulation [78]. Together with a cosine modulation block and a sine modulation block, a near perfect reconstructed  $M$ -channel maximally decimated filter bank is developed.

Other FRM structures for filter banks are mainly developed for two-channel filter banks, including non-uniform filter banks with rational sampling factor [79] and multi-plet perfect reconstruction filter banks [80]. In addition, a non-multirate fast filter bank (FFB) is developed in [81], to realize the function of sliding FFT filter bank with low computational complexity, but achieving high frequency selectivity and passband performance.

#### 1.06.8.4.3 Cosine-modulated transmultiplexer

Due to the same constraints on the selection of the compression factor  $L$ , the prototype bandedge shaping filter for an  $M$  band transmultiplexer (TMUX) is not realizable if  $L$  is an integer multiple of  $M$ . Unlike the variant structure proposed in [74], in [82], an implementation of an  $M$  band cosine-modulated TMUX (CMT) for

$$L = 2K_a M + \frac{M}{K_b}, \quad (6.99)$$

where  $K_a$  is a non-negative integer and  $K_b$  is a positive integer, is proposed.

If only the upper branch in the original FRM structure is used as the prototype filter for the CMT, the transfer function for the analysis filter becomes

$$H_m(z) = \sum_{n=0}^N c_{m,n} \left( f^L * g_1 \right) (n) z^{-n}, \quad (6.100)$$

where  $c_{m,n} = 2 \cos \left[ \frac{(2m+1)(n-\frac{N}{2})}{2M} + (-1)^m \frac{\pi}{4} \right]$  for  $m = 0, 1, \dots, M-1$  and  $n = 0, 1, \dots, N-1$ ,

and the term  $(f^L * g_1)(n)$  denotes the convolution between the impulse responses of  $F(z^L)$  and  $G_1(z)$ . Thus,  $F(z^L)$  can be decomposed to  $Q = 2K_b$  polyphase components assuming that  $N_F = QK_c - 1$  with  $K_c$  being a positive integer, and  $G_1(z)$  can be decomposed to  $2M$  polyphase components assuming that  $N_1 = 2K_d M - 1$  with  $K_d$  being a positive integer. Under the relation of  $L$  and  $M$  in (6.99), and  $c_{m,(n+2kM)} = (-1)^k c_{m,n}$ , the overall polyphase representation of  $H_m(z)$  is given by

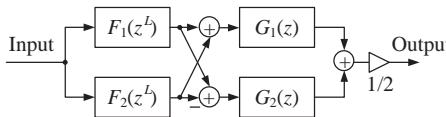
$$H_m(z) = \sum_{q=0}^{Q-1} \left[ z^{-Lq} E_F^q(-z^{LQ}) \times \sum_{p=0}^{2M-1} c_{m,\left(n+\frac{M}{K_b}q\right)} z^{-p} E_G^p(-z^{2M}) \right],$$

where  $E_F^q$  is the  $q$ th polyphase component of the prototype bandedge shaping filter  $F(z)$ , given by

$$E_F^q(z) = \sum_{k=0}^{K_c} (-1)^{K_a q} f(kQ + q) z^{-k}$$

and  $E_G^p(z)$  is the  $p$ th polyphase component of the masking filter  $G_1(z)$  given by

$$E_G^p(z) = \sum_{k=0}^{K_d-1} g_1(2kM + p) z^{-k}.$$

**FIGURE 6.80**

The overall structure of the recursive filter.

If both branches in the FRM design are used, this decomposition can be applied to the lower branch, and by enforcing both masking filters to have the same order  $N_1 = N_2$ , the responses of the two branches can be added together just before the modulation stage.

#### 1.06.8.5 FRM structure for recursive filter

While FRM structures were originally and mainly developed for the design of sharp filters with low computational complexity, a hybrid IIR and FIR filter structure was proposed for the design of high speed-recursive digital filters [83].

In the proposed structures, the bandedge shaping prototype filters are power complementary IIR filters, denoted as  $F(z)$  and  $F_c(z)$ , realized as a parallel connection of two allpass filters, and the masking filters are linear-phase FIR filters. The bandedge shaping prototype filters are expressed as

$$F(z) = \frac{F_1(z) + F_2(z)}{2}$$

and

$$F_c(z) = \frac{F_1(z) - F_2(z)}{2},$$

where  $F_1(z)$  and  $F_2(z)$  are stable allpass filters. When the magnitude responses of the power complementary IIR filter pair  $F(z)$  and  $F_c(z)$  are bounded by unity, the overall transfer function is expressible as

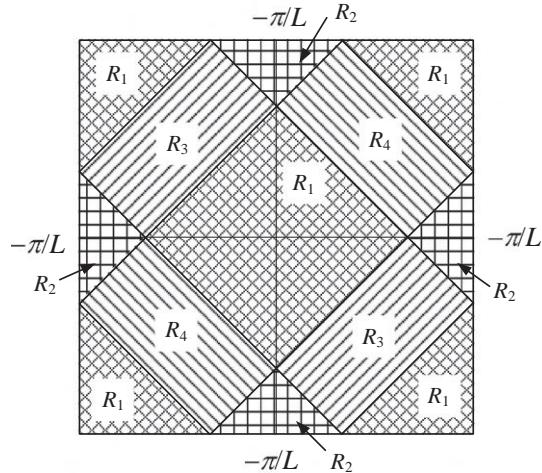
$$H(z) = \frac{F_1(z^L) + F_2(z^L)}{2} G_1(z) + \frac{F_1(z^L) - F_2(z^L)}{2} G_2(z),$$

where  $G_1(z)$  and  $G_2(z)$  are the linear-phase FIR masking filters. The structure of the overall filter is shown in Figure 6.80.

Since the maximum sample frequency of recursive filters is determined by the ratio of the number of delay elements and the operational latency in the critical loop of the filter realization, see (6.102), this structure introduced at least  $L$  delay elements in its critical loop, resulting in an  $L$ -fold increase of the maximal sample frequency.

#### 1.06.8.6 Two-dimensional FRM structure

The extension of the FRM structure to two-dimensional (2D) sharp filters is not straightforward. The main difficulty lies in the generation of complementary bandedge shaping filters, whose frequency responses of the compressed passbands should be separated by stopbands and able to be masked by the subsequent masking filters.

**FIGURE 6.81**

The four frequency regions for the complementary components of the 2D FRM filter.

A 2D diamond-shaped filter using FRM structure was proposed in [84]. The idea is to divide the frequency spectrum into four suitably chosen complementary components  $R_k$  for  $k = 1, 2, 3$ , and 4, as shown in Figure 6.81, in the 2D region  $[-\frac{\pi}{L}, \frac{\pi}{L}]^2$ , for a compression factor of  $L$ . These regions are also defined for the periodic repetitions of the 2D region  $[-\frac{\pi}{L}, \frac{\pi}{L}]^2$ .

These four complementary regions are obtained by four filters  $\hat{F}_k(z_1, z_2)$  for  $k = 1, 2, 3$ , and 4. To make the four filters complementary, an error filter, denoted as  $F_e(z_1, z_2)$ , is introduced, such that

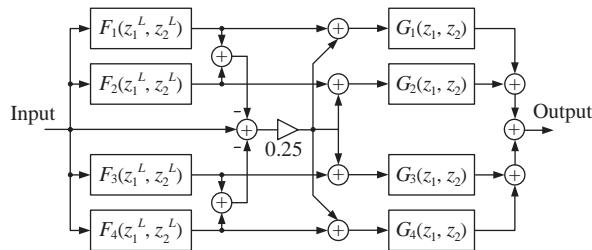
$$\sum_{k=1}^4 \hat{F}_k(z_1, z_2) + F_e(z_1, z_2) = 1. \quad (6.101)$$

Thus, the four bandedge shaping prototype filters, denoted as  $F_k(z_1, z_2)$ , are given by

$$F_k(z_1, z_2) = \hat{F}_k(z_1, z_2) + \frac{F_e(z_1, z_2)}{4}.$$

Each delay, in each dimension of the four bandedge shaping prototype filters, is replaced by  $L$  delays and four complementary bandedge shaping filters  $F_k(z_1^L, z_2^L)$  are obtained. The output of  $F_k(z_1^L, z_2^L)$  are then filtered by respective masking filters  $G_k(z_1, z_2)$ , and the resulting outputs are summed to form the final output. The generation of the complementary bandedge shaping filter and the overall 2D diamond-shaped FRM filter structure is shown in Figure 6.82.

Thus, based on the same principle of one-dimensional FRM filters, the specifications of the bandedge shaping prototype filters and masking filters can be derived for the design procedure.

**FIGURE 6.82**

The overall filter structure of the 2D diamond-shaped FRM filter.

### 1.06.8.7 Summary

As one of the most successful low computational complexity digital filter structures, basic FRM filter consists of an elegant filter network with two branches comprising three subfilters and realizes frequency responses owning sharp transition band. The price paid, for the high computational saving, is a small increase in the filter order. Further developments on the basic FRM structure have extended the technique to various areas of digital filters, including but not limited to recursive filters, multirate filters, filter banks, and 2D filters.

---

## 1.06.9 Computational properties of filter algorithms

In order to compare and evaluate different filter structures we need to discuss a few basic properties of algorithms and their implementation.

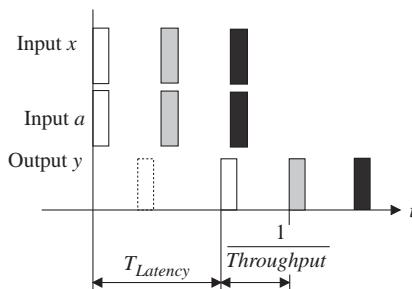
### 1.06.9.1 Latency and throughput

We define latency, as the time it takes to generate an output value from the corresponding input value, as illustrated in Figure 6.83, for an arithmetic operation. The throughput (samples/s) is defined as the reciprocal of the time between successive outputs. In sequential algorithms it is possible to increase the throughput of an algorithm—for example, by using pipelining [1].

### 1.06.9.2 Maximal sample rate

The maximal sample rate of an algorithm is determined only by its recursive loops [1, 85]. Nonrecursive parts of the signal-flow graph, e.g., input and output branches, generally do not limit the sample rate, but to achieve this limit, additional delay elements may have to be introduced into the nonrecursive branches [1]. The maximal sample frequency for an iterative recursive algorithm, described by a fully-specified signal-flow graph [1], is

$$f_{\max} = \min_i \left\{ \frac{N_i}{T_{\text{op}}} \right\}, \quad (6.102)$$

**FIGURE 6.83**

Latency and throughput.

where  $T_{op}$  is the total latency due to the arithmetic operations, and  $N_i$  is the number of delay elements in the directed loop  $i$ . We will later define the latency for different arithmetic operations. The loop with the lowest  $f_{max}$  is called the critical loop. The minimum sample period is also referred to as the iteration period bound. It is, of course, not possible to improve the iteration period bound for a given algorithm. However, a new algorithm with a higher bound can often be derived from the original algorithm using algorithm transformations [1, 86, 87]. We recognize from (6.102) that there are several possibilities to improve the bound:

- Reduce the operation latency,  $T_{op}$ , in the critical loop.
- Introduce additional delay elements into the loop, without changing the behavior.
- Remove superfluous arithmetic or logic operations from the loop.

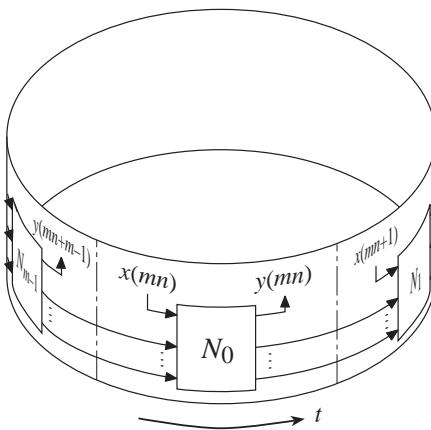
### 1.06.9.3 Cyclic scheduling

In order to achieve a maximally fast implementation that achieves the bound in (6.102), we must generally connect  $m$  computation graphs, that represents the operations within single sample interval [1], as illustrated in Figure 6.84.

This will result in a periodic scheduling formulation that allows a schedule period, that is  $m$  times longer than the sample interval. This cyclic formulation will under certain conditions result in a maximally fast, periodic schedule. Unfortunately, it is not possible to determine the best choice of  $m$  in the general case. In order to attain the minimum sample period, it is necessary to perform cyclic scheduling of the operations belonging to several successive sample intervals if

- The latency for a PE is longer than  $T_{min}$  or
- The critical loop(s) contains more than one delay element.

Generally, the critical loop should be at least as long as the longest latency for any of the PEs in the loop. Bit-serial and digit-serial operations with latencies longer than the minimal sample period can thereby

**FIGURE 6.84**

Cyclic scheduling.

be completed within the scheduling period. The minimum number of sample periods for the schedule is

$$m = \left\lceil \frac{\max_i (T_{\text{latency},i})}{T_{\min}} \right\rceil, \quad (6.103)$$

where  $T_{\text{latency},i}$  is the latency of operation  $i$ . From the cyclic scheduling formulation it is possible to find a resource-optimal schedule with a minimum number of concurrent operations.

### 1.06.10 Architecture

A major problem when implementing an algorithm is to decide upon a suitable hardware architecture. We may select to use one or several processing elements (PE) and one or several memories, or memory ports, and among a variety of communication networks between PEs and the memories [1]. Typically the design target is to minimize the implementation cost, e.g., chip area and power consumption, while meeting the throughput requirement. Normally, in digital signal processing applications there is no benefit in having a higher throughput than required.

We recommend that the arithmetic operations are scheduled cyclically to attain a maximally fast implementation as illustrated in Figure 6.84. A maximally fast schedule and the corresponding implementation is obtained by using an isomorphic mapping between the arithmetic operations and the processing elements [88–90]. This allows the hardware to operate with a low power supply voltage and low power consumption. Moreover, the communication network becomes simple and static. There exist graphic based methods to determine the number of independent memories or memory ports that are required [1]. Recently, it has been shown that a maximally fast implementation is obtainable using a numerically equivalent state-space representation and distributed arithmetic [91].

## 1.06.11 Arithmetic operations

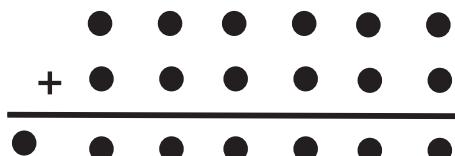
Here we are mainly interested in the arithmetic operations: addition, subtraction, and multiplications as well as sum-of-products. There exist many alternative arithmetic algorithms to perform these operations. Consider, for example, addition. Addition of two numbers, independently of their sign, can be performed using the same circuit, if two's-complement representation is used. Moreover changing the circuit for addition into a subtractor requires only that the bits of one of the inputs are inverted. An advantage of the two's-complement number representation is that several numbers can be added, with possible intermediate overflows, if we can guarantee that the result is within the correct number range. This situation occurs frequently in digital filter algorithms. A drawback is, however, that the carry propagation limits the latency. Several techniques to speed-up the computation of the carries as well as carry-free number representations have been proposed [3].

### 1.06.11.1 Addition and subtraction

The operation of adding two or more numbers is in many ways the most fundamental arithmetic operation, since most other operations in one way or another, are based on addition. The operands of concern here are either two's-complement or unsigned representation. Most DSP applications use fractional arithmetic instead of integer arithmetic [1]. The sum of two  $W$ -bit numbers is a  $(W+1)$ -bit number while the product of two binary  $W$ -bit numbers is a  $2W$ -bit number. In many cases, and always in recursive algorithms, the resulting number needs to be quantized to a  $W$ -bit number. Hence, the question is which bits of the result are to be retained. In fractional arithmetic, two's-complement numbers are interpreted as being in the range  $[-1, 1]$ , i.e.,

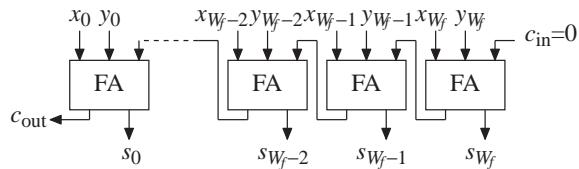
$$x = -x_0 + \sum_{i=1}^{W-1} x_i 2^{-i}. \quad (6.104)$$

Hence, the most significant part of a product is contained in the bits  $(x_0, x_1, \dots)$ . We use the graphic representation shown in Figure 6.85 to represent the operands and the sum bits with the most significant bit to the left.



**FIGURE 6.85**

Illustration of addition of two binary numbers.

**FIGURE 6.86**

Ripple-carry adder.

#### 1.06.11.1.1 Ripple-carry addition

A straightforward way to add two numbers is to sequentially add the two bits of the same significance and the carry from the previous stage using a full adder (FA) and propagate the carry bit to the next stage. This is called ripple-carry addition and is illustrated in Figure 6.86. Note that the operation propagate from right to the left and that only one FA perform have correct inputs at a give moment. Hence, the circuit performs addition sequentially, one bit at a time, even though it is referred to as a parallel adder. This type of adder can add both unsigned and two's-complement numbers.

The major drawback with the ripple-carry adder is that the worst-case delay is proportional to the wordlength. Also, typically the ripple-carry adder will produce many glitches since the full adders perform switching operations without having correct carries. This situation is improved if the delay for the carry bit is smaller than that of the sum bit [92].

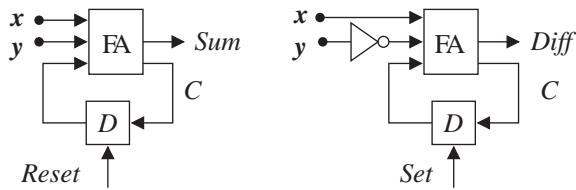
However, due to the simple design the energy per computation is still reasonable small [93]. Alternatively all pairs of two bits of the same significance can be added simultaneously and then the carries are added using some efficient scheme. There are many adder schemes proposed, for more details we refer to e.g., [93]. For carry-save addition we refer to [94, 95]. It is also possible to perform addition in constant time using redundant number systems such as signed-digit or carry-save representations. An alternative is to use residue number systems (RNS), which breaks the carry-chain into several shorter ones [96].

#### 1.06.11.1.2 Bit-serial addition and subtraction

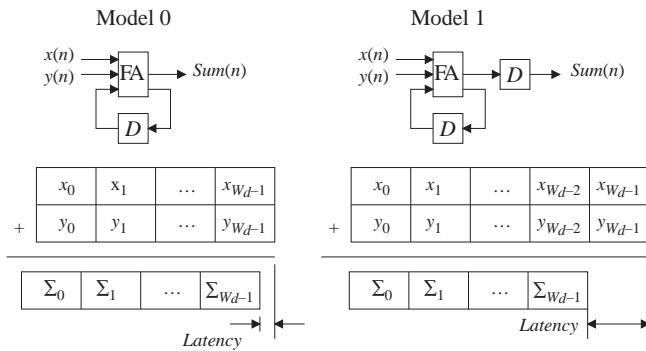
In bit-serial arithmetic the inputs  $x$  and  $y$  are normally processed with the least-significant bit first. Bit-serial numbers in two's-complement representation can be added or subtracted with the circuits shown in Figure 6.87.

Since the carries are saved from one bit position to the next, the circuits are called carry-save adder and carry-save subtractor, respectively. At the start of the addition the D flip-flop is reset (set) for the adder (subtractor), respectively.

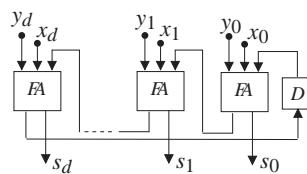
Figure 6.90 shows two latency models for bit-serial adders. The left most is suitable for static CMOS and the right most for dynamic (clocked) logic styles. The time to add two  $W$ -bit numbers is  $W$  or  $W + 1$  clock cycles. Note that the first bit of the sum, which is delayed by the latency given in Figure 6.90, can directly be feed to any subsequent bit-serial processing element (PE). Thus, the throughput of a complex computation may not necessarily be determined by the long addition time [1] (see Figure 6.88).

**FIGURE 6.87**

Bit-serial adder and subtracter.

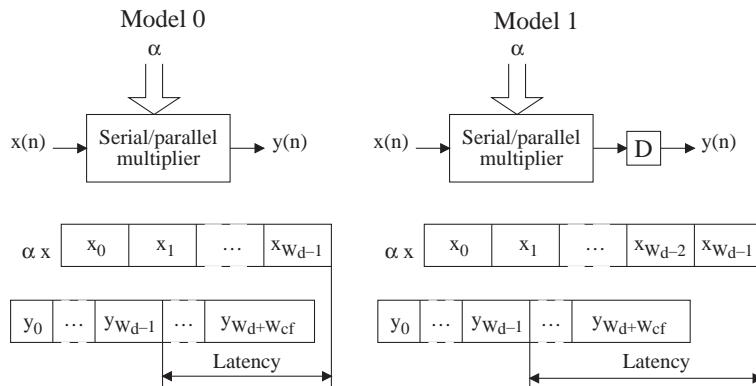
**FIGURE 6.88**

Latency models for a bit-serial adder.

**FIGURE 6.89**

Digit-serial adder with digit-size  $d$ .

Comparing the ripple-carry adder and the bit-serial adder, we note that both operate sequentially with one bit addition at a time. In the ripple-carry adder, each bit addition is mapped to its private FA, while in the bit-serial adder, all bit additions are mapped to a time-multiplexed FA. Hence, the difference in throughput is mainly due to the time lost in the D flip-flop. The chip area for bit-serial circuits is small and consequently the wiring is also small.

**FIGURE 6.90**

Latency models for a serial/parallel multiplier.

#### 1.06.11.1.3 Digit-serial addition and subtraction

From speed and power consumption points of view it may sometimes be advantageous to process several bits at a time, so-called digit-serial processing [97–103]. The number of bits processed in a clock cycle is referred to as the digit size. Figure 6.89 shows a  $d$ -bit digital-serial adder.

The flip-flop transmit the carry between successive digits. In case of subtraction of a two's-complement number, the negative value is instead added by inverting the bits and setting the carry flip-flop. Most of the principles for bit-serial arithmetic can easily be extended to digit-serial arithmetic. An advantage of bit-serial and digit-serial arithmetic is that less chip area is required and therefore the equivalent switched capacitance and leakage current is low [104]. The difference in throughput between a conventional word level and a digit-serial implementation is not great [105]. Both bit-serial and digit-serial are suitable for implementations in FPGA circuits.

#### 1.06.11.2 Multiplication

Bit-parallel multiplication of two numbers can be divided into two main steps:

- partial product generation that determines the bits to be added, and
- accumulation of the partial products.

Two main approaches to perform the accumulation have been proposed, namely array and tree type accumulation [3]. Bit-serial and digit-serial multiplications are usually based on a shift-and-add approach [1, 106]. Here, however, we will focus on implementation techniques that use addition and subtractions as the generic arithmetic operations.

To identify the critical loop in the filter, the total latency of the operations inside the loops must be determined. The total latency of several sequential arithmetic operations depend on how they are realized at the logic level, because the clock frequency is determined by the longest propagation path between any two registers. The propagation paths do not only depend on the internal circuitry in an

arithmetic unit; they also depend on the paths between units. Hence, it may be efficient to use pipelining inside the critical loop in order to avoid long propagation paths between arithmetic units.

Two latency models for a serial/parallel multiplier are shown in Figure 6.90. Denoting the number of fractional bits of the coefficient by  $W_{cf}$ , the latencies are  $W_{cf}$  for latency model 0 and  $W_{cf} + 1$  for latency model 1. The corresponding latency models for digit-serial arithmetic are defined in the same manner.

In model 0, which corresponds to a static CMOS logic style without pipelining of the gates, the latency is equal to the gate delay of a full adder. In model 1, which corresponds to a dynamic CMOS logic style, or a static CMOS logic style with pipelining at the gate level, the full adder followed by a D flip-flop making the latency equal to one clock cycle. Model 1 generally results in faster bit-serial implementations, due to the shorter logic paths between the flip-flops in successive operations.

### 1.06.11.2.1 Example 6

Consider a lattice wave digital filter. The critical loop in a first-order allpass section is indicated in Figure 6.91 [89, 90].

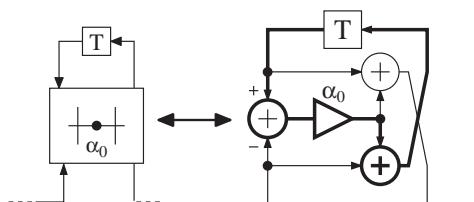
This loop containing two additions, one multiplication with  $\alpha_0$ , and one delay element has the minimal sample period given by

$$T_{\min 1} = \frac{T_{\alpha_0} + 2T_{\text{add}}}{1}. \quad (6.105)$$

In Figure 6.92 the critical loop in a second-order allpass section is indicated. For this loop consisting of four additions, two multiplications, and one delay element, the corresponding minimal sample period becomes

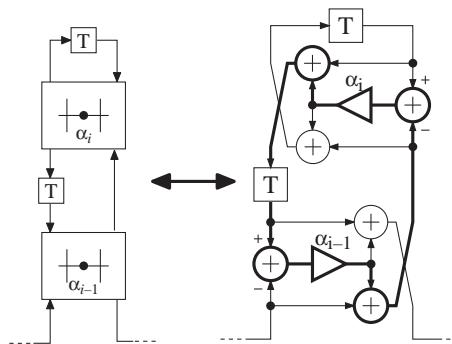
$$T_{\min 2} = \frac{T_{\alpha_i} + T_{\alpha_{i-1}} + 4T_{\text{add}}}{1}. \quad (6.106)$$

The minimal sample period of the filter is then bounded by the section with the lowest throughput, i.e., the loop  $i$  that yields  $T_{\min}$ , since each section has to operate with the same throughput. Figure 6.93 shows the scheduling of the operations belonging to  $m$  sample periods for a first-order allpass section. The shaded areas indicate execution time for the operations and darker shaded areas indicate their latency. Scheduling of second-order allpass sections can be found in [1]. Since a bit-serial processing element only processes 1 bit in each clock cycle, a complete arithmetic operation requires at least  $W_d$  clock cycles. Therefore, if the minimal sample period of  $T_{\min}$  clock cycles, where  $T_{\min} < W_d$ , operations

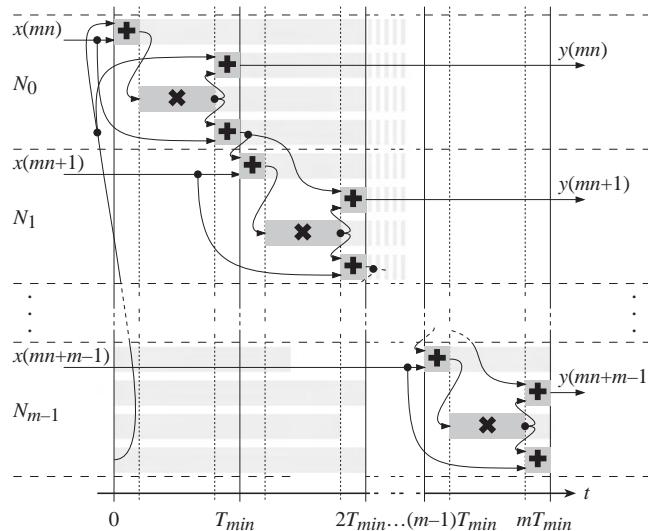


**FIGURE 6.91**

First-order allpass section.

**FIGURE 6.92**

The critical loop in a second-order allpass section.

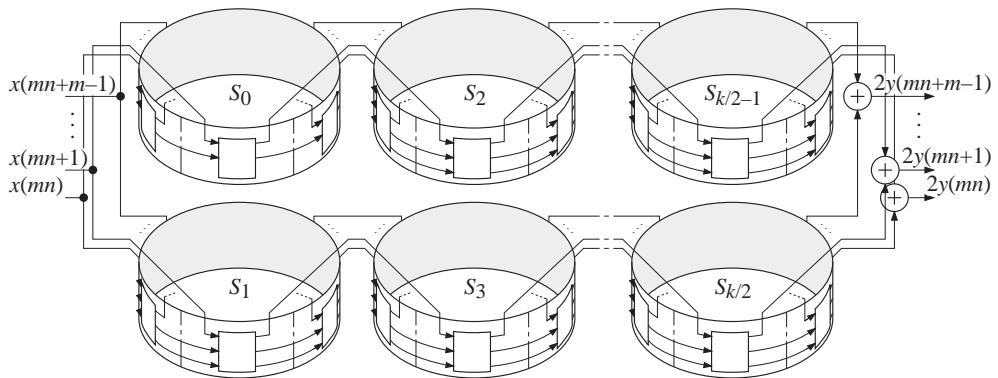
**FIGURE 6.93**

Maximally fast schedule for a first-order allpass section.

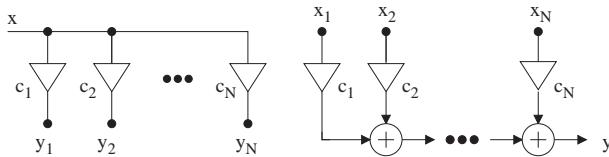
belonging to  $m$  sample periods need to be included in the schedule. It can be shown that  $m$  must be selected as

$$m \geq \left\lceil \frac{W_{cf} + W_d + m_{n0}}{T_{min}} \right\rceil, \quad (6.107)$$

where  $m_{n0}$  is 0 for model 0 and 1 for model 1, respectively in order to allow an arithmetic operation requiring  $W_{cf} + W_d + m_{n0}$  clock cycles to be completed within the scheduling period. Thus, the

**FIGURE 6.94**

Complete lattice wave digital filter.

**FIGURE 6.95**

Substructures suitable for multiple-constant multiplication.

scheduling period for a maximally fast schedule becomes  $mT_{\min}$ . The implementation of the complete filter is illustrated in Figure 6.94.

## 1.06.12 Sum-of-products (SOP)

In this section we discuss the two main techniques to simplify the hardware implementation of sum-of-products and their transpose. Note that the arithmetic networks shown in Figure 6.84 consists of a set of sum-of-products of the type shown in Figure 6.95. Consider the sum-of-products

$$y = \sum_{k=1}^N a_k x_k. \quad (6.108)$$

Typically, the coefficients,  $a_k$ , are fixed and  $x_k$  are variable, but there are several common cases where both  $a_k$  and  $x_k$  are variable, for example, computation of correlation, which require a general multiplier.

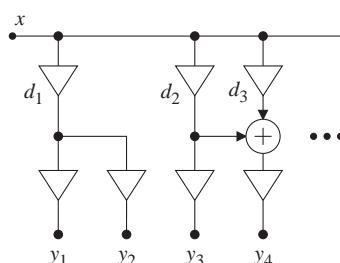
Distributed arithmetic is an efficient procedure for computing sum-of-products between a fixed and a variable data vector. The basic principle is owed to Croisier et al. [107]. Distributed arithmetic is suitable for implementation of relative short sum-of-products, discrete cosine transforms (DCT), as well as complex multiplications [1].

### 1.06.12.1 Multiple-constant multiplication (MCM)

Multiple-constant multiplication techniques [1, 3] are efficient methods to simplify multipliers with fixed coefficients and thereby reduce the power consumption. Single multipliers and substructures of the type shown in Figure 6.95 can be simplified significantly by using multiple-constant multiplication techniques [4–7, 108, 109]. For the multiple-constant multiplication case several effective algorithms have been proposed that avoids the problem of number representation dependency [110, 111]. Theoretical lower bounds for related problems have been presented in [11]. Moreover, using linear programming, the filter coefficient optimization and the MCM implementation can be jointly addressed [112, 113].

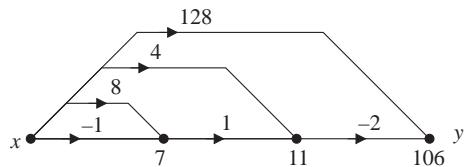
Figure 6.96 shows a typical example of an MCM network. The basic idea is to reduce the number of basic operations (i.e., add/sub-and-shift operations) by factoring out common factors in the coefficients  $c_i$ . By first generating a set of common subexpressions,  $d_1, d_2, d_3$ , that in the next step are multiplied with simple coefficients, or added/subtracted, to generate new subexpressions, and so on [8, 95]. A main drawback of these techniques is that they are highly specialized for a particular SOP and can therefore not be multiplexed to perform other SOPs. For multiple-constant multiplication, with many coefficients (for example for long FIR filters) the average increase of the number of adders/subtractors, approaches one, for an increase of the length of the filter by one. This is due to the fact that most subexpressions have already been generated and only a single addition/subtraction is needed to generate another coefficient. The problem of finding optimal solutions to the multiple-constant multiplication problem is NP complete. Several heuristic algorithms have therefore been derived, e.g., [4–6, 114, 115].

Figure 6.97 shows a graph that simultaneously realize the multiplication with 7, 11, and 106. Only three adder/subtractors and some shift operations are required. These techniques are efficient for implementing bit-parallel, digit-serial as well as bit-serial processing elements [1].



**FIGURE 6.96**

MCM network.

**FIGURE 6.97**

MCM example.

An implementation according to Figure 6.84 will consist of a set of MCM networks where the input to a network is taken from a register and the different subexpressions are SOP to be stored into other registers.

### 1.06.12.2 Distributed arithmetic

Consider the sum-of-products in (6.108) and assuming that the coefficients,  $a_k, k = 1, 2, \dots, N$  are fixed. Two's-complement representation is used for both coefficients and data. The inputs are scaled so that  $|x_k| \leq 1$ . The sum-of-products can be rewritten

$$y = \sum_{k=1}^N a_k \left( -x_{k0} + \sum_{i=1}^{W_d-1} x_{ki} 2^{-i} \right), \quad (6.109)$$

where  $x_{ki}$  is the  $i$ th bit in  $x_k$ . By interchanging the order of the two summations we get

$$y = - \sum_{k=1}^N a_k x_{k0} + \sum_{i=1}^{W_d-1} \sum_{k=1}^N (a_k x_{ki}) 2^{-i} \quad (6.110)$$

which can be written

$$y = -F_0(x_{10}, x_{20}, \dots, x_{N0}) + \sum_{i=1}^{W_d-1} F_i(x_{1i}, x_{2i}, \dots, x_{Ni}) 2^{-i}, \quad (6.111)$$

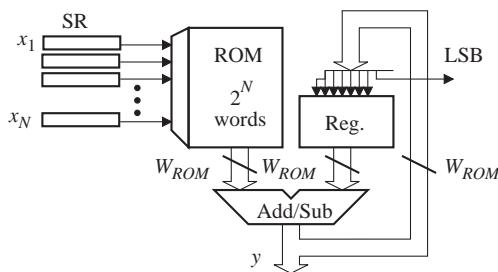
where

$$F_i(x_{1i}, x_{2i}, \dots, x_{Ni}) = \sum_{k=1}^N a_k x_{ki}. \quad (6.112)$$

$F$  is a function of  $N$  binary variables, the  $k$ th variable being the  $i$ th bit in the data  $x_k$ . Since  $F_i$  can take on only a finite number of values,  $2^N$ , it can be computed and stored in a ROM (Read-Only Memory). Equation (6.111) can, using Horner's method for evaluating a polynomial for  $x = 0.5$ , be rewritten as

$$y = (((((0 + F_{W_d-1})2^{-1} + \dots + F_2)2^{-1} + F_1)2^{-1} - F_0)). \quad (6.113)$$

Figure 6.98 shows a block diagram for computing a sum-of-products according to (6.113).

**FIGURE 6.98**

Block diagram for distributed arithmetic.

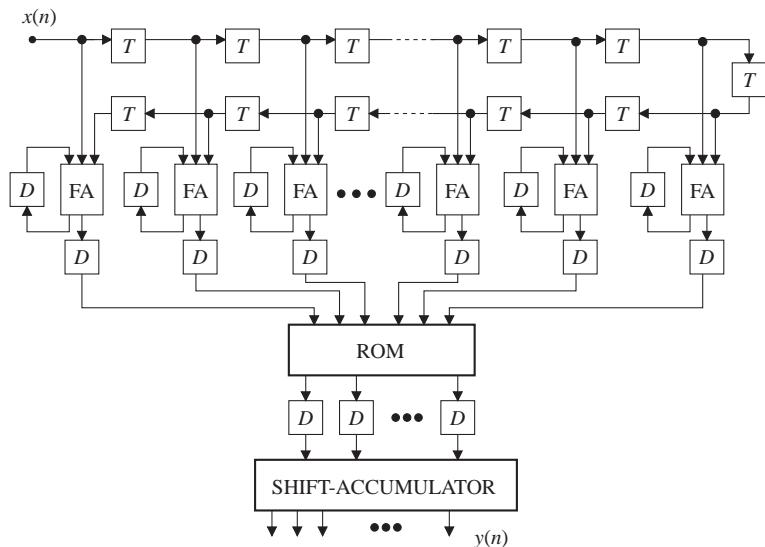
Inputs,  $x_1, x_2, \dots, x_N$  are shifted bit-serially out from the shift registers with the least-significant bit first. The bits  $x_{ki}$  are used as an address to the ROM storing the look-up table. Computation of the sum-of-products starts by adding  $F_{W_d-1}$  to the initially cleared accumulator register, REG. In the next clock cycle, outputs from the shift registers address  $F_{W_d-2}$ , which is added to the value in the accumulator register. The result is divided by 2 and stored back into REG. After  $W_d - 1$  clock cycles,  $F_0$  is subtracted from the value in the accumulator register. The computation time is  $W_d$  clock cycles. The required wordlength in the ROM,  $W_{ROM}$ , depends on the  $F_{\text{imax}}$  with the largest magnitude and the coefficient wordlength,  $W_c$ , and  $W_{ROM} = \lceil W_c + \log_2(N) \rceil$ .

The hardware cost is similar to a multiplier. In fact, most multiplier structures, e.g., serial/parallel, array, and three structures multipliers, can be used for distributed arithmetic. Distributed arithmetic can be used to implement first- and second-order direct form I sections, FIR filters, as well as wave digital filters [20, 107, 116–119].

#### 1.06.12.2.1 Implementation of FIR filters using distributed arithmetic

Figure 6.99 shows an implementation of an eleventh-order linear-phase FIR filter.  $N/2$  bit-serial adders (subtractors) are used to sum the symmetrically placed values in the delay line. This reduces the number of terms in the sum-of-products. Only 64 words are required whereas  $2^{12} = 4096$  words are required for the general case, e.g., a nonlinear-phase FIR filter. For higher-order FIR filters the reduction in the number of terms by 50% is significant. Further, the logic circuitry has been pipelined by introducing  $D$  flip-flops between the adders (subtractors) and the ROM, and between the ROM and the shift-and-add accumulator.

The number of words in the ROM is  $2^N$  where  $N$  is the number of terms in the sum-of-products. The chip area for the ROM is small for sum-of-products with up to 5–6 terms. The basic approach is useful for up to 10–11 terms. However, sum-of-products containing many terms, can be partitioned into a number of smaller sum-of-products, which can be computed and summed by using either distributed arithmetic or an adder tree. A complete sum-of-products PE and several similar types of PEs, for example adaptors and butterflies, can be based on distributed arithmetic. The main parts of a sum-of-products PE are the shift-accumulator, the coefficient ROM with decoder, and the control circuits. Overflow and quantization circuitry are also necessary, but it is often advantageous to move parts of this circuitry to the serial/parallel converters used in the interconnection network [1].

**FIGURE 6.99**

Eleventh-order linear-phase FIR filter.

Distributed arithmetic can, of course, be implemented in parallel form, i.e., by allocating a ROM to each of the terms in (6.113) and use an adder tree for the accumulation.

### 1.06.12.2 Memory reduction techniques

The amount of memory required becomes very large for long sum-of-products. There are mainly two ways to reduce the memory requirements. Namely, partitioning and coding of input signals. The two methods can be applied at the same time to obtain a very small amount of memory.

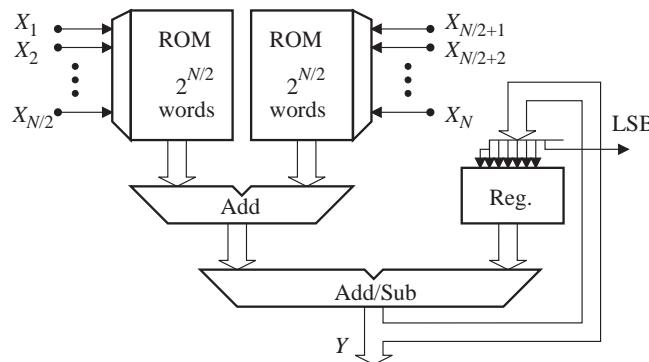
One way to reduce the overall memory requirement is to partition the memory into smaller pieces that are added before the shift-accumulator as shown in Figure 6.100.

The amount of memory is reduced from  $2^N$  words to  $2 \cdot 2^{\frac{N}{2}}$  words if the original memory is partitioned into two parts. For example, for  $N = 10$  we get  $2^{10} = 1024$  words to  $2 \cdot 2^5 = 64$  words. Hence, this approach reduces the memory significantly at the cost of an additional adder. In addition, the partitioning may in some case be done so that long and short values are stored in different partitions. Notice, depending on the values in the ROM, it is often favorable from both speed and area points of view to implement a ROM by logic gates. Large ROMs tend to be slow.

The second approach is based on a special coding of the inputs,  $x_i$ , which result in a symmetric ROM content [1, 107]. Memory size can be halved by using the ingenious scheme [107] based on the identity

$$x = \frac{1}{2}(x - (-x)). \quad (6.114)$$

This approach is also used for implementation of a complex multiplier using only two shift-accumulators and a common ROM [1].

**FIGURE 6.100**

Reducing the memory by partitioning.

### 1.06.13 Power reduction techniques

Many approaches to reduce the power consumption are based on various techniques to decrease the arithmetic workload. This can be done at the algorithmic level by using efficient, low-sensitivity structures that allows few and simple coefficients, and at the hardware level by using simplified multiplier realizations. Usually the power consumption, due to the memory, communication, and control, is neglected in these techniques, although their contribution may be significant. Polyphase structures and frequency masking techniques are commonly used to reduce the arithmetic workload.

Another approach is based on power supply voltage scaling where the implementation is operated with a low supply voltage. Pipelining and interleaving are not applicable to recursive algorithms. However, pipelining inside recursive loops can be performed at the logic level. It is therefore important to exploit all parallelism of the recursive algorithm in order to obtain excess speed that allows a reduction in the power supply voltage.

We propose the following strategy to design recursive digital filters with a low power consumption:

- Select an inherently fast and low-sensitivity filter algorithm, e.g., ladder or lattice wave digital filters, possibly in combination with frequency masking techniques.
- Use an approximation with a diminishing ripple, which reduces the passband sensitivity in ladder WDFs, and use discrete optimization techniques to reduce the operation latencies in the critical loop.
- Use pipelining to split the nonrecursive parts into smaller pieces so that  $T_{\min}$  can be achieved.
- Find a maximally fast and resource minimal schedule.
- Use an isomorphic mapping to an optimal hardware structure with bit- or digit-serial PEs.
- Use CMOS logic circuits that yield bit- or digit-serial PEs with a low power consumption, especially for the D flip-flops in the bit-serial case.
- Convert any excess circuit speed to a reduced power consumption by reducing the power supply voltage.

*Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 5](#) Sampling and Quantization

See this Volume, [Chapter 7](#) Multirate Signal Processing

## References

- [1] L. Wanhammar, DSP Integrated Circuits, Academic Press, 1999.
- [2] A.P. Chandrakasan, R.W. Brodersen, Low Power Digital CMOS Design, Kluwer, 1995.
- [3] I. Koren, Computer Arithmetic Algorithms, Prentice-Hall, 1993.
- [4] D.R. Bull, D.H. Horrocks, Primitive operator digital filters, IEE Proc. G 138 (3) (June 1991) 401–412.
- [5] A.G. Dempster, M.D. Macleod, Constant integer multiplication using minimum adders, IEE Proc. Circuits Devices Systems 141 (5) (1994) 407–413.
- [6] A.G. Dempster, M.D. Macleod, Use of minimum-adder multiplier blocks in FIR digital filters, IEEE Trans. Circuits Syst. II CAS-II-42 (9) (1995) 569–577.
- [7] R.I. Hartley, Subexpression sharing in filters using canonic signed digit multipliers, IEEE Trans. Circuits Syst. II, CAS-II-43, (10) (1996) 677–688.
- [8] M. Martinez-Peiro, E. Boemo, L. Wanhammar, Design of high speed multiplierless filters using a nonrecursive signed common subexpression algorithm, IEEE Trans. Circuits Syst. II, CAS-II-49 (3) (2002) 196–203.
- [9] O. Gustafsson, A.G. Dempster, On the use of multiple constant multiplication in polyphase FIR filters and filter banks, in: Proc. Nordic Signal Processing Symp., vol. 3, Espoo, Finland, June 9–11, 2004, pp. 53–56.
- [10] O. Gustafsson, H. Johansson, L. Wanhammar, MILP design of frequency-response masking FIR filters with few SPT terms, in: Proc. Int. Symp. Control, Communications, Signal Processing, Hammamet, Tunisia, March, 2004, pp. 21–24.
- [11] O. Gustafsson, Lower bounds for constant multiplication problems, IEEE Trans. Circuits Syst. II CAS-II-54 (11) (2007) pp. 974–978.
- [12] L.B. Jackson, Roundoff noise bounds derived from coefficient sensitivities for digital filters, IEEE Trans. Circ. Sys. CAS-23 (8) (1976) pp. 481–485.
- [13] L.O. Chua, T. Lin, Chaos in digital filters, IEEE Trans. Circ. Sys. CAS-35 (6) (1988) pp. 648–658.
- [14] L. Wanhammar, H. Johansson, Digital Filters Using MATLAB, Springer, 2013.
- [15] G. Jovanovic-Dolecek (Ed.), Multirate Systems: Design and Applications, Idea Group Publ, Hersey, USA, 2001.
- [16] L. Milic, Multirate Filtering for Digital Signal Processing: MATLAB Applications, Information Science Reference, 2009.
- [17] L. Wanhammar, H. Johansson, Toolbox: Digital Filters Using MATLAB. <<http://www.es.isy.liu.se/software/>>, 2012.
- [18] R.W. Daniels, Approximation Methods for Electronic Filter Design, McGraw-Hill, New York, 1974.
- [19] M.D. Lutovac, D.V. Tasic, B.L. Evan, Filter Design for Signal Processing Using MATLAB and Mathematica, Prentice Hall, 2001.
- [20] L. Wanhammar, Analog Filters Using MATLAB, Springer, 2009.
- [21] L. Wanhammar, Toolbox: Analog Filters Using MATLAB. <<http://www.es.isy.liu.se/software/>>, 2009.
- [22] A. Antoniou, Digital filters, Analysis, Design and Applications, McGraw-Hill, 2000.
- [23] M.G. Bellanger, G. Bonnerot, M. Coudreuse, Digital filtering by polyphase network: Application to sample-rate alteration and filter banks, IEEE Trans. Acoust., Speech, Signal Process. ASSP-24 (2) (1976) 109–114.
- [24] E.C. Ifeachor, B.W. Jervis, Digital Signal Processing: A Practical Approach, Addison-Wesley, 2002.

- [25] A. Fettweis, Digital filter structures related to classical filter networks, *Archiv fur Elektronik und Übertragungstechnik* 5 (2) (1971) 79–89.
- [26] A. Fettweis, On the connection between multiplier word length limitation and roundoff noise in digital filters, *IEEE Trans. Circuit Theory* CT-19 (5) (1972) pp. 486–491.
- [27] A. Fettweis, Wave digital filters: theory and practice, *Proc. IEEE* 2 (2) (1986) 270–327.
- [28] D.F. Elliott (Ed.), *Handbook of Digital Signal Processing, Engineering Applications*, Academic Press, 1988.
- [29] J.G. Proakis, D.G. Manolakis, *Digital Signal Processing, Principles, Algorithms, and Applications*, third Ed., Prentice Hall, 1996.
- [30] L.R. Rabiner, B. Gold, *Theory and Application of Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1975.
- [31] N.J. Fliege, *Multirate Digital Signal Processing*, John Wiley and Sons, New York.
- [32] P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993.
- [33] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice Hall, NJ, 1983.
- [34] J.H. McClellan, T.W. Parks, L.R. Rabiner, A computer program for designing optimum FIR linear phase digital filters, *IEEE Trans. Audio Electroacoust. AU-21* (1973) 506–526.
- [35] M. Ahsan, T. Saramäki, A MATLAB based optimum multibands FIR filters design program following the original idea of the Remez multiple exchange algorithm, in: *Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS-11*, Rio de Janeiro, Brazil, pp. 137–140.
- [36] MATLAB, *Signal Processing Toolbox User's Guide*, The MathWorks, 2012a.
- [37] C.H. Chen (Ed.), *The Circuits and Filters Handbook*, second ed., CRC Press, New York, 2003.
- [38] S.K. Mitra, J.F. Kaiser (Eds.), *Handbook for Digital Signal Processing*, John Wiley and Sons, 1993.
- [39] T. Saramäki, M. Renfors, A novel approach for the design of IIR filters as a tapped cascaded interconnection of identical allpass subfilters, in: *Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS-87*, vol. 2, Philadelphia, May 4–7, 1987, pp. 629–632.
- [40] A.V. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.
- [41] K. Nakayama, Permuted difference coefficient realization of FIR digital filters, *IEEE Trans. Acoust., Speech, Signal Process. ASSP-30* (2) (1982) 269–278.
- [42] H. Ohlsson, O. Gustafsson, L. Wanhammar, A shifted permuted difference coefficient method, in: *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, Vancouver, Canada, May 23–26, 2004, pp. 161–164.
- [43] A.S. Sedra, P.O. Brackett, *Filter Theory and Design: Active and Passive*, Pitman, London, 1978.
- [44] H.J. Orchard, Inductorless filters, *Electron. Lett.* 2 (1966) 224–225.
- [45] L. Wanhammar, A bound on the passband deviation for symmetric and antimetric commensurate transmission line filters, 1991. <<http://www.es.isy.liu.se/publications/>>
- [46] M.D. Lutovac, D.V. Tosic, B.L. Evans, Advanced filter design, in: *Proc. 34th Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, CA, November 2–5, 1997, pp. 710–715.
- [47] G. Groenewold, Noise and group delay in active filters, *IEEE Trans. Circuits and Syst. I CAS-I-54* (7) (2007) 1471–1480.
- [48] A. Fettweis, Digital filter structures related to classical filter networks, *Archiv fur Elektronik und Übertragungstechnik* 25 (2) (1971) 79–89.
- [49] M. Yaseen, On the design of multiband transmission functions synthesized by one wave digital lattice structure, *Intern. J. Circ Theor App* (2011).
- [50] M. Yaseen, Robust simultaneous amplitude and phase approximations oriented to bandpass wave digital lattice filters, *Intern. J. Circ Theor App* 26 (1998) 179–189.
- [51] J.A. Nossek, M.T. Ivrla, On the relation of circuit theory and signals, systems and communications, in: *Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS-11*, Rio de Janeiro, Brazil, May 15–18, 2011, pp. 603–604.

- [52] A. Fettweis, Some principles of designing digital filters imitating classical filter structures, *IEEE Trans. Circuit Theory* CT-18 (2) (March 1971) 314–316.
- [53] A. Fettweis and K. Meerkötter, On adaptors for wave digital filters, *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-23 (6) (1975) 516–525.
- [54] A. Fettweis, Pseudopassivity, sensitivity, and stability of wave digital filters, *IEEE Trans. Circuit Theory* CT-19 (6) (1972) pp. 668–673.
- [55] A. Fettweis, K. Meerkötter, Suppression of parasitic oscillations in wave digital filters, in: *Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS-74*, San Francisco, April 1974, pp. 682–686.
- [56] A. Fettweis, K. Meerkötter, Suppression of parasitic oscillations in wave digital filters, *IEEE Trans. Circuits and Systems CAS-22* (3) (1975) 239–246.
- [57] A. Fettweis, Passivity and losslessness in digital filtering, *Archiv fur Elektronik und Übertragungstechnik* 42 (1) (1988) pp. 1–8.
- [58] A. Fettweis, Scattering properties of wave digital filters, in: *Proc. Florence Sem. on digital filtering*, Florence, Italy, September 1972, pp. 1–8.
- [59] A.H. Gray, J.D. Markel, Digital lattice and ladder filter synthesis, *IEEE Trans. Audio Electroacoust. AU-21* (6) (1973) 491–500.
- [60] A.H. Gray, J.D. Markel, A normalized digital filter structure, *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-23 (3) (1975) 268–277.
- [61] A.H. Gray, J.D. Markel, Roundoff noise characteristics of a class of orthogonal polynomial structures, *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-23 (October 1975) 473–486.
- [62] A.H. Gray, J.D. Markel, Fixed-point implementation algorithms for a class of orthogonal polynomial filter structures, *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-23 (5) (1975) 486–494.
- [63] A.N. Willson, H.J. Orchard, Insights into digital filters made as a sum of two allpass functions, *IEEE Trans. Circuits Syst. I CAS-I-42* (3) (1995) 129–137.
- [64] V.I. Anzova, J. Yli-Kaakinen, T. Saramäki, An algorithm for the design of multiplierless IIR filters as a parallel connection of two all-pass filters, in: *Proc. IEEE Asia Pacific Conf. on Circuits and Systems, APCCAS*, Singapore, December 2006, pp. 744–747.
- [65] B. Jaworski, T. Saramäki, Linear phase IIR filters composed of two parallel allpass sections, in: *IEEE Intern. Symp. on Circuits and Systems, ISCAS-94*, vol. 2, London, May 1994, pp. 537–540.
- [66] Y.C. Lim, Frequency-response masking approach for the synthesis of sharp linear phase digital filters, *IEEE Trans. Circuits and Systems CAS-33* (4) (1986) 357–364.
- [67] Y.C. Lim, Y. Lian, The optimum design of one- and two-dimensional FIR filters using the frequency response masking technique, *IEEE Trans. Circuits and Syst. II CAS-II-40* (1993) 88–95.
- [68] J. Yli-Kaakinen, T. Saramäki, An efficient algorithm for the optimization of FIR filters synthesized using the multistage frequency-response, *Circ. Syst. Signal. Process.* 30 (1) (2011) 157–183.
- [69] Y. Neuvo, D. Cheng-Yu, and S.K. Mitra, Interpolated finite impulse response filters, *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-32 (3) (1984) pp. 563–570.
- [70] Y.C. Lim, Y. Lian, Frequency-response masking approach for digital filter design: complexity reduction via masking filter factorization, *IEEE Trans. Circuits and Syst. II CAS-II-41* (1994) 518–525.
- [71] O. Gustafsson, H. Johansson, L. Wanhammar, Single filter frequency-response masking FIR filters, *J. Circ. Sys. Comp.* 12 (5) (2003) pp. 601–630.
- [72] Y. Lian, C.Z. Yang, Complexity reduction by decoupling the masking filters from the bandedge shaping filter in the FRM technique, *Circ. Syst. Signal. Process.* 22 (2) (2003) 115–135.
- [73] T. Saramäki, Y.C. Lim, R. Yang, The synthesis of half-band filter using frequency-response masking technique, *IEEE Trans. Circuits Syst. II CAS-II-42* (1) (1995) 58–60.
- [74] Y.C. Lim, Y.J. Yu, Synthesis of very sharp Hilbert transformer using the frequency-response masking technique, *SP-53* (7) (2005) 2595–2597.

- [75] H. Johansson, Two classes of frequency-response masking linear-phase FIR filters for interpolation and decimation, *Circuits, Syst. and Signal Processing*, Special issue on Computationally Efficient Digital Filters: Design and Applications 25 (2) (2006) 175–200.
- [76] H. Johansson, L. Wranhammar, Filter structures composed of allpass and FIR filters for interpolation and decimation by a factor of two, *IEEE Trans. Circuits Syst. II CAS-II-46* (7) (1999) 896–905.
- [77] H. Johansson, P. Löwenborg, On the design of adjustable fractional delay FIR filters, *IEEE Trans. Circuits Syst. II* 50 (4) (2003) 164–169.
- [78] L. Rosenbaum, P. Löwenborg, H. Johansson, An approach for synthesis of modulated M-channel filter banks utilizing the frequency-response masking technique, *EURASIP J. Advances Signal Processing - Special Issue on Multirate Systems and Applications* 2007 (1) (2007).
- [79] R. Bregovic, Y.C. Lim, T. Saramäki, Frequency-response masking-based design of nearly perfect-reconstruction two-channel FIR filterbanks with rational sampling factors, *Trans. Circuits and Syst. I CAS-I-55* (7) (2008) 2002–2012.
- [80] K.M. Tsui, S.C. Chan, Y.C. Lim, Design of multi-plet perfect reconstruction filter banks using frequency-response masking technique, *IEEE Trans. Circuits and Syst. I CAS-I-55* (9) (2008) 2707–2715.
- [81] Y.C. Lim, B. Farhang-Boroujeny, Fast filter bank (FFB), *IEEE Trans. Circuits and Syst. II CAS-II-39* (5) (1992) 316–318.
- [82] P.S.R. Diniz, L.C.R. de Barcellos, S.L. Netto, Design of high-resolution cosine-modulated transmultiplexers with sharp transition band, *Trans. Signal Process.* 5 (2004).
- [83] H. Johansson, L. Wranhammar, High-speed recursive digital filters based on the frequency-response masking approach, *IEEE Trans. Circuits Syst. II CAS-II-47* (1) (2000) 48–61.
- [84] Y.C. Lim, S.H. Lou, Frequency-response masking approach for the synthesis of sharp two-dimensional diamond-shaped filters, *IEEE Trans. Circuits and Syst. II CAS-II-45* (12) (1998) 1573–1584.
- [85] M. Renfors, Y. Neuvo, The maximal sampling rate of digital filters under hardware speed constraints, *IEEE Trans. Circuits and Systems* CAS-28 (3) (1981) 196–202.
- [86] K. Palmkvist, M. Vesterbacka, L. Wranhammar, Arithmetic transformations for fast bit-serial VLSI implementations of recursive algorithms, in: Proc. Nordic Signal Processing Symp., NORSIG'96, Espoo, Finland, September 24–27, 1996, pp. 391–394.
- [87] H. Ohlsson, O. Gustafsson, L. Wranhammar, Arithmetic transformations for increased maximal sample rate of bit-parallel bireciprocal lattice wave digital filters, in: Proc. IEEE Int. Symp. Circuits Syst., Sydney, Australia, May 2001, pp. 6–9.
- [88] M. Vesterbacka, K. Palmkvist, P. Sandberg, L. Wranhammar, Implementation of fast bit-serial lattice wave digital filters, in: Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS '94, vol. 2, London, May 30–June 1, 1994, pp. 113–116.
- [89] M. Vesterbacka, K. Palmkvist, L. Wranhammar, On implementation of fast, bit-serial loops, in: Proc. IEEE 1996 Midwest Symp. on Circuits and Systems, vol. 1, Ames, Iowa, August 18–21, 1996, pp. 190–193.
- [90] M. Vesterbacka, K. Palmkvist, L. Wranhammar, Maximally fast, bit-serial lattice wave digital filters, in: Proc. IEEE DSP Workshop 96, Loen, Norway, September 2–4, 1996, pp. 207–210.
- [91] O. Gustafsson, L. Wranhammar, Maximally fast scheduling of bit-serial lattice wave digital filters using three-port adaptor allpass sections, in: Proc. IEEE Nordic Signal Processing Symp., Kolmårdens, Sweden, June 13–15, pp. 441–444.
- [92] K. Johansson, O. Gustafsson, L. Wranhammar, Power estimation for ripple-carry adders with correlated input data, in: Proc. IEEE Intern. Workshop on Power and Timing Modeling, Optimization and Simulation, Santorini, Greece, September 15–17 2004.
- [93] R. Zimmermann, Binary adder architectures for cell-based VLSI and their synthesis. Ph.D. Thesis, Swiss Federal Institute of Technology (ETH), Zurich, Hartung-Gorre Verlag, 1998.

- [94] O. Gustafsson, L. Wanhammar, Low-complexity constant multiplication using carry-save arithmetic for high-speed digital filters, in: Int. Symp. Image, Signal Processing, Analysis, Istanbul, Turkey, 2007, pp. 27–29.
- [95] O. Gustafsson, A.G. Dempster, L. Wanhammar, Multiplier blocks using carry-save adders, in: Proc. IEEE Intern. Symp. Circuits Systems, Vancouver, Canada, May 2004, pp. 23–26.
- [96] V.G. Oklobdzija, D. Villegas, S.S. Liu, A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach, *IEEE Trans. Computers* C-45 (3) (1996).
- [97] M.J. Irwin, R.M. Owens, Design issues in digit serial signal processors, in: Proc. IEEE Intern. Symp. on Circuits and Systems, vol. 1, May 8–11, 1989, pp. 441–444.
- [98] H. Lee, G.E. Sobelman, Digit-serial reconfigurable FPGA logic block architecture, in: Proc. IEEE Workshop on Signal Processing Systems, SIPS 98, October 8–10, 1998, pp. 469–478.
- [99] K.K. Parhi, A systematic approach for design of digit-serial signal processing architectures, *IEEE Trans. Circ. Sys. CAS-38*, no. 4, pp. 358–375, April 1991.
- [100] H. Lee, G.E. Sobelman, Digit-serial DSP library for optimized FPGA configuration, in: Proc. IEEE Symp. on FPGAs for Custom Computing Machines, April 15–17, 1998, pp. 322–323.
- [101] H. Suzuki, Y.-N. Chang, K.K. Parhi, Performance tradeoffs in digit-serial DSP systems, in: Proc. Thirty-Second Asilomar Conf., vol. 2, Pacific Grove, CA, November 1–4, 1998, pp. 1225–1229.
- [102] Y.-N. Chang, J.H. Satyanarayana, K.K. Parhi, Systematic design of high-speed and low-power digit-serial multipliers, *IEEE Trans. Circuits and Syst. II, CAS-II-45* (12) (1998) pp. 1585–1596.
- [103] Y.-N.C.J.H., Satyanarayana, K.K. Parhi, Low-power digit-serial multipliers, in: Proc. IEEE Intern. Symp. on Circuits and Systems, ISCAS '97, vol. 3, June 9–12, 1997, pp. 2164–2167.
- [104] P. Nilsson, Arithmetic and architectural design to reduce leakage in nano-scale digital circuits, in: Proc. 18th European Conference on Circuit Design, ECCTD 07, Seville, Spain, 2007, pp. 373–375.
- [105] A.N. Willson, H.J. Orchard, A design method for half-band FIR filters, *IEEE Trans. Circuits Syst. ICAS-I-45* (1) (1999) pp. 95–101.
- [106] K. Johansson, O. Gustafsson, L. Wanhammar, Low-complexity bit-serial constant-coefficient multipliers, in: Proc. IEEE Int. Symp. Circuits Syst., vol. 3, Vancouver, Canada, May 23–26, 2004, pp. 649–652.
- [107] A. Croisier, D.J. Esteban, M.E. Levilion, V. Rizo, Digital Filter for PCM Encoded Signals. U.S. Patent 3777130, December 4, 1973.
- [108] A.G. Dempster, M.D. Macleod, O. Gustafsson, Comparison of graphical and sub-expression elimination methods for design of efficient multipliers, in: Proc. Asilomar Conf. Signals Syst. Comp., Monterey, CA, November 7–10, 2004.
- [109] M. Potkonjak, M.B. Srivastava, A.P. Chandrakasan, Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination, *IEEE Trans. Computer-Aided Design CAD-15* (2) (1996) 151–165.
- [110] O. Gustafsson, A difference based adder graph heuristic for multiple constant multiplication problems, in: Proc. IEEE Int. Symp. Circuits Syst., New Orleans, LA, May 27–30, 2007, pp. 1097–1100.
- [111] Y. Voronenko and M. Püschel, Multiplierless multiple constant multiplication, *ACM Trans. Algorithms* 3 (2) (2007).
- [112] Y.J. Yu, Y.C. Lim, Design of linear phase FIR filters in subexpression space using mixed integer linear programming, *IEEE Trans. Circuits Syst. I* 54 (10) (2007) 2330–2338.
- [113] D. Shi, Y.J. Yu, Design of linear phase FIR filters with high probability of achieving minimum number of adders, *IEEE Trans. Circuits Syst. I* 58 (1) (2011) 126–136.
- [114] K. Johansson, O. Gustafsson, L. Wanhammar, Multiple constant multiplication for digit-serial implementation of low power FIR filters, *WSEAS Trans. Circuits Syst.* 5 (7) (2006) 1001–1008.

- [115] K. Johansson, O. Gustafsson, L.S. DeBrunner, L. Wanhammar, Minimum adder depth multiple constant multiplication algorithm for low power FIR filters, in: Proc. IEEE Int. Symp. Circuits Syst, Rio de Janeiro, Brazil, May, 2011, pp. 15–18.
- [116] M. Büttner, H.W. Schüssler, On structures for the implementation of the distributed arithmetic, Nachrichtentechn. Z. 29 (6) (1976) 472–477.
- [117] A. Peled, B. Liu, Digital Signal Processing: Theory, Design and Implementation, John Wiley and Sons, 1976.
- [118] L. Wanhammar, Implementation of wave digital filters using vector-multipliers, in: Proc. First European Signal Processing Conf., EUSIPCO-80, Lausanne, Switzerland, September 1980, pp. 21–26.
- [119] S. White, On applications of distributed arithmetic to digital signal processing, a tutorial review, IEEE ASSP Magazine, pp. 4–19, July 1989.

# Multirate Signal Processing for Software Radio Architectures

7

**Fred Harris, Elettra Venosa, and Xiaofei Chen***San Diego State University, Department of Electrical and Computer Engineering,  
San Diego, CA, USA*

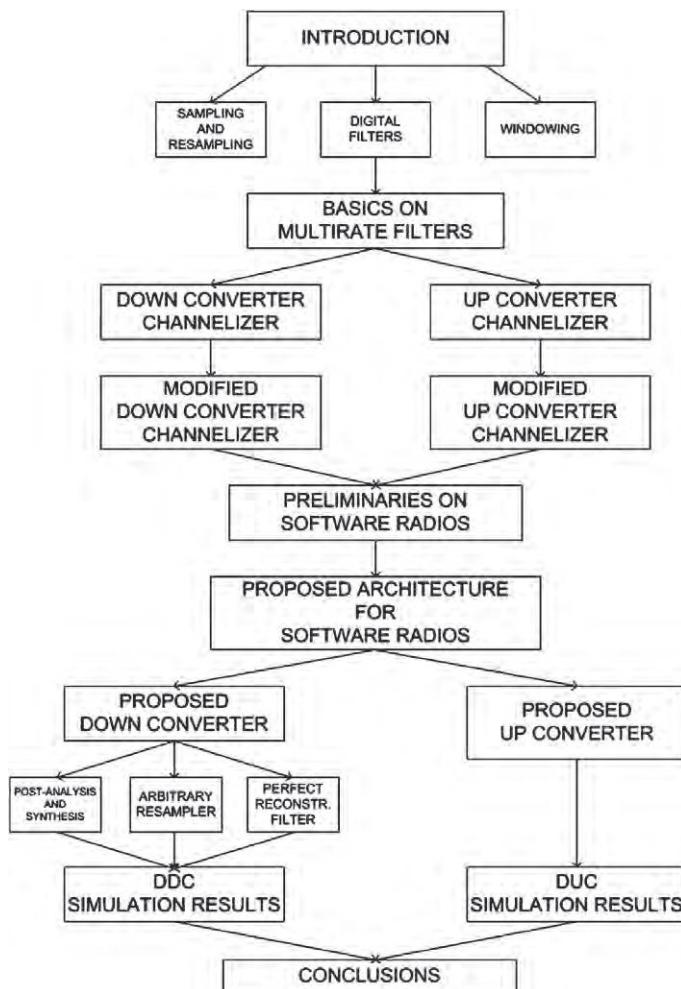
After the introductory sections on resampling theory and basics of digital filters, the architectures of up and down converter channelizers are presented. Those paragraphs are followed by the main core of this chapter in which the authors present novel digital up converter (DUC) and digital down converter (DDC) architectures for software defined radios (SDRs) which are based on variations of standard polyphase channelizers.

The proposed DUC is able to simultaneously up convert multiple signals, having arbitrary bandwidths, to arbitrarily located center frequencies. On the other side of the communication chain, the proposed DDC is able to simultaneously down convert multiple received signals with arbitrary bandwidth and arbitrarily located in the frequency domain. Both the proposed structures avoid the digital data section replication, which is necessary in the current digital transmitters and receivers, when multiple signals have to be handled. Due to the inverse discrete Fourier transform (IDFT), performed by using inverse fast Fourier transform (IFFT) algorithm, embedded in the channelizers, the proposed architectures are very efficient in terms of total workload.

This chapter is structured into two main parts which are composed, in total, of 10 sections. Its structure is described in Figure 7.1. The first part provides preliminary concepts on the sampling process, resampling process of a digital signal, digital filters, multirate structures, standard  $M$ -path polyphase up and down converter channelizers as well as their modified versions that represent the core of the structures we present in the second part of this document. These are well known topics in the digital signal processing (DSP) area and they represent the necessary background for people who want to start learning about software radios. The second part of this chapter goes through new frontiers of the research presenting the novel transmitting and receiving designs for software radios and demonstrating, while explaining the way in which they work, the reasons that make them the perfect candidates for the upcoming cognitive radios.

## 1.07.1 Introduction

Signal processing is the art of representing, manipulating, and transforming wave shapes and the information content that they carry by means of hardware and/or software devices. Until 1960s almost entirely continuous time, analog technology was used for performing signal processing. The evolution of digital systems along with the development of important algorithms such as the well known fast

**FIGURE 7.1**

Structure of the chapter.

Fourier transform (FFT), by Cooley and Tukey in 1965, caused a major shift to digital technologies giving rise to new digital signal processing architectures and techniques. The key difference between analog processing techniques and digital processing techniques is that while the first one processes analog signals, which are continuous functions of time; the second one processes sequences of values. The sequences of values, called digital signals, are series of quantized samples (discrete time signal values) of analog signals.

The link between the analog signals and their sampled versions is provided by the sampling process. When sampling is properly applied it is possible to reconstruct the continuous time signal from

its samples preserving its information content. Thus the selection of the sampling rate is crucial: an insufficient sampling frequency causes, surely, irrecoverable loss of information, while a more than necessary sampling rate causes useless overload on the digital systems.

The string of words multirate digital signal processing indicates the operation of changing the sample rate of digital signals, one or multiple times, while processing them. The sampling rate changes can occur at a single or multiple locations in the processing architecture. When the multirate processing applied to the digital signal is a filtering then the digital filter is named multirate filter; a multirate filter is a digital filter that operates with one or more sample rate changes embedded in the signal processing architecture. The opportunity of selecting the most appropriate signal sampling rate at different stages of the processing architecture, rather than having a single (higher) sampling rate, enhances the performance of the system while reducing its implementation costs. However, the analysis and design of multirate systems could result complicated because of the fact that they are time-varying systems. It is interesting to know that the first multirate filters and systems were developed in the context of control systems. The pioneer papers on this topic were published in the second half of 1950s [1–3]. Soon the idea spilled in the areas of speech, audio, image processing [4,5], and communication systems [6]. Today, multirate structures seem to be the optimum candidates for cognitive and software defined radio [7–11] which represents the frontier of innovation for the upcoming communication systems.

One of the well known techniques that find its most efficient application when embedded in multirate structures is the polyphase decomposition of a prototype filter. The polyphase networks, of generic order  $M$ , originated in the late 1970s from the works by Bellanger et al. [12,13]. The term polyphase is the aggregation of two words: poly, that derives from the ancient greek word polys, which means many, and phase. When applied to an  $M$ -path partitioned filter these two words underline the fact that, on each path, each aliased filter spectral component, experiences a unique phase rotation due to both their center frequencies and the time delays, which are different in each path because of the way in which the filter has been partitioned. When all the paths are summed together the undesired spectral components, having phases with opposite polarity, cancel each other while only the desired components, which experience the same phase on all the arms of the partitions, constructively add up. By applying appropriate phase rotators to each path we can arbitrarily change the phases of the spectral components selecting the spectral component that survives as a consequence of the summation. It is interesting to know that the first applications of the polyphase networks were in the areas of real-time implementation of decimation and interpolation filters, fractional sampling rate changing devices, uniform DFT filter banks as well as perfect reconstruction analysis/synthesis systems. Today polyphase filter banks, embedded in multirate structures, are used in many modern DSP applications as well as in communication systems where they represent, according to the author's belief, one of the most efficient options for designing software-based radio.

The processing architecture that represents the starting point for the derivation of the standard polyphase down converter channelizer is the single channel digital down converter. In a digital radio receiver this engine performs the operations of filtering, frequency translation and resampling on the intermediate frequency (IF) signals. The resampling is a down sampling operation for making the signal sampling rate commensurate to its new reduced bandwidth. When the output sampling rate is selected to be an integer multiple of the signal's center frequency, the signal spectrum is shifted, by aliasing, to base-band and the complex heterodyne defaults to unity value disappearing from the processing path. The two remaining operations of filtering and down sampling are usually performed in a single processing architecture (multirate filter). After exchanging the positions of the filter and resampler, by

applying polyphase decomposition, we achieve the standard  $M$ -path polyphase down converter channelizer which shifts a single narrowband channel to base-band while reducing its sampling rate. By following a similar procedure the standard up converter channelizer can also be obtained. It performs the operation of up converting while interpolating a single narrowband channel.

When the polyphase channelizer is used in this fashion the complex phase rotators can be applied to each arm to select, by coherent summation, a desired channel arbitrarily located in the frequency domain. Moreover, the most interesting and efficient application of this engine is in the multichannel scenario. When the inverse discrete Fourier transform block is embedded, the polyphase channelizer acquires the capability to simultaneously up and down convert, by aliasing, multiple, equally spaced, narrowband channels having equal bandwidths. Due to its computational efficiency, the polyphase channelizer, and its modified versions, represents the best candidate for building up new flexible multichannel digital radio architectures [14] like software defined radio promises to be.

Software defined radio represents one of the most important emerging technologies for the future of wireless communication services. By moving radio functionality into software, it promises to give flexible radio systems that are multi-service, multi-standard, multi-band, reconfigurable, and reprogrammable by software. The goal of software defined radio is to solve the issue of optimizing the use of the radio spectrum that is becoming more and more pressing because of the growing deployment of new wireless devices and applications [15–18].

In the second part of this chapter we address the issue of designing software radio architectures for both the transmitter and the receiver. The novel structures are based on modified versions of the standard polyphase channelizer, which provides the system with the capability to optimally adapt its operating parameters according to the surrounding radio environment [7, 19–21]. This implies, on the transmitter side, the capability of the radio to detect the available spectral holes [22, 23] in the spanned frequency range and to dynamically use them for sending signals having different bandwidths at randomly located center frequencies. The signals could be originated from different information sources or they could be spectral partitions of one signal, fragmented because of the unavailability of free space in the radio spectrum or for protecting it from unknown and undesired detections.

The core of the proposed transmitter is a synthesis channelizer [24]. It is a variant of the standard  $M$ -path polyphase up converter channelizer [6, 8] that is able to perform 2-to- $M$  up sampling while shifting, by aliasing, all the base-band channels to the desired center frequencies. Input signals with bandwidths wider than the synthesis channelizer bandwidth are pre-processed through small down converter channelizers that disassemble their bandwidths into reduced bandwidth sub-channels. The proposed transmitter avoids the need to replicate the sampled data section when multiple signals have to be simultaneously transmitted and it also allows partitioning of the signal spectra, when necessary, before transmitting them. Those spectral fragments will be reassembled in the receiver after being shifted to base-band without any loss of energy because perfect reconstruction filters are used as low-pass prototype in the channelizer polyphase decompositions.

On the other side of the communication chain, a cognitive receiver has to be able to simultaneously detect multiple signals, recognize, when necessary, all their spectral partitions, filter, down convert and recompose them without energy losses [7] independently of their bandwidths or center frequencies. An analysis channelizer is the key element of the proposed receiver [7]. This engine is able to perform  $M$ -to-2 down sampling while simultaneously demodulating, by aliasing, all the received signal spectra having

arbitrary bandwidths residing at arbitrary center frequencies [9]. Post-processing up converter channelizers are used for reassembling, from the analysis channelizer base-line channels, signal bandwidths wider than the analysis channelizer channel bandwidth.

In the transmitter complex frequency rotators apply the appropriate frequency offsets for arbitrary center frequency positioning of the spectra. When the frequency rotators are placed in the proposed receiver, they are responsible for perfect DC alignment of the down converted signals.

This chapter is composed of 10 main sections. The first four sections are dedicated to the basics of digital signal processing, multirate signal processing, and polyphase channelizers. These sections contain preliminary concepts necessary for completely understanding the last sections of this work in which actual research issues on software radio (SR) architecture design are discussed and innovative results are presented. In particular, Section 1.07.2 recalls basic concepts on the resampling process of a discrete time signal as opposed to the sampling process of an analog, continuous time, signal. Section 1.07.3 introduces the readers to digital filters providing preliminaries on their design techniques. Section 1.07.5 presents the standard single path architectures for down sampling and up sampling digital signals. Section 1.07.6 introduces the standard version of the  $M$ -path polyphase down and up converter channelizers. Both of them are derived, step by step, by the single path structures that represent the current state of the technology. In Section 1.07.7 we present the modified versions of these engines. These are the structures that give form to the proposed digital down and up converters for SDRs which are presented in Section 1.07.9 which also provides the simulation results. Section 1.07.8 introduces the readers to the concept of software radio while Section 1.07.10 gives concluding remarks along with suggestions for future research works in this same area.

## 1.07.2 The Sampling process and the “Resampling” process

In the previous section we mentioned that the sampling process is the link between the continuous time world and the discrete time world. By sampling a continuous time signal we achieve its discrete time representation. When sampling frequency is properly selected the sampling process becomes invertible and it is possible to re-shift from the discrete time representation to the continuous time representation of a signal maintaining its information content during the process.

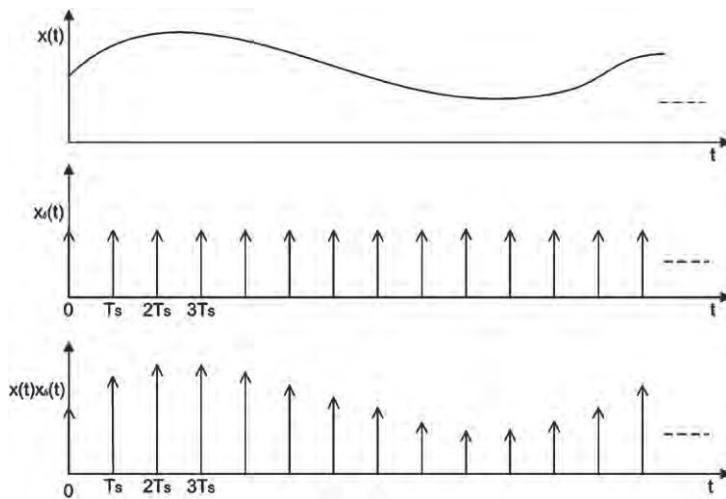
It is well known that, when the signal is bandlimited and has no frequency components above  $f_{\text{MAX}}$ , it can be uniquely described by its samples taken uniformly at frequency

$$f_s \geq 2f_{\text{MAX}}. \quad (7.1)$$

Equation (7.1) is known as Nyquist sampling criterion (or uniform sampling theorem) and the sampling frequency  $f_s = 2f_{\text{MAX}}$  is called the Nyquist sampling rate. This theorem represents a theoretically sufficient condition for reconstructing the analog signal from its uniformly spaced samples.

Even though sampling is practically implemented in a different way, it is convenient, in order to facilitate the learning process, to represent it as a product of the analog waveform,  $x(t)$ , with a periodic train of unit impulse functions defined as

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s),$$

**FIGURE 7.2**

Sampling process as a product of the analog waveform,  $x(t)$ , with a periodic train of unit impulse functions  $x_\delta(t)$ .

where  $T_s = 1/f_s$  is the sampling period. By using the shifting property of the impulse function we obtain

$$x_s(t) = x(t)x_\delta(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s). \quad (7.2)$$

A pictorial description of Eq. (7.2) is given in Figure 7.2.

For facilitating the readers' understanding of the uniform sampling effects on the bandlimited continuous time signal,  $x(t)$ , we shift from time domain to the frequency domain where we are allowed to use the properties of the Fourier transform. The product of two functions in time domain becomes the convolution of their Fourier transforms in the frequency domain. Thus, if  $X(f)$  is the Fourier transform of  $x(t)$  and  $X_\delta(f)$  is the Fourier transform of  $x_\delta(t)$ , then the Fourier transform of  $x_s(t)$  is

$$X_s(f) = X(f) * X_\delta(f),$$

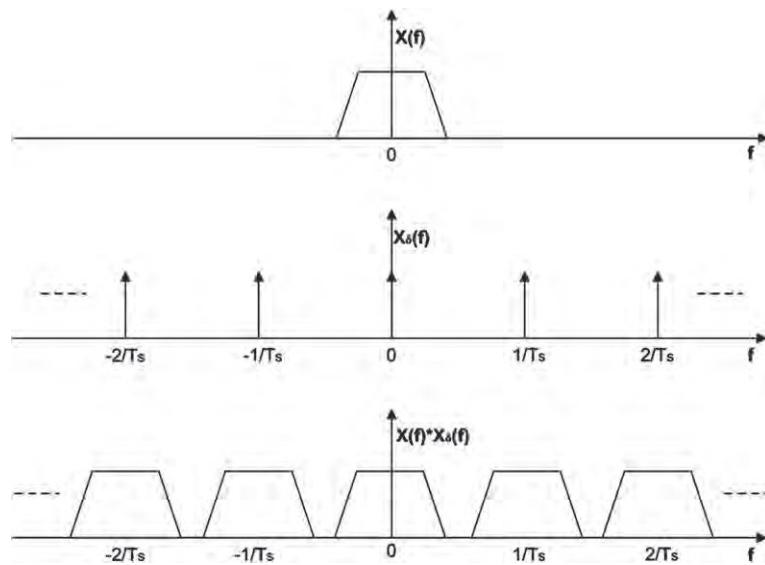
where  $*$  indicates linear convolution and

$$X_\delta(f) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} \delta(f - kf_s).$$

Note that the Fourier transform of a impulse train is another impulse train with the values of the periods reciprocally related to each other. Then in the frequency domain Eq. (7.2) becomes

$$X_s(f) = X(f) * X_\delta(f) = X(f) * \frac{1}{T_s} \sum_{k=-\infty}^{\infty} \delta(f - kf_s) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X(f - kf_s), \quad (7.3)$$

whose pictorial view is shown in Figure 7.3.

**FIGURE 7.3**

Sampling process in the frequency domain.

From Eq. (7.3) we conclude that the spectrum of the sampled signal, in the original signal bandwidth ( $[-f_s/2, f_s/2]$ ), is the same as the continuous time one (except for a scale factor,  $\frac{1}{T_s}$ ) however, as a consequence of the sampling process, this spectrum periodically repeats itself with a period of  $f_s = \frac{1}{T_s}$ . We can easily recognize that it should be possible to recover the original spectrum (associated to the spectral replica which resides in  $[-f_s/2, f_s/2]$ ), by filtering the periodically sampled signal spectrum with an appropriate low-pass filter.

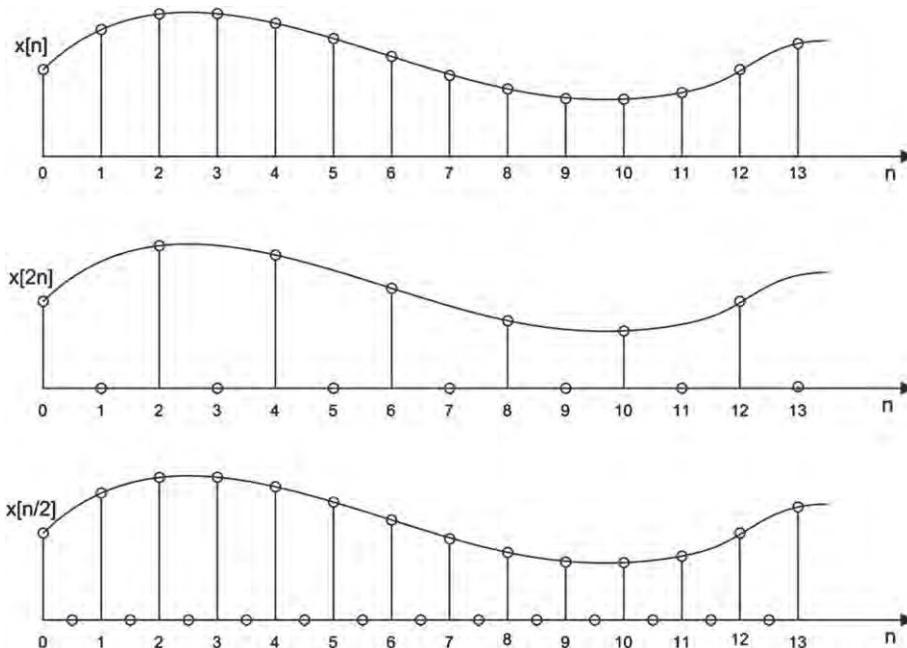
Notice, from Eq. (7.3) that the spacing,  $f_s$ , between the signal replicas is the reciprocal of the sampling period  $T_s$ . A small sampling period corresponds to a large space between the spectral replicas.

A fundamental observation, regarding the selection of the uniform sampling frequency, needs to be done at this point: when the sampling rate is selected to be less than the maximum frequency component of the bandlimited signal ( $f_s < 2f_{MAX}$ ) the periodic spectral replicas overlap each other. The amount of the overlap depends of the selected sampling frequency. Smaller the sampling frequency, larger the amount of overlap experienced by the replicas. This phenomenon is well known as aliasing. When aliasing occurs it is impossible to recover the analog signal from its samples. When  $f_s = 2f_{MAX}$  the spectral replicas touch each other without overlapping and it is theoretically (but not practically) possible to recover the analog signal from its samples; however a filter with infinite number of taps would be required. As a matter of practical consideration, we need to specify here that the signals (and filters) of interest are never perfectly bandlimited and some amount of aliasing always occurs as effect of sampling however some techniques can be used to limit the phenomena making it less harmful.

After the sampling has been applied the amplitude of each sample is one from an infinite set of possible values. This is the reason for which the samples are not compatible with a digital system yet. A digital

system can, in fact, only deal with a finite number of values. The digital samples need to be quantized before being sent to the digital data system. The quantization process limits the amplitude of the samples to a finite set of values. After the quantization process the signal can still be recovered, however some additional imprecision is added to it. The amount of imprecision depends of the quantization levels used in the process and it has the same effect on the signal as white noise. This is the reason for which it is referred to as quantization noise. The sampling of a continuous time signal and the quantization of its discrete time samples are both performed with devices called analog-to-digital converters (ADCs).

The selection of the appropriate sampling rate is a fundamental issue faced when designing digital communication systems. In order to preserve the signal, the Nyquist criterion must be always satisfied. Also, it is not difficult to find tasks for which, at some points in the digital data section of the transmitter and the receiver, it is recommended to have more than two samples per signal bandwidth. On the other hand, large sample rates cause an increase in the total workload of the system. Thus the appropriate sampling rate must be selected according to both the necessities: to have the required number of samples and to minimize the total workload. Most likely the optimum sampling frequency changes at different points in the digital architecture. It would be a huge advantage to have the option of changing the signal sampling rate at different parts in the systems while processing it, thus optimizing the number of samples as a function of the requirements. The process of changing the sampling rate of a digital signal is referred to as resampling process. The resampling process is the key concept in multirate signal processing.

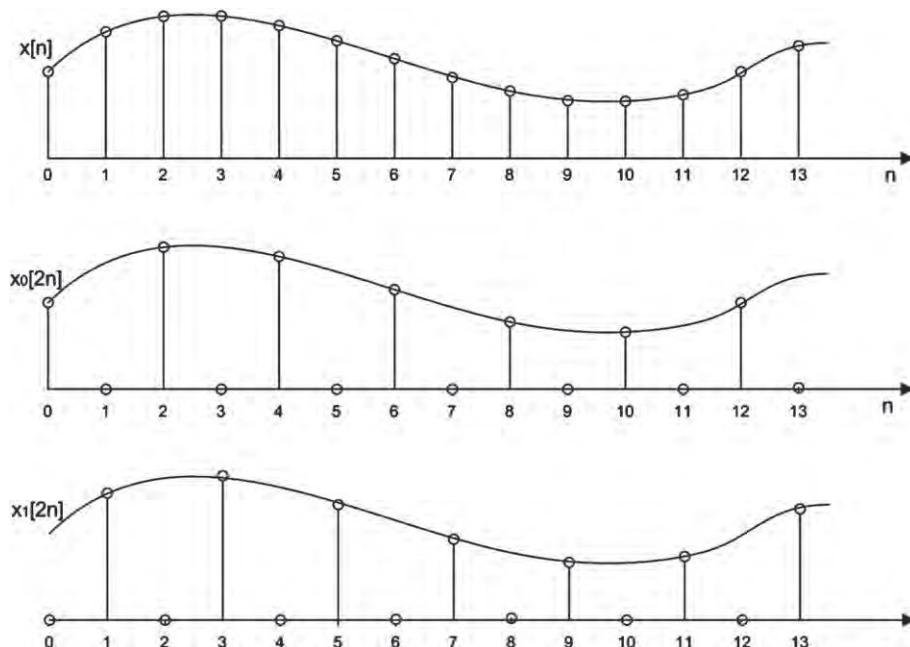


**FIGURE 7.4**

Resampling by zeroing sample values and by inserting zero valued samples.

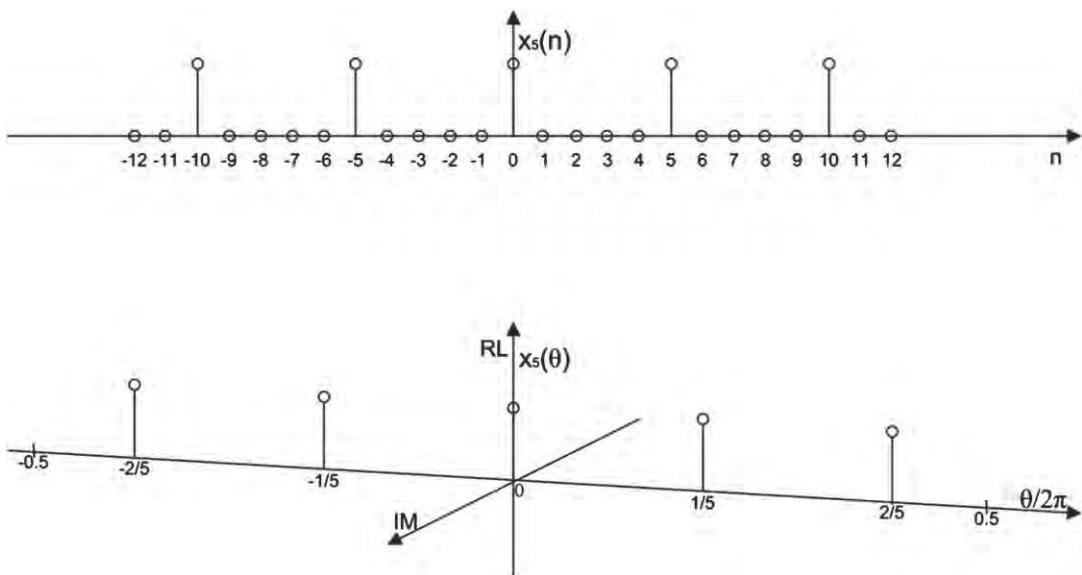
After the brief previous discussion about the sampling process of a continuous time signal, we address now the process of resampling an already sampled signal. Notice that when a continuous time signal is sampled there are no restrictions on the sample rate or the phase of the sample clock relative to the time base of the continuous time signal. On the other hand, when we resample an already sampled signal, the output sample locations are intimately related to the input sample positions. A resampled time series contains samples of the original input time series separated by a set of zero valued samples. The zero valued time samples can be the result of setting a subset of input sample values to zero or the result of inserting zeros between existing input sample values. Both options are shown in Figure 7.4. In the first example shown in this figure, the input sequence is resampled 2-to-1, keeping every second input sample starting at sample index 0 while zeroing the interim samples. In the second example, the input sequence is resampled 1-to-2, keeping every input sample but inserting a zero valued sample between each input samples. These two processes are called down sampling and up sampling respectively.

The non-zero valued samples of two sequences having the same sample rate can occur at different time indices (i.e., the same sequence can have different initial time offset) as in Figure 7.5 in which the two 2-to-1 down sampled sequences,  $s_2(n)$  and  $s_2(n - 1)$ , have different starting time index, 0 and 1, which gives equal magnitude spectral components with different phase profiles. This is explicitly shown in Figures 7.6 and 7.7, in which the time domain and the frequency domain views of the resampled sequences  $s_5(n)$  and  $s_5(n - 1)$  are depicted respectively. Remember, in fact, the time shifting property of

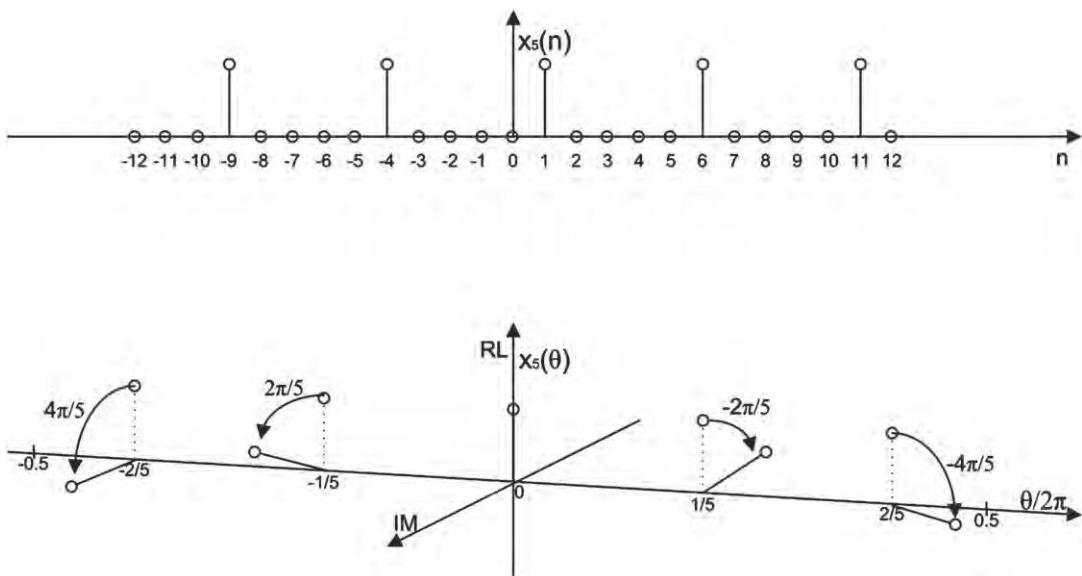


**FIGURE 7.5**

Two examples of 2-to-1 down sampling of a time series with different time offsets.

**FIGURE 7.6**

Sampling sequence  $s_5(n)$  in time and frequency domain.

**FIGURE 7.7**

Sampling sequence  $s_5(n - 1)$  in time and frequency domain.

the discrete Fourier transform (DFT) for which a shift in time domain is equivalent to a linear phase shift in the frequency domain. The different phase profiles play a central role in multirate signal processing.

The  $M$  zero valued samples inserted between the signal samples create  $M$  periodic replicas of the original spectrum at the frequency locations  $k/M$ , with  $k = 0, \dots, M-1$  in the normalized domain. This observation suggests that resampling can be used to affect translation of spectral bands, up and down conversion, without the use of sample data heterodynes.

In the following paragraphs we clarify this concept showing how to embed the spectral translation of narrowband signals in resampling filters and describe the process as aliasing.

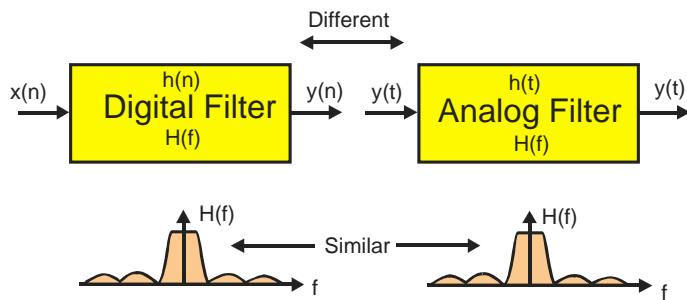
A final comment about the resampling process is that it can be applied to a time series or to the impulse response of a filter, which, of course, is simply another time series. When the resampling process is applied to a filter, the architecture of the filter changes considerably and the filter is called multirate filter.

### 1.07.3 Digital filters

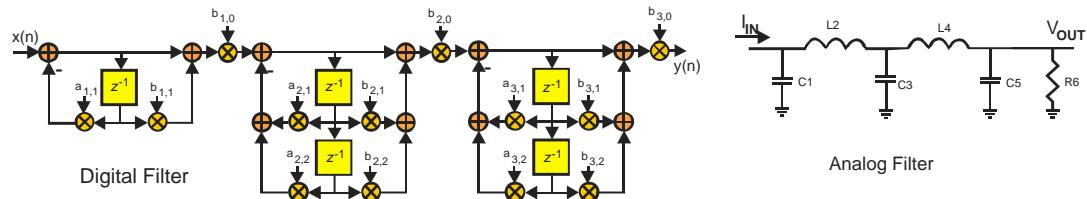
Filtering is the practice of processing signals which results in some changes of their spectral contents. Usually the change implies a reduction or filtering out some undesired input spectral components. A filter allows certain frequencies to pass while attenuating others. While analog filters operate on continuous-time signals, digital filters operate on sequences of discrete sampled value (see Figure 7.8) although digital filters perform many of the same functions as analog filters, they are different!

The two classes of filters share many of the same or analogous properties. In the standard order of our educational process we first learn about analog filters and later learn about digital filters. To ease the entry into the digital domain we emphasize the similarities of the two classes of filters. This order is due to fact that an analog filter is less abstract than a digital filter. We can touch the components of an analog filter; capacitors, inductors, resistors, operational amplifiers and wires. On the other hand a digital filter doesn't have the same physical form because a digital filter is merely a set of instructions that perform arithmetic operations on an array of numbers. The operations can be weighted sums or inner products (see Figure 7.9). It is convenient to visualize the array as a list of uniformly spaced sample values of an analog waveform. The instructions to perform these operations can reside as software in a computer or microprocessor or as firmware in a dedicated collection of interconnected hardware elements.

Let us examine the similarities that are emphasized when we learn about digital filters with some level of prior familiarity with analog filters and the tools to describe them. Many analog filters are formed by interconnections of lumped linear components modeled as ideal capacitors, inductors, and resistors while others are formed by interconnections of resistors, capacitors, and operational amplifiers. Sampled data filters are formed by interconnections of registers, adders, and multipliers. Most analog filters are designed to satisfy relationships defined by linear time invariant differential equations. The differential equations are recursive which means the initial condition time domain response, called the homogeneous or undriven response, is a weighted sum of exponentially decaying sinusoids. Most sampled data filters are designed to satisfy relationships defined by linear time invariant difference equations. Here we find the first major distinction between the analog and digital filters: the difference equations for the sampled data filters can be recursive or non recursive. When the difference equations are selected to be recursive the initial condition response is, as it was for the analog filter, samples of a

**FIGURE 7.8**

Digital and analog filters are different even though they perform similar tasks.

**FIGURE 7.9**

Digital filter: registers, adders, and multipliers. Analog filter, capacitor, inductors, and resistors.

weighted sum of exponentially decaying sinusoids. On the other hand, when the difference equation is non recursive the initial condition response is anything the designer wants it to be and is limited only by her imagination but is usually designed to satisfy some frequency domain specifications.

When we compare the two filter classes, analog and digital, we call attention to the similarities of the tools we use to describe, analyze, and design them. They are described by differential and difference equations which are weighted sums of signal derivates or weighted sums of delayed sequences. They both have linear operators or transforms, the Laplace transform  $L\{h(t)\} = H(s)$ , and the  $z$  transform  $z\{h(n)\} = H(z)$ , that offer us insight into the internal structure of the differential or difference equations (see Figure 7.10). They both have operator descriptions that perform the equivalent functions. These are the integral operator, denoted by  $s^{-1}$ , and the delay operator, denoted by  $z^{-1}$ , in which reside the system memories or state of the analog and digital filters respectively. Both systems have transfer functions  $H(s)$  and  $H(z)$ , ratios of polynomials in the operator variable  $s$  and  $z$ . The roots of the denominator polynomial, the operator version of the characteristic equation, are the filter poles. These poles describe the filter modes, the exponentially damped sinusoids, of the recursive structure. The roots of numerator polynomial are the filter zeros. The zeros describe how the filter internal mode responses are connected to the filter input and output ports. They tell us the amplitude of each response component, (called residues to impress us) in the output signal's initial condition response. The transforms of the two filter

$$\begin{aligned}
 H(z) &= \sum_{n=0}^{\infty} h(n)z^{-n} & H(s) &= \int_0^{\infty} h(t) e^{-st} dt \\
 H(z) &= b_0 \frac{z^3 + b_1 z^2 + b_2 z^1 + b_3}{z^3 + a_1 z^2 + a_2 z^1 + a_3} & H(s) &= b_0 \frac{s^3 + b_1 s^2 + b_2 s^1 + b_3}{s^3 + a_1 s^2 + a_2 s^1 + a_3} \\
 H(z) &= b_0 \frac{(z - z_1)(z - z_2)(z - z_3)}{(z - p_1)(z - p_2)(z - p_3)} & H(s) &= b_0 \frac{(s - z_1)(s - z_2)(s - z_3)}{(s - p_1)(s - p_2)(s - p_3)} \\
 H(z) &= c_0 + c_1 \frac{z}{z - p_1} + c_2 \frac{z}{z - p_2} + c_3 \frac{z}{z - p_3} & H(s) &= c_0 + c_1 \frac{p_1}{s - p_1} + c_2 \frac{p_2}{s - p_2} + c_3 \frac{p_3}{s - p_3} \\
 h(n) &= c_0 + c_1 p_1^n + c_2 p_2^n + c_3 p_3^n & h(t) &= c_0 \delta(t) + c_1 e^{-p_1 t} + c_2 e^{-p_2 t} + c_3 e^{-p_3 t}
 \end{aligned}$$

**FIGURE 7.10**

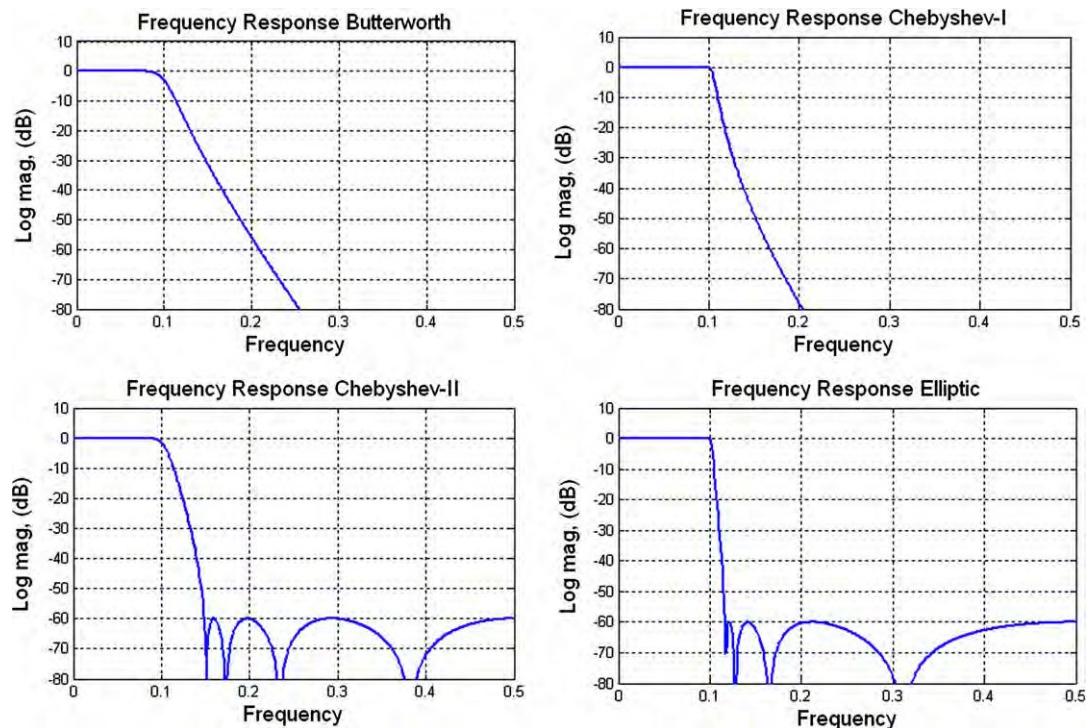
Similarity of system equations for digital and analog filters.

classes have companion transforms the Fourier transform  $H(\omega)$  and the sampled data Fourier series  $H(\theta)$  which describe, when they exist, the frequency domain behavior, sinusoidal steady state gain and phase, of the filters. It is remarkable how many similarities there are in the structure, the tools, and the time and frequency responses of the two types of filters. It is no wonder we emphasize their similarities.

We spend a great deal of effort examining recursive digital filters because of the close relationships with their analog counterparts. There are digital filters that are mappings of the traditional analog filters. These include maximally flat Butterworth filters, equal ripple pass-band Tchebychev-I filters, equal ripple stop-band Tchebychev-II filters, and both equal ripple pass-band and equal ripple stop-band Elliptic filters. The spectra of these infinite impulse response (IIR) filters are shown in Figure 7.11. All DSP filter design programs offer the DSP equivalents of their analog recursive filter counterparts.

In retrospect, emphasizing similarities between the analog filter and the equivalent digital was a poor decision. We make this claim for a number of reasons. The first is there are limited counterparts in the analog domain to the non-recursive filter of the digital domain. This is because the primary element of the non recursive filter is the pure delay, represented by the delay operator  $z^{-1}$ . We cannot form a pure delay response, represented by the analog delay operator  $e^{-sT}$ , in the analog domain as the solution of a linear differential equation. To obtain pure delay, we require a partial differential equation which offers the wave equation whose solution is a propagating wave which we use to convert distance to time delay. The propagation velocity of electromagnetic waves is too high to be of practical use in most analog filter designs but the velocity of sound waves in crystal structures enables an important class of analog filters known as acoustic surface wave (SAW) devices. Your cell phone contains one or more such filters. In general, most of us have limited familiarity with non-recursive analog filters so our first introduction to the properties and design techniques for finite duration impulse response (FIR) filters occurs in our DSP classes.

The second reason for not emphasizing the similarities between analog and digital filters is that we start believing the statement that “a digital filter is the same as an analog filter” and that all the properties of one are embedded in the other. That seemed like an ok perspective the first 10 or 20 times we heard that claim. The problem is, it is not true! The digital filter has a resource the analog filter does not have! The digital filter can perform tasks the analog filter cannot do! What is that resource? It is the sampling clock! We can do applied magic in the sampled data domain by manipulating the sample clock. There

**FIGURE 7.11**

Frequency response of digital IIR filter realizations of standard analog filters.

is no equivalent magic available in the analog filter domain! Analog designers shake their heads in disbelief when they see what a multirate filter can do. Actually digital designers also shake their heads in disbelief when they learn that manipulating the clock can accomplish such amazing things.

We might ask how can the sampling clock affect the performance or more so, enhance the capabilities of a digital filter? Great question! The answer my friend is written in the way we define frequency of a sampled data signal. Let us review how frequency as we understand it in the analog world is converted to frequency in the digital world. We now present a concise review of complex and real sinusoids with careful attention paid to their arguments.

The exponential function  $e^x$  has the interesting property that it satisfies the differential equation shown in Eq. (7.4). This means the exponential function replicates under the differential operator, an important property inherited by real and complex sinusoids. It is easy to verify that the Taylor series shown in Eq. (7.5) satisfies Eq. (7.4). When we replace the argument “ $x$ ” of Eq. (7.5) with the argument “ $j\theta$ ” we obtain the series shown in Eq. (7.6) and when we gather the terms corresponding to the even and odd powers respectively of the argument “ $j\theta$ ” we obtain the partitioned series shown in Eq. (7.7). We recognize that the two Taylor series of the partitioned Eq. (7.7) are the Taylor series of the cosine and sine which we show explicitly in Eq. (7.8). We note that the argument of both the real exponential and of

the complex exponential must be dimensionless otherwise we would not be able to sum the successive powers of the series. We also note that the units of the argument “ $\theta$ ” are radians, a dimensionless unit formed by the ratio of arc length on a circle normalized by the radius of the circle. When we replace the “ $\theta$ ” argument of Eq. (7.8) with a time varying angle “ $\theta(t)$ ” we obtain the time function shown in Eq. (7.9). The simplest time varying angle is one the changes linearly with time,  $\theta(t) = \omega_c t$ , which when substituted in Eq. (7.9) we have Eq. (7.10). All of this discussion leads us to the next statement. Since the argument of a sinusoid has the dimensionless units radians and the units of “ $t$ ” has units of seconds, then the units of  $\omega_c$  must be rad/s, a velocity. In other words, the frequency  $\omega_c$  is the time derivative of the linearly increasing phase angle  $\omega_c t$

$$\frac{d}{dx} f(x) = f(x), \quad (7.4)$$

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \dots, \quad (7.5)$$

$$e^{j\theta} = \sum_{k=0}^{\infty} \frac{(j\theta)^k}{k!} = 1 + j\theta + \frac{(j\theta)^2}{2} + \frac{(j\theta)^3}{6} + \frac{(j\theta)^4}{24} + \dots, \quad (7.6)$$

$$e^{j\theta} = \left[ 1 - \frac{\theta^2}{2} + \frac{\theta^4}{24} - \frac{\theta^6}{720} + \dots \right] + j \left[ \theta - \frac{\theta^3}{6} + \frac{\theta^5}{120} - \frac{\theta^7}{5040} + \dots \right], \quad (7.7)$$

$$e^{j\theta} = \cos(\theta) + j \sin(\theta), \quad (7.8)$$

$$s(t) = e^{j\theta(t)} = \cos(\theta(t)) + j \sin(\theta(t)), \quad (7.9)$$

$$s(t) = e^{j\omega_c t} = \cos(\omega_c t) + j \sin(\omega_c t). \quad (7.10)$$

We now examine the phase argument of the sampled data sinusoid. The successive samples of a sampled complex sinusoid are formed as shown in Eq. (7.11) by replacing the continuous argument  $t$  with the sample time positions  $nT$  as was done in Eq. (7.12). We now note that while the argument of the sampled data sinusoid is dimensionless, the independent variable is no longer “ $t$ ” with units of seconds but rather “ $n$ ” with units of sample. This is consistent with the dimension of “ $T$ ” which of course is s/sample. Consequently the product  $\omega_c T$  has units of (rad/s) · (s/smpl) or rad/smpl, thus digital frequency is rad/smpl or emphatically the angle change per sample! A more useful version of Eq. (7.12) is obtained by replacing  $\omega_c$  with  $2\pi f_c$  as done in Eq. (7.13) and then replace  $T$  with  $1/f_s$  as was done in Eq. (7.14). We finally replace  $2\pi f_c / f_s$  with  $\theta_c$  as was done in Eq. (7.15). Here we clearly see that the parameter  $\theta_c$  has units of rad/smpl, which described the fraction of the circle the argument traverses per successive sample

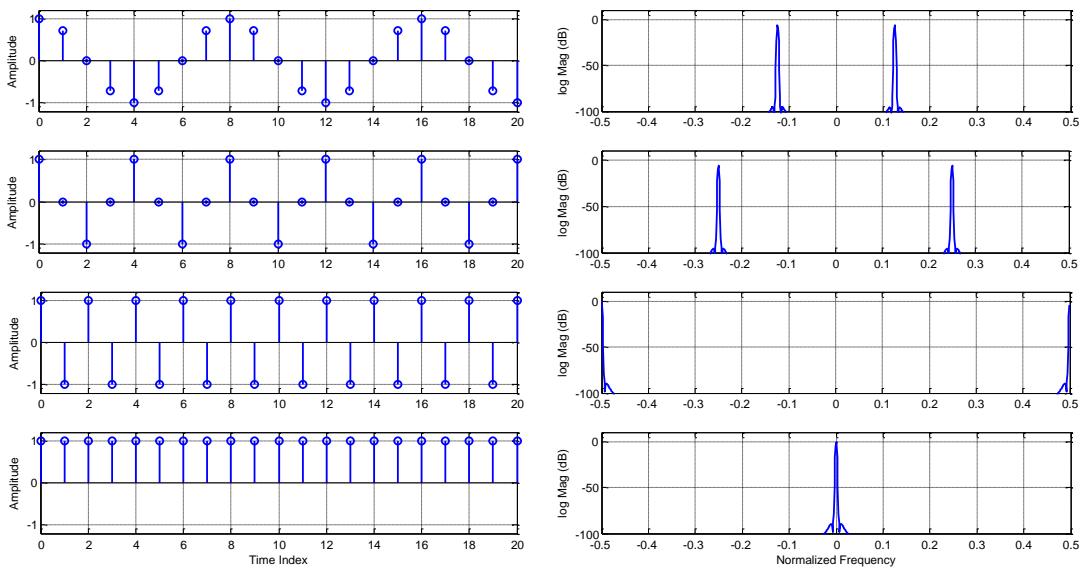
$$s(n) = s(t)|_{t=nT} = \cos(\omega_c t)|_{t=nT} + j \sin(\omega_c t)|_{t=nT}, \quad (7.11)$$

$$s(n) = \cos(\omega_c nT) + j \sin(\omega_c nT), \quad (7.12)$$

$$s(n) = \cos(2\pi f_c nT) + j \sin(2\pi f_c nT), \quad (7.13)$$

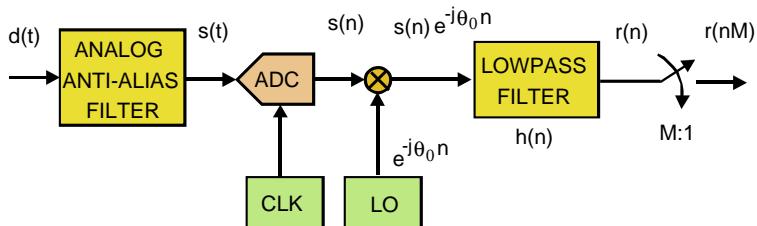
$$s(n) = \cos\left(2\pi \frac{f_c}{f_s} n\right) + j \sin\left(2\pi \frac{f_c}{f_s} n\right), \quad (7.14)$$

$$s(n) = \cos(\theta_c n) + j \sin(\theta_c n). \quad (7.15)$$

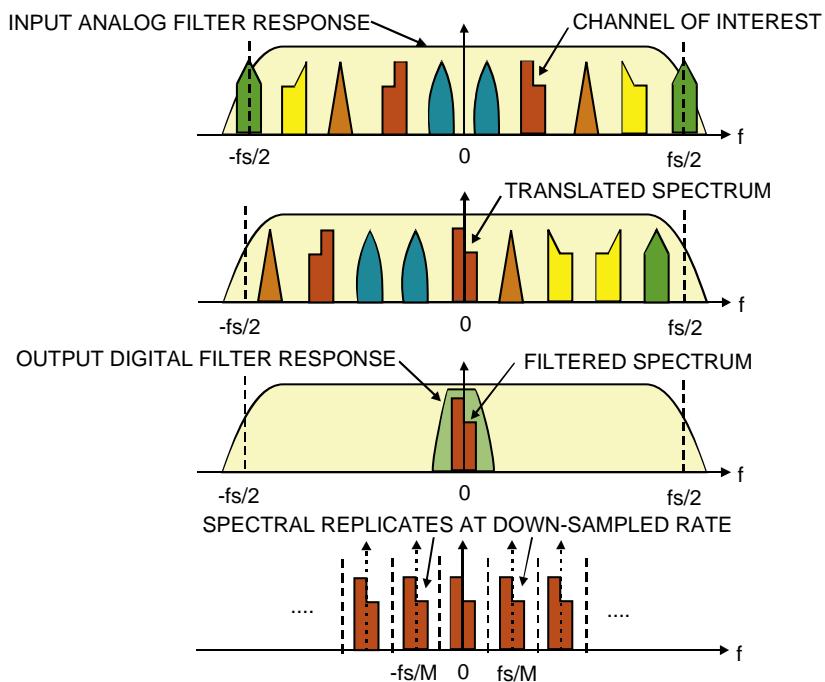
**FIGURE 7.12**

Time series and spectra of sinusoid aliased to new digital frequencies as a result of down-sampling.

Now suppose we have a sampled sinusoid with sample rate 8-times the sinusoid's center frequency so that there are 8-samples per cycle. This is equivalent to the sampled sinusoid having a digital frequency of  $2\pi/8$  rad/smpl. We can visualize a spinning phasor rotating 1/8th of the way around the unit circle per sample. Figure 7.12 subplots 1 and 2 show 21 samples of this sinusoid and its spectrum. Note the pair of spectral lines located at  $\pm 0.125$  on the normalized frequency axis. We can reduce the sample rate of this time series by taking every other sample. The spinning rate for this new sequence is  $2 \cdot 2\pi/8$  or  $2\pi/4$ . The down sampled sequence has doubled its digital frequency. The newly sampled sequence and its spectrum are shown in Figure 7.12 subplots 3 and 4. Note the pair of spectral lines are now located at  $\pm 0.25$  on the normalized frequency axis. We can further reduce the sample rate of by again taking every other sample (or every 4th sample of the original sequence). The spinning rate for this new sequence is  $4 \cdot 2\pi/8$  or  $2\pi/2$ . The down sampled sequence has again doubled its digital frequency. The newly sampled sequence and its spectrum are shown in Figure 7.12 subplots 5 and 6. Note the pair of spectral lines are now located at  $\pm 0.5$  on the normalized frequency axis. We once again reduce the sample rate of by taking every other sample (or every 8th sample of the original sequence). The spinning rate for this new sequence is  $8 \cdot 2\pi/8$  or  $2\pi$ , but since a  $2\pi$  rotation is equivalent to no rotation, the spinning phasor appears to be stationary or rotating at 0 rad/smpl. The newly sampled sequence and its spectrum are shown in Figure 7.12 subplots 7 and 8. Note the pair of spectral lines are now located at  $\pm 1$  which is equivalent to 0 on the normalized frequency axis. Frequency shifts due to sample rate changes are attributed to aliasing. Aliasing enables us to move spectrum from one frequency location to another location without the need for the complex mixers we normally use in a digital down converter to move a spectral span to base-band.

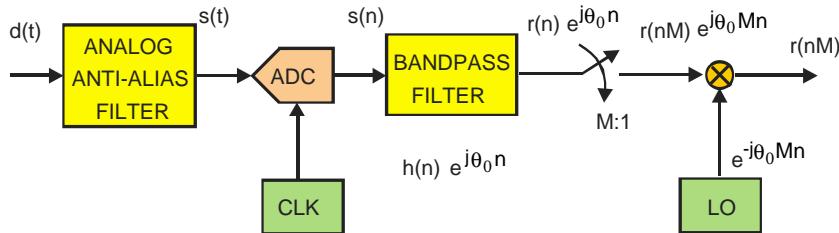
**FIGURE 7.13**

Building blocks of conventional digital down converter.

**FIGURE 7.14**

Spectra at consecutive points in conventional digital down converter.

Speaking of the standard digital down converter Figure 7.13 presents the primary signal flow blocks based on the DSP emulation of the conventional analog receiver. Figure 7.14 shows the spectra that will be observed at successive output ports of the DDC. We can follow the signal transformations of the DDC by following the spectra in the following description. The top figure shows the spectrum at the output of the analog to digital converter. The second figure shows the spectrum at the output of the quadrature mixer. Here we see the input spectrum has been shifted so that the selected channel band resides at zero

**FIGURE 7.15**

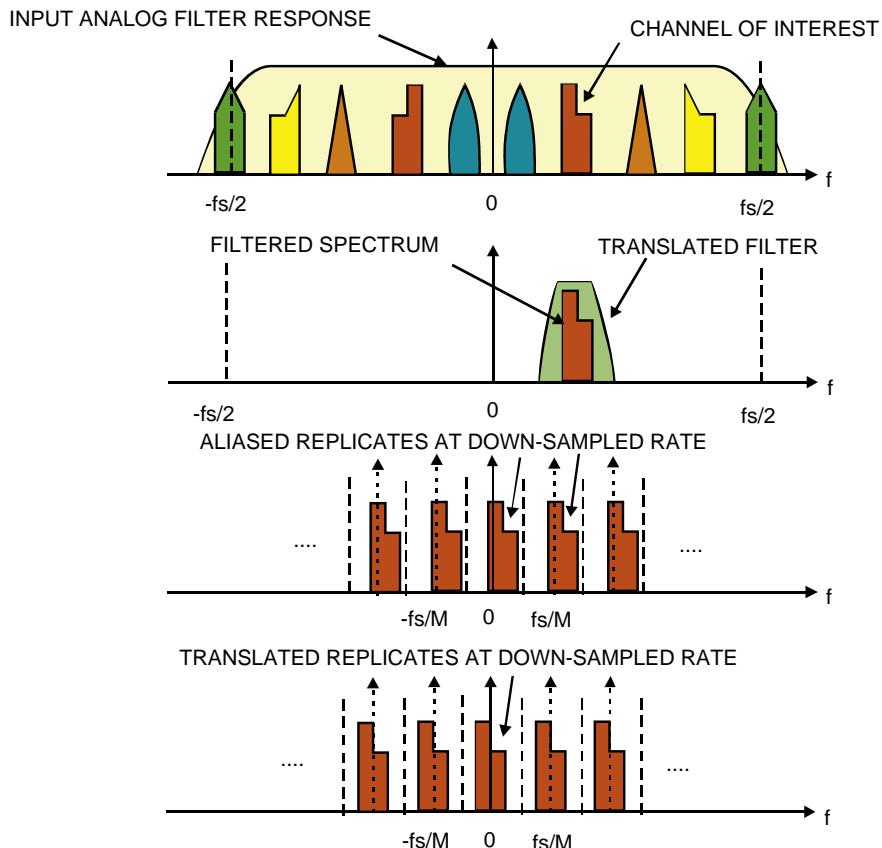
Building blocks of aliasing digital down converter with  $M$ -to-1 down sampler following filter.

frequency. The third figure shows the output of the low-pass filter pair. Here we see that a significant fraction of the input bandwidth has been rejected by the stop band of the low pass filter. Finally, the last figure shows the spectrum after the  $M$ -to-1 down sampling that reduced the sample rate in proportion to the reduction in bandwidth performed by the filter. We will always reduce sample rate if we are reducing bandwidth. We have no performance advantage for over satisfying the Nyquist criterion, but incur significant penalties by operating the DSP processors at higher than necessary sample rates.

Figure 7.15 presents the primary signal flow blocks based on aliasing the selected band pass span to base-band by reducing the output sample rate of the band pass filter. Figure 7.16 shows the spectra that will be observed at successive output ports of the aliasing DDC. We can follow the signal transformations of the aliasing DDC by following the spectra in the following description. The top figure shows the spectrum at the output of the analog to digital converter. The second figure shows the spectrum of the complex band pass filter and the output spectrum from this filter. The band pass filter is formed by up-converting, by a complex heterodyne, the coefficients of the prototype low pass filter. We note that this filter does not have a mirror image. Analog designers sigh when they see this digital option. The third figure shows the output of the band pass filter pair following the  $M$ -to-1 down sampling. If the center frequency of the band is a multiple of the output sample rate the aliased band resides at base band. For the example shown here the center frequency was  $\Delta f$  above  $k f_s/M$  and since all multiples of  $f_s/M$  alias to base-band, our selected band has aliased to  $\Delta f$  above zero frequency. Finally, the last figure shows the spectrum after the output heterodyne from  $\Delta f$  to base-band. Note that the heterodyne is applied at the low output rate rather than at the input at the high input rate as was done in Figure 7.13.

Our last section of this preface deals with the equivalency of cascade operators. Here is the core idea. Suppose we have a cascade of two filters, say a low pass filter followed by a derivative filter as shown in Figure 7.17. The filters are linear operators that commute and are distributive. We could reorder the two filters and have the same output from their cascade or we could apply the derivative filter operator to the low pass filter to form a composite filter and obtain the same output as from the cascade. This equivalency is shown in Figure 7.18. The important idea here is that we can filter the input signal and then apply an operator to the filter output or we can apply the operator to the filter and have the altered filter perform both operators to the input signal simultaneously.

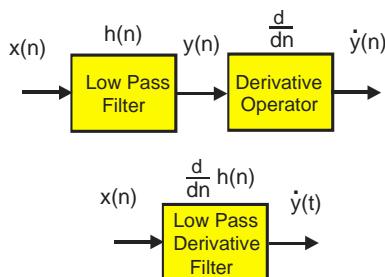
Here comes the punch-line! Let us examine the aliasing digital down converter of Figure 7.15 which contains a band pass filter followed by an  $M$ -to-1 resampler. This section is redrawn in the upper half of Figure 7.19. If we think about it, this almost silly: Here we compute one output sample for each input sample and then discard  $M-1$  of these samples in the  $M$ -to-1 down sampler. Following the lead of the

**FIGURE 7.16**

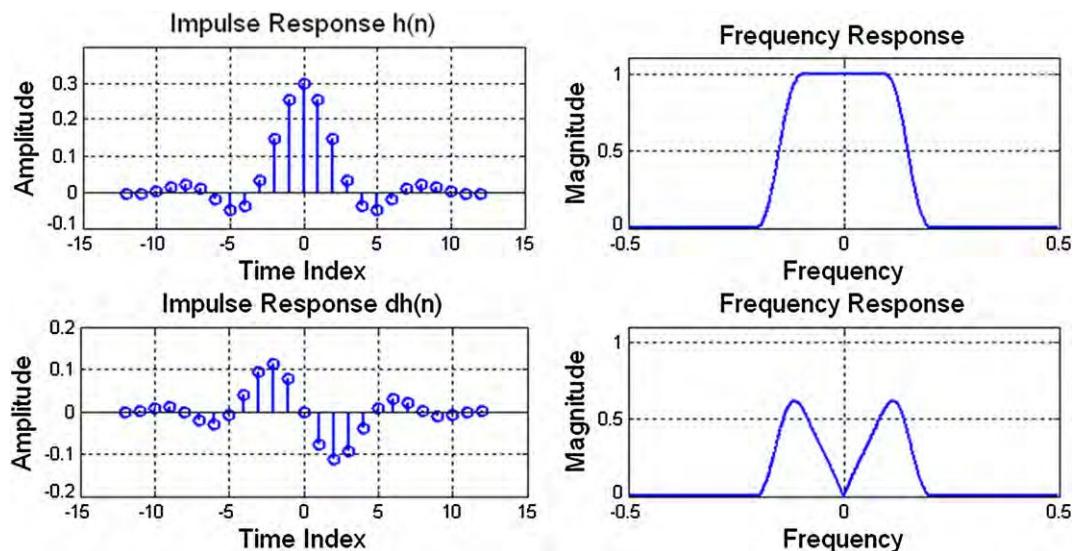
Spectra at consecutive points in aliasing digital down converter with down sampling following filter.

cascade linear operators, the filter and resampler, being equivalent to a composite filter containing the resampler applied to the filter we replace the pair of operators with the composite operator as shown in the lower half of Figure 7.19.

Embedding the  $M$ -to-1 resampler in the band pass filter is accomplished by the block diagram shown in Figure 7.20. This filter accepts  $M$  inputs, one for each path, and computes 1 output. Thus we do not waste operations computing output sample scheduled to be discarded. An interesting note here is that the complex rotators applied to the prototype low pass filter in Figure 7.15 have been factored out of each arm and are applied once at the output of each path. This means that if the input data is real, the samples are not made complex till they leave the filter. This represents a 2-to-1 savings over the architecture of Figure 7.13 in which the samples are made complex on the way into the filter which means there are 2 filters, one for each path of the complex heterodyne. The alias based DDS with

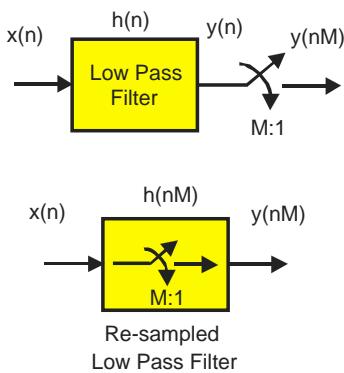
**FIGURE 7.17**

A cascade of two filters is same as a single filter formed by a composite filter.

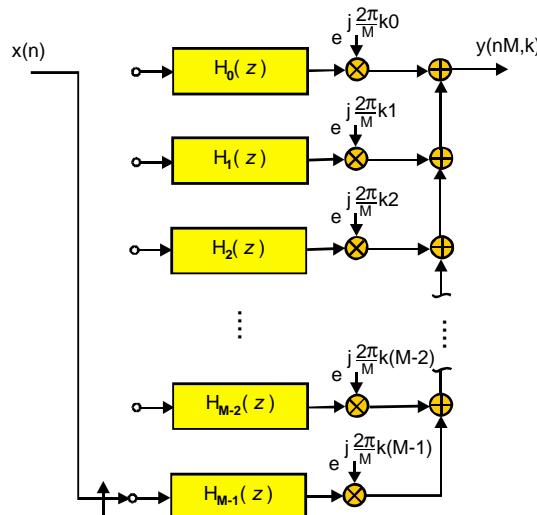
**FIGURE 7.18**

Spectrum of cascade filters is same as spectrum of single filter containing combined responses.

resampler embedded in the filter is not a bad architecture! It computes output samples at the output rate and it only uses one partitioned filter rather than 2. Not bad is understatement! The cherry on top of this dessert is the rotator vector applied to the output of the path filters. This rotator can extract any band from the aliased frequency spans that have aliased to base-band by the resampling operator. If it can extract any band then we have the option of extracting more than one band, and in fact we can extract every band from the output of the partitioned filter. That's amazing! This filter can service more than one output channel simultaneously; all it needs is additional rotator vectors. As we will see shortly, the single filter structure can service all the channels that alias to base-band and the most efficient bank of

**FIGURE 7.19**

Cascade of filter and  $M$ -to-1 resampler equivalent to  $M$ -to-1 resample filter.

**FIGURE 7.20**

$M$ -path  $M$ -to-1 resample band pass filter performs aliased digital down converter.

rotators is implemented by an  $M$ -point IFFT. More to come! Be sure to read on! We have only scratched the surface and the body of material related to multirate filters if filled with many wonderful properties and applications.

### 1.07.4 Windowing

As specified in the previous section, digital filters can be classified in many ways, starting with the most general frequency domain characteristics such as low pass, high pass, band pass, band stop and finishing with, secondary characteristics such as uniform and non-uniform group delay. We now also know that an important classification is based on the filter's architectural structure with a primary consideration being that of finite impulse response and infinite impulse response filter. Further sub-classifications, such as canonic forms, cascade forms, lattice forms are primarily driven by consideration of sensitivity to finite arithmetic, memory requirements, ability to pipeline arithmetic, and hardware constraints.

The choice to perform a given filtering task with a recursive or a non-recursive filter is driven by a number of system considerations, including processing resources, clock speed, and various filter specifications. Performance specifications, which include operating sample rate, pass band and stop band edges, pass band ripple, and out-of-band attenuation, all interact to determine the complexity required of the digital filter.

In filters with infinite impulse response each output sample depends on previous input samples and on previous filter output samples. That is the reason for which their practical implementation always requires a feedback loop. Thus, like all feedback based architectures, IIR filters are sensitive to input perturbations that could cause instability and infinite oscillations at the output. However infinite impulse response filters are usually very efficient and require far few multiplications than an FIR filter per output sample. Notice that an IIR filter could have an infinite sequence of output samples even if the input samples become all zeros. It is this characteristic which gives them their name.

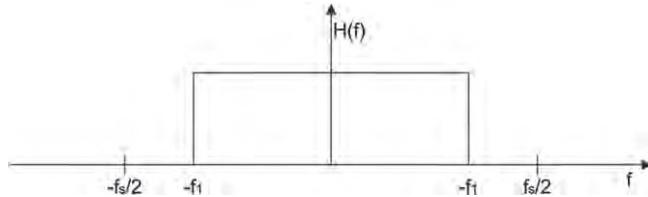
Finite impulse response filters use only current and past input samples and none of the filter's previous output samples to obtain a current output sample value (remember that they are also sometimes referred to as non-recursive filters). Given a finite duration of the input signal, the FIR filter will always have a finite duration of non-zero output samples and this is the characteristic which gives them their name.

The procedure by which the FIR filters calculate the output samples is the convolution of the input sequence with the filter impulse response which is shown in Eq. (7.16)

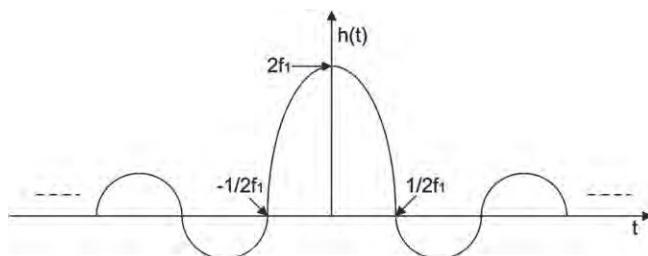
$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k), \quad (7.16)$$

where  $x(n)$  is the input sequence,  $h(n)$  is the filter impulse response and  $M$  is the number of filter taps. The convolution is nothing but a series of multiplications followed by the addition of the products while the impulse response of a filter is nothing but what the name tells us: it is the time domain view of the filter's output values when the input is an impulse (a single unity-valued sample preceded and followed by zero-valued samples).

In the following paragraphs we introduce the basic window design method for FIR filters. Because low pass filtering is the most common filtering task, we will introduce the window design method for low pass FIR filters. However the relationships between filter length and filter specifications and the interactions between filter parameters remain valid for other filter types. Many other techniques can be also used to design FIR filters and, at the end of this section, we will introduce the Remez algorithm,

**FIGURE 7.21**

Frequency response of a prototype ideal low pass filter.

**FIGURE 7.22**

Impulse response of a prototype ideal low pass filter.

which is one of the most used filter design technique, as well as one of its modified versions which allows us to achieve  $1/f$  type of decay for the out of band side-lobes.

The frequency response of a prototype low pass filter is shown in Figure 7.21. The pass band is seen to be characterized by an ideal rectangle with unity gain between the frequencies  $\pm f_1$  Hz and zero gain elsewhere.

The attraction of the ideal low pass filter  $H(f)$  as a prototype is that, from its closed form inverse Fourier transform, we achieve the exact expression for its impulse response  $h(t)$  which is shown in Eq. (7.17)

$$h(t) = 2f_1 \frac{\sin(2\pi f_1 t)}{2\pi f_1 t}. \quad (7.17)$$

The argument of the  $\sin(x)/x$  function is always the product of  $2\pi$ , half the spectral support  $(2f_1)/2$ , and the independent variable  $t$ . The numerator is periodic and becomes zero when the argument is a multiple of  $\pi$ . The impulse response of the prototype filter is shown in Figure 7.22. The  $\sin(x)/x$  filter shown in Figure 7.22 is a continuous function which we have to sample to obtain the prototype sampled data impulse response. To preserve the filter gain during the sampling process we scale the sampled function by the sample rate  $1/f_s$ . The problem with the sample set of the prototype filter is that the number of samples is unbounded and the filter is non-causal. If we had a finite number of samples, we could delay the response to make it causal and solve the problem. Then our first task is to reduce the unbounded set of filter coefficients to a finite set. The process of pruning an infinite sequence to a finite sequence is called windowing. In this process, a new limited sequence is formed as the product

of the finite sequence  $w(n)$  and the infinite sequence as shown in Eq. (7.18) where the  $\cdot^*$  operator is the standard MATLAB point-by-point multiply

$$h_w(n) = w(n) \cdot^* h(n). \quad (7.18)$$

Applying the properties of the Fourier transforms we obtain the expression for the spectrum of the windowed impulse response which is the circular convolution of the Fourier transform of  $h(n)$  and  $w(n)$ .

The symmetric rectangle we selected as our window abruptly turns off the coefficient set at its boundaries. The sampled rectangle weighting function has a spectrum described by the Dirichlet kernel which is the periodic extension of the transform of a continuous time rectangle function. The convolution between the spectra of the prototype filter with the Dirichlet kernel forms the spectrum of the rectangle windowed filter coefficient set. The contribution to the corresponding output spectrum is seen to be the stop band ripple, the transition bandwidth, and the pass band ripple. The pass band and stop band ripples are due to the side-lobes of the Dirichlet kernel moving through the pass band of the prototype filter while the transition bandwidth is due to the main lobe of the kernel moving from the stop band to the pass band of the prototype. A property of the Fourier series is that their truncated version forms a new series exhibiting the minimum mean square (MMS) approximation to the original function. We thus note that the set of coefficients obtained by a rectangle window exhibits the minimum mean square approximation to the prototype frequency response. The problem with MMS approximations in numerical analysis is that there is no mechanism to control the location or value of the error maxima. The local maximum errors are attributed to the Gibbs phenomena, the failure of the series to converge in the neighborhood of a discontinuity. These errors can be objectionably large. A process must now be invoked to control the objectionably high side-lobe levels. We have two ways to approach the problem. First we can redefine the frequency response of the prototype filter so that the amplitude discontinuities are replaced with a specified tapering in the transition bandwidth. In this process, we trade-off the transition bandwidth for side-lobe control. Equivalently, knowing that the objectionable stop band side-lobes are caused by the side-lobes in the spectrum of the window, we can replace the rectangle window with other even symmetric functions with reduced amplitude side-lobe levels. The two techniques, side-lobe control and transition-bandwidth control are tightly coupled. The easiest way to visualize control of the side-lobes is by destructive cancellation between the spectral side-lobes of the Dirichlet kernel associated with the rectangle and the spectral side-lobes of translated and scaled versions of the same kernel. The cost we incur to obtain reduced side-lobe levels is an increase in main lobe bandwidth. Remembering that the window's two-sided main lobe width is an upper bound to the filter's transition bandwidth, we can estimate the transition bandwidth of a filter required to obtain a specified side-lobe level. The primary reason we examined windows and their spectral description as weighted Dirichlet kernels was to develop a sense of how we trade the main lobe width of the window for its side-lobe levels and in turn filter transition bandwidth and side-lobe levels. Some windows perform this trade-off of bandwidth for side-lobe level very efficiently while others do not.

The Kaiser-Bessel window is very effective while the triangle (or Fejer) window is not. The Kaiser-Bessel window is in fact a family of windows parameterized over the time-bandwidth product,  $\beta$ , of the window. The main lobe width increases with  $\beta$  while the peak side-lobe level decreases with  $\beta$ . The Kaiser-Bessel window is a standard option in filter design packages such as Matlab and QED-2000.

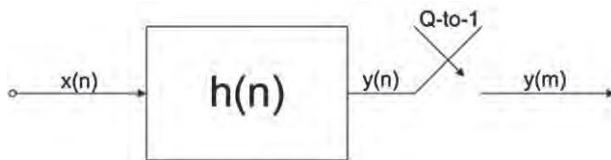
Other filter design techniques include the Remez algorithm [6], sometimes also referred to as the Parks-McClellan or P-M, the McClellan, Parks and Rabiner or MPR, the Equiripple, and the Multiple Exchange algorithm, which found large acceptance in practice. It is a very versatile algorithm capable

of designing FIR filters with various frequency responses, including multiple pass band and stop band frequency responses with independent control of ripple levels in the multiple bands. The desired pass band cut off frequencies, the frequencies where the attenuated bands begin and the desired pass band and stop band ripple are given as input parameters to the software which will generate  $N$  time domain filter coefficients.  $N$  is the minimum number of taps required for the desired filter response and it is selected by using the Harris approximation or the Hermann approximation (in Matlab). The problem with the Remez algorithm is that it shows equiripple side-lobes which is not a desirable characteristic. We would like to have a filter frequency response which has a  $1/f$  out-of-band decay rate rather than exhibit equiripple. The main reasons for desiring that characteristic are related to system performance. We often build systems comprising a digital filter and a resampling switch. Here the digital filter reduces the bandwidth and is followed by a resampling switch that reduces the output sample commensurate with the reduced output bandwidth. When the filter output is resampled, the low level energy residing in the out-of-band spectral region aliases back into the filter pass band. When the reduction in sample rate is large, there are multiple spectral regions that alias or fold into the pass band. For instance, in a 16-to-1 reduction in sample rate, there are 15 spectral regions that fold into the pass band. The energy in these bands is additive and if the spectral density in each band is equal, as it is in an equiripple design, the folded energy level is increased by a factor of  $\sqrt{15}$ . The second reason for which we may prefer FIR filters with  $1/f$  side-lobe attenuation as opposed to uniform side-lobes is finite arithmetic. A filter is defined by its coefficient set and an approximation to this filter is realized by a set of quantized coefficients. Given two filter sets  $h(n)$  and  $g(n)$ , the first with equiripple side-lobes, the second with  $1/f$  side-lobes, we form two new sets,  $h_Q(n)$  and  $g_Q(n)$ , by quantizing their coefficients. The quantization process is performed in two steps: first we rescale the filters by dividing with the peak coefficient. Second, we represent the coefficients with a fixed number of bits to obtain the quantized approximations. The zeros of an FIR filter residing on the unit circle perform the task of holding down the frequency response in the stop band. The interval between the zeros contains the spectral side-lobes. When the interval between adjacent zeros is reduced, the amplitude of the side-lobe between them is reduced and when the interval between adjacent zeros is increased the amplitude of the side-lobe between them is increased. The zeros of the filters are the roots of the polynomials  $H(z)$  and  $G(z)$ . The roots of the polynomials formed by the quantized set of coefficients differ from the roots of the non-quantized polynomials. For small changes in coefficient size, the roots exhibit small displacements along the unit circle from their nominal positions. The amplitude of some of the side-lobes must increase due to this root shift. In the equiripple design, the initial side-lobes exactly meet the designed side-lobe level with no margin for side-lobe increase due to root shift caused by coefficient quantization. On the other hand, the filter with  $1/f$  side-lobe levels has plenty of margin for side-lobe increase due to root shift caused by coefficient quantization.

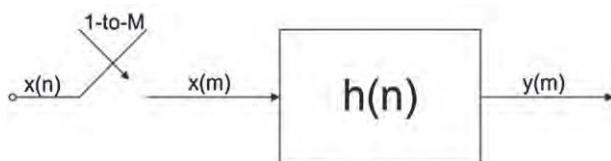
More details on the modified Remez algorithm can be found in [6].

## 1.07.5 Basics on multirate filters

Multirate filters are digital filters that contain a mechanism to increase or decrease the sample rate while processing input sampled signals. The simplest multirate filter performs integer up sampling of 1-to- $M$  or integer down sampling of  $Q$ -to-1. By extension, a multirate filter can employ both up sampling and down sampling in the same process to affect a rational ratio sample rate change of  $M$ -to- $Q$ . More sophisticated techniques exist to perform arbitrary and perhaps slowly time varying sample rate changes.

**FIGURE 7.23**

Down sampling process; filtering and sample rate reduction.

**FIGURE 7.24**

Up sampling process; sample rate increasing and filtering.

The integers  $M$  and  $Q$  may be selected to be the same so that there is no sample rate change between input and output but rather an arbitrary time shift or phase offset between input and output sample positions of the complex envelope. The sample rate change can occur at a single location in the processing chain or can be distributed over several subsections.

Conceptually, the process of down sampling can be visualized as a two-step progression indicated in Figure 7.23. There are three distinct signals associated with this procedure. The process starts with an input series  $x(n)$  that is processed by a filter  $h(n)$  to obtain the output sequence  $y(n)$  with reduced bandwidth. The sample rate of the output sequence is then reduced  $Q$ -to-1 to a rate commensurate with the reduced signal bandwidth. In reality the processes of bandwidth reduction and sample rate reduction are merged in a single process called multirate filter. The bandwidth reduction performed by the digital filter can be a low-pass or a band-pass process.

In a dual way, the process of up sampling can be visualized as a two-step process as indicated in Figure 7.24. Here too there are three distinct time series. The process starts by increasing the sample rate of an input series  $x(n)$  by resampling it 1-to- $M$ . The zero-packed time series with  $M$ -fold replication of the input spectrum is processed by a filter  $h(n)$  to reject the spectral replicas and output the sequence  $y(m)$  with the same spectrum as the input sequence but sampled at  $M$  times higher sample rate. In reality the processes of sample rate increase and selected bandwidth rejection are also merged in a single process again called multirate filtering.

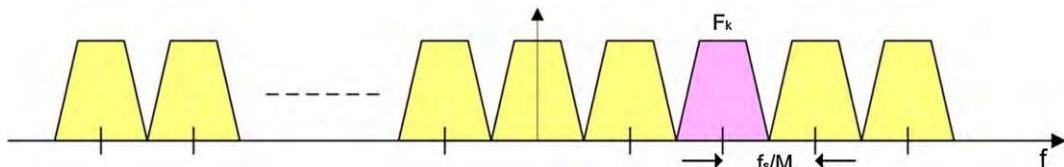
The presented structures are the most basic forms of multirate filters; in the following we present more complicated multirate architecture models. In particular we focus on the polyphase decomposition of a prototype filter which is very efficient when embedded in multirate structures. We also propose the derivation, step by step, of both the polyphase down converter and up converter channelizers. These are the two standard engines that we will modify for designing a novel polyphase channelizer that better fits the needs of the future software defined radio.

## 1.07.6 From single channel down converter to standard down converter channelizer

The derivation of the standard polyphase channelizer begins with the issue of down converting a single frequency band, or channel, located in a multi channel frequency division multiplexed (FDM) input signal whose spectrum is composed of a set of  $M$  equally spaced, equal bandwidth channels, as shown in Figure 7.25. Note that this signal has been band limited by analog filters and has been sampled at a sufficiently high sample rate to satisfy the Nyquist criterion for the full FDM bandwidth.

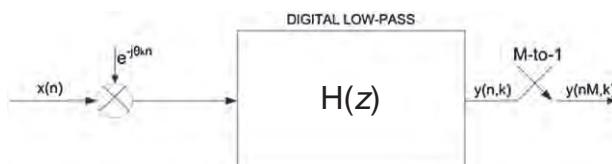
We have many options available for down converting a single channel; The standard processing chain for accomplishing this task is shown in Figure 7.26. This structure performs the standard operations of down converting a selected frequency band with a complex heterodyne, low pass filtering to reduce the output signal bandwidth to the channel bandwidth, and down sampling to a reduced rate commensurate with the reduced bandwidth. The structure of this processor is seen to be a digital signal processor implementation of a prototype analog  $I$ - $Q$  down converter. We mention that the down sampler is commonly referred to as a decimator, a term that means to destroy one sample every tenth. Since nothing is destroyed and nothing happens in tenths, we prefer, and will continue to use, the more descriptive name, down sampler.

The output data from the complex mixer is complex, hence it is represented by two time series,  $I(n)$  and  $Q(n)$ . The filter with real impulse response  $h(n)$  is implemented as two identical filters, each processing one of the quadrature time series. The convolution process between a signal and a filter is often performed by simply multiply and sum operations between signal data samples and filter coefficients extracted from two sets of addressed memory registers. In this form of the filter, one register set contains the data samples while the other contains the coefficients that define the filter impulse response.



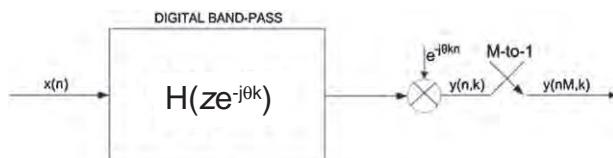
**FIGURE 7.25**

Spectrum of multichannel input signal, processing task: extract complex envelope of selected channel.



**FIGURE 7.26**

Standard single channel down converter.

**FIGURE 7.27**

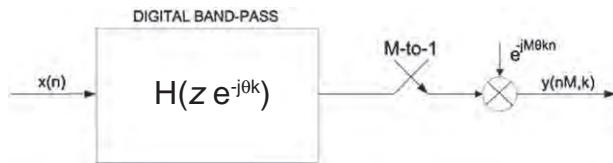
Band-pass filter version of single channel down converter.

By using the equivalency theorem, which states that *the operations of down conversion followed by a low-pass filter are equivalent to the operations of band-pass filtering followed by a down conversion*, we can exchange the positions of the filter and of the complex heterodyne achieving the block diagram shown in Figure 7.27.

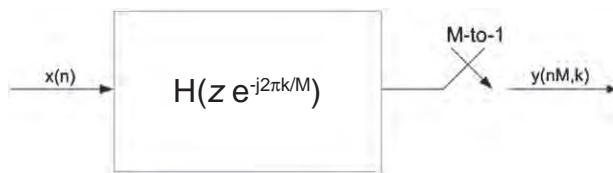
Note here that the up converted filter,  $h(n) \exp(j\theta_k n)$ , is complex and as such its spectrum resides only on the positive frequency axis without a negative frequency image. This is not a common structure for an analog prototype because of the difficulty of forming a pair of analog quadrature filters exhibiting a  $90^\circ$  phase difference across the filter bandwidth. The closest equivalent structure in the analog world is the filter pair used in image-reject mixers and even there, the phase relationship is maintained by a pair of complex heterodynes.

Applying the transformation suggested by the equivalency theorem to an analog prototype system does not make sense since it doubles the required hardware. We would have to replace a complex scalar heterodyne (two mixers) and a pair of low-pass filters with a pair of band-pass filters, containing twice the number of reactive components, and a full complex heterodyne (four mixers). If it makes no sense to use this relationship in the analog domain, why does it make sense in the digital world? The answer is found in the fact that we define a digital filter as a set of weights stored in the coefficient memory. Thus, in the digital world, we incur no cost in replacing the pair of low-pass filters  $h(n)$  required in the first option with the pair of band-pass filters  $h(n) \cos(n\theta_k)$  and  $h(n) \sin(n\theta_k)$  required for the second one. We accomplish this task by a simple download to the coefficient memory.

An interesting historical perspective is worth noting here. In the early days of wireless, radio tuning was accomplished by sliding a narrow band filter to the center frequency of the desired channel to be extracted from the FDM input signal. These radios were known as tuned radio frequency (TRF) receivers. The numerous knobs on the face of early radios adjusted reactive components of the amplifier chain to accomplish the tuning process. Besides the difficulty in aligning multiple tuned stages, shifting the center frequency of the amplifier chain often initiated undesired oscillation due to the parasitic coupling between the components in the radio. Edwin Howard Armstrong, an early radio pioneer, suggested moving the selected channel to the fixed frequency filter rather than moving the filter to the selected channel. This is known as the *superheterodyne principle*, a process invented by Armstrong in 1918 and quickly adopted by David Sarnoff of the Radio Corporation of America (RCA) in 1924. Acquiring exclusive rights to Armstrong's single-tuning dial radio invention assured the commercial dominance of RCA in radio broadcasting as well the demise of hundreds of manufacturers of TRF radio receivers. It seems we have come full circle. We inherited from Armstrong a directive to move the desired spectrum to the filter and we have readily applied this legacy to DSP-based processing. We are now proposing that, under appropriate conditions, it makes more sense to move the filter to the selected spectral region.

**FIGURE 7.28**

Down sampled band-pass down converter.

**FIGURE 7.29**

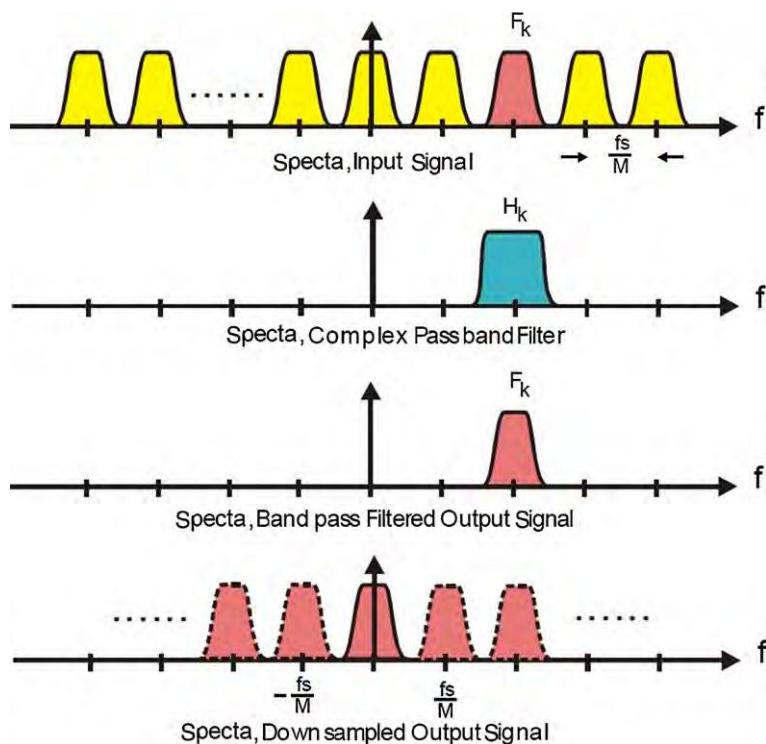
Band-pass down converter aliased to base-band by down sampler.

We still have to justify the full complex heterodyne required for the down conversion at the filter output rather than at the filter input, which is done in the subsequent paragraphs.

Examining Figure 7.27, we note that following the output down conversion, we perform a sample rate reduction in which we retain one sample out of  $M$ -samples. Because we find it useless to down convert the samples we discard in the next down sample operation, we move the down sampler before the down converter achieving the demodulator scheme shown Figure 7.28. We note in this figure that also the time series of the complex sinusoid has been down sampled. The rotation rate of the sampled complex sinusoid is  $\theta_k$  and  $M\theta_k$  radians per sample at the input and output of the  $M$ -to-1 resampler respectively. This change in observed rotation rate is due to aliasing. When aliased, a sinusoid at one frequency or phase slope appears at another phase slope due to the resampling.

We now invoke a constraint on the sampled data center frequency of the down converted channel. We choose center frequencies  $\theta_k$ , which will alias to DC as a result of down sampling to  $M\theta_k$ . This condition is assured if  $M\theta_k$  is congruent to  $2\pi$ , which occurs when  $M\theta_k = k2\pi$ , or more specifically, when  $\theta_k = k2\pi/M$ . The modification to Figure 7.28 to reflect this provision is seen in Figure 7.29. The constraint, that the center frequencies be limited to integer multiples of the output sample rate, assures aliasing to base-band by the sample rate change. When a channel aliases to base-band because the resampling operation the corresponding resampled heterodyne defaults to a unity-valued scalar, which consequently is removed from the signal processing path.

Note that if the center frequency of the aliased signal is offset by  $\Delta_\theta$  rad/smpl from a multiple of the output sample rate, the aliased signal will reside at an offset of  $\Delta_\theta$  rad/smpl from zero frequency at base-band and a complex heterodyne, or base-band converter, is needed to shift the signal by the residual  $\Delta_\theta$  offset. This base-band mixer operates at the output sample rate rather than at the input sample rate for a conventional down converter. We can consider this required final mixing operation as a post conversion task and allocate it to the next processing block.

**FIGURE 7.30**

Spectrum, down sampled output signal.

The spectral effect of the signal processing performed by the structure in Figure 7.29 is shown in Figure 7.30. The savings realized by this form of down conversion is due to the fact that we no longer require a quadrature oscillator or the pair of input mixers to effect the required frequency translation.

Applying again the idea that it is useless to filter those samples that will be later discarded by the down sampler we apply the noble identity to exchange the operations of filtering and down sampling. The noble identity states that *a filter processing every Mth input sample followed by an output M-to-1 down sampler is the same as an input M-to-1 down sampler followed by a filter processing every input sample*. Its interpretation is that the M-to-1 down sampled time series from a filter processing every  $M$ th input sample presents the same output by first down sampling the input by  $M$ -to-1, to discard the samples not used by the filter when computing the retained output samples, and then operating the filter on only the retained input samples. The noble identity works because samples of delay at the input clock rate is the same interval as one-sample delay at the output clock rate.

At this point, with only a filter followed by a resampler, we can apply the polyphase decomposition to the filter and separate it into  $M$ -parallel paths for achieving the standard  $M$ -path polyphase down

converter channelizer. In order to make the understanding process easier for the reader, we first perform the polyphase decomposition of a low-pass prototype filter and then we extend the results to the band-pass filter case. Equation (7.19) represents the  $z$  transform of a digital low-pass prototype filter  $h(n)$

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n} = h(0) + h(1)z^{-1} + h(2)z^{-2} + \cdots + h(N-1)z^{-(N-1)}. \quad (7.19)$$

In order to achieve the polyphase partition of  $H(z)$ , we rewrite the sum in Eq. (7.19), as shown in Eq. (7.20), partitioning the one-dimensional array of weights in a two-dimensional array. In this mapping we load an array by columns but process it by rows. The partition forms columns of length  $M$  containing  $M$  successive terms in the original array, and continues to form adjacent  $M$ -length columns until we account for all the elements of the original one-dimensional array

$$\begin{aligned} H(z) = & h(0) + h(M+0)z^{-M} + h(2M+0)z^{-2M} + \cdots \\ & h(1)z^{-1} + h(M+1)z^{-(M+1)} + h(2M+1)z^{-(2M+1)} + \cdots \\ & h(2)z^{-2} + h(M+2)z^{-(M+2)} + h(2M+2)z^{-(2M+2)} + \cdots \\ & \vdots \\ & h(M-1)z^{-(M-1)} + h(2M-1)z^{-(2M-1)} + h(3M-1)z^{-(3M-1)} + \cdots \end{aligned} \quad (7.20)$$

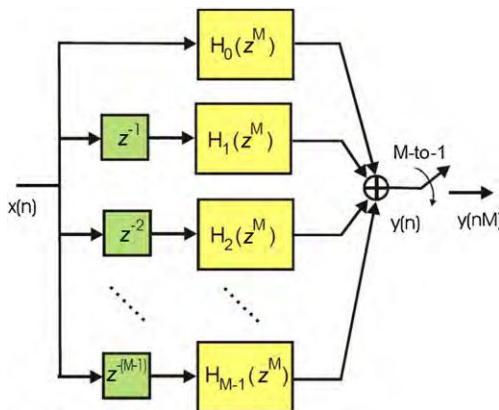
We note that the first row of the two-dimensional array is a polynomial in  $z^M$ , which we will denote  $H_0(z^M)$  a notation to be interpreted as an addressing scheme to start at index 0 and increment in strides of length  $M$ . The second row of the same array, while not a polynomial in  $z^M$ , is made into one by factoring the common term  $z^{-1}$  and then identifying this row as  $z^{-1}H_1(z^M)$ . It is easy to see that each row of the two-dimensional array described by Eq. (7.8) can be written as  $z^{-r}H_r(z^M)$  achieving the more compact form as shown in Eq. (7.9) where  $r$  is the row index which is coincident with the path index

$$H(z) = \sum_{r=0}^{M-1} z^{-r} H_r(z^M) = \sum_{r=0}^{M-1} z^{-r} \sum_{n=0}^{\frac{N}{M}-1} h(r+nM)z^{-nM}. \quad (7.21)$$

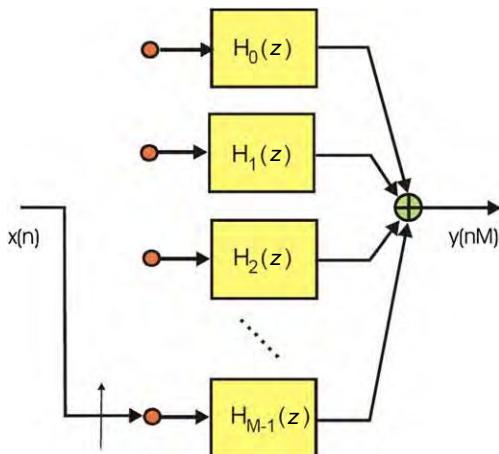
The block diagram depicting the  $M$ -path polyphase decomposition of the resampled low-pass filter of Eq. (7.21) is depicted in Figure 7.31.

At this point, in the structure depicted in Figure 7.31, we can pull the resampler on the left side of the adder and down sample the separate filter outputs performing the sum only for the retained filter output samples. With the resamplers at the output of each filter we can invoke again the noble identity and place them at the input of each filter.

The resamplers operate synchronously, all closing at the same clock cycle. When the switches are closed, the signal delivered to the filter on the top path is the current input sample. The signal delivered to the filter one path down is the content of the one stage delay line, which, of course, is the previous input sample. Similarly, as we traverse the successive paths of the  $M$ -path partition, we find upon switch closure, that the  $k$ th path receives a data sample delivered  $k$  samples ago. We conclude that the interaction of the delay lines in each path with the set of synchronous switches can be likened to an input

**FIGURE 7.31**

*M*-path partition of prototype low-pass filter with output resampler.

**FIGURE 7.32**

*M*-path partition of prototype low-pass filter with input delays and *M*-to-1 resamples replaced by input commutator.

commutator that delivers successive samples to successive arms of the *M*-path filter. This interpretation is shown in Figure 7.32 which depicts the final structure of the polyphase decomposition of a low-pass prototype filter.

At this point we can easily achieve the band-pass polyphase partition from the low-pass partition by applying the frequency translation property of the *z*-Transform that states the following.

If

$$H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + \cdots + h(N-1)z^{N-1} = \sum_{n=0}^{N-1} h(n)z^{-n}$$

and

$$\begin{aligned} G(z) &= h(0) + h(1)e^{j\theta}z^{-1} + h(2)e^{j2\theta}z^{-2} + \cdots + h(N-1)e^{j(N-1)\theta}z^{N-1} \\ &= h(0) + h(1) \left[ e^{-j\theta}z \right]^{-1} + h(2) \left[ e^{-j\theta}z \right]^{-2} + \cdots + h(N-1) \left[ e^{-j(N-1)\theta}z \right]^{-(N-1)} \\ &= \sum_{n=0}^{N-1} h(n) \left[ e^{-j\theta}z \right]^{-n}, \end{aligned}$$

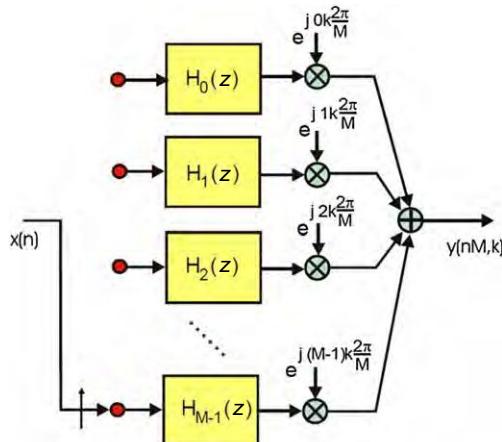
then

$$G(z) = H(z)|_{z=e^{-j\theta}z} = H \left( e^{j\theta}z \right).$$

By using this property and replacing in Eq. (7.21) each  $z^{-r}$  with  $z^{-r}e^{jr\theta}$ , where  $\theta = k \frac{2\pi}{M}$ , we achieve

$$H(ze^{-j(2\pi k/M)}) = \sum_{r=0}^{M-1} z^{-r} e^{jr(2\pi k/M)} H_r(z^M). \quad (7.22)$$

The complex scalars  $e^{jr(\frac{2\pi k}{M})}$  attached to each path of the  $M$ -path filter can be placed anywhere along the path. We choose to place them after the down sampled path filter segments  $H_r(z^M)$ . This change is shown in Figure 7.33 that depicts the standard polyphase filter bank used for down converting a single channel. The computation of the time series obtained from the output summation in Figure 7.33 is shown



**FIGURE 7.33**

Resampling  $M$ -path down converter.

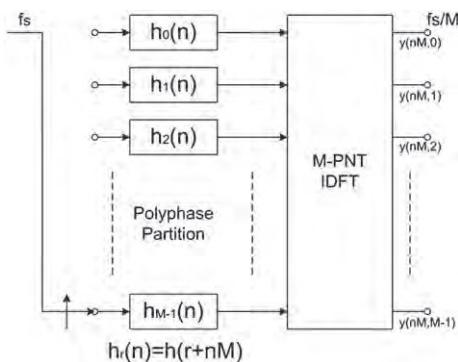
in Eq. (7.23). Here, the argument  $nM$  reflects the down sampling operation, which increments through the time index in strides of length  $M$ , delivering every  $M$ th sample of the original output series. The variable  $y_r(nM)$  is the  $nM$ th sample from the filter segment in the  $r$ th path, and  $y(nM, k)$  is the  $nM$ th time sample of the time series from the  $r$ th center frequency. Remember that the down converted center frequencies located at integer multiples of the output sample frequency alias to zero frequency after the resampling operation. Note that the output  $y(nM, k)$  is computed as a phase coherent summation of the output series  $y_r(nM)$ . This phase coherent sum is, in fact, an inverse discrete Fourier transform of the  $M$ -path outputs, which can be likened to beam forming the output of the path filters

$$y(nM, k) = \sum_{r=1}^{M-1} y_r(nM) e^{j(2\pi/M)rk}. \quad (7.23)$$

The beam forming perspective offers an interesting insight to the operation of the resampled down converter system we have just examined; the reasoning proceeds as follows: the commutator delivering consecutive samples to the  $M$  input ports of the  $M$ -path filter performs a down sampling operation. Each port of the  $M$ -path filter receives data at  $1/M$ th of the input rate. The down sampling causes the  $M$ -to-1 spectral folding, effectively translating the  $M$  multiples of the output sample rate to base-band. The alias terms in each path of the  $M$ -path filter exhibit unique phase profiles due to their distinct center frequencies and the time offsets of the different down sampled time series delivered to each port. These time offsets are, in fact, the input delays shown in Figure 7.31 and Eq. (7.3). Each of the aliased center frequency experiences a phase shift shown in Eq. (7.24) equal to the product of its center frequency and the path time delay.

$$\varphi(r, k) = w_k \Delta T_r = 2\pi \frac{f_s}{M} kr T_s = 2\pi \frac{f_s}{M} kr \frac{1}{f_s} = \frac{2\pi}{M} kr. \quad (7.24)$$

The phase shifters of the IDFT perform phase coherent summations, very much like that performed in narrowband beam forming, extracting from the myriad of aliased time series, the alias with the particular matching phase profile. This phase sensitive summation aligns contributions from the desired alias to realize the processing gain of the coherent sum while the remaining alias terms, which exhibit rotation rates corresponding to the  $M$  roots of unity, are destructively canceled in the summation. The inputs to the  $M$ -path filter are not narrowband, and phase shift alone is insufficient to cause the destructive cancellation over the full bandwidth of the undesired spectral contributions. Continuing with our beam-forming perspective, to successfully separate signals with unique phase profiles due to the input commutator delays, we must perform the equivalent of time-delay beam forming. The  $M$ -path filters, obtained by  $M$ -to-1 down sampling of the prototype low-pass filter supply the required time delays. The  $M$ -path filters are approximations to all-pass filters, exhibiting, over the channel bandwidth, equal ripple approximation to unity gain and the set of linear phase shifts that provide the time delays required for the time-delay beam-forming task. The filter achieves this property by virtue of the way it is partitioned. Each of the  $M$ -path filters,  $h_r(n)$ , for instance, with weights  $h_r(r + nM)$ , is formed by starting with an initial offset of  $r$  samples and then incrementing in strides of  $M$  samples. The initial offsets, unique to each path, are the source of the different linear phase-shift profiles. It is because of the different linear phase profiles, that the filter partition is known as *polyphase* filter. An useful perspective is that the phase rotators following the filters perform phase alignment of the band center for each aliased spectral band while the polyphase filters perform the required differential phase shift across these same channel bandwidths.

**FIGURE 7.34**

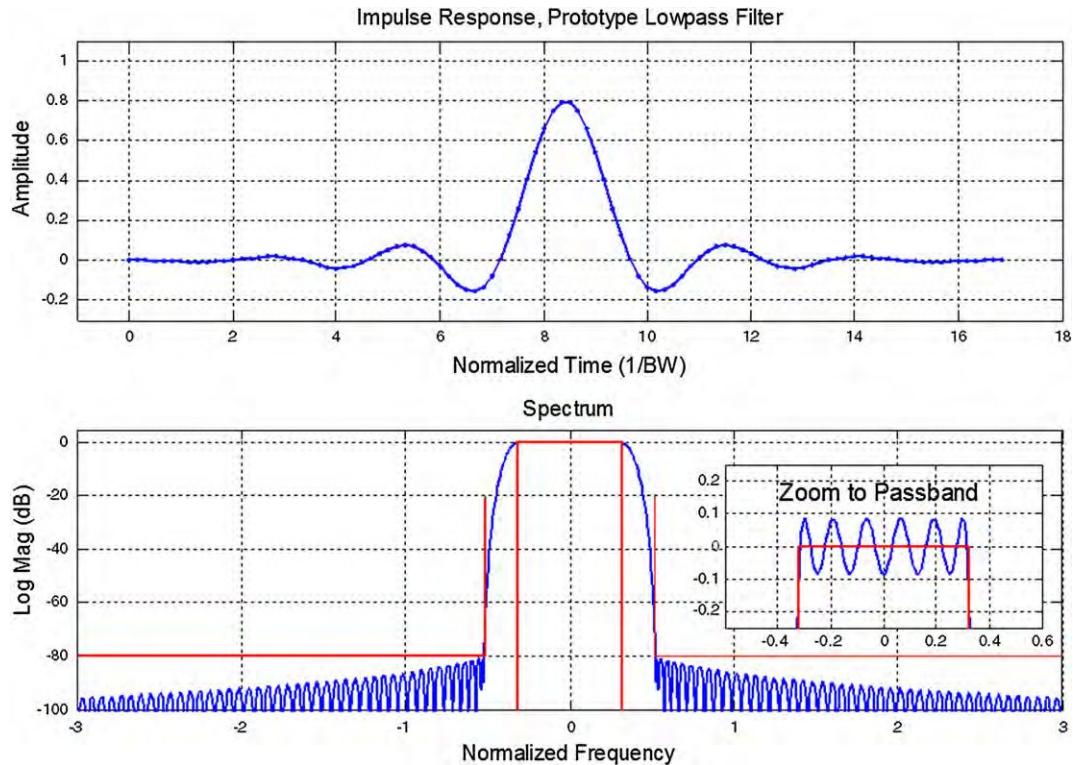
Standard polyphase down converter channelizer: commutator,  $M$ -path polyphase filter and  $M$ -point IFFT.

When the polyphase filter is used to down convert and down sample a single channel, the phase rotators are implemented as external complex products following each path filter. When a small number of channels are being down converted and down sampled, appropriate sets of phase rotators can be applied to the filter stage outputs and summed to form each channel output. Therefore the most advantageous applications of that structure are those in which the number of channels to be down converted becomes sufficiently large. Sufficiently large means on the order of  $\log_2(N)$ . Since the phase rotators following the polyphase filter stages are the same as the phase rotators of an IDFT, we can use the IDFT to simultaneously apply the phase shifters for all the channels we wish to extract from the aliased signal set. This is reminiscent of phased-array beam forming. For computational efficiency, the IFFT algorithm implements the IDFT.

The complete structure of a standard  $M$ -path polyphase down converter channelizer is shown in Figure 7.34.

To summarize: in this structure the commutator performs an input sample rate reduction by commuting successive input samples to selected paths of the  $M$ -path filter. Sample rate reduction occurring prior to any signal processing causes spectral regions residing at multiples of the output sample rate to alias to base-band. The partitioned  $M$ -path filter performs the task of aligning the time origins of the offset sampled data sequences delivered by the input commutator to a single common output time origin. This is accomplished by the all-pass characteristics of the  $M$ -path filter sections that apply the required differential time delay to the individual input time series. The IDFT performs the equivalent of a beam-forming operation; the coherent summation of the time-aligned signals at each output port with selected phase profiles. The phase coherent summation of the outputs of the  $M$ -path filters separates the various aliases residing in each path by constructively summing the selected aliased frequency components located in each path, while simultaneously destructively canceling the remaining aliased spectral components.

We conclude the reasoning on the standard  $M$ -path down converter channelizer by stating that it simultaneously performs the following three basic operations: sample rate reduction, due to the input commutator; bandwidth reduction, due to the  $M$ -path partitioned filter weights and Nyquist zone selection, due to the IFFT block.

**FIGURE 7.35**

Impulse response and frequency response of prototype low-pass filter.

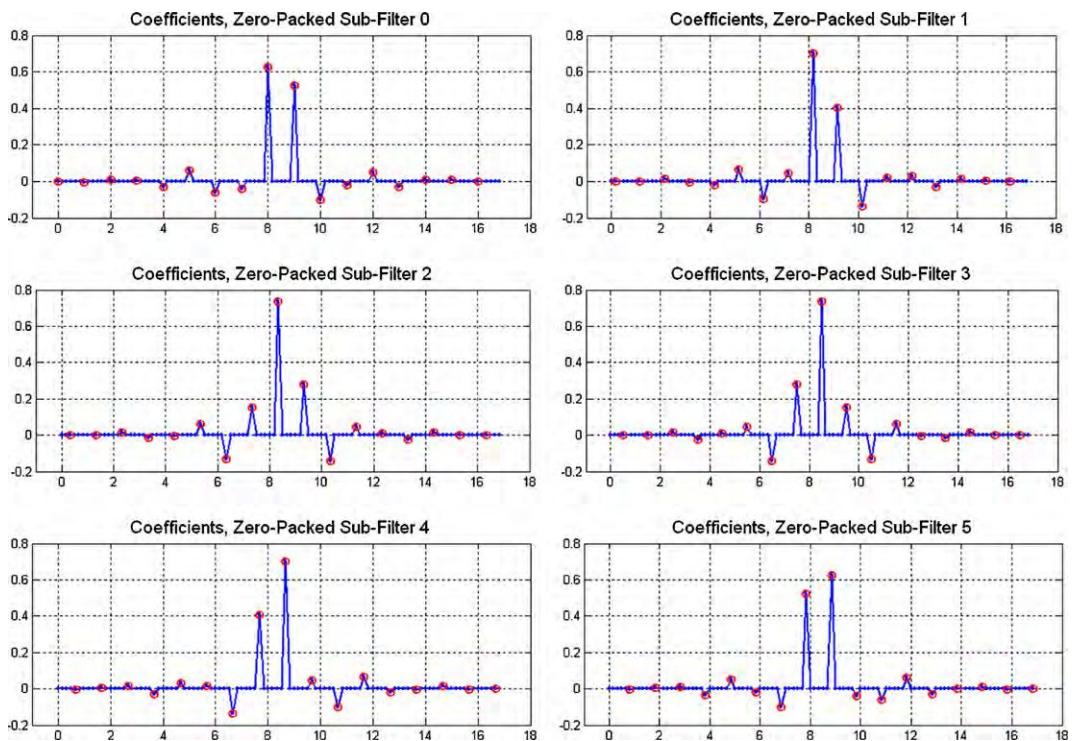
In the following, in order to facilitate the understanding process for the reader, we show some figures, which are results of Matlab simulation, to support the theoretical reasoning of the previous paragraphs. The example concerns a 6-path down converter channelizer that performs 6-to-1 down sampling. In particular Figure 7.35 shows the impulse response and the frequency response of the low-pass prototype filter used for the polyphase partition.

Figure 7.36 shows the impulse responses of the zero-packed filter on each arm of the 6-path polyphase partition while Figure 7.37 shows the spectra corresponding to the filters of Figure 7.36. It is evident from this figure that the zero packing process has the effect of producing spectral copies of the filter frequency response.

In Figure 7.38 we show the phase of each spectral copy of the zero packed prototype filter on each arm of the partition. In this figure the phases of the spectral copies of each arm are overlaid and each line of the plot corresponds to a specific arm in the partition.

In Figure 7.39 the same phase profiles of Figure 7.38 are de-trended, zoomed, and made causal.

In Figure 7.40 the 3-D view of the filter spectral copies, with their phase profiles, is shown for each path, while, in Figure 7.41, the final result of the channelization process is shown, again in a 3-D fashion.

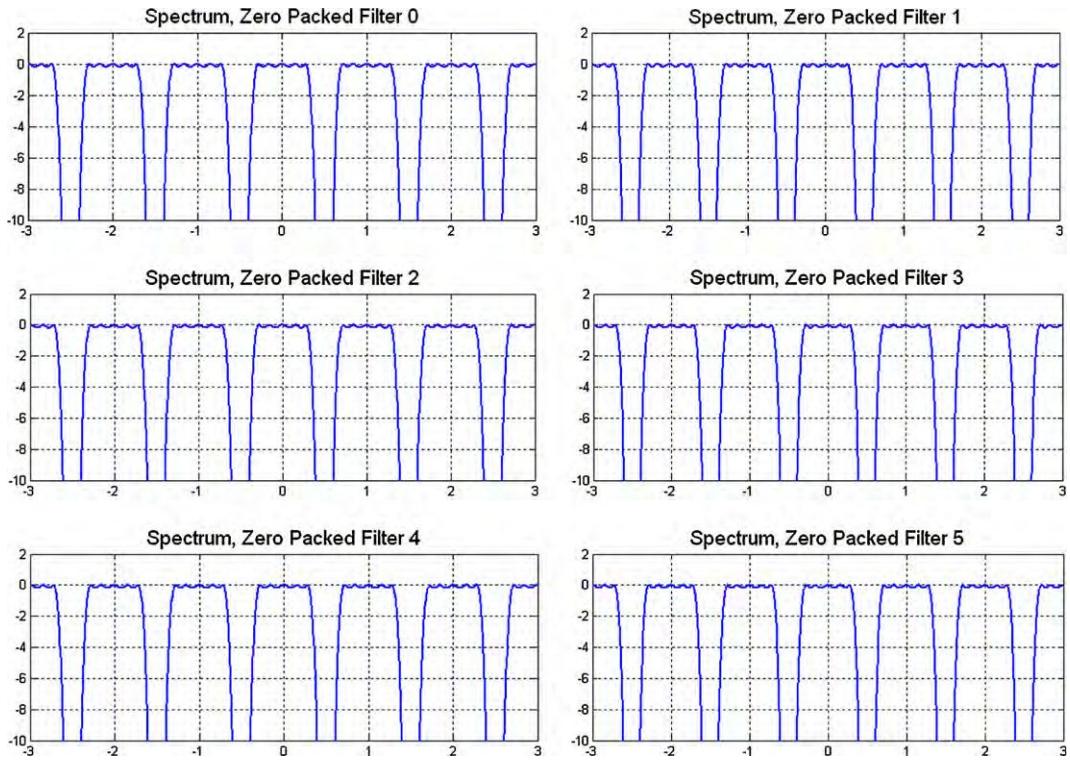
**FIGURE 7.36**

Impulse response of 6-path polyphase partition prior to 6-to-1 resampling.

### 1.07.6.1 From single channel up converter to standard up converter channelizer

The standard  $M$ -path up converter channelizer that performs 1-to- $M$  up sampling while up converting the input time series is derived from the basic structure, shown in Figure 7.11, of a single channel interpolator formed of an up sampler followed by an appropriate low-pass filter. In this configuration, the up sampler converts the Nyquist interval, which is the observable frequency span, from the input sample rate to a span  $M$  times wider, which is the output sample rate. The 1-to- $M$  up sampler zero packs the input time series, effectively decreasing the distance between the input samples without modifying the spectral content of the series. The wider Nyquist interval, spanning  $M$  input Nyquist intervals, presents  $M$  spectral copies of the input spectrum to the low-pass filter. The amplitude of each of the  $M$  copies is  $1/M$ th of the amplitude of the input signal spectrum. When a low-pass filter is scaled so that the peak coefficient is unity, the filter exhibits a processing gain inversely proportional to its fractional bandwidth. Thus, as the filter eliminates the  $M-1$  spectral copies, reducing the bandwidth by a factor of  $1/M$ , the filter gain precisely compensates for the attenuation of the input spectra due to the zero packing of the input series.

We start the modifications of this basic architecture by observing that the zero valued samples of the zero-packed input time series do not contribute to the weighted sums formed at the filter output. Since they do not contribute, there is no need to perform the product and sum from the input data registers

**FIGURE 7.37**

Frequency response of 6-path polyphase partition prior to 6-to-1 resampling.

containing the known zero-valued samples. Since only the non-zero packed samples contribute to the filter output, we can track their location in the filter and perform the weighted sum only from the register locations containing these samples. These locations are separated by  $M$  sample values, and their position shifts through the filter as each new zero-valued input is presented to the input of the filter. Keeping track of the coefficient stride and the position of each coefficient set is automatically performed by the polyphase partition of the filter which is represented in Eq. (7.25). This equation describes the filter as a sum of successively delayed sub-filters with coefficients separated by the stride of  $M$  samples. Equation (7.26) is a compact representation of Eq. (7.25) where the  $r$ th stage  $H_r(z^M)$  of the polyphase filter is formed by the coefficient set that starts at index  $r$  and increments in steps of length  $M$

$$H(z) = \sum_{r=0}^{M-1} z^{-r} \sum_{n=0}^{\left(\frac{N}{M}\right)-1} h(r + nM) z^{-nM}, \quad (7.25)$$

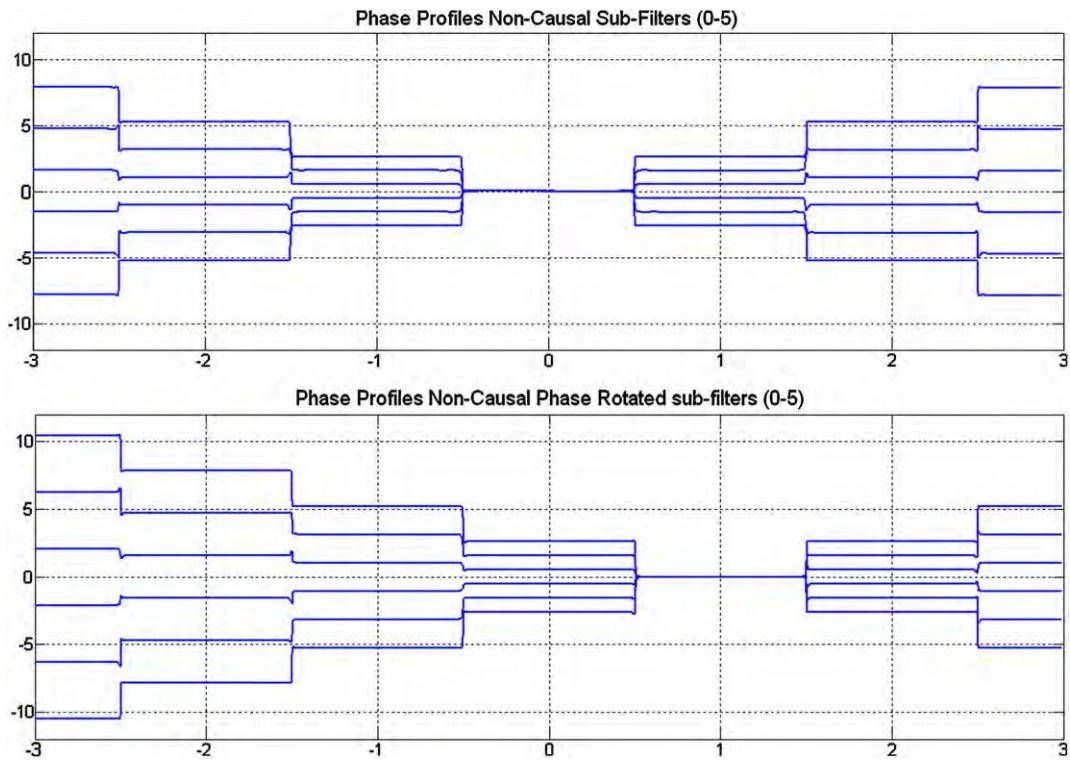
**FIGURE 7.38**

Overlaid phase response of 6-path polyphase partition prior to 6-to-1 resampling.

$$H(z) = \sum_{r=0}^{M-1} z^{-r} H_r(z^M). \quad (7.26)$$

The pictorial form of the filter in the  $M$ -path polyphase partition is shown in Figure 7.42. This structure enables the application of the noble identity in which we slide the resampler through the filter and replace the  $M$  units of delay at the output clock rate with one unit of delay at the input clock rate. Note that the resampler cannot slide through the delays  $z^{-r}$  following each filter segment  $H_r$ .

The resamplers following the separate filter stages up sample each time series by a factor of  $M$ , and the delays in each arm shift each resulting time series at different time increments so that only one non-zero time sample is presented to the summing junction at each output time. Thus, rather than performing the sum with multiple zeros, we can simply point to the arm that sequentially supplies the non-zeros samples. The output commutator, shown in Figure 7.43, performs this selective access. Figure 7.42 represents the standard polyphase interpolator or up sampler. Due to the low-pass nature of the filter on which the polyphase decomposition has been applied, this structure does not perform

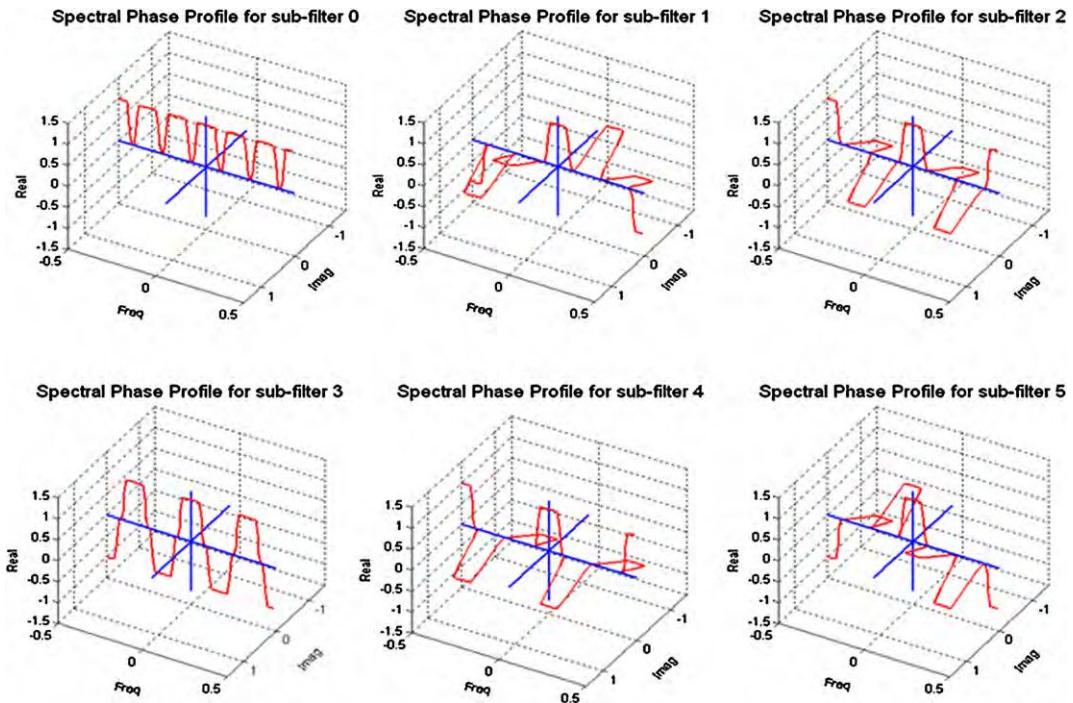
**FIGURE 7.39**

De-trended overlaid phase response; 6-path partition prior to 6-to-1 resampling.

the up conversion of the interpolated signal yet. Its output is the same low-pass input spectrum with a sample rate that is  $M$  times higher than the input sample rate.

We recall here that the purpose of the interpolation process is to increase the input sample rate while translating the input spectrum to a higher carrier frequency. If used in a communication system, such a structure directly translates the input signal to an intermediate frequency (IF), and then outputs the digital IF signal via a single DAC. This option reduces the system costs by using a single DAC and a band-pass filter to replace the standard base-band process requiring matched DACs, matched low-pass filters, matched balanced mixers, and a quadrature oscillator to form the IF frequency band.

In the structure of Figure 7.24, the 1-to- $M$  interpolating process zero packed the input data giving us access to  $M$  spectral copies of the input time series. The spectral copies reside at multiples of the input sample rate. The low-pass filter rejected the spectral copies, retrieving only the base-band copy centered at DC that is then sent through a digital up converter for translation to the desired center frequency. It is possible to perform the spectral translation as a part of the interpolation process when the desired center frequency coincides with one of the multiples of the input sample rate. Rather than extracting the spectral copy at base-band from the replicated set of spectra and then translating it by means of a

**FIGURE 7.40**

3-D Paddle-Wheel phase profiles; 6-path partition prior to 6-to-1 resampling.

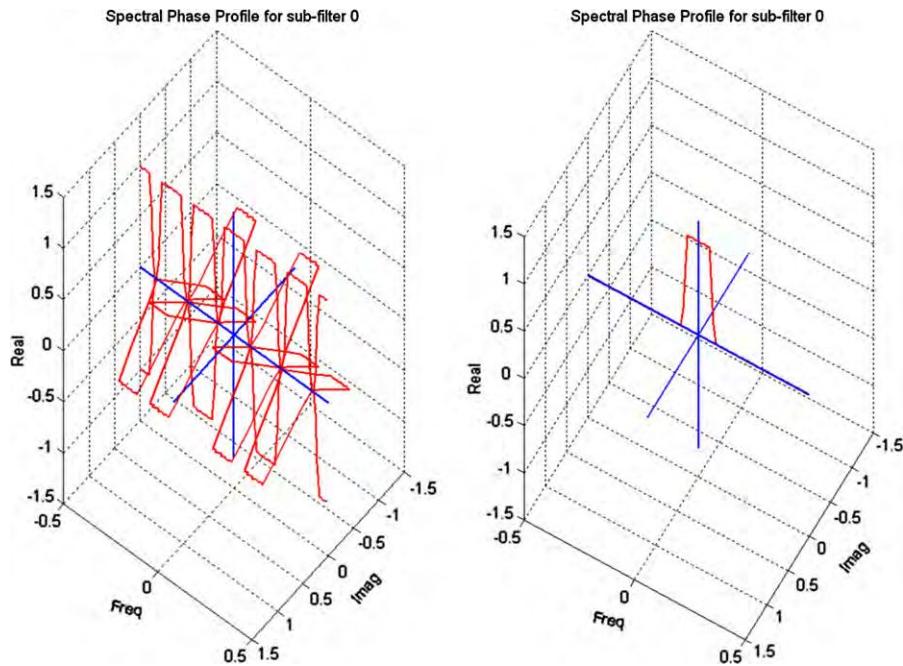
complex heterodyne, we can directly extract one of the spectral copies by using a band-pass filter as opposed to a low-pass filter. The band-pass filter is simply an up converted version of the low-pass filter  $h(n)$  with weights shown in Eq. (7.27). Here the center frequency is interpreted as the  $k$ th multiple of  $\frac{1}{M}$  of the output sample rate.

$$g(n, k) = h(n) \exp\left(j \frac{2\pi}{M} kn\right). \quad (7.27)$$

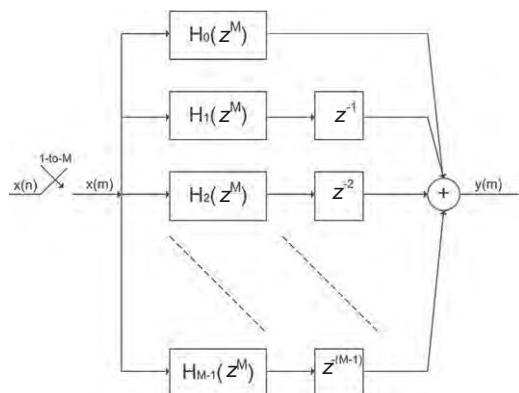
The  $z$ -transform of the band-pass filter  $g(n, k)$ , is shown in Eq. (7.28)

$$G_k(z) = \sum_{n=0}^{N-1} h(n) \exp\left(j \frac{2\pi}{M} kn\right) z^{-n}. \quad (7.28)$$

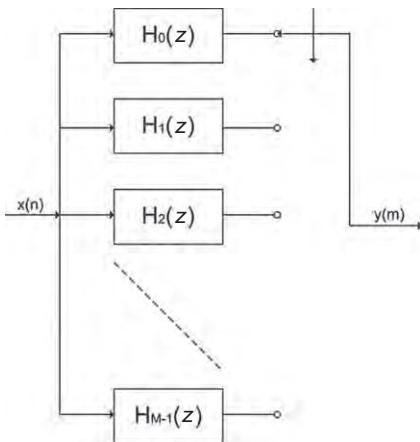
The  $z$ -transform of the polyphase decomposition of this filter is shown in Eq. (7.29). Here we see that the length  $M$  stride in coefficient index due to the 1-to- $M$  resampling aliases the phase rotators in the polyphase filter stages to DC and hence has no effect on the polyphase weights. The phase rotator does, however, have a contribution related to the delay associated with each arm of the partition. The output commutator process absorbs the delays while the phase rotators are applied to each arm to obtain the

**FIGURE 7.41**

Overlaid 3-D Paddle-Wheel phase profiles; 6-path partition prior to 6-to-1 resampling.

**FIGURE 7.42**

Initial structure of 1-to- $M$  polyphase interpolator.

**FIGURE 7.43**

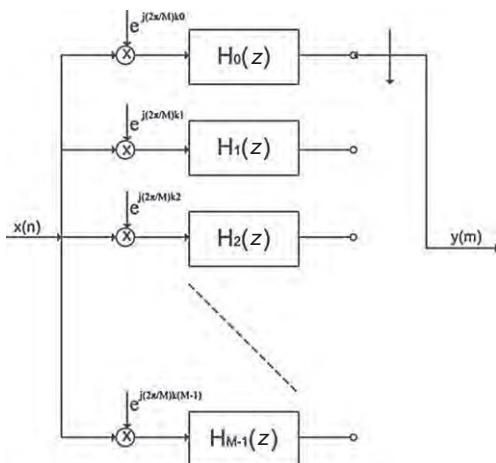
Standard structure of polyphase interpolator.

spectral translation as a part of the interpolation.

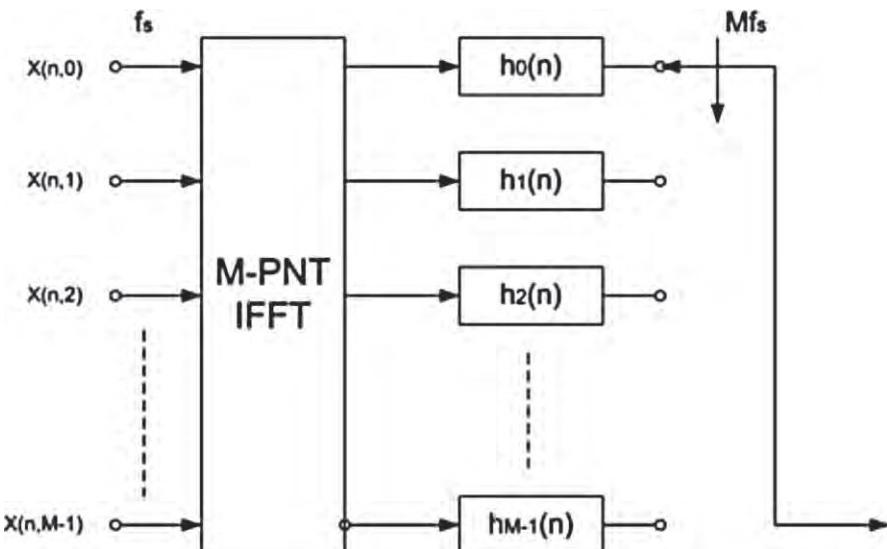
$$\begin{aligned}
 G_k(z) &= \sum_{r=0}^{M-1} z^{-r} \exp\left(j \frac{2\pi}{M} rk\right) \sum_{n=0}^{\frac{N}{M}-1} h(r + nM) \exp\left(j \frac{2\pi}{M} nM k\right) z^{-nM} \\
 &= \sum_{r=0}^{M-1} z^{-r} \exp\left(j \frac{2\pi}{M} rk\right) \sum_{n=0}^{\frac{N}{M}-1} h(r + nM) z^{-nM}.
 \end{aligned} \tag{7.29}$$

The polyphase filter that accomplishes the task of simultaneous interpolation and up conversion to the  $k$ th Nyquist zone is shown in Figure 7.44. This engine up samples and up converts, by aliasing, a single narrowband channel to a particular center frequency which is a multiple of the input sample rate. By noticing, as we did in the previous section of this chapter for the dual down converter channelizer, that the phase rotators applied in each arm are the coefficients of an inverse discrete Fourier transform it is easy to generalize this structure to the standard multichannel polyphase up converter channelizer by applying an IDFT block at the input of the filter bank.

Figure 7.45 shows the complete structure of the standard  $M$ -path polyphase up converter channelizer. It is composed of an  $M$  point IDFT block, an  $M$ -path partitioned filter and an output commutator. In this structure, we enter the  $M$ -channel process at the IDFT block and leave the process by the output commutator. The  $M$ -point IDFT performs two simultaneous tasks; an initial up sampling of 1-to- $M$ , forming an  $M$ -length vector for each input sample  $x(n, k)$  and further imparts a complex phase rotation of  $k$  cycles in  $M$ -samples on the up sampled output vector. It generates a weighted sum of complex vectors containing integer number of cycles per  $M$ -length vector. The polyphase filter forms a sequence of column coefficient weighted, MATLAB's dot-multiply, versions of these complex spinning vectors. The sum of these columns, formed by the set of inner products in the polyphase partitioned filter, is the

**FIGURE 7.44**

Structure of 1-to- $M$  polyphase interpolator with phase rotators for selecting the spectrum centered in  $M$ th nyquist zone.

**FIGURE 7.45**

Polyphase up converter channelizer:  $M$ -point IFFT,  $M$ -path polyphase filter and commutator.

shaped version of the up converted  $M$ -length vector output from the IFFT. On each output port of this engine we find the input base-band channel aliased to the specific Nyquist zone with a new sampling rate that is commensurate with its reduced bandwidth.

A closing comment on both the standard polyphase up converter and down converter channelizer is that the operations of sampling rate change, spectral shaping and Nyquist zone selection are completely independent of each other. The channel bandwidth, the channel spacing and the output sampling rate do not have necessarily to be equal but they can be all modified according to the applications.

---

### 1.07.7 Modifications of the standard down converter channelizer— $M:2$ down converter channelizer

In the standard polyphase down converter channelizer shown in Figure 7.34 the channel bandwidth  $f_{BW}$ , the channel spacing  $f_\Delta$ , and the sampling frequency  $f_s$  are fixed to be equal.

This configuration could represent a good choice when the polyphase channelizer is used for communication systems. In these kind of applications, in fact, an output sample rate matching the channel spacing is sufficient to avoid adjacent channel cross talk since the two-sided bandwidth of each channel is less than the channel spacing. An example of a signal that would require this mode of operation is the Quadrature Amplitude Modulation (QAM) channels of a digital cable system. In North America, the channels are separated by 6 MHz centers and operate with square-root cosine tapered Nyquist-shaped spectra with 18% or 12% excess bandwidth, at symbol rates of approximately 5.0 MHz. The minimum sample rate required of a cable channelizer to satisfy the Nyquist criterion would be 6.0 MHz (The European cable plants have channel spacing of 8.0 MHz and symbol rates of 7.0 MHz). The actual sample rate would likely be selected as a multiple of the symbol rate rather than as a multiple of the channel spacing. Systems that channelize and form samples of the Nyquist-shaped spectrum often present the sampled data to an interpolator to resample the time series collected at bandwidth-related Nyquist rate to a rate offering two samples per symbol or twice symbol rate. For the TV example just cited, the 6 Ms/s, 5 MHz symbol signal would have to be resampled by 5/3 to obtain the desired 10 Ms/s. This task is done quite regularly in the communication receivers and it may represent a significant computational burden. It would be appropriate if we could avoid the external interpolation process by modifying the design of the standard polyphase channelizer for directly providing us the appropriate output sampling frequency.

Many others applications desire channelizers in which the output sampling frequency is not equal to the channel spacing and the channel bandwidth. The design of software defined radio receiver and transmitter is only one of them. Another one can be found in [8].

We have concluded the previous section mentioning that the channel spacing, the channel bandwidth and the output sampling frequency of the polyphase channelizer are completely independent of each other and that they can be arbitrary selected based on the application. Figure 7.46 shows some possible options in which the channel bandwidth, the output sampling rate and the channel spacing of the channelizer are not equal. In this chapter, for the purpose of designing flexible radio architectures, the third option, when the selected low-pass prototype filter is a Nyquist filter (perfect reconstruction filter), results to be the most interesting one.

In this section of the chapter we derive the reconfigured version of the standard down converter channelizer that is able to perform the sample rate change from the input rate  $f_s$  to the output sampling rate  $2f_s/M$  maintaining both the channel spacing and the channel bandwidth equal to  $f_s$ . Similar reasoning can be applied for implementing different selections.

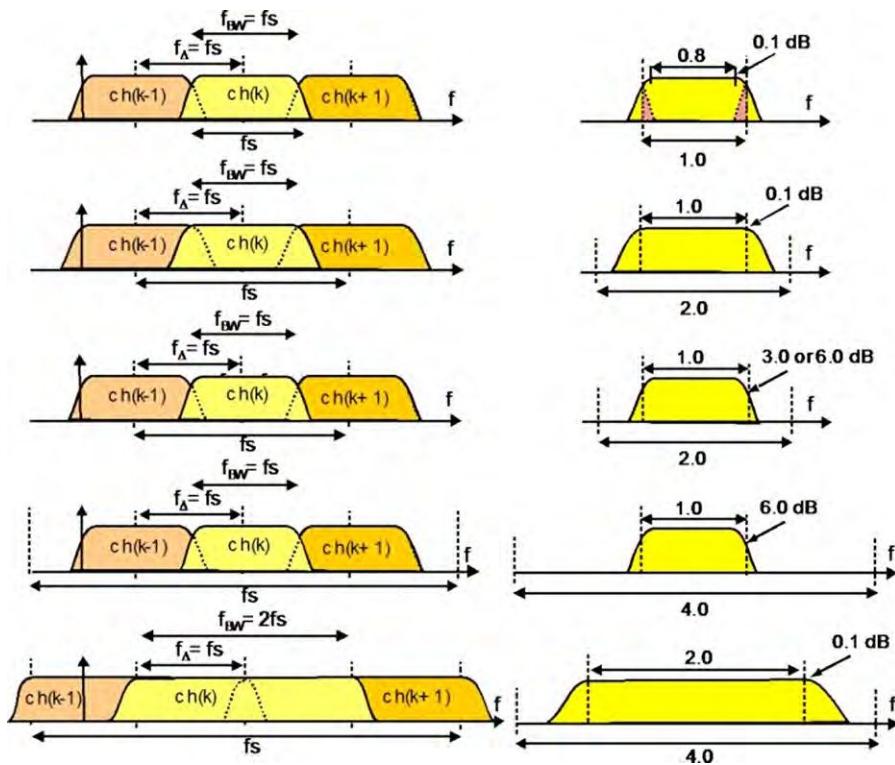


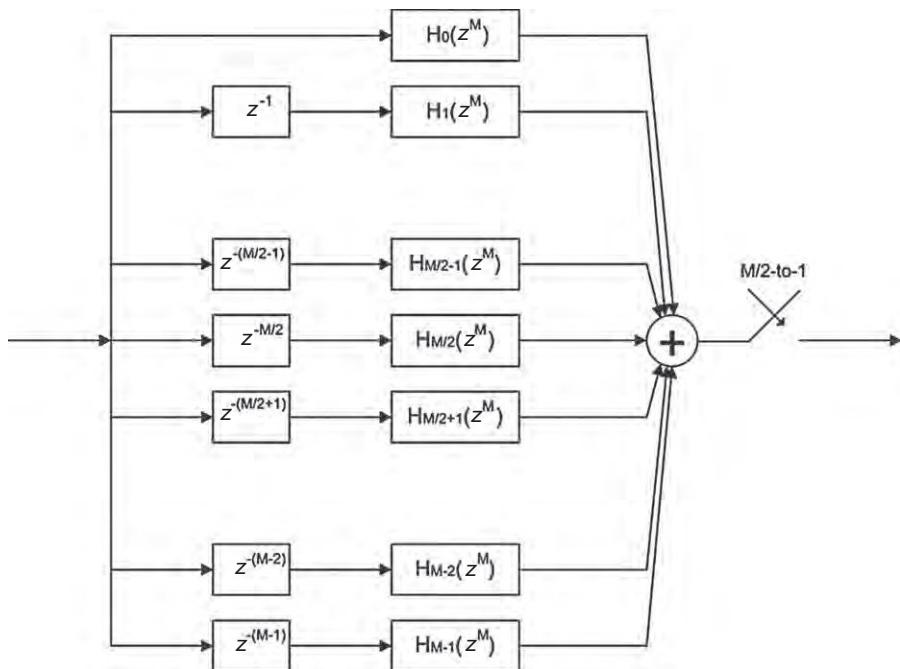
FIGURE 7.46

Some possible channel width, channel spacing, and output sampling frequency.

Note that the standard down converter channelizer is critically sampled when the channel spacing and the channel bandwidth are both equal to  $f_s$  and when the output sample rate is also equal to  $f_s/M$ . This particular choice, in fact, causes the transition band edges of the channelizer filters to alias onto itself which would prevent us from further processing the signals when they are arbitrarily located in the frequency domain which is the case of software radio. This problem can be visualized in the upper example depicted in Figure 7.46.

For the record, we remind that a polyphase filter bank can be operated with an output sample rate which can be any rational ratio of the input sample rate. With minor modifications the filter can be operated with totally arbitrary ratios between input and output sample rates. This is true for the sample rate reduction imbedded in a polyphase receiver as well as for the sample rate increase embedded in a polyphase modulator.

We have control on the output sampling rate of the down converter channelizer by means of the input commutator that delivers input data samples to the polyphase stages. We normally deliver  $M$  successive input samples to the  $M$ -path filter starting at port  $M-1$  and progressing up the stack to port 0 and by doing so we deliver  $M$  inputs per output which means to perform an  $M$ -to-1 down sampling operation.

**FIGURE 7.47**

*M*-path filter and *M*/2:1 down sampler.

To obtain the desired ( $M/2$ )-to-1 down sampling, we have to modify the input commutator in a way that it delivers  $M/2$  successive input samples starting at port  $(M/2)-1$  and progressing up the stack to port 0. We develop and illustrate the modifications with the aid of Figures 7.47–7.50.

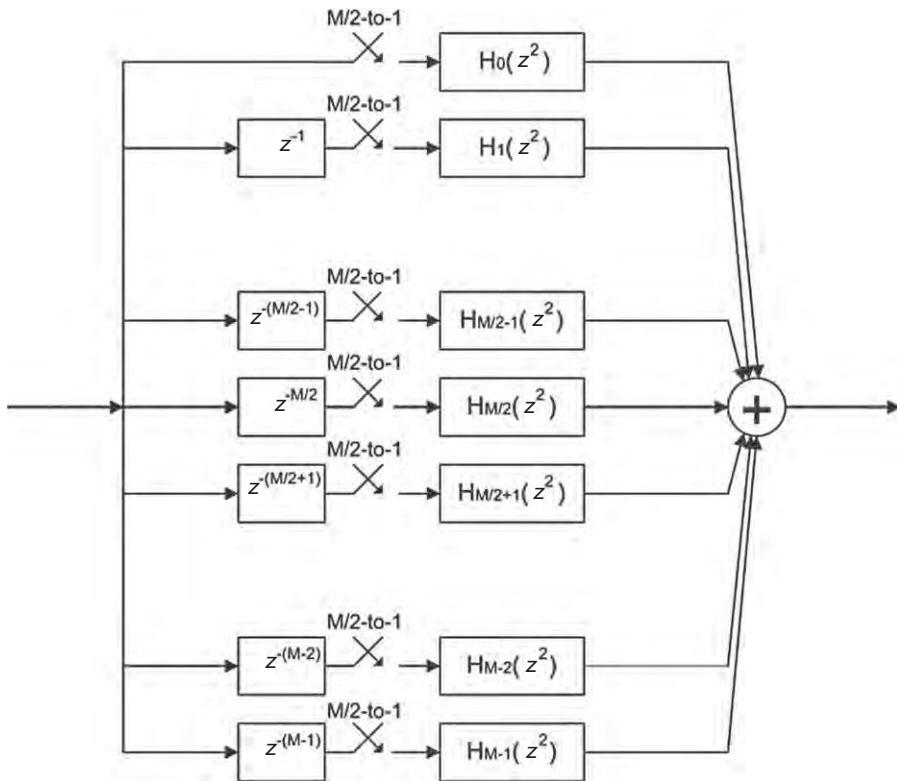
Figure 7.47 represents the polyphase  $M$ -path filter partition shown in Eq. (7.9) with an  $M/2$ -to-1 rather than the conventional  $M$ -to-1 down sample operation after the output summing junction. We need it in order to perform the desired sampling rate change.

In Figure 7.48 we apply the noble identity to the polyphase paths by pulling the  $M/2$ -to-1 down sampler through the path filters which converts the polynomials in  $z^M$  operating at the high input rate to polynomials in  $z^2$  operating at the lower output rate.

Figure 7.49 shows the second application of the noble identity in which we again take the  $M/2$ -to-1 down samplers through the  $z^{-M/2}$  parts of the input path delays for the paths in the second or bottom half of the path set.

In Figure 7.50 the  $M/2$ -to-1 down samplers switch and their delays are replaced with a two pronged commutator that delivers the same sample values to path inputs with the same path delay. Here we also merged the  $z^{-1}$  delays in the lower half of filter bank with their path filters.

Figure 7.51 shows and compares the block diagrams of the path filters in the upper and lower half of this modified polyphase partition.

**FIGURE 7.48**

Noble identity applied to  $M$ -path filters.

When the input commutator is so designed, the  $M/2$  addresses to which the new  $M/2$  input samples are delivered have to be first vacated by their former contents, the  $M/2$  previous input samples. All the samples in the two-dimensional filter undergo a serpentine shift of  $M/2$  samples with the  $M/2$  samples in the bottom half of the first column sliding into the  $M/2$  top addresses of the second column while the  $M/2$  samples in the top half of the second column slide into the  $M/2$  addresses in the bottom half of the second column, and so on. This is equivalent to performing a linear shift through the prototype one-dimensional filter prior to the polyphase partition. In reality, we do not perform the serpentine shift but rather perform an addressing manipulation that swaps two memory banks. This is shown in Figure 7.52.

After each  $M/2$ -point data sequence is delivered to the partitioned  $M$ -stage polyphase filter, in the standard channelizer configuration, the outputs of the  $M$  stages are computed and conditioned for delivery to the  $M$ -point IFFT. What we need to do at this point is the time alignment of the shifting time origin of the input samples in the  $M$ -path filter with the stationary time origin of the phase rotator outputs of the IFFT. We can understand the problem by visualizing, in Figure 7.53, a single cycle of

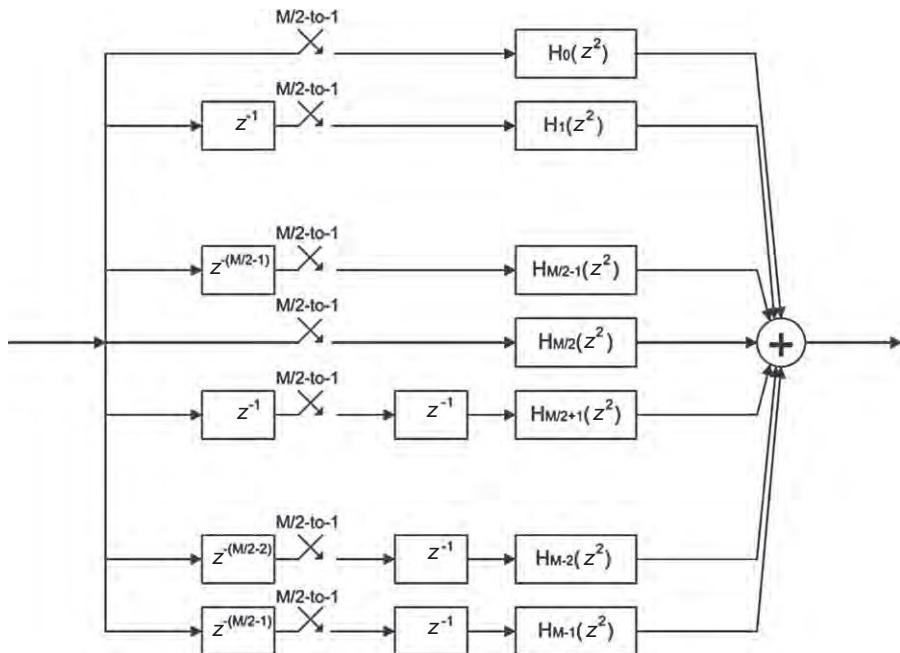
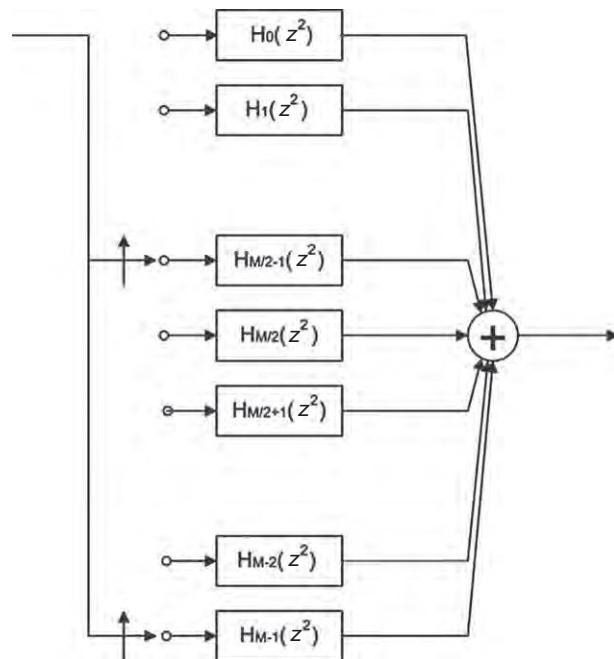


FIGURE 7.49

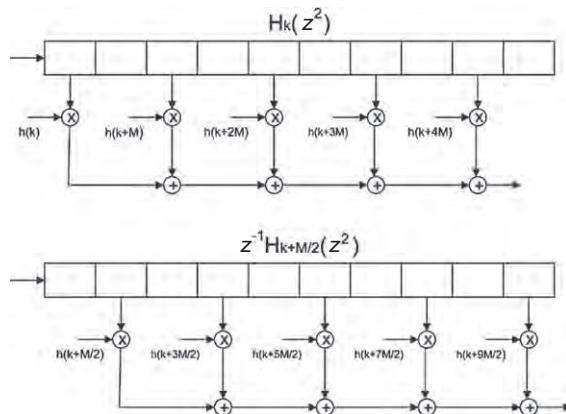
Noble identity applied to delays.

a sine wave extending over  $M$  samples being inserted in the input data register, the first column of the polyphase filter in segments of length  $M/2$ . We can assume that the data in the first  $M/2$  addresses are phase aligned with the first  $M/2$  samples of a single cycle of the sine wave offered by the IFFT.

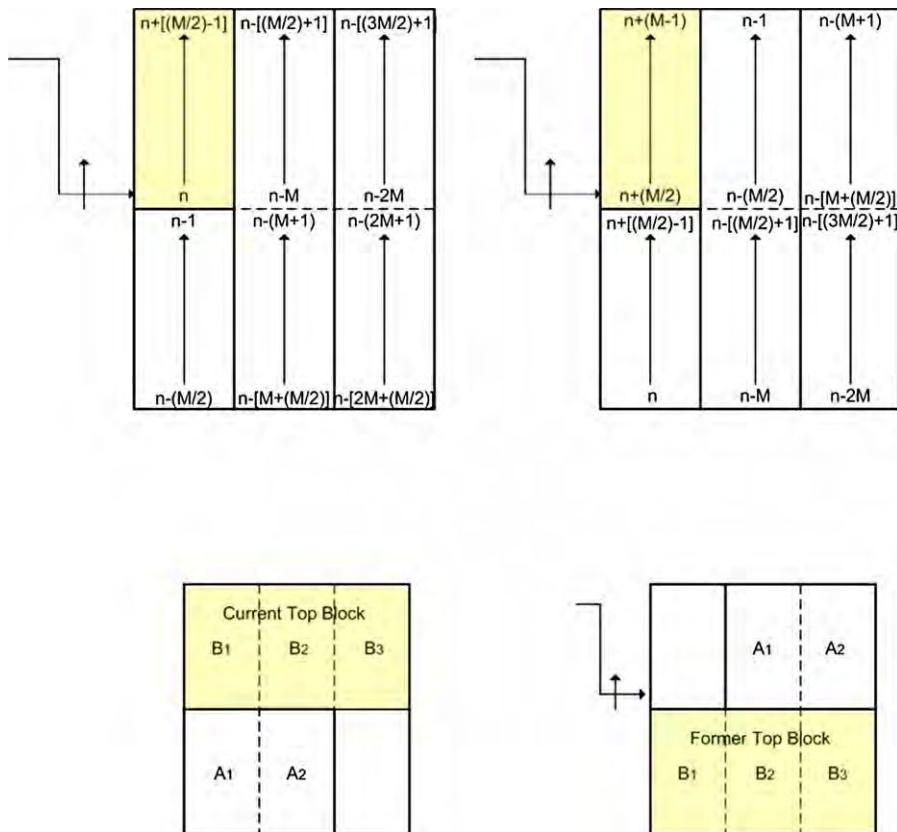
When the second  $M/2$  input samples are delivered to the input data register the first  $M/2$  input samples shift to the second half of the  $M$ -length array. Its original starting point is now at address  $M/2$  but the IFFT's origin still resides at address 0. The shift of the origin causes the input sine wave in the register to have the opposing phase of the sine wave formed by the IFFT; in fact the data shifting into the polyphase filter stages causes a frequency dependent phase shift of the form shown in Eq. (7.30). The time delay due to shifting is  $nT$  where  $n$  is the number of samples, and  $T$  is the time interval between samples. The frequencies of interest are integer multiple  $k$  of  $1/M$ th of the sample rate  $2\pi/T$ . Substituting these terms in Eq. (7.30) and canceling terms, we obtain the frequency dependent phase shift shown in Eq. (7.31). From this relationship we see that for time shifts  $n$  equal to multiples of  $M$ , as demonstrated in Eq. (7.32), the phase shift is a multiple of  $2\pi$  and contributes zero offset to the spectra observed at the output of the IFFT. The  $M$ -sample time shift is the time shift applied to the data in the normal use of the polyphase filter. Now suppose that the time shift is  $M/2$  time samples. When substituted in Eq. (7.31) we find, as shown in Eq. (7.33), as frequency dependent phase shift of  $k\pi$  from which we conclude that odd-indexed

**FIGURE 7.50**

Path delays replaced by input commutator.

**FIGURE 7.51**

M-path filters with and without extra delays.

**FIGURE 7.52**

Data memory loading for successive  $M/2$ -point sequences in an  $M$ -stage polyphase channelizer.

frequency terms experience a phase shift of  $\pi$  radians for each successive  $N/2$  shift of input data

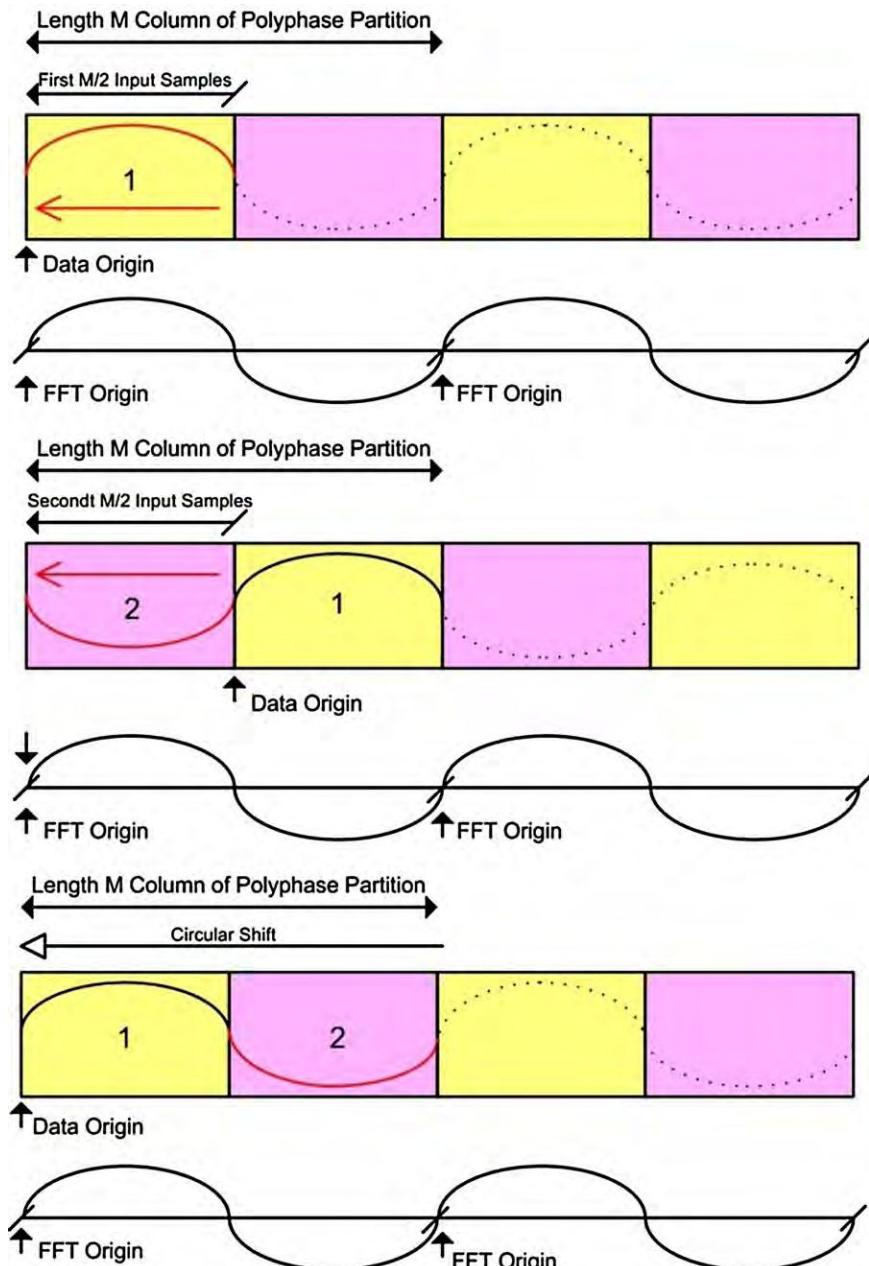
$$\theta(\omega) = \Delta t \cdot \omega, \quad (7.30)$$

$$\theta(\omega_k) = nT \cdot k \frac{1}{M} \frac{2\pi}{T} = \frac{nk}{M} 2\pi, \quad (7.31)$$

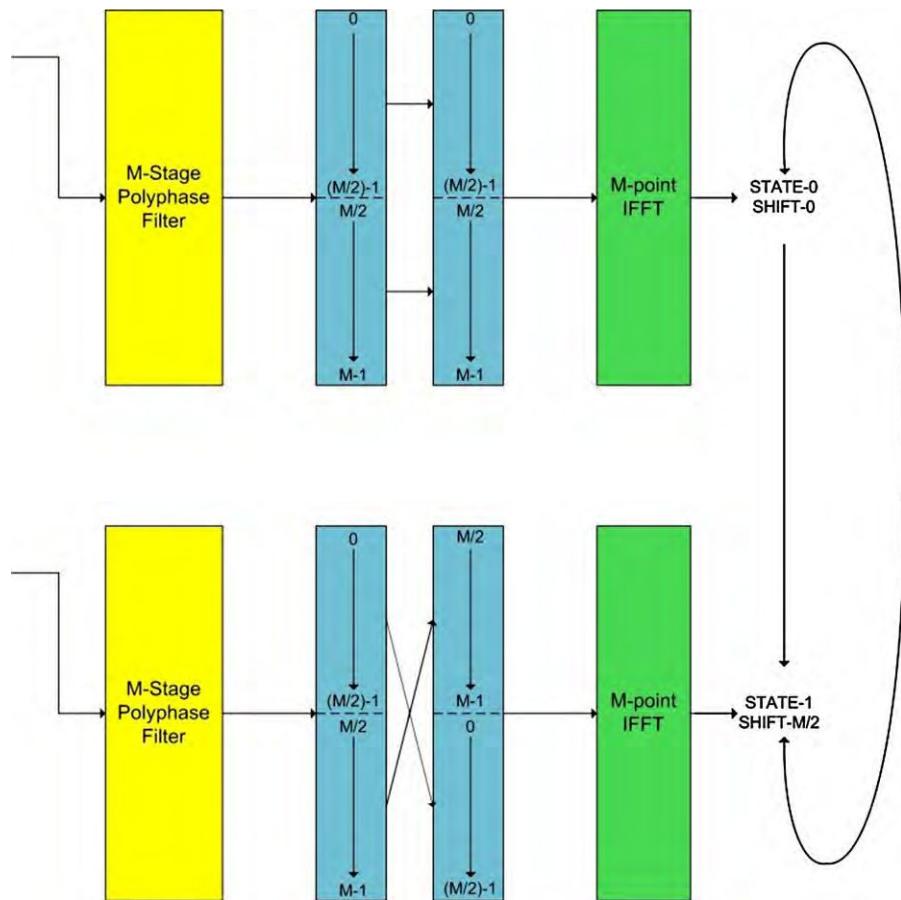
$$\theta(\omega_k)|_{n=M} = \frac{nk}{M} 2\pi \Big|_{n=M} = 2\pi k, \quad (7.32)$$

$$\theta(\omega_k)|_{n=M/2} = \frac{nk}{M} 2\pi \Big|_{n=M/2} = \pi k. \quad (7.33)$$

This  $\pi$  radians phase shift is due to the fact that the odd-indexed frequencies alias to the half sample rate when the input signal is down sampled by  $M/2$ . What we are observing is the sinusoids with an odd

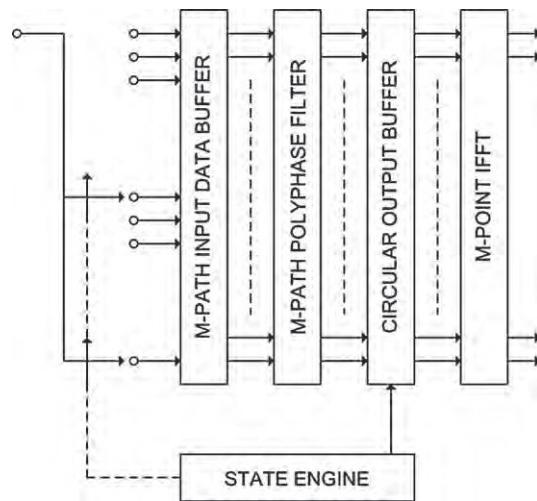
**FIGURE 7.53**

Illustrating phase reversal of  $M$ -point sinusoid input to  $M/2$  path polyphase filter.

**FIGURE 7.54**

Cyclic shift of input data to IFFT to absorb phase shift due to linear time shift of data through polyphase filter.

number of cycles in the length  $M$  array alias to the half sample rate when down sampled  $M/2$ -to-1. Note that, when down sampled  $M/2$ -to-1, the sinusoids with an even number of cycles in the length  $M$  array alias to DC. We can compensate for the alternating signs in successive output samples by applying the appropriate phase correction to the spectral data as we extract successive time samples from the odd-indexed frequency bins of the IFFT. The phase correction here is trivial, but for other down-sampling ratios, the residual phase correction would require a complex multiply at each transform output port. Alternatively, since time delay imposes a frequency dependent phase shift, we can use time shifts to cancel the frequency dependent phase shifts. We accomplish this by applying a circular time shift of  $N/2$  samples to the vector of samples prior to their presentation to the IFFT. As in the case of the serpentine shift of the input data, the circular shift of the polyphase filter output data is implemented as an

**FIGURE 7.55**

M-to-2 modified down converter channelizer.

address-manipulated data swap. This data swap occurs on alternate input cycles and a simple two-state machine determines for which input cycle the output data swap is applied. This is shown in Figure 7.54.

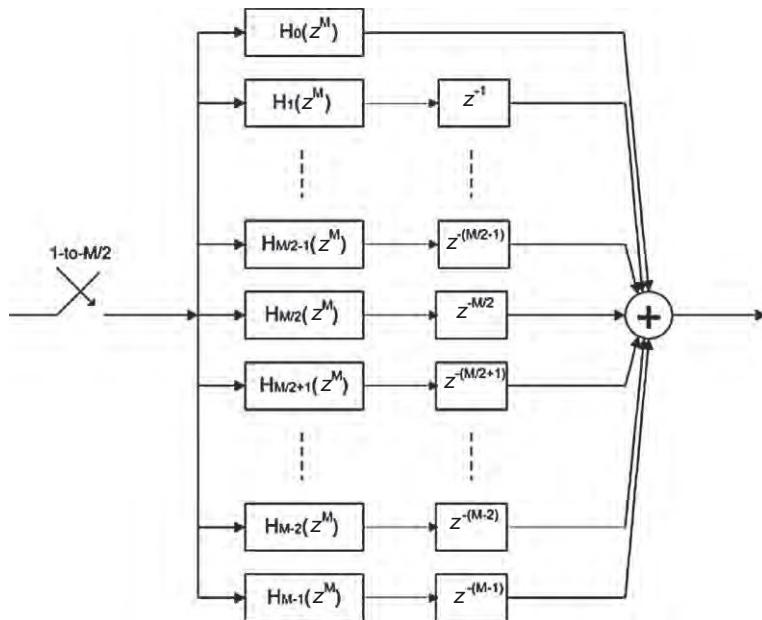
The complete structure of the modified version of the  $M$ -to-2 down converter channelizer with the input data buffer and the circular data buffer is shown in Figure 7.55.

For brevity of notation, we avoid reporting here all the dual mathematical derivations that led at the final structure of the modified up converter channelizer. We briefly explain its block diagram in the next subsection.

### 1.07.7.1 Modifications of the standard up converter channelizer—2: $M$ up converter channelizer

Dual reasoning drives us to the reconfigured version of the  $M$ -path polyphase up converter channelizer that is able to perform the sample rate change from  $2f_s/M$  to  $f_s$  maintaining both the channel spacing and the channel bandwidth equal to  $f_s$ . The choice of using a 2-to- $M$  up sampler avoids the difficulty of having the sample rate that precisely matches the two sided bandwidth of the input signals as well as permitting a shorter length prototype channelizer filter due to an allowable wider transition bandwidth. In this chapter we briefly derive its block diagram that is shown in Figure 7.45. More details on this structure can be found in [6,8,9]. We develop and illustrate the modifications with the aid of Figures 7.56–7.61. In these figures we do not consider the IFFT block because, for now, it does not affect our reasoning. Moreover we will introduce it later motivating the reason for which it is important to include a circular buffer in the design of the modified up converter channelizer.

Figure 7.56 presents the structure of the  $M$ -path filter implementation of the polyphase partition shown in Eq. (7.7). Note the 1-to- $M/2$  up-sample operation at the input port normally described as the

**FIGURE 7.56**

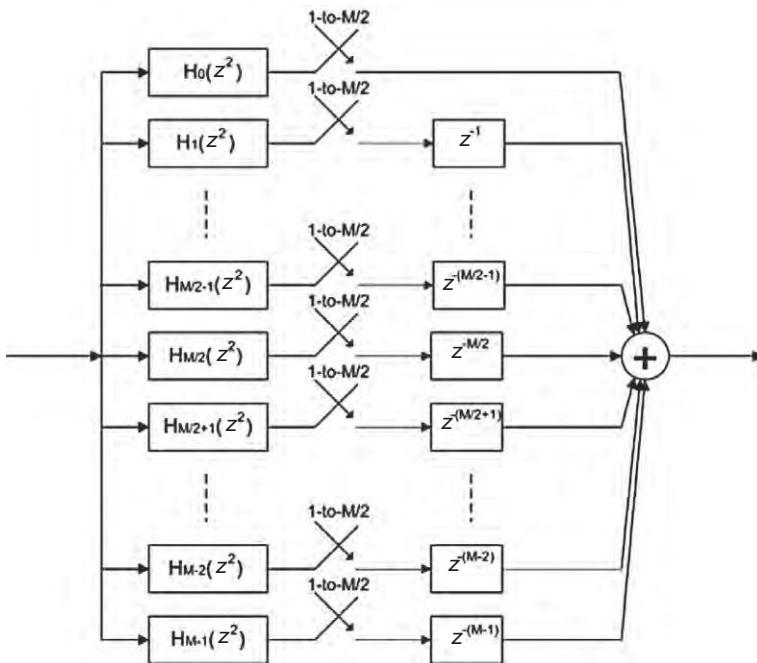
M-path polyphase filter.

zero-packing process. In Figure 7.57 we apply the noble identity to the polyphase paths and pull the 1-to- $M/2$  up sampler through the path filters which convert the polynomials in  $z^M$  operating at the high output rate to polynomials in  $z^2$  operating at the low input rate. Note that the paths are now polynomials in  $z^2$  rather than  $z^M$  as is the normal mode that we identify as the dual of the maximally decimated filter bank.

Figure 7.58 shows the second application of the noble identity in which we again take the 1-to- $M/2$  up sampler through the  $z^{-M/2}$  part of the output path delays for the paths in the second or bottom half of the path set. The resultant delay,  $z^{-1}$ , now operating at the input clock rate, is then interchanged with its path filter as shown in Figure 7.59.

In Figure 7.60 the 1-to- $M/2$  up samplers switch and their delays are replaced with a pair of commutators that add the path outputs with the same path delay. Finally, in Figure 7.61 we fold the single delay in front of the lower set of path filters into the path filter. After having changed the output sampling rate of the polyphase channelizer, the final modification we need to introduce is the time alignment of the phase rotators from the input IFFT and the shifted time origin in the  $M$ -path filter.

From Figure 7.62, we note the locations of the non-zero coefficients in the polynomials in  $z^2$  and conclude that the input sine wave only contributes to the output time samples in the path filters located in the upper half of the path set. When the next scaled sine wave output from the IFFT is inserted in the first column of the path filter memory, the previous scaled sine wave is shifted one column in the memory and it will now contribute to the output time samples from the filter paths in the lower half of the path filter set. The problem is that the sine wave samples in the lower half of the path filter set have opposite polarity of the sine wave samples in the upper half of the path filter set. The samples from the two half

**FIGURE 7.57**

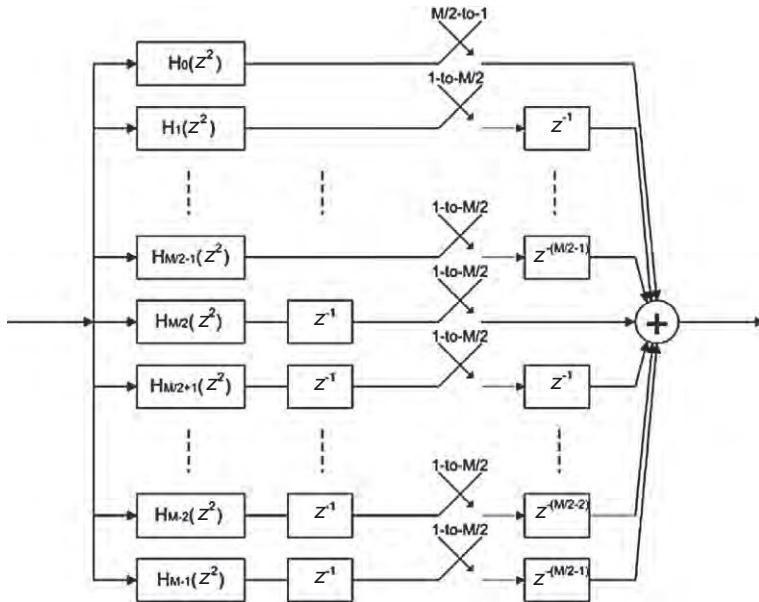
Noble identity applied to  $M$ -path filter.

filter sets are added and to do so they must have the same phase. We note that this alternating sign effect only occurs for the odd indexed IFFT frequency bins which have an odd number of cycles per interval.

An alternate description of the sign reversal is that in the  $M/2$  resampling of the modified  $M$ -path filter the even indexed frequencies alias to multiples of the input sample rate and the odd indexed frequencies alias to odd multiples of the half sample rate.

We recall here that there are two methods for performing the phase alignment of the successive output vectors from the IFFT. In the first method we simply invert the input phase of successive input samples to the odd indexed IFFT bins. In the second method, recognizing that equivalency of phase shift and time delay for sinusoids, on alternate outputs from the IFFT we apply an  $M/2$  circular shift to its output buffer prior to delivering the phase aligned vector to the path filter memory. This end around shift of the output buffer occurs during the data transfer in memory and requires no additional manipulation of the time samples.

Figure 7.63 shows the complete block diagram of the modified  $M$ -path up converter channelizer. In this engine, the input commutator delivers the samples to the  $M$ -point IFFT. It applies the complex phase rotation to the separate base-band input signals as well as performs the initial 1-to- $M$  up sampling of the input samples. The circular output buffer performs the correct data offset of the two  $M/2$  point halves of the IFFT output vector to maintain phase alignment with the  $M/2$  chan-

**FIGURE 7.58**

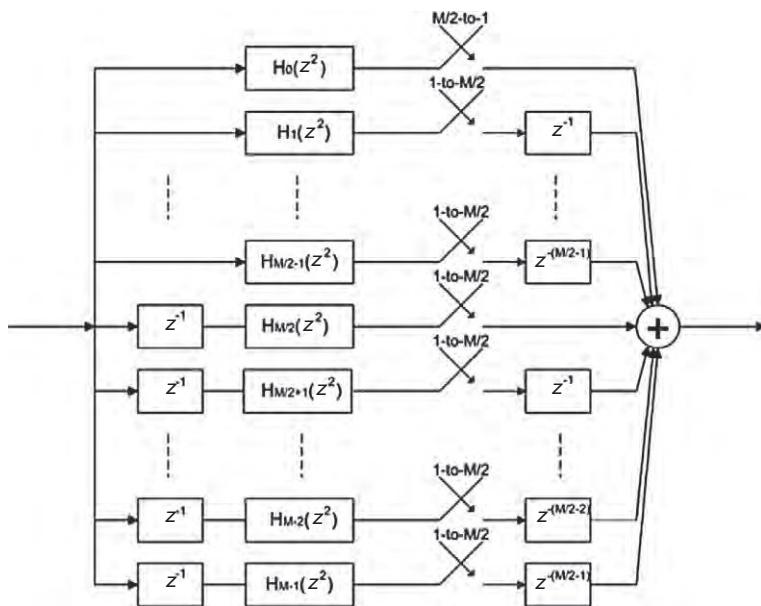
Noble identity applied to delays.

nelizer output vector. The complex sinusoid output by the IFFT always defines its time origin as the initial sample of its output vector. The output of the polyphase filter exhibits a time origin that shifts due to the  $M/2$  time sample shift embedded in the output commutator. The  $M/2$  time sample shift of the output time series causes sinusoids with an odd number of cycles in the length  $M$  array to alternate sign on successive shifts. The alternating sign is the reason that the odd indexed frequency bins up convert to a frequency  $k + N/2$  rather than frequency index  $k$ . Rather than reverse phase alternate input samples to the odd indexed IFFT bins we perform an  $M/2$  point circular shift of alternate  $M$ -length vectors from the IFFT for applying the correct phase alignment to all frequencies simultaneously.

The polyphase down converter and up converter channelizers shown in Figures 7.55 and 7.63 respectively are the key elements of the proposed receiver and transmitter structures that we present in the following sections. Because of the signal processing tasks they handle, in the following, we refer to them as analysis and synthesis channelizers respectively.

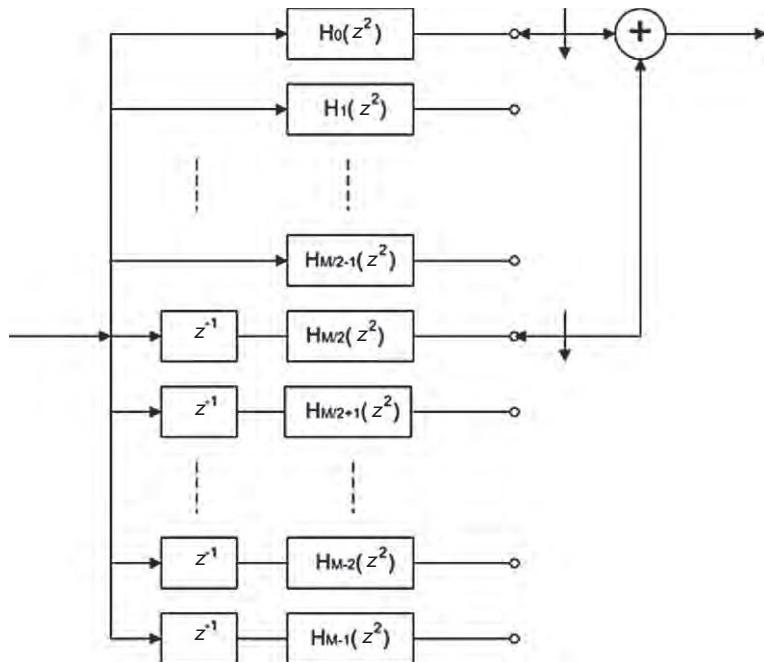
## 1.07.8 Preliminaries on software defined radios

The 20th century saw the explosion of hardware defined radio as a means of communicating all forms of data, audible and visual information over vast distances. These radios have little or no software control. Their structures are fixed in accordance with the applications; the signal modulation formats, the carrier

**FIGURE 7.59**

Interchange unit delay and path filters.

frequencies and bandwidths are only some of the factors that dictate the radio structures. The smallest change to one of these parameters could imply a replacement of the entire radio system. A consequence of this is, for example, the fact that a television receiver purchased in France does not work in England. The reason, of course, is that the different geographical regions employ different modulation standards for the analog TV as well as for digital TV. Then, the citizens cannot use the same TV for receiving signals in both countries; they need to buy a new television for each country in which they decide to live. Sometimes, even if the communication devices are designed for the same application purposes and they work in the same geographical area, they are not able to communicate between each other. One of the most evident examples of this is that the city police car radio cannot communicate with the city fire truck radio, or with the local hospital ambulance radio even if they have the common purpose of helping and supporting the citizen. Also, the city fire truck radio cannot communicate with the county fire truck radio, or with the radios of the fire truck operated by the adjacent city, or by the state park service, or the international airport. None of these services can communicate with the National Guard, or with the local Port Authority, or with the local Navy base, or the local Coast Guard base, or the US Border Patrol, or US Customs Service. In an hardware defined radio, if we decide to change one of the parameters of the transmitted signal, like bandwidth or carrier frequency (for example because the carrier frequency we want to use is the only one available at that particular moment), we need to change the transmitter. On the other side, every time we want to receive a signal having different bandwidth or center frequency, we need to change the receiver. Hardware defined transmitter and receiver devices

**FIGURE 7.60**

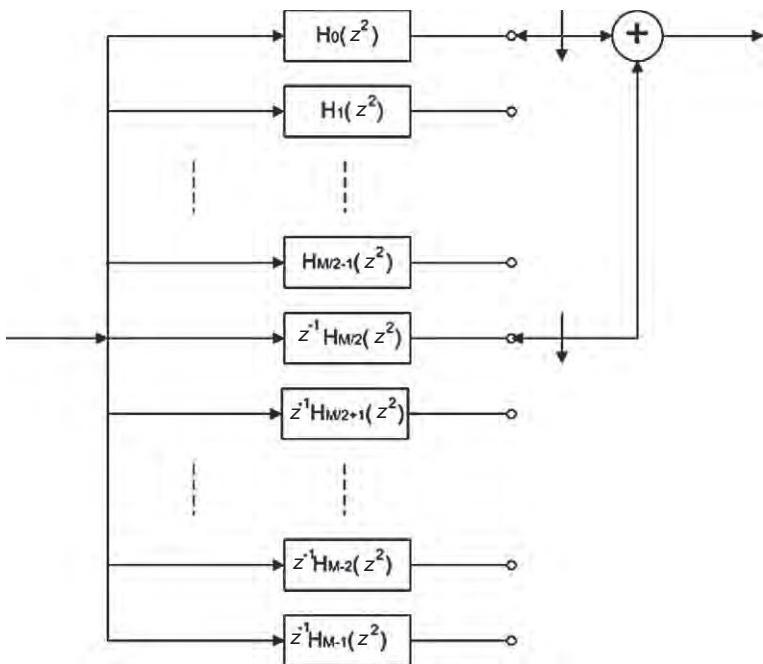
Insert commutator add same delay paths.

are not flexible at all; we must modify their structure every time we change even one of the transmitting and receiving signal parameters.

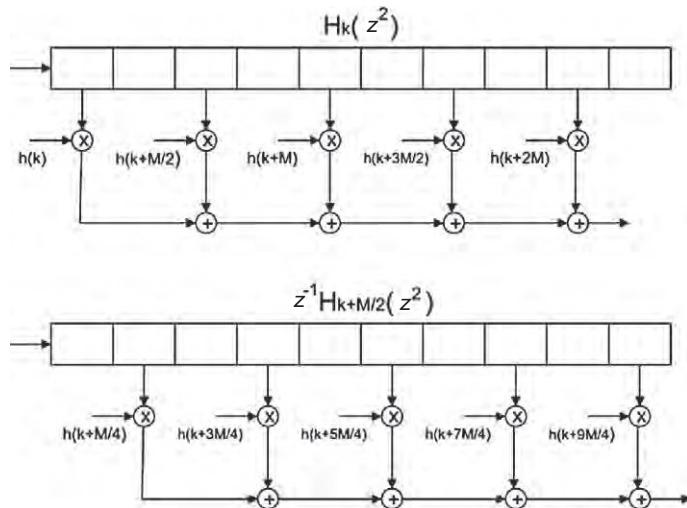
In 1991, Joe Mitola coined the term software defined radio. It was referred to a class of reprogrammable (and reconfigurable) devices. At that time it was not clear at which level the digitization should occur to define a radio as software but the concept sounded pretty interesting, and the dream of building a completely reconfigurable radio device involved scientists from all over the world. Today the exact definition of software defined radio is still controversial, and no consensus exists about the level of reconfigurability needed to qualify a radio as software.

Figure 7.64 shows, in a simple block diagram, all the possible places in which the digitization can occur in a radio receiver. The exact dual block diagram can be portrayed for the radio transmitter. Current radios, often referred to as digital but sometimes referred as software defined radios (depending on their particular structure), after shifting the signals to intermediate frequency, digitize them and assign all the remaining tasks to a digital signal processor. One of the main reasons for shifting the signals to intermediate frequency, before digitizing them, is to reduce their maximum frequency so that a smaller number of samples can be taken for preserving the information content.

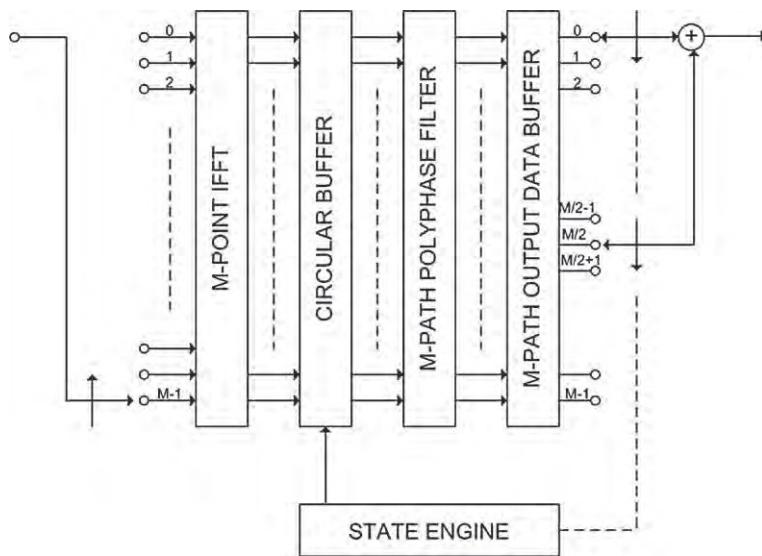
Implementation of ideal software radios requires digitization at the antenna, allowing complete flexibility in the digital domain. Then it requires both the design of a flexible and efficient DSP-based

**FIGURE 7.61**

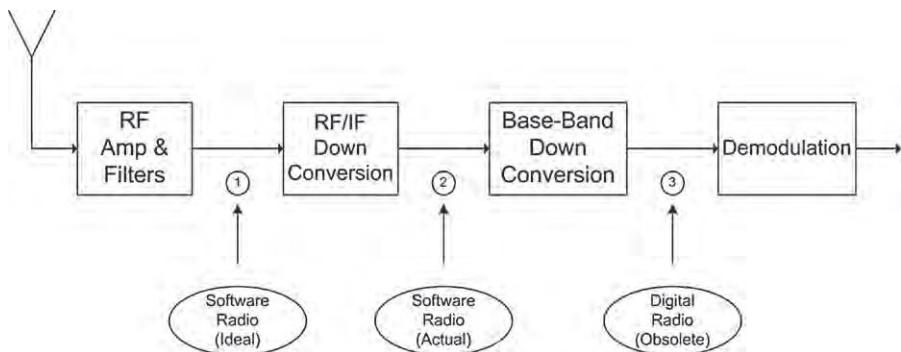
Fold unit delays into path filters.

**FIGURE 7.62**

Comparison of path filter polynomials in  $z^2$  with and without the additional input delay.

**FIGURE 7.63**

2-to- $M$  modified up converter channelizer.

**FIGURE 7.64**

Simple block diagram indicating all the possible places in which the digitization can occur in a radio receiver.

structure and the design of a completely flexible radio frequency front-end for handling a wide range of carrier frequencies, bandwidths and modulation formats. These issues have not been exploited yet in the commercial systems due to technology limitations and cost considerations.

As pointed out in the previous section, in a current software defined radio receiver the signals are digitized in intermediate frequency bands. The receiver employs a super heterodyne frequency down conversion, in which the radio frequency signals are picked up by the antenna along with other spurious,

unwanted signals (noise and interferences), filtered, amplified with a low noise amplifier and mixed with a local oscillator to shift it to intermediate frequency. Depending on the application, the number of stages of this operation may vary. Digitizing the signal in the IF range eliminates the last analog stage in the conventional hardware defined radios in which problems like carrier offset and imaging are encountered. When sampled, digital IF signals give spectral replicas that can be placed accurately near the base-band frequency, allowing frequency translation and digitization to be carried out simultaneously. Digital filtering and sample rate conversion are often needed to interface the output of the ADC to the processing hardware to implement the receiver. Likewise, on the transmitter side, digital filtering and sample rate conversion are often necessary to interface the digital hardware, that creates the modulated waveforms, to the digital to analog converter. Digital signal processing is usually performed in radio devices using field programmable gate arrays (FPGAs), or application specific integrated circuits (ASICs).

Even if the dream of building a universal radio is still far away, current software defined radio architectures are quite flexible, in the sense that they usually down convert to IF a collection of signals and, after sampling, they are able to shift these signals to base-band via software. Changes in the signal bandwidths and center frequencies are performed by changing some parameters of the digital data section. The flexibility of such a structure can be improved by moving the analog to digital converter closest to the receiver antenna. By digitizing the signals immediately after (and before in the transmitter) the antenna, which is the ideal software radio case, the down conversion (and up conversion) processes are performed completely in software and the radio acquires the capability of changing its personality, possibly in real-time, guaranteeing a desired quality of service (QoS). The digitization after the receiver antenna, in fact, allows service providers to upgrade the infrastructure and market new services quickly. It promises multi-functionality, global mobility, ease of manufacture, compactness and power efficiency. The flexibility in hardware architectures combined with flexibility in software architectures, through the implementation of techniques such as object oriented programming, can provide software radio also with the ability to seamlessly integrate itself into multiple networks with widely different air and data interfaces.

---

### 1.07.9 Proposed architectures for software radios

A digital transmitter accepts binary input sequences and outputs radio frequency amplitude and phase modulated wave shapes. The digital signal processing part of this process starts by accepting  $b$ -bit words from a binary source at input symbol rate. These words address a look-up table that outputs gray coded ordered pairs, i.e.,  $I$ - $Q$  constellation points, that control the amplitude and phase of the modulated carrier. The  $I$ - $Q$  pair is input to DSP-based shaping filters that form 1-to-4 up sampled time series designed to control the wave shape and limit the base-band modulation bandwidth. The time series from the shaping filter are further up sampled by a pair of interpolating filters to obtain a wider spectral interval between spectral replicas of the sampled data. The interpolated data are then heterodyned by a digital up converter to a convenient digital intermediate frequency and then moved from the sampled data domain to the continuous analog domain by a digital to analog converter and analog IF filter. Further analog processing performs the spectral transformations required to couple the wave shape to the channel.

On the other side of the communication chain, a digital receiver has to perform the tasks of filtering, spectral translation and analog to digital conversion to reverse the dual tasks performed at the transmitter. The receiver must also perform a number of other tasks absent in the transmitter for estimating

the unknown parameters of the received signal such as amplitude, frequency and timing alignment. It samples the output of the analog IF filter and down converts the intermediate frequency centered signal to base-band with a digital down converter (DDC). The base-band signal is down sampled by a decimating filter and finally processed in the matched filter to maximize the signal-to-noise ratio (SNR) of the samples presented to the detector. The DSP portion of this receiver includes carrier alignment, timing recovery, channel equalization, automatic gain control, SNR estimation, signal detection, and interference suppression blocks. In order to suppress the undesired artifacts introduced by the analog components, the receiver also incorporates a number of digital signal processing compensating blocks.

Figure 7.65 shows the first tier processing block diagrams of a typical digital transmitter-receiver chain. The depicted transmitter is designed for sending, one per time, signals having fixed bandwidths at precisely located center frequencies. On the other hand, the depicted receiver is also designed for down converting to base-band fixed bandwidth signals having fixed center frequencies. Therefore, the goal of a cognitive radio is quite different: cognitive transmitter and receiver should be completely flexible. The transmitter has, in fact, to be able to simultaneously up convert, at desired center frequencies, which are selected based on the temporary availability of the radio spectrum, a collection of signals

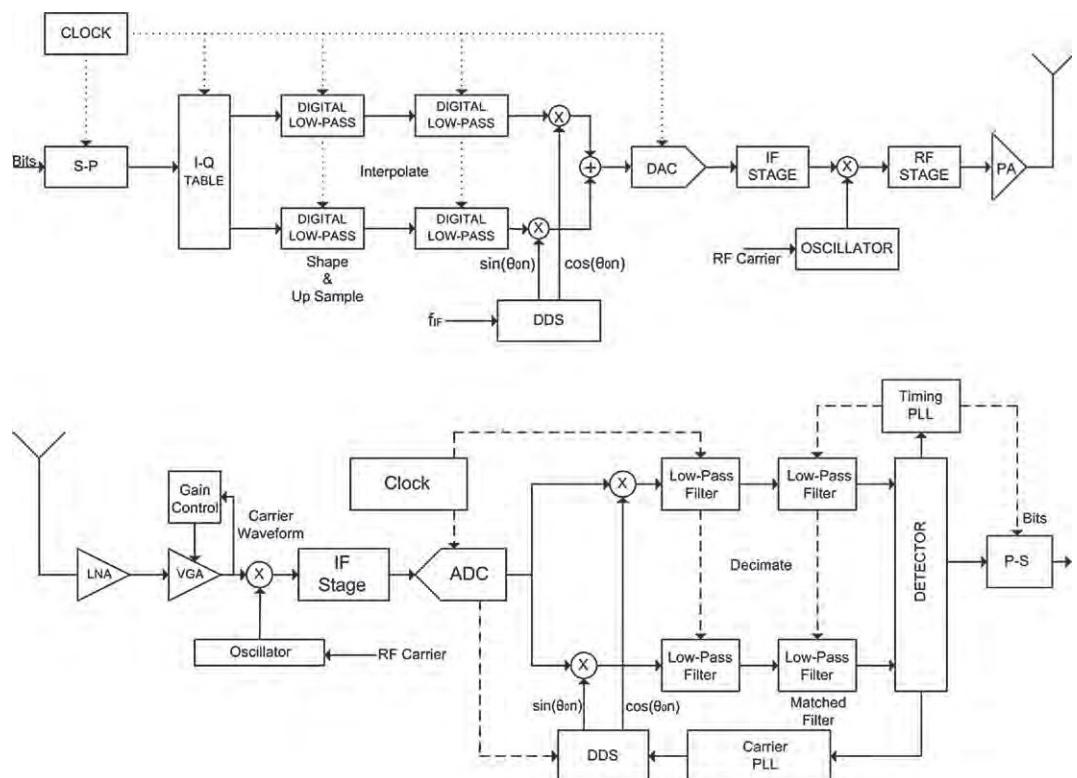


FIGURE 7.65

Block diagram of primary signal processing tasks in a typical transmitter and receiver.

having arbitrary bandwidths. It should also be able to decompose a signal in spectral fragments when a sufficiently wide portion of the radio spectrum is not available for transmission. Of course no energy loss has to occur in the partitioning process. A cognitive receiver has to be able to simultaneously down convert the transmitted signal spectra wherever they are positioned in the frequency domain and whatever bandwidths they have. Also it should have the capability of recognizing spectral segments belonging to the same information signal and recompose them, after the base-band translation, without energy losses. It could be possible to reach this goal by using the current technology but the cost would be very high: for every signal we want to transmit we have to replicate the digital data section of both the transmitter and the receiver. The more signals we have to simultaneously up and down convert, the more sampled data sections we need to implement! Also, it is not possible, by using the current technology, to fragment a single signal, transmit its spectral partitions using the temporary available spectral holes and to perfectly recombine them at the receiver. Today the transmission happens under the constraint that the available space in the radio spectrum is large enough to accommodate the entire bandwidth of the signal to be transmitted.

In the next sections of this chapter, we present novel channelizers to simultaneously up and down convert arbitrary bandwidth signals randomly located in the frequency domain. Differently from the previous contents of this document, from this point onwards the presented material reproduces the results of the authors' research on software defined radio design, thus it is new. The proposed architectures can be used for the transmitter and for the receiver devices of a cognitive radio respectively. By using them we avoid the need to replicate the sampled data sections for simultaneously transmitting and receiving more signals with arbitrary bandwidths over multiple desired center frequencies. We also gain the capability to partition the signal spectra before transmitting them and to perfectly reassemble them in the receiver after the base-band shifting.

The core of the proposed transmitter structure is the variant of the standard  $M$ -path polyphase up converter channelizer that is able to perform 2-to- $M$  up sampling while shifting, by aliasing, all the channels to desired center frequencies. It has been presented in Section 1.07.5 of this same chapter. In the following we use the term synthesis channelizer for referring to this structure. This name has been given for the tasks that this structure accomplishes when embedded in a SDR transmitter. When the input signals have bandwidths wider than the synthesis channelizer bandwidth they are pre-processed through small down converter channelizers that disassemble the signals' bandwidths into reduced bandwidth sub-channels which are matched to the base-line synthesis channelizer bandwidths. Complex sampled frequency rotators are used to offset the signal spectra by arbitrary fractions of the sample frequency before processing them through the channelizers. This is necessary for the completely arbitrary center frequency positioning of the signals.

On the other side the variation of the standard  $M$ -path down converter channelizer, presented in Section 1.07.6, represents the key element of the proposed receiver. It is able to perform  $M$ -to-2 down sampling while simultaneously down converting, by aliasing, all the received signal spectra having arbitrary bandwidths. In the following, we refer to this structure as analysis channelizer. Post-processing channelizers, that are a smaller version of the synthesis channelizer composing the transmitter, are used for reassembling, from the analysis channelizer base-line channels, signal bandwidths wider than the analysis channelizer channel bandwidth. Possible residual frequency offsets can be easily solved with the aid of a digital complex heterodyne while a post-analysis block performs further channelization,

when it is required, for separating signal spectra falling in the same channelizer channel after the analysis processing.

### 1.07.9.1 Proposed digital down converter architecture

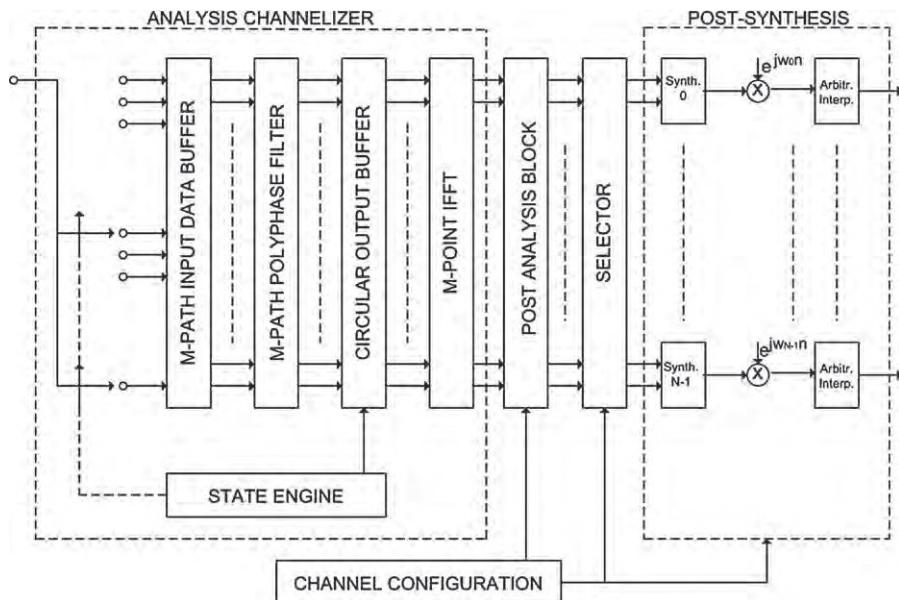
The standard  $M$ -path polyphase down converter channelizer is able to shift to base-band, by aliasing, the input signals that are exactly located on the channel's center frequencies,  $kf_c$ , where  $k = 0, 1, \dots, M-1$ . In this engine the center frequencies of the channels are integer multiples of the channelizer output sampling frequency as well as the channelizer channel bandwidth.

We recall here that the standard  $M$ -path down sampling channelizer simultaneously down converts, by aliasing, from fixed center frequencies to base-band,  $M$  bands narrower than the channelizer channel bandwidth. If  $kf_c \pm \Delta f_k$  are the center frequencies of the input spectra, where  $\pm \Delta f_k$  are the arbitrary frequency offsets from the channel center frequencies,  $kf_c$ , after the down conversion process the output spectra will be shifted by the same offset from DC, i.e., the down converted signals will be centered at the frequency locations  $\pm \Delta f_k$  because the polyphase down converter channelizer only compensates the  $kf_c$  frequency terms. It is unable to compensate the frequency offsets,  $\pm \Delta f_k$ , that are responsible for the arbitrary center frequency positioning of the input signals. The frequency offset compensation is one of the motivations for which we need to add further processing blocks in the digital down converter design when it is intended to be used in a software radio receiver. Other issues, like receiver resolution, base-band reassembly of wide bandwidths and arbitrary interpolation of the base-band shifted signals need to be considered too. All these issues are addressed and solved in the following paragraphs.

We also recall here that the standard  $M$ -to-1 down converter channelizer with the channel spacing and channel bandwidth equal to the output sampling frequency is critically sampled (see upper plot of Figure 7.29). In order to use it in a cognitive radio receiver, we need to modify it, redesigning the channel spacing, channel bandwidth, and output sampling frequency. We presented one of these modifications in Section 1.07.5, where the  $M$ -to-2 down converter channelizer has been derived. Also the selection of the low-pass prototype filter is an important issue that has to be addressed in order to avoid energy losses while the signal is processed.

Figure 7.66 shows the complete block diagram of the proposed down converting chain. The input signal is processed, at first, by an analysis channelizer which is the modified version of the standard  $M$ -to-1 down converter channelizer presented in Section 1.07.5. By performing an  $M/2$ -to-1 down sampling of the input time series this engine simultaneously shifts all the aliased channels to base-band presenting an output sampling rate that is twice the sample rate of the standard  $M$ -to-1 channelizer. Nyquist filters are used, as low-pass prototype, for avoiding energy losses while processing the signals. They are briefly described in one the next subsections. The interested reader can find more details on Nyquist filter design in [6].

Note that if more signals or even spectral fragments belonging to different signals, are channelized in the same channel, further processing is needed to separate them at the output of the analysis down converter. The extra filtering process is performed in the post-analysis block that is connected to a channel configuration block that provides the receiver with the necessary information about the input signal bandwidths and center frequencies. Different options can be implemented for simplifying the design of the post-analysis block. One of the possible options is to decrease the bandwidth of the perfect reconstruction prototype low-pass filter and to increase the number of points of the IFFT block in the

**FIGURE 7.66**

Block diagram of the proposed down converter; analysis down converter channelizer, post analysis block, synthesis up converter channelizers, complex frequency rotators and arbitrary interpolators.

analysis channelizer. By designing the channel spacing and bandwidth to accommodate the most likely expected signal width we minimize the possibility of having more than one signal in each channel. It is also possible to modify the analysis channelizer for having a more convenient frequency positioning of the aliased channels. The optimal choice, of course, depends on the receiver application and on the workload requirements.

At the output of the analysis channelizer, most of the signals with bandwidths narrower than the channel bandwidth are already down converted by means of the channelization process. However the spectra wider than the channel bandwidth, or the spectra that, as a consequence of the arbitrary center frequency positioning, are processed by two or more adjacent channels, have been fragmented and their segments have all been aliased to the first Nyquist zone; these segments need to be recomposed before being translated to DC. The recombination of spectral fragments is the task performed by the synthesis channelizers. They have been presented in Section 9.5.1 and their block diagram is depicted in Figure 7.59. These are small 2-to- $P_n$  polyphase up converters in which the IFFT size,  $P_n$ , is properly chosen in order to span the bandwidth of the  $n$ th received signal spectrum. We summarize the reason for including the synthesizers in our design in this way: at the output of the  $M$ -to-2 analysis channelizer all the signals have been down sampled and their spectra, or their spectral fragments, have been translated to the first Nyquist zone by the channelizing process. In order to reassemble the segments into a wider bandwidth super channel, the time series from each segment must be up sampled and frequency shifted to their appropriate positions so that they can be added together for forming the time series corresponding to the wider bandwidth assembled signal.

At the end of the down conversion and up conversion processes, all the received spectra have been frequency translated and, when necessary, recomposed in the first Nyquist zone. The analysis down converter channelizer shifted to base-band the signal spectra and their fragments, that are exactly centered on its channels' center frequencies. This engine was not able to compensate the frequency offsets deriving from the arbitrary frequency positioning of the received signals.

The frequency offset compensation task is accomplished by the complex frequency rotators that follow the small up converter synthesizers. They are connected with the channel configuration block that provides them proper information about the signal center frequencies.

Once the signals are perfectly centered at DC, arbitrary interpolators are used to adjust their sampling rate to provide us exactly two samples per symbol needed for the further processing stages that are performed in the digital receiver.

#### **1.07.9.1.1 Post-analysis block and synthesis up converter channelizers**

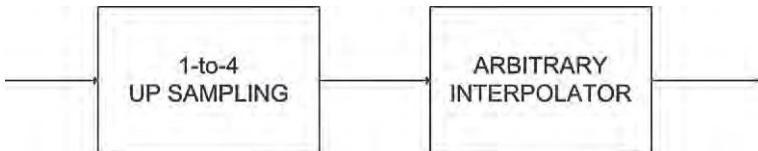
At the output of the analysis channelizer all the channels have been aliased to base-band. As a consequence of that, the channelized segments of the received spectrum have been aliased to the first Nyquist zone (see Figure 7.61).

The following three options, that cover all the possible cases of bandwidths positioning in the channelized spectrum, have to be considered and solved, by further processing the analysis channelizer outputs, in order to achieve, at the end of the analysis-synthesis chain, all the received bands shifted to base-band:

1. The aliased base-band channel could contain only one signal spectrum whose bandwidth is narrower than the channel bandwidth. The carrier frequency of the signal, generally, does not coincide with the center frequency of the channel.
2. The aliased base-band channel could contain two or more spectra, or also their fragments, belonging to different signals. These spectra are arbitrarily positioned in the channel bandwidth.
3. The base-band channel could contain only one spectral fragment belonging to one of the received signals which has bandwidth larger than the channel bandwidth.

In the first option, at the output of the analysis channelizer, the signal spectra that are entirely contained in one single channel reside in the first Nyquist zone. Eventually the receiver has to compensate the frequency offsets derived from their arbitrary center frequency positioning and resample them for obtaining the desired output sampling rate of two samples per symbol. The complex frequency rotators and the arbitrary interpolators perform these tasks at the end of the receiver chain.

For the case in which two or more spectra are processed by one single channel, more tasks need to be performed before frequency offset compensation and arbitrary interpolation occur. We, in fact, have to separate, by filtering, the bands, or their fragments, belonging to different signals before processing all of them independently. The separation task is performed by the post-analysis block. It performs a further channelization process that filters, from every base-band aliased channel, the bands belonging to different signals. Note that some of the filtered signals, and precisely the ones with bandwidths narrower than the channel bandwidth and entirely contained in the channel, only have to be frequency shifted before being delivered to the arbitrary interpolator; else, the spectral fragments belonging to signals processed in more than one channel have to be passed through the up converter synthesizers before being frequency shifted and resampled.

**FIGURE 7.67**

Two stage arbitrary interpolator.

In the third case, that concerns the signals with bandwidths wider than the channel spacing, the analysis channelizer partitioned the bandwidths into several fragments and aliased every fragments to baseband. In order to recombine them we must first, up sample each input time series and second, translate them to their proper spectral region. We can then form the sum to obtain the super channel representation of the original signal bandwidth. Those are exactly the tasks performed by the synthesis channelizers.

#### **1.07.9.1.2 High quality arbitrary interpolator**

After all the bands have been translated to zero frequency, we need to resample them for obtaining exactly the two samples per symbol needed for the subsequent processing performed in the receiver.

In this subsection we present the high quality interpolator structure used for obtaining two samples per symbol needed for the second and third tier processing tasks in a digital receiver. It is well known that the dynamic range of an arbitrary interpolator should match the system's quantization noise level [6]. The error due to the linear interpolation process, in fact, is not observable if it is below the noise level attributed to the signal quantization process. Since the error due to the  $b$ -bit quantized signal is  $2^{-b}$ , the interpolation error or, equivalently, the level of residual spectral artifacts has to be below this threshold. In other words, if the oversampling factor,  $N$ , satisfies Eq. (7.34), then the interpolation error will not be noticeable:

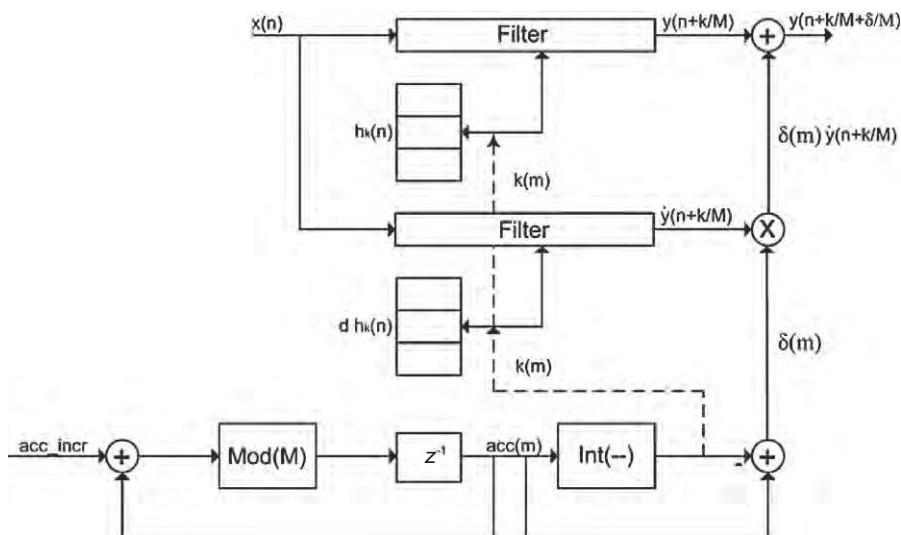
$$\left[ \frac{1}{2N} \right]^2 \leq \frac{1}{2^b}. \quad (7.34)$$

Thus, if we interpolate a 16-bit data set, we keep the spectral artifacts below the quantization noise level if  $N$  is greater than or equal to 128. To interpolate the signals by a factor of  $N = 128$ , we break the up sampling process of the base-band centered spectra in two stages.

As depicted in Figure 7.67, we perform an initial 1-to-4 up sampling followed by a 1-to-32 up sampling obtained by using the high quality arbitrary interpolator shown in Figure 7.68. The initial 4-times oversampling has the effect of strongly decreasing the length of the polyphase filters used in the interpolator which significantly reduces the total workload of this structure. More details on this topic can be found in [6].

Note that the initial 1-to-4 signal up sampling can either be achieved by using two half-band filters in cascade or we can previously up sample the signals by properly choosing  $P_n$ , the IFFT size of the synthesis channelizers. Also, at the output of the synthesizers some of the signals could be already up sampled by some factors because the IFFT size of the synthesizer has to be an even number [6] and it also has to satisfy Nyquist criteria for every signal band.

The arbitrary interpolator shown in Figure 7.68 performs the linear interpolation between two available adjacent signal samples (two neighbor interpolation). The interpolation process operates as follows:

**FIGURE 7.68**

Block diagram of high quality arbitrary interpolator.

A new input data sample is delivered to the interpolation filter register on each accumulator overflow. The integer part of the accumulator content selects one of the  $M$  filter weights,  $h_k(n)$ , and one of the  $M$  derivative filter weights,  $d h_k(n)$ , while the filters compute the amplitude  $y(n + k/M)$  and the first derivative  $\dot{y}(n + k/M)$  at the  $k$ th interpolated point in the  $M$ -point interpolated output grid. Then the structure uses the local Taylor series to form the interpolated sample value between grid points. The output clock directs the accumulator process to add the desired increment  $\Delta$  (acc\_incr in Figure 7.68) to the modulo- $M$  accumulator and increments the output index  $m$ . The increment  $\Delta$  can be computed according to the following equation:  $\Delta = M f_{\text{in}} / f_{\text{out}}$ . This process continues until the accumulator overflows at one of the output clock increments. Upon this overflow the input index is incremented, a new input data point is shifted into the filter, and the process continues as before.

### 1.07.9.1.3 Nyquist filters

Particular attention has to be paid in designing the low-pass prototype filters used in the analysis channelizer. The signal spectra have to be randomly located in the frequency domain and their bandwidths can easily span and occupy more than one base-line channel; also, in the signal disassembling and reassembling processes, we need to collect all the energy corresponding to a single signal without losses.

Nyquist filter presents the interesting property of having a band edge gain equal to 0.5 (or  $-6$  dB). By using this filter as prototype in our channelizers, we place  $M$  of them across the whole spanned spectrum with each filter centered on  $k f_s / M$ . All adjacent filters exhibit  $-6$  dB overlap at their band-edges. The channelizer working under this configuration is able to collect all the signal energy across its full operating spectrum range even if signals occupy more than one adjacent channel or it resides

in the channel's overlapping transition bandwidths. In the following paragraphs, for completeness, we provide a brief introduction on Nyquist filters.

Nyquist pulses is the name given to the wave shapes  $f(n)$  required to communicate over band-limited channels with no inter-symbol interference (ISI). When sampled at equally spaced time increments they have to verify the requirement of Eq. (7.35) which is known as Nyquist pulse criterion for zero ISI:

$$f(n) = \begin{cases} 0 & \text{if } n \neq 0, \\ 1 & \text{if } n = 0. \end{cases} \quad (7.35)$$

There are infinite such functions that satisfy this set of restrictions. The one with minimum bandwidth is the ubiquitous  $\sin(x)/x$  which is variously known as the cardinal pulse when used for band limited interpolation and the Nyquist pulse when used in pulse shaping. The transform of this wave shape,  $R(f)$ , is the unit area rectangle with spectral support 1/T Hz. Unfortunately this waveform is non-causal and further it resides on an infinite support. If the pulse resided on a finite support we could delay the response sufficiently for it to be causal. We have to form finite support approximations to the  $\sin(x)/x$  pulse. The first approximation to this pulse is obtained by convolving the rectangular spectrum,  $R(f)$ , with an even symmetric, continuous spectrum  $W(f)$  with finite support  $\alpha/T$ . The convolution between  $R(f)$  and  $W(f)$  in the frequency domain is equivalent to a product in the time domain between the  $r(t)$  and  $w(t)$ , where  $w(t)$  is the inverse transform of  $W(f)$ . The effect of the spectral convolution is to increase the two-sided bandwidth from  $1/T$  to  $(1+\alpha)/T$ . The excess bandwidth  $\alpha/T$  is the cost we incur to form filters on finite support. The term  $\alpha$  is called the roll-off factor and is typically on the order of 0.5–0.1 with many systems using values of  $\alpha = 0.2$ . The transition bandwidth caused by the convolution is seen to exhibit odd symmetry about the half amplitude point of the original rectangular spectrum. This is a desired consequence of requiring even symmetry for the convolving spectral mass function. When the windowed signal is sampled at the symbol rate 1/T Hz, the spectral component residing beyond the  $1/T$  bandwidth folds about the frequency  $\pm 1/2T$  into the original bandwidth. This folded spectral component supplies the additional amplitude required to bring the spectrum to the constant amplitude of  $R(f)$ .

Following this reasoning, we note that the significant amplitude of the windowed wave shape is confined to an interval of approximate width  $4T/\alpha$  so that a filter with  $\alpha = 0.2$  spans approximately  $20T$ , or 20 symbol durations. We can elect to simply truncate the windowed impulse response to obtain a finite support filter, and often choose the truncation points at  $\pm 2T/\alpha$ . A second window, a rectangle, performs this truncation. The result of this second windowing operation is a second spectral convolution with its transform. This second convolution induces pass-band ripple and out-of-band side-lobes in the spectrum of the finite support Nyquist filter. The description of this band-limited spectrum normalized to unity pass-band gain is presented in Eq. (7.36):

$$H_{\text{NYQ}}(w) = \begin{cases} 1 & \text{for } \frac{|w|}{w_{\text{SYM}}} \leq (1 - \alpha), \\ 0.5 * \left\{ 1 + \cos \left\{ \frac{2\pi}{\alpha} \left[ \frac{w}{w_{\text{SYM}}} - (1 - \alpha) \right] \right\} \right\} & \text{for } (1 - \alpha) \leq \frac{|w|}{w_{\text{SYM}}} \leq (1 + \alpha), \\ 0 & \text{for } \frac{|w|}{w_{\text{SYM}}} \geq (1 + \alpha). \end{cases} \quad (7.36)$$

The continuous time domain expression for the cosine-tapered Nyquist filter is shown in Eq. (7.37). Here we see the windowing operation of the Nyquist pulse as a product with the window that is the

transform of the half-cosine spectrum:

$$h_{\text{NYQ}}(t) = f_{\text{SYM}} \frac{\sin(\pi f_{\text{SYM}} t)}{(\pi f_{\text{SYM}} t)} \frac{\cos(\pi \alpha f_{\text{SYM}} t)}{[1 - (2\alpha f_{\text{SYM}} t)^2]}. \quad (7.37)$$

Since the Nyquist filter is band limited, we can form the samples of a digital filter by sampling the impulse response of the continuous filter. Normally this involves two operations. The first is a scaling factor applied to the impulse response by dividing by the sample rate, and the second is the sampling process in which we replace  $t$  with  $nT_s$  or  $n/f_s$ . The sample rate must exceed the two-sided bandwidth of the filter that, due to the excess bandwidth, is wider than the symbol rate. It is standard to select the sample rate  $f_s$  to be an integer multiple of the symbol rate  $f_{\text{SYM}}$  so that the filter operates at  $M$ -samples per symbol. It is common to operate the filter at 4 or 8 samples per symbol.

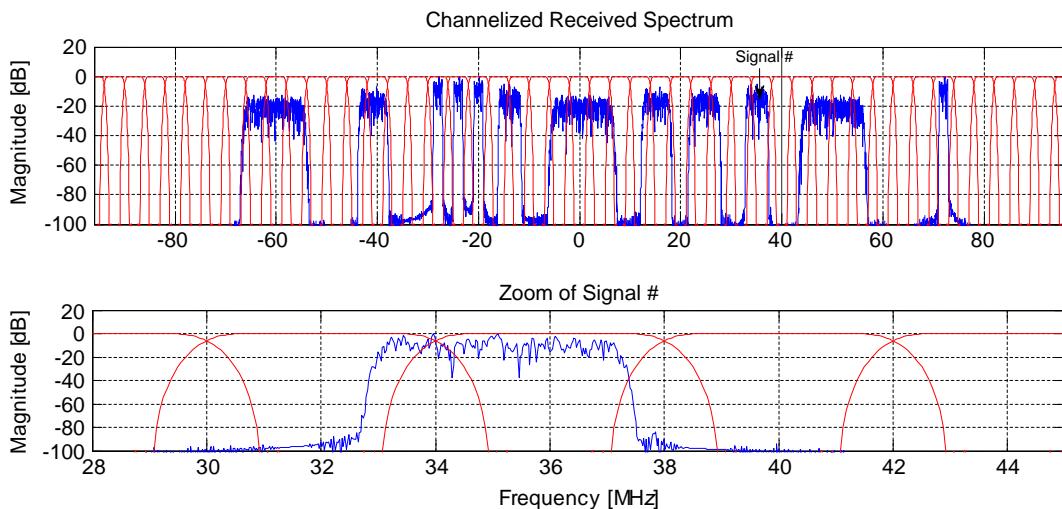
In the design of the proposed analysis channelizer, the Nyquist prototype low-pass filter has to be designed with its two sided 3 dB bandwidth equal to  $1/M$ th of the channelizer input sampling frequency. This is equivalent to the filter impulse response having approximately  $M$  samples between its peak and first zero crossing and having approximately  $M$  samples between its zero crossings. The integer  $M$  is also the size of the IFFT as well as the number of channels in the analysis channelizer. The prototype filter must also exhibit reasonable transition bandwidth and sufficient out of band attenuation or stop-band level. We designed our system for a dynamic range of 80 dB which is the dynamic range of a 16-bit processor.

### 1.07.9.2 Digital down converter simulation results

For simulation purposes, in this section we consider, at the input to the  $M$ -to-2 analysis channelizer, a composite spectrum that contains twelve QAM signals with five different bandwidths randomly located in the frequency domain. In particular, the signal constellations are 4-QAM, 16-QAM, 64-QAM, and 256-QAM while the signal bandwidths are 1.572132 MHz, 3.892191 MHz, 5.056941 MHz, 5.360537 MHz, and 11.11302 MHz respectively. It is clear that we used two different signal bandwidths for the 256-QAM constellation (5.360537 MHz and 11.11302 MHz).

The signals are shaped by square-root Nyquist filters with 20% excess bandwidth. At the input of the analysis channelizer all the signals are resampled for achieving 192 MHz sample rate. The spectrum is shown in Figure 7.56. In particular, the upper subplot of Figure 7.69 shows, superimposed on the composite received spectrum, the 61 channels of the analysis channelizer. It is easy to recognize that the received spectra are arbitrarily located. Their center frequencies do not coincide with the channel center frequencies. That is clearly shown in the lower subplot of Figure 7.69 in which the enlarged view of one of the received signals is presented. The arbitrary signal positioning is the reason for which the polyphase down converter channelizer, by itself, is not able to directly shift them to DC. The IFFT size of the analysis  $M$ -to-2 down converter channelizer is  $M = 48$  with an output sample rate of 8 MHz.

Figure 7.70 shows the impulse response and the magnitude response of the designed prototype low-pass Nyquist filter. It is designed to have 48 samples per symbol. Its length is 1200 taps while its stop band attenuation is  $-80$  dB. Note that, since this filter is  $M$ -path partitioned, the length of each filter in the  $M$ -path bank is only 25 taps.

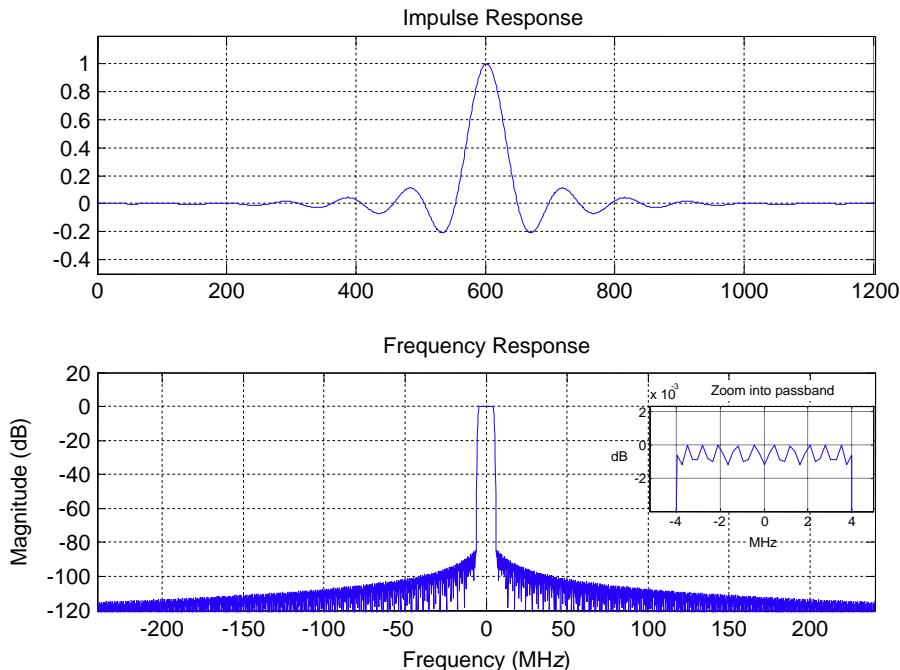
**FIGURE 7.69**

Channelized received spectrum and zoom of one of the received signals.

The 61 spectra, at the output of the analysis channelizer, are depicted in Figure 7.71. The down converter channelizer has partitioned the entire frequency range into 48 segments. It is easy to recognize in this figure the different spectra composing the received signal. Note that, because of the arbitrary frequency positioning, it is possible that also the signals having bandwidths narrower than the channelizer channel bandwidth occupy more than one analysis channelizer channel. Also in this case, before shifting these signals to zero frequency by using complex frequency rotators, we need to pass them through a synthesis channelizer that reassembles their fragments.

Before delivering the analysis channelizer outputs to the synthesis channelizers, we need to separate, if necessary, by filtering, those spectra that belong to different signals lying in the same channel. An example of this is represented by channel 30 in Figure 7.71. It contains fragments of two spectra belonging to different signals. The filter design in the post analysis block, of course, depends on the bands that have to be resolved. Their sample rate has to be the same as the analysis channelizer output rate (8 MHz). An example of post analysis filters along with the filtered signals, for the 30th analysis channelizer channel, is shown in Figure 7.72. In particular, the signal spectra and the filters used for separating them are shown in the upper subplot while the separated spectra are shown in the lower subplots.

At the output of the post analysis block all the spectra, with bandwidths narrower than the analysis channel bandwidth, lying in a single analysis channel, can be directly delivered to the complex heterodyne that translates them to DC. All the other signals, the ones with bandwidths wider than the analysis channel bandwidth and, the ones with bandwidths narrower than the analysis channelizer channel bandwidth that, as consequence of the analysis process, are delivered to two different channelizer outputs, need to be processed by the small synthesis channelizers. We have, in fact, to up sample, frequency shift, and recombine the time series from coupled analysis channelizer outputs. In the example of Figure 7.69

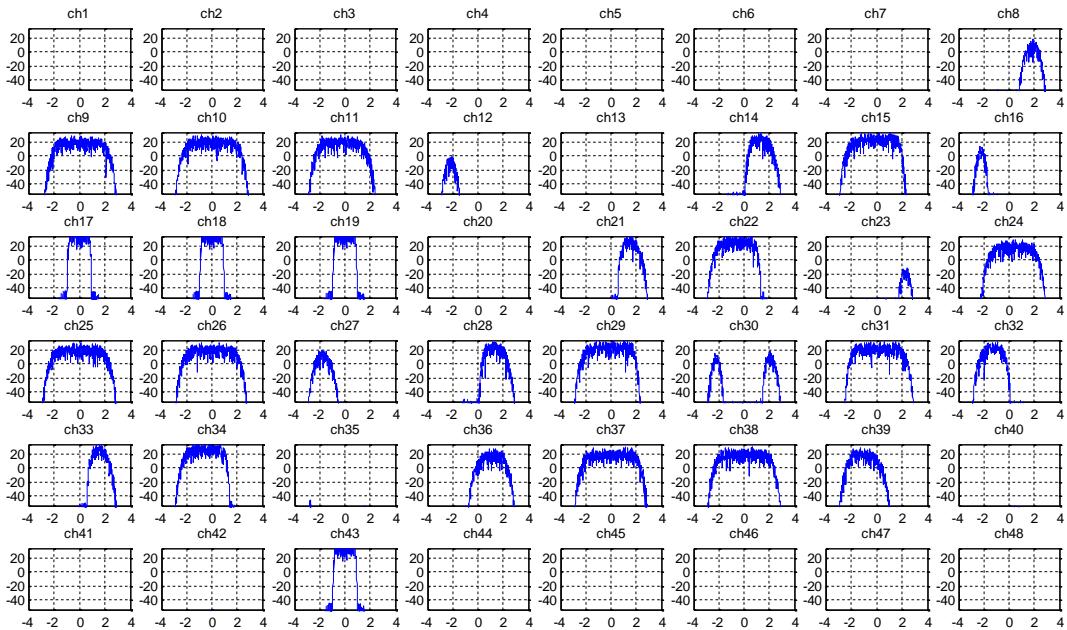
**FIGURE 7.70**

Nyquist prototype filter.

four of the received spectra, the narrowest ones, are directly sent to the frequency rotators. Eight of them are processed through the synthesis channelizer. The IFFT sizes  $P_n$ , with  $n = 0, 1, 2$ , of the synthesizers we selected to process the three remaining bandwidths are:  $P_0 = 6$ ,  $P_1 = 4$ , and  $P_2 = 2$  points. These sizes are the minimum possible chosen to satisfy Nyquist sampling criterion for each output signal. Note that because of the structure of the synthesizers, we can only have an even number of IFFT points [6].

At the output of the synthesizer, all the signal fragments are recombined in base-band but still some of them have a residual frequency offset that needs to be compensated. The signal spectra, before frequency offset compensation are shown in Figure 7.73. Here it is clearly visible that many of the signals are not centered at DC (the red<sup>1</sup> line in Figure 7.73 represents the signals' center frequency). We compensate these frequency offsets by using complex frequency rotators. Note that by estimating the signal energy in the synthesizer channels, we could easily recover the carrier offsets affecting the received spectra. The frequency offset recovery is another of the many applications of the polyphase channelizer. Other papers are being prepared for addressing this topic.

<sup>1</sup>For interpretation of color in Figure 7.73, the reader is referred to the web version of this book.

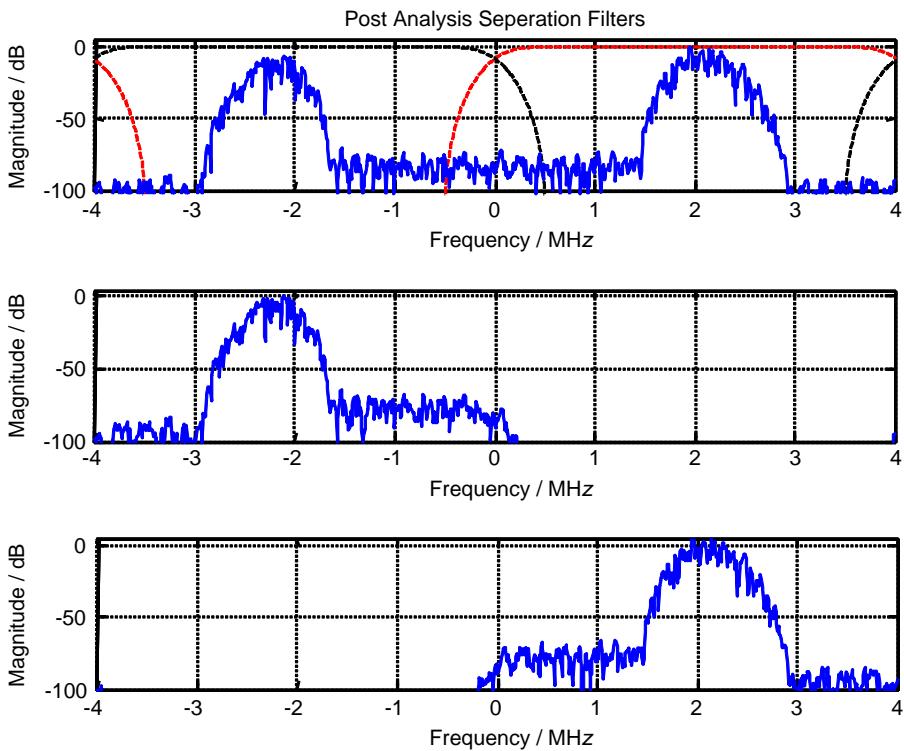
**FIGURE 7.71**

Analysis channelizer outputs.

When they are DC centered, the signals need to be resampled. The signals at the outputs of the synthesis channelizers have different sampling frequencies. All that we need at this point is to interpolate them for obtaining exactly two samples per symbol for each of them. We use an arbitrary interpolator for achieving the sample rate conversion. The interpolated, DC shifted, signals are shown in Figure 7.74. We also match-filtered each of the eight channelized and reconstructed time signals and present their constellations in Figure 7.75, here we see that all of the QAM constellations are perfectly reconstructed which demonstrates the correct functionality of the proposed receiver.

### 1.07.9.3 Proposed up converter architecture

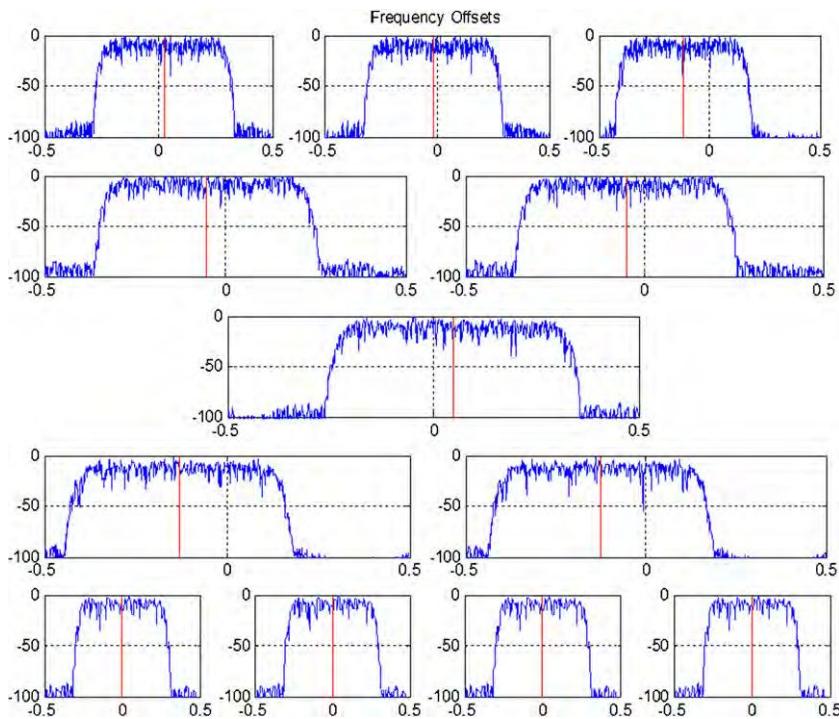
Figure 7.76 shows the complete block diagram of the proposed up converter, which has the structure that is a dual of the down converter structure presented in the previous sections. It is composed of a 2-to- $M$  up sampler synthesis channelizer, which has been presented in Section 9.5.1, preceded by  $N$  pre-processing analysis blocks which are small  $P_n$ -to-2 analysis down converter channelizers. Their number,  $N$ , corresponds to the number of the signal spectra wider than the synthesis channelizer channel bandwidth. Their IFFT size,  $P_n$ , is chosen to completely span the bandwidth of the  $n$ th input signal spectrum. Their task is to decompose the wider input spectra into  $P_n$  partitions matching the bandwidth and sample rate of the base-line synthesis channelizer which will coherently recombine them in the receiver.

**FIGURE 7.72**

Post analysis filters and filtered signals for channel 30 of the analysis channelizer.

All the input signals to the 2-to- $M$  synthesis channelizer with bandwidths less than or equal to the channel spacing are to be up sampled and translated from base-band to the selected center frequency by the channelizing process. For these signals we may only have to filter and resample to obtain the desired sampling rate of two samples per channel bandwidth. However, we expect that many of the spectra we presented to the synthesis channelizer have bandwidths that are wider than the synthesizer channel spacing. For accommodating these signals we need to use the small analysis channelizers that partition their bandwidths into several fragments, translate all of them to base-band, and reduce their sample rate to twice the channel bandwidth. The analysis channelizers are designed as  $P_n$ -to-2 polyphase down converter channelizers where  $P_n$  is approximately twice the number of base-line channels spanned by the wideband spectrum. They have been presented in Section 1.07.5 of this same document and Figure 7.55 shows their block diagram.

We briefly recall here that such a system accepts  $N/2$  input samples and outputs time samples from  $N$  output channels. The  $N$ -point input buffer is fed by a dual input commutator with an  $N/2$  sample offset. The  $N$ -path filter contains polynomials of the form  $H_r(z^2)$  and  $z^{-1}H_{r+N/2}(z^2)$  in its upper and lower halves respectively. The circular output buffer performs the phase rotation alignment of the IFFT block

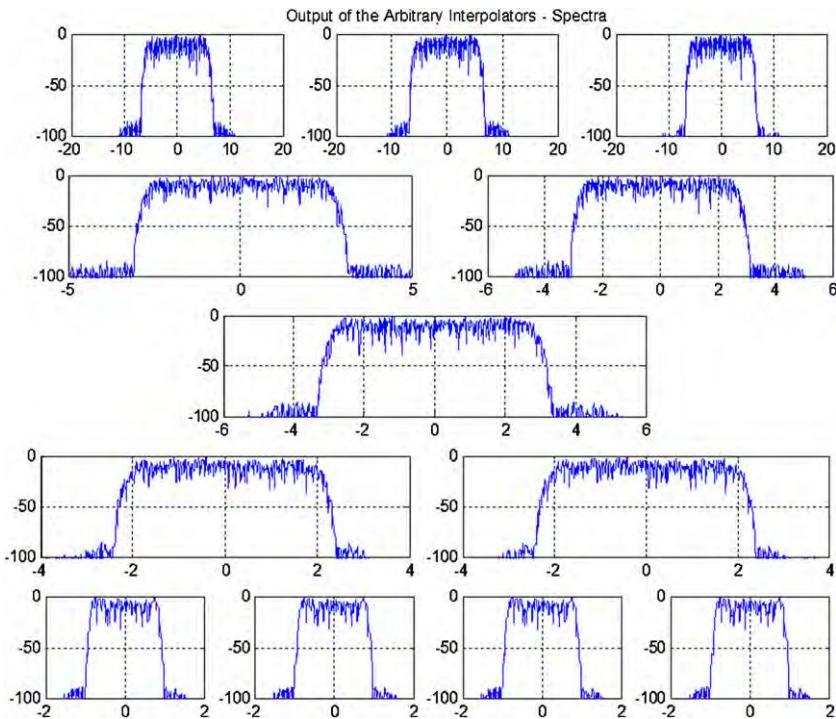
**FIGURE 7.73**

Log magnitude of synthesizer outputs with frequency offsets in the normalized frequency domain.

with the  $N/2$  stride shifting time origin of the  $N$ -path polyphase filter. The IFFT size for these analysis channelizers,  $P_n$ , is also the number of their filter outputs.

In order to recombine the segmented signal components we have to satisfy the Nyquist criteria for their sum. Since these are pre-processor analysis channelizers that feed the  $M$ -path synthesis channelizer we must have their output sample rate two times their channel bandwidth. We can achieve this by selecting  $P_n$  to be approximately twice the number of channels being merged in the synthesizer and setting its input sample rate to be  $2f_s P_n$  so that the pre-processor output rate per channel is the required  $2f_s$ . Remember that the IFFT block sizes must be even to perform the 2-to- $N$  resampling by the technique described in Section 1.07.5; also remember that, an actual system may have a few standard size IFFT's to be used for the analysis channelizer and the user may have to choose from the small list of available block sizes.

The channel selector placed between the analysis channelizer bank and the synthesis channelizer also connected with the input channel configuration block provides the correct outputs from the pre-processor analysis channelizer to the input synthesizer while the channel configuration block provides the necessary information to the selector block that is connected to the synthesis input series. The

**FIGURE 7.74**

Log magnitude of the heterodyned and interpolated spectra in the frequency domain [MHz].

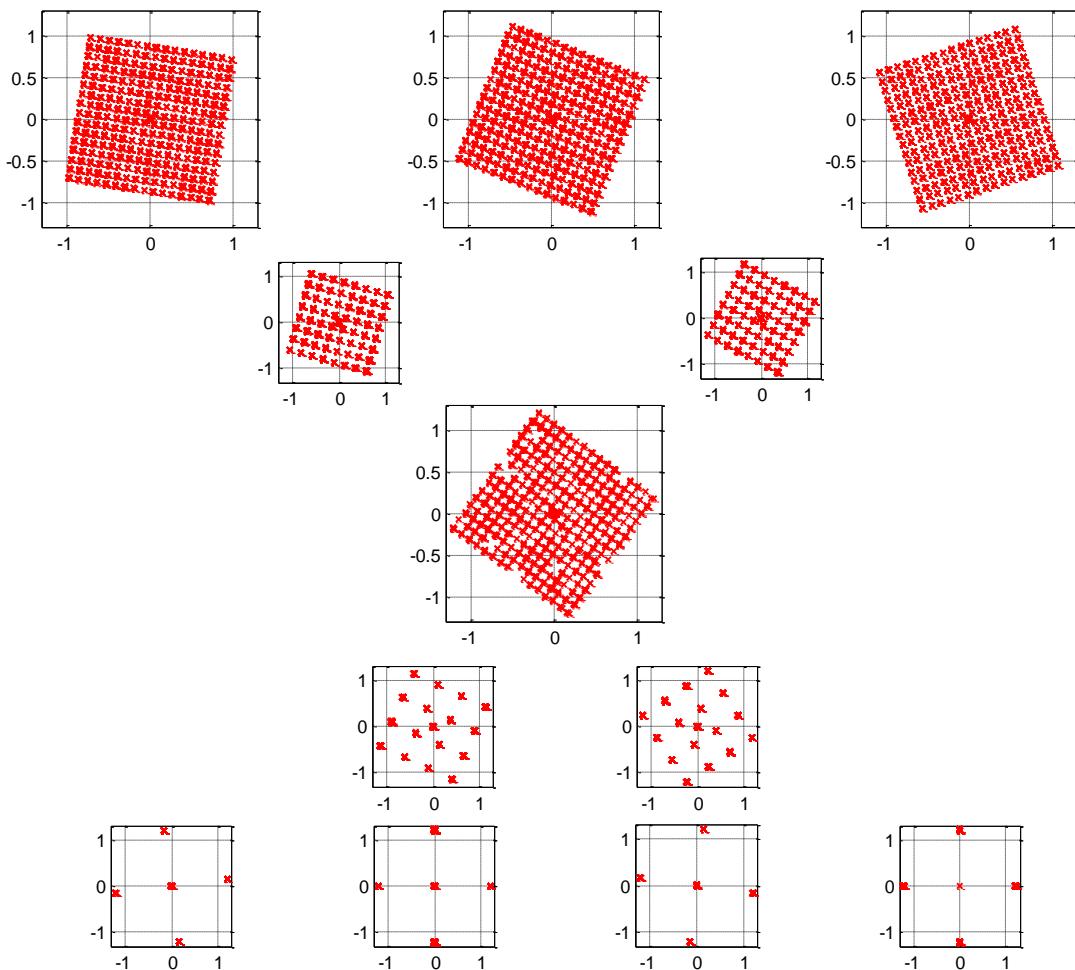
selector routes all the segments required to assemble the wider bandwidth channel to the synthesizer which performs their frequency shift and reassembly.

Depending on the desired center frequency of the disassembled spectrum an earlier complex heterodyne may be required before the analyzers to shift of the about to be disassembled signals with the proper frequency offset.

#### 1.07.9.4 Digital up converter simulation results

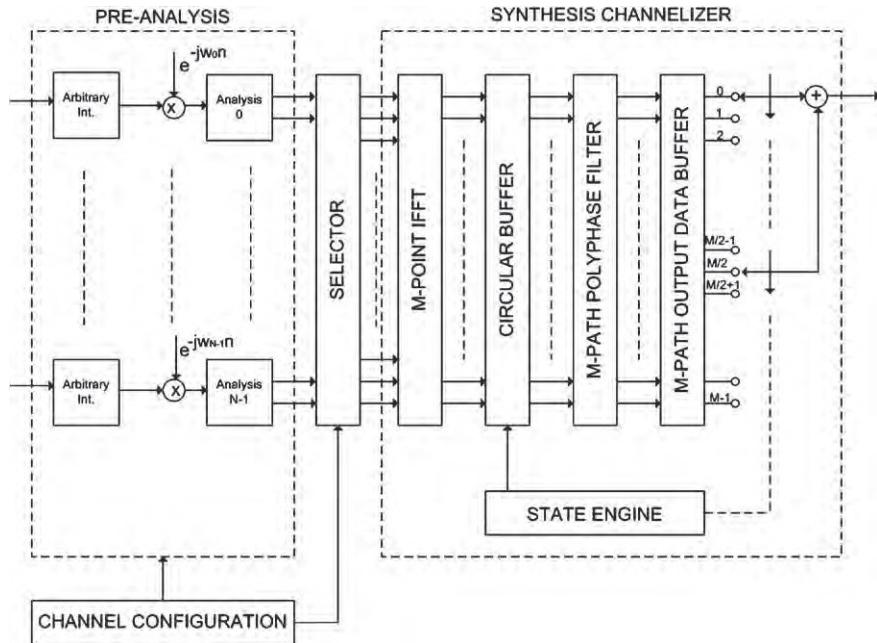
In the simulation results shown in this section we consider a set of eight distinct base-band input signals to be delivered to the 2-to- $M$  up converter synthesis channelizer. These are QPSK signals with three different bandwidths as shown in Figure 7.77. The symbol rates chosen for the three signals denoted 1, 2, and 3, are 7.5, 15.0, and 30.0 MHz. These signals are shaped by square-root Nyquist filters with 25% excess bandwidth, hence the two sided bandwidths are 7.5·1.25, 15.0·1.25, and 30.0·1.25 MHz respectively.

The IFFT size of the base-line 2-to- $M$  up converter channelizer is  $M = 48$  with 10 MHz channel spacing for which the required input sample rate per input signal is 20 MHz and for which the output

**FIGURE 7.75**

Heterodyned and interpolated constellations.

sample rate will be 480 MHz. For ease of signal generation, all three signals were shaped and up-sampled to 60 MHz sample rate with shaping filters designed for 8, 4, and 2 samples per symbol. Signal 1, represented in the first line of Figure 7.77, is down sampled 3-to-1 to obtain the desired 20 MHz sample rate for the synthesis channelizer. Signals 2 and 3, respectively on the second and third line of Figure 7.77, are down sampled 6-to-2 in the six point IFFT analysis channelizers which form 10 MHz channels at 20 MHz sample rate. Signal 2 is spanned by three 10 MHz channels which will feed 3 input ports of the 48 point synthesizer IFFT while signal 3 is spanned by five 10 MHz channels which will

**FIGURE 7.76**

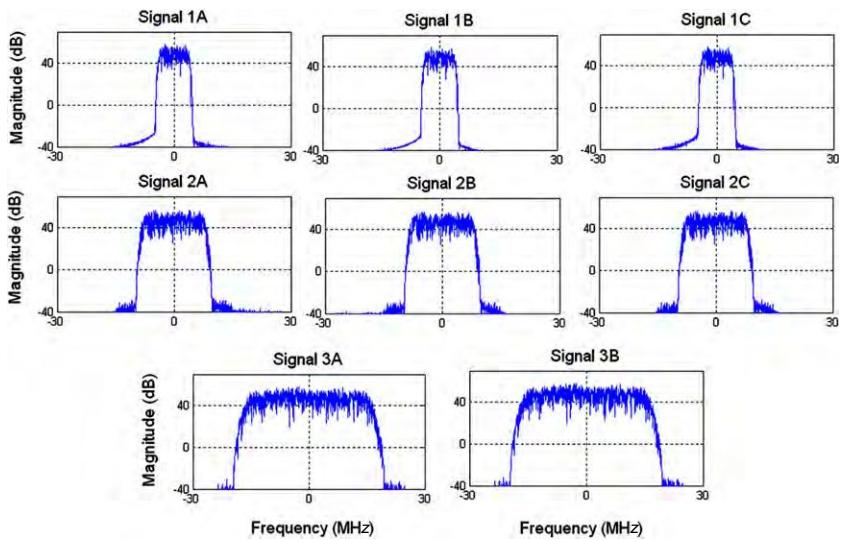
Block diagram of the proposed up converter; arbitrary interpolators, complex frequency rotators, analysis down converter channelizers and synthesis up converter channelizer.

feed 5 input ports of the 48 point IFFT. The IFFT inputs are controlled by a channel control block that routes them to the proper synthesis channelizer ports.

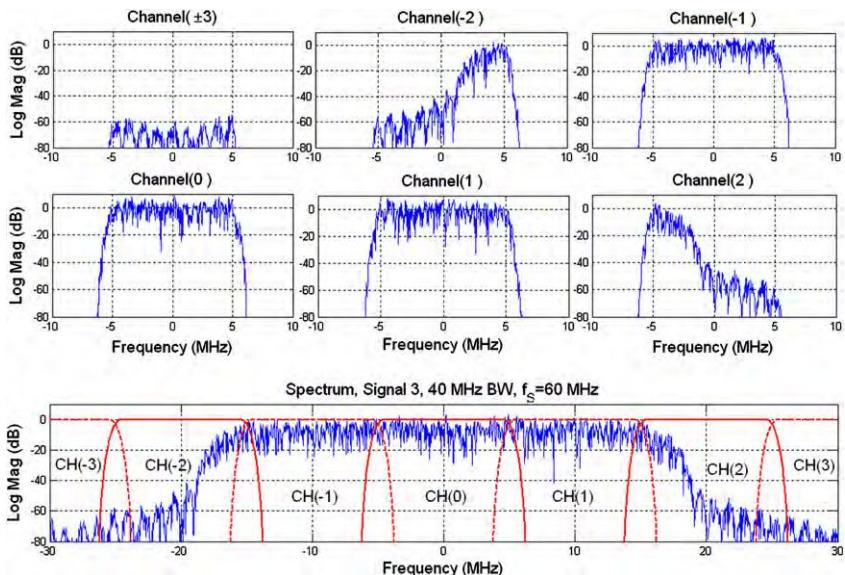
Any of the signals presented to the analysis channelizers is previously shifted to a desired frequency offset, by means of a complex heterodyne if required. The output channels that span the offset input bandwidth of the analysis channelizer are the channels passed on the synthesizer and these change due to a frequency offset. The inserted base-band frequency offset will survive the synthesis channelizer.

Figure 7.78 shows all the spectra of the output time series from the 6-channel polyphase 6-to-2 analysis channelizer engine processing signal 3 which is the signal with the widest band. Also seen, in the same figure, is the spectrum of signal 3 and the frequency response of all the 6 channels formed by one of the analysis channelizers. Note the spectra in the upper subplots have been filtered by the channelizer with the 10 MHz Nyquist pass band frequency response, have been translated to base-band and have been sampled at 20 MHz. Five of these segments are presented to the five input ports of the 2-to-48 synthesis channelizer centered on the desired frequency translation index.

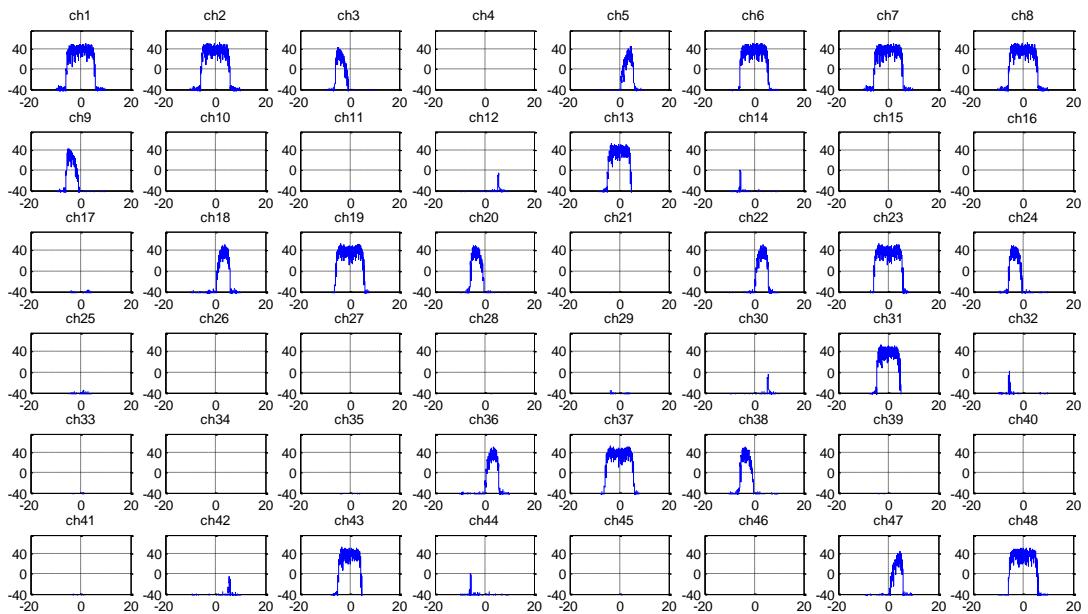
The up converter polyphase synthesis channelizer accepts time sequences sampled at 20 MHz with bandwidths less than 10 MHz. We have delivered three signals that satisfy these constraints along with four signals that were conditioned by analysis channelizers that partitioned their bandwidths into segments that also satisfied the input signal constraints. The spectra of the separate components delivered to the synthesis channelizer are shown in Figure 7.79. It is easy to recognize in this figure the different

**FIGURE 7.77**

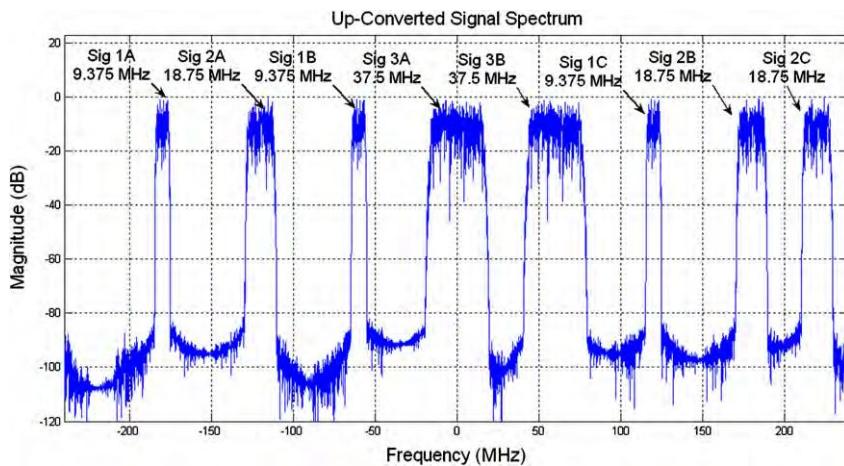
Example of base-band spectra to be up converted.

**FIGURE 7.78**

Spectral fragments formed by 6-channel 6-to-2 down sample analysis channelizer processing signal 3.

**FIGURE 7.79**

2-to- $M$  up converter channelizer inputs.

**FIGURE 7.80**

Up converted synthesized spectrum with unequal bandwidths fragmented and defragmented spectral components.

spectra composing the received signal. Here we see that filters 4-through-8 are segments of a single frequency band fragmented in Figure 7.78.

At this point, we up sample, frequency shift, and recombine the time series from coupled channel outputs using the synthesis channelizer. The frequency shifted spectra of the eight signals, including six that have been fragmented in analysis channelizers and then defragmented in the synthesis channelizer are plotted in Figure 7.80.

### 1.07.10 Closing comments

This chapter had the intention of providing the basic concepts on multirate signal processing and polyphase filter banks to which, when an IDFT block and a commutator are applied, are known as polyphase channelizer because of the processing task that they perform on the input signals. This chapter also had the intention of presenting innovative results on software and cognitive radio designs. While the polyphase filter banks and polyphase channelizers are well known topics in the DSP area, the software radio design is still an open research topic.

The document started with preliminaries, in Section 1.07.2, on the resampling process of a digital signal as opposed to the sampling process of an analog signal. In Section 1.07.3 an introduction on digital filters has been provided and the differences with the analog filters have been explained. In Section 1.07.4 an introduction on the window method for digital filter design has been provided. Multirate filters have been defined and explained in Section 1.07.5 while the polyphase decomposition of a prototype filter has been introduced in Section 1.07.6 along with the standard up sampler and down sampler polyphase channelizers. In Section 1.07.7 the modifications of the standard polyphase channelizer that allow us to change the output sampling rate have been presented. These engines are the basic components of the proposed architectures (presented in Sections 1.07.8 and 1.07.9): the synthesis and analysis channelizers, that are suitable for being used as software defined transmitter and receiver respectively.

The synthesis channelizer, in fact, when supported by small analysis channelizers, is able to simultaneously up convert multiple signals having arbitrary bandwidths to randomly located center frequencies. In this engine small analysis channelizers pre-process wider bandwidth signals into segments acceptable to the following synthesis channelizer that up samples, translates to the proper center frequency, and reassembles them. A channel selector block, connected with a channel configuration block, is used to properly deliver the analysis channelizer outputs to the synthesis channelizer.

On the other side of the communication chain, the analysis channelizer, that is thought for being embedded in a SDR receiver, when supported by small synthesis channelizers, is able to simultaneously demodulate these signals. A channel configuration block, inserted in the receiver, communicates with a selector block that properly delivers the analysis channelizer outputs to the synthesizer up converters that follow it. These synthesizers up sample, translate and reassemble the spectral fragments when they belong to the same source signal. Nyquist prototype low-pass filters, which are perfect reconstruction filters, are used to allow the reconstruction of the signal fragments without energy losses.

Complex frequency rotators are used for compensating residual frequency offsets and arbitrary interpolators provide us exactly two samples per symbol required for the following processing tasks of the receiver.

Theoretical reasoning and simulation results are presented, for both the receiver and the transmitter, in order to demonstrate their correct functionality.

Note that because in both of the proposed DUC and DDC structures, the IFFT sizes of the small synthesis and analysis channelizers are chosen in order to give us at least two samples per symbol for every processed bandwidth, they result to be very efficient, from a computational point of view, when the received signal is composed of spectra with widely varying bandwidths.

Slightly different versions of channelizers can be used based on different input signals and/or for adding functionalities to the proposed architectures. Channelizers are, in fact, highly efficient and very flexible structures. Among their most common applications we find spectral analysis, modem synchronization (phase, frequency and time) and channel equalization.

## Glossary

Filter	calculation procedure that transforms the input signals into others
Digital filter	filter that operates on digital signals
Multirate filter	digital filter that contains a mechanism to increase or decrease the sampling rate while processing input signals
Polyphase filter	digital filter partitioned by following Eq. (7.3)
Polyphase channelizer	flexible digital device which can arbitrarily change the sample rate and the bandwidth of the input signal and can also select randomly located Nyquist zones
Software radio	radio device in which the digitization of the signal occurs before the intermediate frequency translation
Cognitive radio	software radio with the capability to adapt its operating parameters according to the interactions with the surrounding radio environment

## References

- [1] George M. Kranc, Input-output analysis of multirate feedback systems, *IRE Trans. Automat. Control* AC-2 (1956) 21–28.
- [2] E.I. Jury, F.J. Mullin, The analysis of sampled-data control systems with a periodically time-varying sampling rate, *IRE Trans. Automat. Control*-4 (1959) 15–20.
- [3] E.I. Jury, F.J. Mullin, The analysis of sampled data control system with a periodically time varying sampling rate, *IRE Trans. Automat. Control*. AC-4 (1959) 15–21.
- [4] P. Vary, U. Heute, A short-time spectrum analyzer with polyphase-network and DFT, *Signal Process.* 2 (1) (1980) 55–65.
- [5] R.W. Schafer, L.R. Rabiner, Design of digital filter banks for speech analysis, *Bell Syst. Tech. J.* 50 (1971) 3097–3115.
- [6] F.J. Harris, *Multirate Signal Processing for Communication systems*, Prentice Hall, Upper Saddle River, New Jersey, 2004.
- [7] F. Harris, W. Lowdermilk, Software defined radio, *IEEE Instrum. Meas. Mag.* (2010).

- [8] F. Harris, C. Dick, X. Chen, E. Venosa, Wideband 160 channel polyphase filter bank cable TV channelizer, IET Signal Process. (2010).
- [9] F. Harris, C. Dick, X. Chen, E. Venosa, *M*-path channelizer with arbitrary center frequency assignments, in: WPMC 2010, March 2010.
- [10] X. Chen, E. Venosa, F. Harris, Polyphase synthesis filter bank up-converts unequal channel bandwidths with arbitrary center frequencies-design II, in: SDR 2010, Washington, DC, December 2010.
- [11] E. Venosa, X. Chen, F. Harris, Polyphase analysis filter bank down-converts unequal channel bandwidths with arbitrary center frequencies-design II, in: SDR 2010, Washington, DC, December 2010.
- [12] M. Bellanger, G. Bonnerot, M. Coudreuse, Digital filtering by polyphase network: application to sample-rate alteration and filter bank, IEEE Trans. Acoust. Speech Signal Process. 24 (2) (1976) 109–114.
- [13] M. Bellanger, J. Daguet, TDM-FDM transmultiplexer: digital polyphase and FFT, IEEE Trans. Commun. 22 (9) (1974) 1199–1205.
- [14] M. Ribey, Exploration of transmultiplexers in telecommunication networks, IEEE Trans. Commun. COM-30 (1982) 1493–1497.
- [15] B. Wang, K.J. Ray Liu, Advances in cognitive radio networks: a survey, IEEE J. Sel. Top. Signal Process. 5 (1) (2011).
- [16] A. Sahai, S.M. Mishra, R. Tandra, K.A. Woyach, Cognitive radios for spectrum sharing, IEEE Signal Process. Mag. 26 (1) (2009).
- [17] Matthew Sherman, Christian Rodriguez, Ranga Reddy, IEEE standards supporting cognitive radio and networks, dynamic spectrum access, and coexistence, IEEE Commun. Mag. 46 (7) (2008).
- [18] Jun Ma, G. Ye Li, Biing Hwang (Fred) Juang, Signal processing in cognitive radio, Proc. IEEE 97 (7) (2009).
- [19] T. Ulversøy, Software defined radio: challenges and opportunities, IEEE Commun. Surv. 2 (4) (2010).
- [20] R. Bagheri, A. Mirzaei, M.E. Heidari, Software-defined radio receiver: dream to reality, IEEE Commun. Mag. 44 (8) (2006).
- [21] J. Mitola, Cognitive radio architecture evolution, Proc. IEEE 97 (4) (2009).
- [22] R. Tandra, S.M. Mishra, A. Sahai, What is a spectrum hole and what does it take to recognize one? Proc. IEEE 97 (5) (2009).
- [23] Fredric J. Harris, On detecting white space spectra for spectral scavenging in cognitive radios, Springer J. Wireless Pers. Commun. <http://dx.doi.org/10.1007/s11277-008-9460-y>.
- [24] F. Harris, C. Dick, M. Rice, Digital receivers and transmitters using polyphase filter banks for wireless communications, Microwave Theory Tech. 51 (4) (2003) 1395–1412 (special issue).

## 8

# Modern Transform Design for Practical Audio/Image/Video Coding Applications

Trac D. Tran

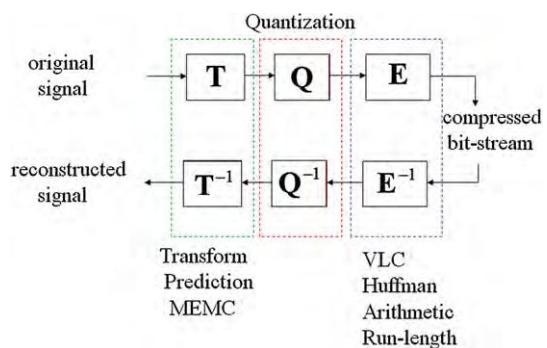
*Department of Electrical and Computer Engineering, The Johns Hopkins University, MD, USA*

## 1.8.1 Introduction

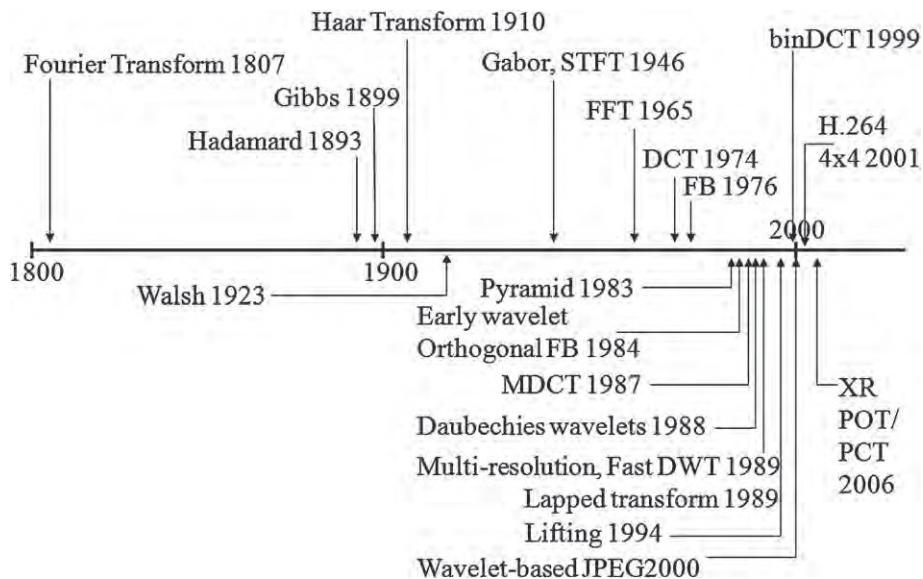
With the recent explosion in popularity of the Internet, wireless communication, and portable computing, the demands for and the interests in digital multimedia (digitized speech, audio, image, video, computer graphics, and their combination) are growing exponentially. A high-performance filter bank is typically at the heart of every state-of-the-art digital multimedia system whose compression paradigm is illustrated in Figure 8.1. Current popular international image/video compression standards such as JPEG [1], MPEG2 [2], and H.263 [3] are all based on the  $8 \times 8$  Discrete Cosine Transform (DCT), an 8-channel 8-tap orthogonal filter bank with fast computational algorithms based on sparse factorizations. The latest image compression JPEG2000 standard [4] is employing the wavelet transform, an iterated 2-channel bi-orthogonal filter bank, as its de-correlation engine. Of extreme importance is the ability to design a filter bank that can fully exploit (i) the statistical properties of a particular signal or class of signals; (ii) the goals of the applications; and (iii) the computational resources available.

Although the roots of signal transformations can be traced back to works in the 19th century by prominent mathematicians such as Laplace and Fourier, the exciting development of modern, practical digital transforms and filter banks only have a few years of history [5–8] as illustrated in the timeline of Figure 8.2. These new systems provide more effective tools to represent digital signals not only for analysis but also for processing and compression purposes. For multimedia signals, representations by transform coefficients are usually more compact and efficient than the time representations, but are just as informative. Taking advantage of the normally sparse transform coefficient matrix, we can perform a majority of signal processing tasks directly in the transform domain at a lower level of computational complexity.

On the other hand, fast, efficient, and low-cost transforms have played a crucial role in revolutionary advances throughout the history of digital signal processing. The Discrete Fourier Transform (DFT) is an excellent signal analysis tool. However, its popularity in practical systems was not widespread until the discovery of the Fast Fourier Transform (FFT) by Cooley and Tukey in 1965 that reduces the complexity level from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(n \log n)$  [9]. Another example, the  $8 \times 8$  Discrete Cosine Transform [10], has a theoretically elegant closed-form expression. It was chosen as the transform-of-choice in most of current image and video coding standards mainly because it possesses various fast algorithms [11] that can reduce the transform complexity from 64 multiplications and 56 additions to as low as 5 multiplications and 29 additions per 8 transform coefficients.

**FIGURE 8.1**

Standard compression paradigm in current digital multimedia systems.

**FIGURE 8.2**

A historical timeline of significant developments in signal decomposition and transformation.

In this chapter, we review the design and development progress of modern transforms for application in digital image/video coding and processing. These new state-of-the-art transforms provide versatility and flexibility in time-frequency mapping, coding performance, cost of implementation as well as integration into modern digital multimedia processing/communication systems. We assert that the most successful philosophy in transform design is to construct highly-complex systems from modular

cascades of similar, simple building blocks, each propagating a set of desired transform properties. These novel transforms will be designed to have as many of the following features as possible:

- orthogonal or at least near-orthogonal,
- symmetric/anti-symmetric (linear-phase) basis functions,
- good stop-band attenuation,
- smoothness of the basis functions for perceptually pleasant reconstruction,
- integer-to-integer mapping capability with exact recovery for a unifying lossless/lossy coding framework,
- fast computation and efficient implementations in both software and hardware: multiplier-less property for low-power real-time systems; in-place computation; low memory buffering; VLSI-friendliness with high modularity and regularity in construction; facilitating region-of-interest coding/decoding and computational parallelism.

The organization of the chapter is as follows. In Section 1.8.2, we offer a review of important background materials, concepts, motivations, and previous related works in transform design. Common design strategy and desirable cost functions are discussed in Section 1.8.3. Next, Section 1.8.4 describes the direct scaling-based design method for modern integer-coefficient transforms. Section 1.8.5 presents a totally different philosophy in transformation design via general parameterization and construction of polyphase matrices based on lifting steps (also known as ladder structures). The section also discusses in details the subset of solutions that allows the construction and implementation of various dyadic-coefficient transforms purely from shift-and-add operations. The design procedure of spectral factorization of maxflat half-band filters that lead to popular wavelet filter pairs is described in Section 1.8.6. Advanced design mainly via optimization with a larger number of channels and/or longer filter lengths is covered in Section 1.8.7. Numerous design examples along with current practical applications will be demonstrated throughout along with discussions on each design's advantages as well as disadvantages and other interesting properties. Finally, we briefly summarize the chapter with a few concluding remarks in Section 1.8.8.

## 1.8.2 Background and fundamentals

We provide in this section the notation, common background, a brief description of the historical development of transforms (especially for compression applications), and a discussion on desirable properties of a state-of-the-art modern transform in multimedia coding and processing.

### 1.8.2.1 Notation

Let  $\mathcal{R}$ ,  $\mathcal{Q}$ , and  $\mathcal{Z}$  denote the sets of real numbers, rational numbers, and integers, respectively. Also, let  $\mathcal{D}$  denote the set of dyadic rationals, i.e., all rational numbers that can be represented in the form of  $\frac{k}{2^m}$  where  $k, m \in \mathcal{Z}$ . Bold-faced lower case characters are used to denote vectors while bold-faced upper case characters are used to denote matrices. Let  $\mathbf{A}^T$ ,  $\mathbf{A}^{-1}$ ,  $|\mathbf{A}|$ , and  $a_{ij}$  denote respectively the transpose, the inverse, the determinant, and the  $i$ th  $j$ th element of the matrix  $\mathbf{A}$ . Several special matrices with reserved symbols are: the polyphase matrix of the analysis bank  $\mathbf{E}(z)$ , the polyphase matrix of the synthesis bank  $\mathbf{R}(z)$ , the identity matrix  $\mathbf{I}$ , the reversal or anti-diagonal matrix  $\mathbf{J}$ , the null matrix

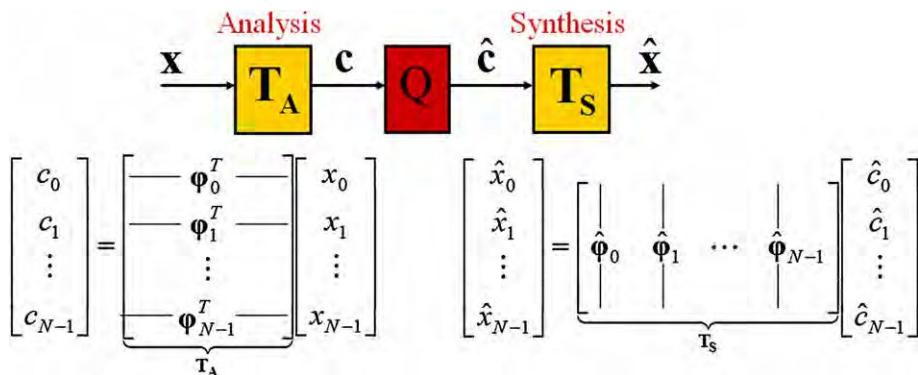
(or vector) **0**, the unity vector **1**, the permutation matrix **P**, and the diagonal matrix **D**. The letter  $M$  is usually reserved for the number of channels of the filter bank or the size of the transform. Finally, we denote the  $\ell_p$  norm of an  $N$ -point vector  $\mathbf{x}$  as  $\|\mathbf{x}\|_p = \left(\sum_{i=0}^{N-1} |x_i|^p\right)^{1/p}$  where the  $\ell_2$  norm  $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \mathbf{x}}$  is of special interest.

### 1.8.2.2 Transform fundamentals

A transform is most often thought of as a linear mapping which can be carried out by a simple matrix multiplication. A block transform  $\mathbf{T}_A$  of size  $M$  is simply an  $M \times M$  scalar matrix, mapping the signal vector  $\mathbf{x}$  to its corresponding transform coefficients  $\mathbf{c}$ . Here, the subscript  $A$  is used to denote the analysis transform often employed in an encoder to analyze or to de-correlate input data samples. At the decoder, we need the inverse operator  $\mathbf{T}_S$  to recover  $\mathbf{x}$  back where the subscript  $S$  denotes the synthesis or signal reconstruction operation. The series of mappings are depicted in Figure 8.3. The operation  $\mathbf{Q}$  in the middle of Figure 8.3 represents either approximation (in the form of quantization for instance), processing, coding, communication or any combination thereof. Obviously, if the coefficients can be recovered losslessly or  $\hat{\mathbf{c}} = \mathbf{c}$ , we simply require  $\mathbf{T}_S$  to be the exact inverse of  $\mathbf{T}_A$  to reconstruct the input signal perfectly.

The columns  $\phi_i$  of  $\mathbf{T}_A^T$  are often called the analysis basis functions while the columns  $\hat{\phi}_i$  of  $\mathbf{T}_S$  are called the synthesis basis functions.  $\mathbf{T}_A$  is said to be an invertible transform or a transform that has perfect reconstruction or bi-orthogonal when  $\mathbf{T}_S \mathbf{T}_A = \mathbf{T}_A \mathbf{T}_S = \mathbf{I}$ . In the special case that the synthesis transform has real coefficients and is simply the transpose of the analysis transform  $\mathbf{T}_S = \mathbf{T}_A^{-1} = \mathbf{T}_A^T$ , then the transform  $\mathbf{T}_A$  is said to be orthogonal. In this case, we can conceptually think of representing the  $N$ -point input signal as

$$\mathbf{x} = c_0 \phi_0 + c_1 \phi_1 + \cdots + c_{N-1} \phi_{N-1} = \sum_{i=0}^{N-1} c_i \phi_i, \quad \text{where } c_i = \langle \phi_i, \mathbf{x} \rangle. \quad (8.1)$$



**FIGURE 8.3**

Signal decomposition and reconstruction as matrix operations.

However, our interest is to design the transform that yields the sparsest and most compact representation

$$\mathbf{x} = \sum_{i \in \mathcal{S}; |\mathcal{S}|=K \ll N} c_i \boldsymbol{\phi}_i. \quad (8.2)$$

In (8.2) above, the set of significant coefficients are indexed by the set  $\mathcal{S}$  whose cardinality  $K$  is much smaller than the signal dimension  $N$ .

For two-dimensional signals such as still images or frames in video sequences, we typically employ the separable transformation approach where all rows are transformed and then all columns are transformed or vice versa. Mathematically, if we let  $\mathbf{X}$  be the image of interest in the form of a matrix, then the transform coefficients  $\mathbf{C}$  of  $\mathbf{X}$  can be computed as

$$\mathbf{C} = \mathbf{T}_A \mathbf{X} \mathbf{T}_A^T, \quad (8.3)$$

while the inverse synthesis mapping can be computed as

$$\mathbf{X} = \mathbf{T}_S \mathbf{C} \mathbf{T}_S^T. \quad (8.4)$$

Note that the computation order of rows or columns in 8.3 and 8.4 is not critical and the strategy can be straightforwardly extended to signals of any higher dimension.

### 1.8.2.3 Optimal orthogonal transform

Obtaining the sparsest representation as in (8.2) above is certainly a signal-dependent task. Let  $\mathbf{c}$  be the coefficients of the input signal  $\mathbf{x}$ , obtained from a certain linear transformation  $\mathbf{T}$ . Let us further assume that the input signal  $\mathbf{x}$  can be modeled as a white-sense stationary stochastic process. Then, the correlation matrix of the output signal  $\mathbf{c}$  is

$$\mathbf{R}_{cc} = E\{\mathbf{T}\mathbf{x}\mathbf{T}^T\} = \mathbf{T}\mathbf{R}_{xx}\mathbf{T}^T.$$

The optimal orthogonal linear transform  $\mathbf{T}$  here is the one that can fully de-correlate  $\mathbf{x}$ . In order to achieve this,  $\mathbf{R}_{cc}$  must be a diagonal matrix with the eigenvalues of  $\mathbf{R}_{xx}$  being its diagonal entries, and each row of  $\mathbf{T}$  being an eigenvector of  $\mathbf{R}_{xx}$ . This optimal transform is often called the Karhunen-Loëve Transform (KLT) of signal  $\mathbf{x}$ , also known as principal component analysis (PCA) or the Hotelling transform. Obviously, the KLT depends on the auto-correlation matrix  $\mathbf{R}_{xx}$ . Thus it is signal dependent, computationally expensive, and costly to communicate to the decoder if the signal of interest is not stationary. To find fast algorithms and signal-independent transforms for practical applications, a typical approach is to start with a simplified stationary signal model, try to find the approximation of its KLT, and then develop fast computational algorithms of the resulting approximation. The Discrete Cosine Transform (DCT) is developed following this approach [10,11].

### 1.8.2.4 Popular transforms in signal processing: DFT, WHT, DCT

The problem of transform design has undergone several important evolutions. In the beginning, transformation is mainly thought of as a continuous-time continuous-amplitude change of basis operation

(and often in infinite-dimension functional space) to solve complicated set of equations. Discrete-time transforms are often obtained from sampling the continuous-time version. Four popular discrete-time transforms in signal processing are the Discrete Fourier Transform (DFT), the Walsh-Hadamard Transform (WHT), the type-II Discrete Cosine Transform (DCT) and the type-IV Discrete Cosine Transform (DCT), shown below in the matrix representation format respectively.

$$[\mathbf{F}] = \frac{1}{\sqrt{M}} [W_M^{mn}]; \quad 0 \leq m, n \leq M - 1; \quad W_M = e^{-j \frac{2\pi}{M}}. \quad (8.5)$$

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad \mathbf{H}_{2^m} = \mathbf{H}_2 \otimes \mathbf{H}_{2^{m-1}} = \begin{bmatrix} \mathbf{H}_{2^{m-1}} & \mathbf{H}_{2^{m-1}} \\ \mathbf{H}_{2^{m-1}} & -\mathbf{H}_{2^{m-1}} \end{bmatrix}; \quad m \in \mathcal{Z}, m \geq 2. \quad (8.6)$$

$$[\mathbf{C}_M^{II}] = \sqrt{\frac{2}{M}} \left[ K_m \cos \left( \frac{m(n + 1/2)\pi}{M} \right) \right]; \quad 0 \leq m, n \leq M - 1; \quad K_i = \begin{cases} \frac{1}{\sqrt{2}} & i = 0 \\ 1 & i > 0 \end{cases}. \quad (8.7)$$

$$[\mathbf{C}_M^{IV}] = \sqrt{\frac{2}{M}} \left[ \cos \left( \frac{(m + 1/2)(n + 1/2)\pi}{M} \right) \right]; \quad 0 \leq m, n \leq M - 1. \quad (8.8)$$

Out of the three mentioned above, only the Hadamard transform has integer coefficients. Unfortunately, it is too simplistic to offer reasonable coding performance. On the other hand, the DCT offers very good compression performance but it has irrational coefficients. The DFT is not only irrational, but it also has complex-valued coefficients.

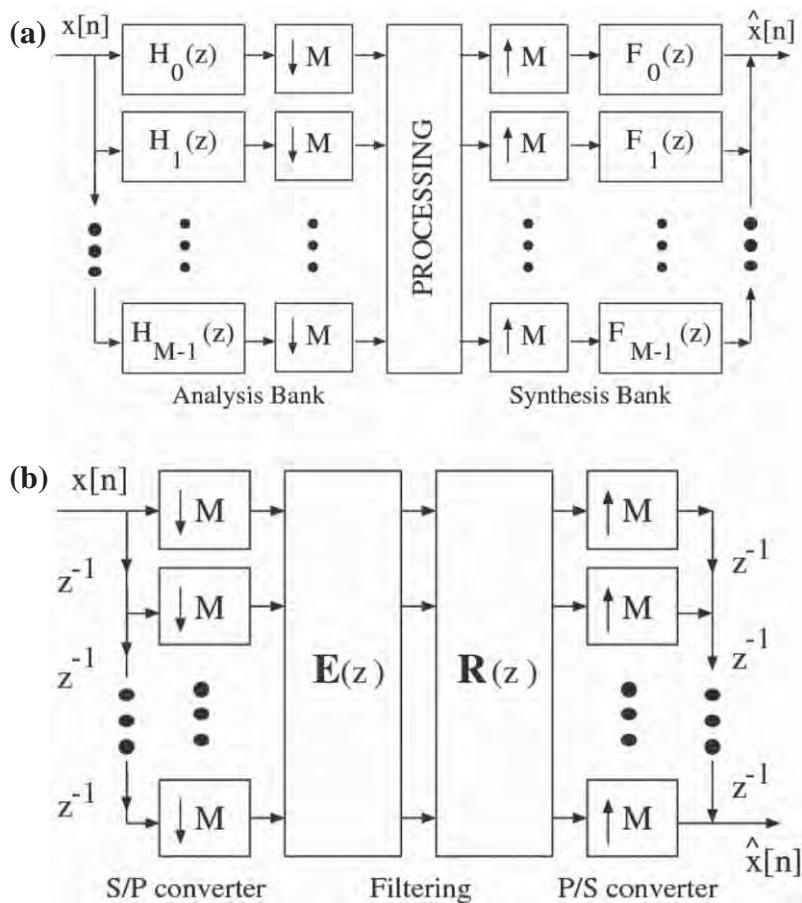
### 1.8.2.5 The filter bank connection

A typical discrete-time, maximally-decimated  $M$ -channel filter bank is depicted in Figure 8.4. At the analysis stage, the input signal  $x[n]$  is passed through a bank of  $M$  analysis filters  $H_i(z)$ ,  $i = 0, 1, \dots, M - 1$ , each of which preserves a frequency band. These  $M$  filtered signals are then decimated by  $M$  to preserve the system's overall sampling rate. The resulting subband signals can be coded, processed, and/or transmitted independently or jointly. At the synthesis stage, the subbands are combined by a set of upsamplers and  $M$  synthesis filters  $F_i(z)$ ,  $i = 0, 1, \dots, M - 1$ , to form the reconstructed signal  $\hat{x}[n]$ . In the absence of processing errors (e.g., quantization), filter banks that yield the output  $\hat{x}[n]$  as a pure delayed version of the input  $x[n]$ , i.e.,  $\hat{x}[n] = x[n - n_0]$ , are called *perfect reconstruction* filter banks. From a traditional transform perspective, the analysis filter  $h_i[n]$  is the  $i$ th analysis basis function  $\phi_i$ . Similarly, the synthesis filter  $f_i[n]$  yields the  $i$ th synthesis basis function  $\hat{\phi}_i$ .

The filter bank in Figure 8.4a can also be represented in terms of its polyphase matrices as shown in Figure 8.4b.  $\mathbf{E}(z)$  is the analysis bank's polyphase matrix and  $\mathbf{R}(z)$  is the synthesis bank's polyphase matrix. Note that both  $\mathbf{E}(z)$  and  $\mathbf{R}(z)$  are  $M \times M$  matrices whose elements are polynomials in  $z$  [7]. If  $\mathbf{E}(z)$  is invertible with a minimum-phase determinant (stable inverse), one can obtain perfect reconstruction by simply choosing  $\mathbf{R}(z) = \mathbf{E}^{-1}(z)$ . In other words, any choice of  $\mathbf{R}(z)$  and  $\mathbf{E}(z)$  that satisfies

$$\mathbf{R}(z)\mathbf{E}(z) = \mathbf{E}(z)\mathbf{R}(z) = z^{-l}\mathbf{I}, \quad l \geq 0 \quad (8.9)$$

yields perfect reconstruction. Since filter banks with finite impulse response (FIR) filters are very valuable in practice, we can restrict the determinants of both polyphase matrices need to be monomials

**FIGURE 8.4**

$M$ -channel filter bank. (a) Conventional representation. (b) Polyphase representation.

[6–8] as well:

$$|\mathbf{R}(z)| = z^{-m} \quad \text{and} \quad |\mathbf{E}(z)| = z^{-n} \quad m, n \in \mathbb{Z}. \quad (8.10)$$

A popular choice of  $\mathbf{R}(z)$  yielding *paraunitary* or *orthogonal* systems is

$$\mathbf{R}(z) = z^{-K} \mathbf{E}^T(z^{-1}), \quad (8.11)$$

where  $K$  is the order of a properly-designed  $\mathbf{E}(z)$ . In the case where  $\mathbf{E}(z)$  may not be paraunitary but (8.9) still holds, the filter bank is said to be *bi-orthogonal*. Now, the design of a complicated  $M$ -band FIR perfect reconstruction filter bank is reduced to choosing appropriate polynomial matrices  $\mathbf{E}(z)$  and  $\mathbf{R}(z)$  such that (8.9) and (8.10) are satisfied.

When the filter length equals the number of channel  $M$ , the polyphase matrices  $\mathbf{E}(z)$  and  $\mathbf{R}(z)$  become scalar matrices of zero order and they become equivalent to the conventional transforms above, i.e.,  $\mathbf{E}(z) = \mathbf{T}_A$  and  $\mathbf{R}(z) = \mathbf{T}_S$ . When the filter length is higher than the number of channel, we have to deal with matrices whose entries are polynomials in  $z$ . In the wavelet transform,  $M = 2$  and we almost never have a scalar polyphase matrix, except the Haar wavelet. When  $M$  equals the support of the input signal, we have a global transform and maximum frequency resolution becomes achievable. On the other hand, when  $M$  is much smaller than the input size (for example, most imaging applications have the transform size of  $16 \times 16$  or less), we have a block transform coding framework where the down-samplers and the delay chain on the left of Figure 8.4b serve as serial-to-parallel blocking mechanism whereas the up-samplers and the delay (or advance) chain on the right of Figure 8.4b implement the parallel-to-serial unblocking mechanism. In this case, the global transform matrices  $\mathbf{T}_A$  and  $\mathbf{T}_S$  have block-diagonal structure and maximum time/space resolution becomes achievable.

### 1.8.2.6 The lapped transform connection

The lapped transform (LT) is defined as a linear transformation that partitions the input signal into small *overlapped* blocks and then processes each block independently. The equivalence between the lapped transform, the filter bank in Section 1.8.2.5, as well as the general linear transformation in Section 1.8.2.2 has been well-established in [12].

Consider the  $M$ -channel perfect-reconstruction filter bank with FIR filters, all of length  $L = KM$  for presentation elegance. The polyphase matrix  $\mathbf{E}(z)$  as shown in Figure 8.4b is of order  $K - 1$  and can be expressed as  $\mathbf{E}(z) = \sum_{i=0}^{K-1} \mathbf{E}_i z^{-1}$ . Similarly, the synthesis polyphase matrix  $\mathbf{R}(z)$  can be represented as  $\mathbf{R}(z) = \sum_{i=0}^{K-1} \mathbf{R}_i z^{-1}$ . With  $\mathbf{J}$  as the time-reversal matrix, if we define

$$\mathbf{E} = [\mathbf{E}_{K-1}\mathbf{J} \cdots \mathbf{E}_1\mathbf{J} \mathbf{E}_0\mathbf{J}] \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} \mathbf{JR}_0 \\ \mathbf{JR}_1 \\ \vdots \\ \mathbf{JR}_{K-1} \end{bmatrix},$$

then the global transform matrices can be shown to take the following forms

$$\begin{aligned} \mathbf{T}_A &= \begin{bmatrix} \ddots & & & \mathbf{0} \\ & \mathbf{E} & & \\ & & \mathbf{E} & \\ & \mathbf{0} & & \ddots \end{bmatrix} \\ &= \begin{bmatrix} \ddots & & & \ddots & & & & \mathbf{0} \\ & \mathbf{E}_{K-1}\mathbf{J} & \mathbf{E}_{K-2}\mathbf{J} & \cdots & \mathbf{E}_0\mathbf{J} & & & \\ & \mathbf{E}_{K-1}\mathbf{J} & \mathbf{E}_{K-2}\mathbf{J} & \cdots & & \mathbf{E}_0\mathbf{J} & & \\ & & \mathbf{E}_{K-1}\mathbf{J} & \mathbf{E}_{K-2}\mathbf{J} & \cdots & & \mathbf{E}_0\mathbf{J} & \\ & \mathbf{0} & & & & & & \ddots & \ddots & \ddots \end{bmatrix} \end{aligned} \tag{8.12}$$

and

$$\begin{aligned}
 \mathbf{T}_S &= \begin{bmatrix} \ddots & & & \mathbf{0} \\ & \mathbf{R} & & \\ & & \mathbf{R} & \\ & & & \ddots \\ \mathbf{0} & & & & \end{bmatrix} \\
 &= \begin{bmatrix} \ddots & & & \mathbf{0} \\ \ddots & \mathbf{JR}_0 & & \\ & \mathbf{JR}_1 & \mathbf{JR}_0 & \\ \vdots & & \mathbf{JR}_1 & \mathbf{JR}_0 \\ & \mathbf{JR}_{K-1} & \vdots & \mathbf{JR}_1 \\ & & & \vdots \\ & & & \mathbf{JR}_{K-1} \\ \mathbf{0} & & & & \ddots \end{bmatrix}. \tag{8.13}
 \end{aligned}$$

Although the blocking windows overlap, the total number of produced transform coefficients is the same as the total number of input samples. Hence, the lapped transform does not introduce any redundancy. In traditional block-transform processing, the number of input samples from each block is the same as the number of basis functions, i.e.,  $L = M$ . In this case, the transform matrix  $\mathbf{P}$  becomes square ( $M \times M$ ) and there is no overlapping between neighboring blocks. However, as the basis vectors of block transforms do not overlap, there may be discontinuities along the boundary regions of the blocks when heavy quantization is involved. Different approximations of those boundary regions in each side of the border may cause an artificial “edge” in between blocks. This is the so-called *blocking* effect. Allowing overlapping block boundary leads to coding improvement over non-overlapped transforms such as the DCT on two counts: (i) from the analysis viewpoint, the overlapping basis takes into account inter-block correlation, hence, provides better energy compaction; (ii) from the synthesis viewpoint, the overlapping basis either eliminates or significantly reduces blocking discontinuities.

However, the global transform matrices  $\mathbf{T}_A$  and  $\mathbf{T}_S$  remain to be very sparse, leading to fast computation of transform coefficients  $\mathbf{c}$ . The philosophy is that a few borrowed samples are enough to eliminate blocking artifacts and to improve coding performance.

### 1.8.3 Design strategy

All of the transforms presented in the previous sections are designed to have high practical value. They all have perfect reconstruction. Some of them even have real and symmetric basis functions. However, for the transforms to achieve high coding performance, several other properties are also needed. Transforms can be obtained using unconstrained nonlinear optimization where some of the popular cost criteria are:

coding gain  $C_{CG}$ , DC leakage  $C_{DC}$ , attenuation around mirror frequencies  $C_M$ , and stop-band attenuation in both analysis and synthesis bank— $C_A$  and  $C_S$ . In the particular field of image compression, all of these criteria are well-known desired properties in yielding the best reconstructed image quality [6]. The cost function in the optimization process can be a weighted linear combination of these measures as follows

$$C_{\text{Overall}} = \alpha_1 C_{CG} + \alpha_2 C_{DC} + \alpha_3 C_M + \alpha_4 C_A + \alpha_5 C_S, \quad (8.14)$$

where each of the individual cost function will be further elaborated in the next section.

### 1.8.3.1 Desirable transform properties

- *Symmetry:*  $\mathbf{J}\phi_i = \pm\phi_i$ . Linear-phase basis functions are critical in image/video processing applications (but not so much in audio processing applications).
- *Orthogonality:* offers mathematically elegance and norm preservation property. However, practical codecs often employ bi-orthogonal transforms (e.g., the 9/7-tap wavelet in JPEG2000). In other words, the relaxation of the tight orthogonal constraint can sometimes provide a much needed flexibility in transform design.
- *Energy compaction:* also known as coding gain

$$C_{CG} = 10 \times \log_{10} \left( \frac{\sigma_{\text{input}}^2}{\left[ \prod_{i=1}^M \sigma_i^2 \|\hat{\phi}_i\|^2 \right]^{1/M}} \right), \quad (8.15)$$

where  $\sigma_{\text{input}}^2$  is the variance of the input signal;  $\sigma_i^2$  is the variance of the  $i$ th subband (generated from the  $i$ th analysis basis function  $\phi_i$ ); and  $\|\hat{\phi}_i\|^2$  is the norm-squared of the  $i$ th synthesis basis function. Note that for orthogonal transforms, this term disappears and Eq. 8.15 reduces down to the ratio (in dB) of the arithmetic mean over the geometric mean of the sub-band variances. The signal model is the commonly used AR(1) process with intersample autocorrelation coefficient  $\rho = 0.95$ . As aforementioned, the coding gain can be thought of as an approximate measure of the transform's energy compaction capability. Among the listed criteria, higher coding gain correlates most consistently with higher objective performance (measured in MSE or PSNR) in coding applications. Transforms with higher coding gain compact more signal energy into a fewer number of coefficients, leading to a sparser, more compact, representation and more efficient entropy coding.

- *Stop-band attenuation:* defined as the summation of the stop-band energy of all analysis and/or synthesis basis functions. It can be formulated as

$$\text{Analysis stop-band attenuation } C_A = \sum_{i=0}^{M-1} \int_{\omega \in \Omega_i} |\phi_i(e^{j\omega})|^2 d\omega, \quad (8.16)$$

$$\text{Synthesis stop-band attenuation } C_S = \sum_{i=0}^{M-1} \int_{\omega \in \Omega_i} |\hat{\phi}_i(e^{j\omega})|^2 d\omega, \quad (8.17)$$

where  $\Omega_i$  denotes the stop-band of the  $i$ th basis function.

Stop-band attenuation is a classical performance criterion in filter design. On the analysis side, the stop-band attenuation cost helps in improving the signal decorrelation and decreasing the amount of aliasing. In meaningful images, we know *a priori* that most of the energy is concentrated in the low frequency region. Hence, besides the few basis functions that are designated to cover this low frequency range, high stop-band attenuation for the remaining in this part of the frequency spectrum becomes extremely desirable. On the contrary, at the synthesis side, the synthesis basis functions (or filters) covering low-frequency bands need to have high stop-band attenuation near and/or at  $\omega = \pi$  to enhance their smoothness.

- *Zero DC leakage:* The DC leakage cost function measures the amount of DC energy that leaks out to the bandpass and highpass subbands. The main idea is to concentrate all signal energy at DC into the DC coefficients. This proves to be advantageous in both signal decorrelation and in the prevention of discontinuities in the reconstructed signals. Low DC leakage can prevent the annoying checkerboard artifact that usually occurs when high frequency bands are severely quantized. Zero DC leakage is equivalent to attaining one vanishing moment in wavelet design (a necessary condition for the convergence of the wavelet construction):  $\langle \phi_i, \mathbf{1} \rangle = 0; \forall i \neq 0$ . Depending on whether the focus is on the transform coefficients or the filter bank coefficients, it can be stated as

$$C_{DC} = \sum_{i \neq 0} \langle \phi_i, \mathbf{1} \rangle \quad \text{or} \quad C_{DC} = \sum_{i=1}^{M-1} \sum_{n=0}^{L-1} h_i[n]. \quad (8.18)$$

- *Attenuation at mirror frequencies:* The mirror frequency cost function is a generalization of  $C_{DC}$  above. The concern is now at every aliasing frequencies  $\omega_m = \frac{2m\pi}{M}; m \in \mathcal{Z}, 1 \leq m \leq \frac{M}{2}$ , instead of just at DC. Frequency attenuation at mirror frequencies are very important in the further reduction of blocking artifacts: the filter responses (except the only high-pass filter in the system) should vanish at these mirror frequencies as well. The corresponding cost function is:

$$C_M = \sum_{i=0}^{M-2} |H_i(e^{j\omega_m})|^2; \quad \omega_m = \frac{2m\pi}{M}, \quad m \in \mathcal{Z}, \quad 1 \leq m \leq \frac{M}{2}. \quad (8.19)$$

Low DC leakage and high attenuation near the mirror frequencies are not as essential to the coder's objective performance as coding gain. However, they do improve the visual quality of the reconstructed signal significantly.

- *Integer-to-integer mapping* with exact invertibility and tight dynamic range extension: these two properties are critical for lossless coding applications.
- *Practical considerations:* integer or dyadic-rational coefficients; hardware and software friendly; low latency; small bus-width; parallelizability; structural regularity; low memory buffering; and in-place computation.

## 1.8.4 Approximation approach via direct scaling

The simplest approach to design an integer transform is to approximate a well-known infinite-precision (represented using floating-point precision) transform by a close finite-precision (fixed-point) version.

For instance, an integer-coefficient transform  $\mathbf{T}_M$  can be obtained from a well-known irrational-coefficient transform  $\mathbf{C}_M$  as follows

$$\mathbf{T}_M(\alpha, \mathbf{C}_M) = \text{round}(\alpha \mathbf{C}_M), \quad (8.20)$$

where  $\alpha$  is a parameter to control the precision of the approximation. Obviously, the larger  $\alpha$  is, the closer the approximation gets along with a higher implementation cost and a larger output dynamic range.

### 1.8.4.1 H.264 4 × 4 Transform design

The most well-known integer transform successfully designed from this scaling approach is the 4-point bit-exact DCT approximation in the latest video coding international standard H.264 or MPEG-4 Part 10 [13, 14]. In this particular example, the target transform is the 4-point type-II DCT in (8.7), which can be expressed as follows

$$\mathbf{C}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ c & s & -s & -c \\ 1 & -1 & -1 & 1 \\ s & -c & c & -s \end{bmatrix}; \quad c = \sqrt{2} \cos \frac{\pi}{8}, \quad s = \sqrt{2} \sin \frac{\pi}{8}. \quad (8.21)$$

In the earlier design, the approximation  $\mathbf{T}_4$  transform is designed with the scaling parameter  $\alpha = 26$  yielding the following integer version

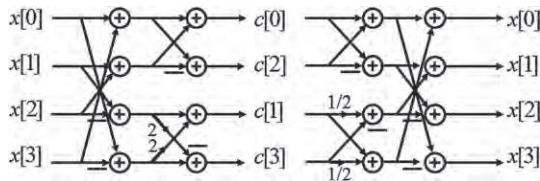
$$\mathbf{T}_4(26) = \text{round}(26\mathbf{C}_4) = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix}, \quad (8.22)$$

which happens to retain symmetry, orthogonality, as well as the coding gain of the DCT. The only drawback here is the extended dynamic range which requires the transformation stage to be implemented on a 32-bit architecture. The standardization committee later selected the following integer transform, proposed independently by Microsoft and Nokia [14]

$$\mathbf{T}_4\left(\frac{5}{2}\right) = \text{round}\left(\frac{5}{2}\mathbf{C}_4\right) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \quad (8.23)$$

whose inverse turns out to be

$$\mathbf{T}_4^{-1}\left(\frac{5}{2}\right) = \frac{1}{20} \begin{bmatrix} 5 & 4 & 5 & 2 \\ 5 & 2 & -5 & -4 \\ 5 & -2 & -5 & 4 \\ 5 & -4 & 5 & -2 \end{bmatrix}. \quad (8.24)$$

**FIGURE 8.5**

Fast implementation of the H.264 transform (left) and inverse transform (right). No multiplications are needed, only binary additions and shifts.

The current transform in H.264 is designed with the scaling parameter  $\alpha = 2.5$  (actually, any parameter in the range  $2.3 \leq \alpha < 3$  also yields exactly the same result). Its coding gain as defined in (8.15) is only 0.02 dB lower than the original DCs. Similar to the earlier design, this transform retains symmetry as well as orthogonality, and offers an efficient multiplier-less implementation as depicted in Figure 8.5. Most importantly, the lower dynamic range allows the entire transformation stage to fit within a 16-bit architecture. It is interesting to note that when the binary shifts of 2 and  $\frac{1}{2}$  in the structures of Figure 8.5 are removed, we are left with the Walsh-Hadamard transform  $\mathbf{H}_4$  as defined in (8.6). Despite the important improvements, several drawbacks still exist: (i) it is not an integer-to-integer transform with exact invertibility (its inverse involves a division by 5); and (ii) its dynamic range control is still not tight enough. Hence, H.264 loses the capability of lossy and lossless coding based on the same compression framework.

### 1.8.4.2 Integer DCT design via the principle of dyadic symmetry

The direct scaling approach is actually studied extensively by Cham, whose focus is on the two cases  $M = 8$  and  $M = 16$  [15–17]. Cham's approach can be characterized as direct scaling with orthogonal constraint. In other words, since there is no guarantee that scaling retains the orthogonality property, we seek additional condition(s) on the integer transform coefficients such that the scaled-up integer transform is always orthogonal. Let us illustrate this design procedure with the  $M = 8$  case, in which the free integer parameters come from the set of  $\{a, b, c, d, e, f\}$ .

$$\text{ICT}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{bmatrix}. \quad (8.25)$$

In this particular case, it can be proven that the only condition they have to satisfy in order for the resulting integer cosine transform (ICT) to be orthogonal [15] is

$$ab = ac + bd + cd. \quad (8.26)$$

An exhaustive search, where the cost function is set to be the transform coding gain  $C_{CG}$  as defined in (8.15), up to a certain maximum resolution is performed to identify a rather large family of ICTs. One example of a parameter set that yields a high-performance ICT is  $a = 230$ ,  $b = 201$ ,  $c = 134$ ,  $d = 46$ ,  $e = 3$ , and  $f = 1$ .

### 1.8.4.3 Direct scaling of a rotation angle

It is worth noting that although direct scaling of a given orthogonal matrix does not generally retain orthogonality, it always holds in the  $2 \times 2$  case. In this special case, any orthogonal matrix can be characterized by a single rotation angle, and it is easy to see that orthogonality is robust under direct scaling. For any rotation angle  $\mathbf{R}_\theta$ , the direct scaling method produces the approximation

$$\hat{\mathbf{R}}_\theta = \text{round}(\alpha \mathbf{R}_\theta) = \text{round}\left(\alpha \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}\right) = \begin{bmatrix} \hat{c} & \hat{s} \\ -\hat{s} & \hat{c} \end{bmatrix} \quad (8.27)$$

that always maintain orthogonality regardless of the scaling parameter choice:

$$\hat{\mathbf{R}}_\theta^T \hat{\mathbf{R}}_\theta = \begin{bmatrix} \hat{c} & -\hat{s} \\ \hat{s} & \hat{c} \end{bmatrix} \begin{bmatrix} \hat{c} & \hat{s} \\ -\hat{s} & \hat{c} \end{bmatrix} = \begin{bmatrix} \hat{c}^2 + \hat{s}^2 & 0 \\ 0 & \hat{c}^2 + \hat{s}^2 \end{bmatrix}. \quad (8.28)$$

The observation above leads to the following systematic integer-approximating design procedure which does not require complicated orthogonality constraints on the parameter set:

- Obtain a rotation-based sparse factorization of the target transform.
- Apply direct scaling to each rotation.

Note that each rotation angle can be approximated independently with a different degree of accuracy. This independent approximation approach above is actually very well-known in practical fixed-point implementations. Most fixed-point IDCT structures are obtained following this procedure [18].

The direct scaling design approach in this section offers a few practical advantages:

- Several desirable properties of the original transform can be robustly retained (most importantly, symmetry, orthogonality, and high coding gain).
- The design procedure leads to efficient integer implementations that are often equivalent to fixed-point hardware implementations.

However, this approach is only applicable to transform approximation, not construction of new ones. Furthermore, a few disadvantages still remains in this ad-hoc trial-and-error method:

- There is in general no exact inverse (not useful for lossless coding).
- In fact, the inverse operation often involves division operations.
- It is difficult to control the dynamic range expansion and to apply when the transform size/complexity increases.

---

### 1.8.5 Approximation approach via structural design

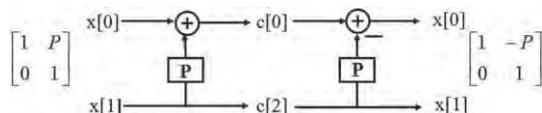
The structural design approach seeks to capture structurally as many desirable properties as possible and then optimize the parameters of the structure for high coding gain (or any other desirable property

for the targeted application). In other words, we use basic low-order building blocks (such as butterflies, lifting steps, scaling factors, delay elements) to construct much more complicated high-order transforms with certain predetermined properties (such as integer-mapping, perfect invertibility, symmetry). This design approach can be applied to transform approximation as well as new transform construction. Integer- or dyadic-rational-coefficient transforms can be easily designed by approximating the optimal structural parameters by (dyadic) rationals.

### 1.8.5.1 Lifting step

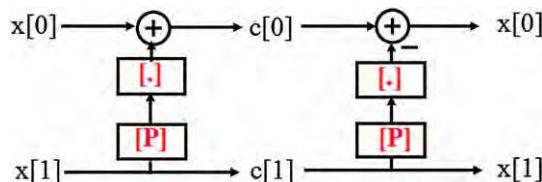
One critical component in modern transform design is the lifting step [19], which is simply an elementary upper-triangular or a lower-triangular matrix with unity diagonal elements, also known as a shear operation in matrix theory or the ladder structure in signal processing literature [20]. Figure 8.6 depicts a typical lifting step on the left and its corresponding inverse on the right. The upper-triangular matrix as shown in Figure 8.6 is often referred to as the update lifting step while the lower-triangular matrix is called the prediction lifting step in the literature. The operator  $P$  in Figure 8.6 is very general as far as perfect reconstruction is concerned:  $P$  can certainly be any scalar value;  $P$  actually can be any polynomial with delays (as popularized in fast wavelet transform implementation); in fact,  $P$  can even be any non-linear operator (to generate morphological wavelet for instance). Invertibility is structurally guaranteed since to invert a lifting step, we only need to subtract out (add in) what has been added in (subtracted out) at the forward transform.

One advantage that the lifting scheme offers is the versatility and the simplicity in constructing fast transforms that can map integers to integers. If a *floor* (or *round*, or *ceiling*, or any other quantization) operator is placed after  $P$  in the lifting step in Figure 8.6, the transformation can now map integers to integers with perfect reconstruction regardless of the lifting step has integer coefficients or not. This property can be confirmed through a simple inspection as demonstrated in Figure 8.7.



**FIGURE 8.6**

A lifting step (left) and its corresponding inverse (right).



**FIGURE 8.7**

Integer-to-integer mapping with exact invertibility of a lifting step.

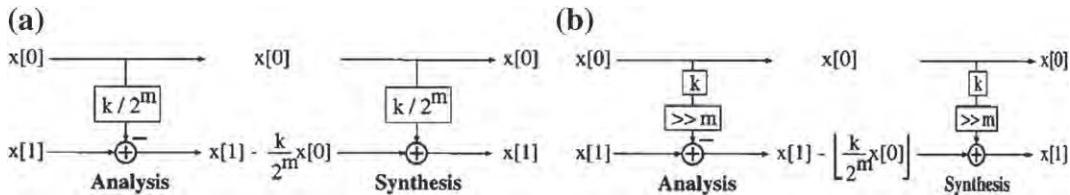


FIGURE 8.8

Lifting step with (a) dyadic rational coefficient and (b) the resulting multiplier-less implementation.

Moreover, if the lifting step is chosen to be dyadic, the nonlinear operation can be incorporated into the division using binary shift as illustrated in Figure 8.8. A scalar lifting step of value  $\frac{k}{2^m}$  can be implemented as a multiplication by  $k$  followed by a division by  $2^m$ . Division by  $2^m$  followed by a truncation is equivalent to a binary right-shift by  $m$  places. The numerator  $k$  can be easily implemented using bit shift and add operations as well, or we can split the fraction  $\frac{k}{2^m}$  into a cascade of pure power-of-two lifting steps, i.e.,  $\frac{k}{2^m} = \sum_i \frac{1}{2^i}$ . The latter approach leads to an implementation with solely binary right-shifts, preventing the bit-depth expansion in intermediate results. With all scaling factors set to unity, multiplier-less transforms can be easily constructed via a cascade of multiple lifting steps. If the scaling factors are powers of two, we can still retain the multiplier-less feature on both analysis and synthesis side. Besides the integer-friendly property, the lifting scheme also offers several other interesting advantages: simplicity of wavelet design on irregular intervals, in-place computation, connection to spatial prediction, and capability of incorporating nonlinear filtering.

It is quite straightforward to extend the lifting result above to the higher dimension case of any arbitrary  $M \times M$  matrix since it is well-known from the LDU matrix factorization that every  $M \times M$  invertible matrix  $\mathbf{V}$  can be completely characterized by  $M(M - 1)$  elementary matrices,  $M$  diagonal scaling factors, and a permutation matrix. In other words, any invertible matrix  $\mathbf{V}$  can be factorized as  $\mathbf{V} = \mathbf{P}\mathbf{L}\mathbf{D}\mathbf{U}$  where the upper-triangular matrix  $\mathbf{U}$  can be constructed from  $\frac{M(M-1)}{2}$  elementary update lifting steps labeled  $u_{ij}$ , the lower-triangular matrix  $\mathbf{L}$  can be constructed from  $\frac{M(M-1)}{2}$  elementary prediction lifting steps labeled  $p_{ij}$ , and the diagonal matrix  $\mathbf{D}$  contains the scaling factors  $\alpha_i$ . The construction of  $\mathbf{V}$  as described above is depicted in Figure 8.9. Note that the parameterization can be accomplished with rotation angles and diagonal scaling factors as well (see Figure 8.10).

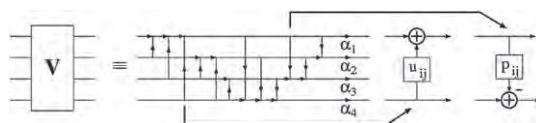
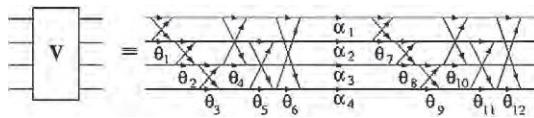
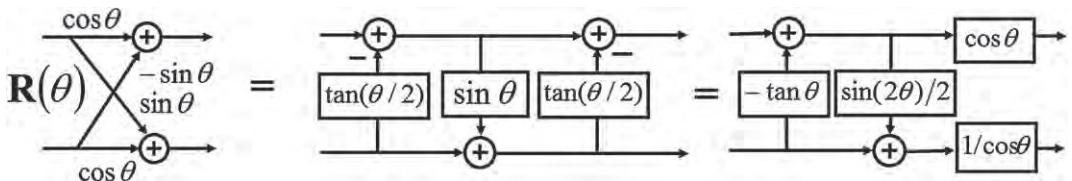


FIGURE 8.9

General parameterization of any invertible matrix via lifting steps.

**FIGURE 8.10**

General parameterization of any invertible matrix via rotation angles.

**FIGURE 8.11**

Representation of a rotation angle via either 3 lifting steps or 2 lifting steps and 2 scaling factors.

### 1.8.5.2 Lifting-based approximation

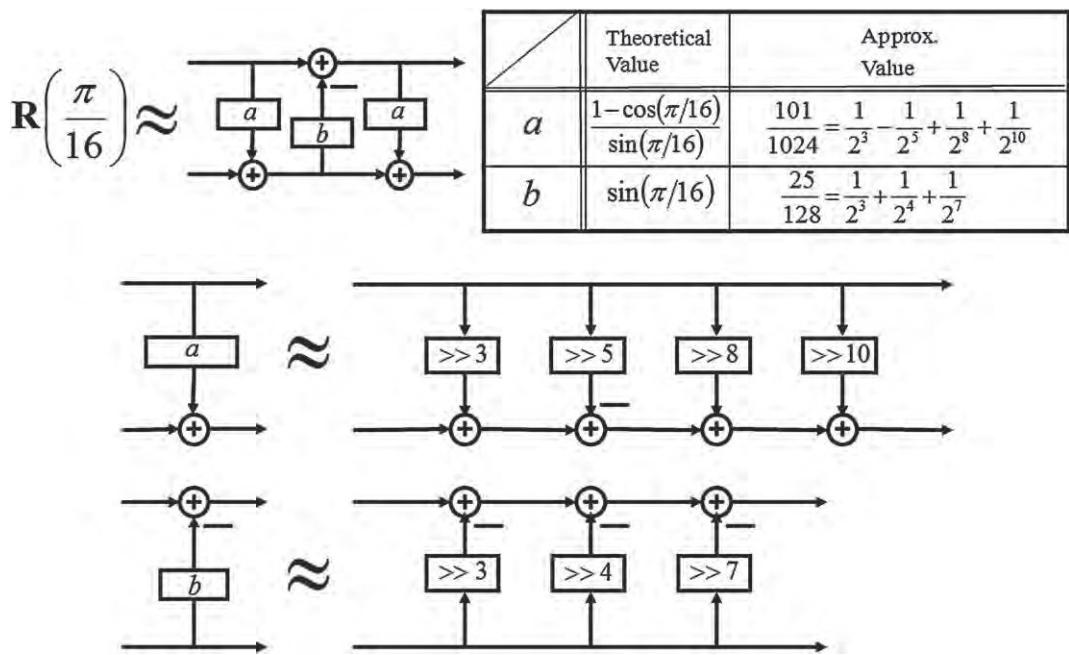
Lifting steps are also very convenient in transform approximation as well. Figure 8.11 shows the equivalent representation of the popular rotation angle by 3 lifting steps or 2 lifting steps and 2 scaling factors. Sometimes, the latter representation can lead to a more efficient approximation since the 2 scaling factors can be absorbed into the quantization stage. Figure 8.12 demonstrates a systematic approach to rotation approximation via multiplier-free lifting:

- first compute the theoretical values of the lifting steps,
- obtain dyadic-rational approximating values from theoretical ones (the resolution of the approximating values is the trade-off between computational complexity and coding performance),
- obtain the multiplier-less implementation from the dyadic-rational lifting steps.

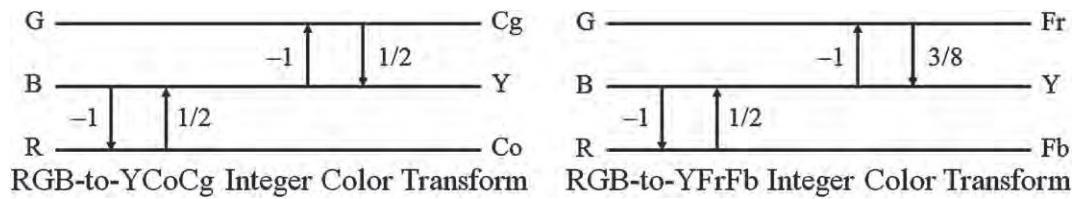
Since most existing fast transform algorithms are obtained from sparse factorization of the transform matrix as a cascade of rotation angles, we can systematically break down the design procedure one rotation independently at a time. An example of a successful application of this lifting-based approximation approach is the design of multiplier-less IDCT for video coding in an effort to end the drifting effects from IDCT mismatch. Liu et al shows that standard-compliance IDCT can be achieved on a 16-bit architecture whereas on a 32-bit architecture, approximation via lifting and butterfly can be so accurate that the reconstructed video sequences are virtually drifting-free when pairing with a 64-bit floating-point IDCT [21].

### 1.8.5.3 Lossless color transform design

We demonstrate in this section the simplest design example based on lifting construction for  $M > 2 - 3 \times 3$  color transforms. Figure 8.13 depicts two different designs developed independently and submitted to the JVT standardization committee at the same time: the RGB-to-YCoCg transform

**FIGURE 8.12**

Example of a rotation approximation by dyadic-rational lifting steps.

**FIGURE 8.13**

Multiplierless  $3 \times 3$  color transforms with exact invertibility.

(on the left of Figure 8.13) by Malvar et al. [22] and the RGB-to-YFbFr transform (on the right of Figure 8.13) by Topiwala et al. [23].

It is interesting to note that two designs are remarkably similar since they are both designed using our lifting approximation framework described in the previous sections. First, the optimal KLT is obtained where the auto-correlation matrix is derived from a set of high-quality test color images acquired by Kodak. Lifting-based approximation is then applied to obtain the dyadic lifting coefficients. Practical experiments show that both transforms de-correlate the three color planes more efficiently than existing

popular color transforms such as the RGB-to-YUV and the RGB-to-YCbCr transform while additionally providing the pivotal feature of lossless integer mapping with low dynamic range (there is only a 1-bit expansion on the two chrominance components).

Finally, both transforms only utilize four lifting steps (instead of theoretically six) without any diagonal scaling. At the cost of one more addition and one more bit-shift operation per luminance sample, the RGB-to-YFbFr transform attains a coding gain of 4.812 dB while the RGB-to-YCoCg transform achieves  $CCG = 4.619$  dB (the optimal KLT in this case has a coding gain of 4.966 dB).

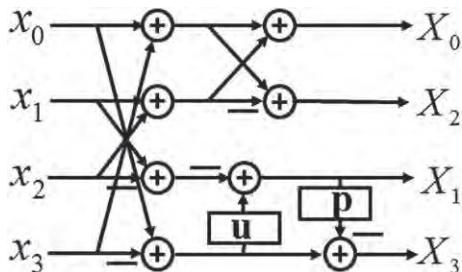
#### 1.8.5.4 Integer DCT design

The DCT is the most widely used transform in current international image/video compression standards. This 4-point dyadic-coefficient design example is a close approximation of the 4-point DCT and it follows the DCT's factorization very closely. The resulting structure is depicted in Figure 8.14.

In the sparse 4-point DCT factorization case, there are three butterflies and only one irrational rotation angle of  $\frac{\pi}{8}$  [24]. We can employ only 2 lifting steps for this rotation angle (since the scaling factors associated with the 2-lifting-step structure can be combined with the quantization step sizes to save computational complexity further) while keeping the butterflies intact. Many different transforms can be represented using this structure the difference between them lies only in the parameter choices as demonstrated below:

- $u = 1; p = 1/2$ : essentially the Walsh-Hadamard transform,
- $u = 7/16; p = 3/8$ : Liang et al. submission to H.264 [25, 26],
- $u = 1/2; p = 2/5$ : the current 4-point H.264 transform (the last 2 basis functions are off by 2, 5/2 scaling),
- $u = 1/2; p = 1/2$ : the current 4-point Photo Core Transform (PCT) in HD-Photo/JPEG-XR [27, 28].

It is interesting to note that the last two options yield two transforms with identical coding gain of 7.55 dB. The lifting implementation allows the PCT [27, 28] to map integers to integers losslessly and to have a much tighter dynamic range control comparing to the rotation-based implementation of H.264 transform (this is where H.264 loses its lossless coding capability). The second option with  $u = 7/16$



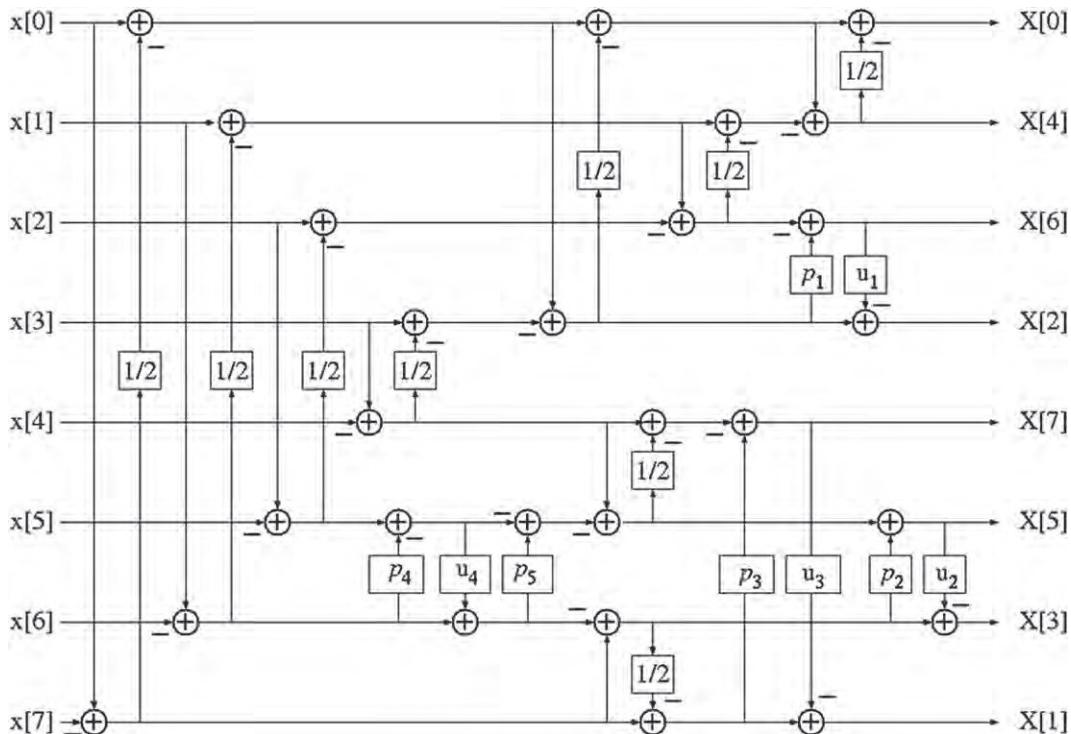
**FIGURE 8.14**

4 × 4 integer DCT structure.

and  $p = 3/8$  essentially achieves the same coding gain as the theoretical 4-point DCT (7.57 dB for  $AR(1)$  signal model with  $\rho = 0.95$ ). Note that all four options allow 16-bit implementation with 9-bit input data.

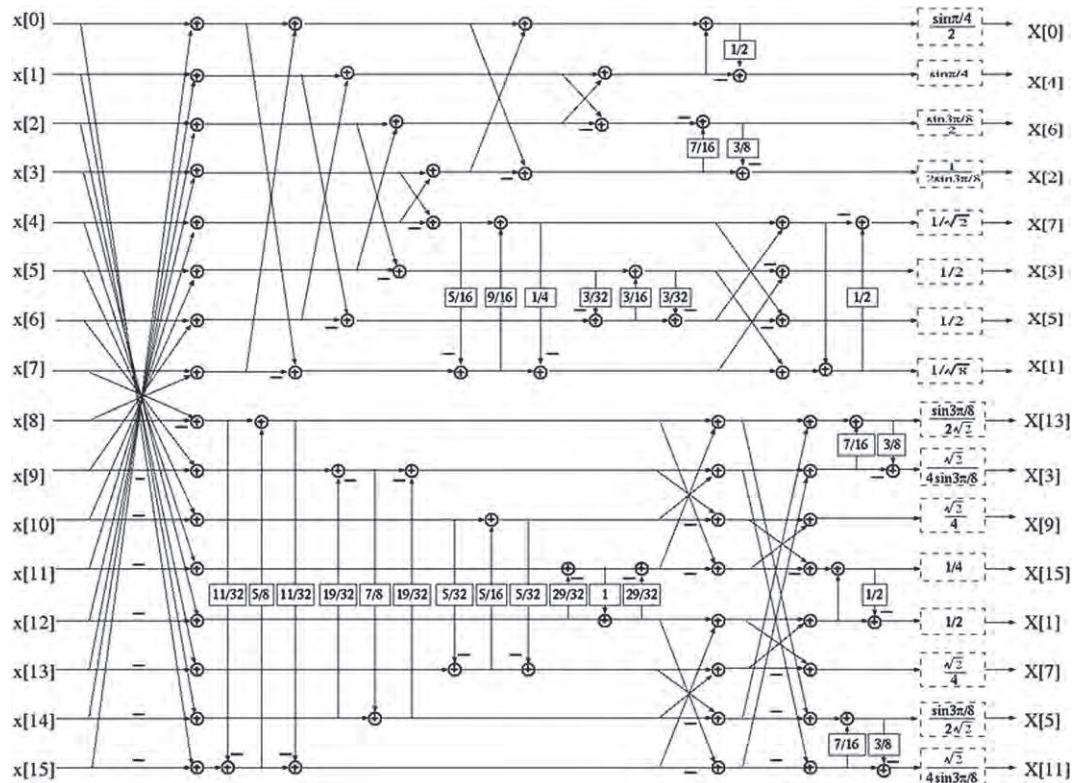
Integer DCTs of larger sizes ( $M > 4$ ) can be easily designed using the same procedure. The rotation-factorized representation of the transform of interest is first obtained, then each rotation angle can be approximated methodically as demonstrated in Section 1.8.5.2. With various degrees of approximation accuracy (more accuracy obviously requires more complexity), these integer DCT can be tuned to cover the gap between the Walsh-Hadamard transform and the original DCT. The corresponding solutions often allows a 16-bit implementation, enables lossless compression, and maintains satisfactory compatibility with the floating-point DCT. The  $8 \times 8$  and  $16 \times 16$  integer DCT are depicted in Figures 8.15 and 8.16, respectively.

Table 8.1 lists the analytical values of all the lifting parameters and some configurations of the integer DCT family (called binary DCT or binDCT in short) as shown in Figure 8.15. The dyadic lifting values are obtained by truncating or rounding the corresponding analytical values with different accuracy levels.  $C_g(8)$  and  $C_g(4)$  are the coding gains of these  $8 \times 8$  binDCTs and the  $4 \times 4$  binDCTs embedded



**FIGURE 8.15**

$8 \times 8$  integer DCT.

**FIGURE 8.16**

16 × 16 integer DCT.

in them. Table 8.1 also shows the MSE between the binDCT output and the true DCT output for the popular  $8 \times 8$  case. The MSE here is the expected  $\ell_2$ -norm of the error in the produced coefficients and it is computed as follows. Assume that  $\mathbf{C}$  is the true  $M \times M$  DCT, and  $\hat{\mathbf{C}}$  is its approximated integer version. For an input column vector  $\mathbf{x}$ , the difference between the DCT coefficients and the binDCT coefficients is:

$$\mathbf{e} = \mathbf{Cx} - \hat{\mathbf{C}}\mathbf{x} = (\mathbf{C} - \hat{\mathbf{C}})\mathbf{x} \triangleq \mathbf{Dx} \quad (8.29)$$

and the MSE of each coefficient is

$$\epsilon \triangleq \frac{1}{M} E[\mathbf{e}^T \mathbf{e}] = \frac{1}{M} \text{trace}\{\mathbf{e} \mathbf{e}^T\} = \frac{1}{M} \text{trace}\{\mathbf{DR}_{xx} \mathbf{D}^T\}, \quad (8.30)$$

where  $\mathbf{R}_{xx} \triangleq \mathbf{E}[\mathbf{xx}^T]$  is the autocorrelation matrix of the input signal. The low MSE level as depicted in Table 8.1 indicate that the integer transform approximation is very accurate. The interested reader is referred to [21, 26] for more details on the design of this particular family of multiplier-less cosine transforms.

**Table 8.1** Different Configurations of the  $8 \times 8$  Integer DCT in Figure 8.15

	Floating-point	binDCT-C1	C2	C3	C4	C5
$p_1$	0.4142135623	13/32	7/16	3/8	1/2	1/2
$u_1$	0.3535533905	11/32	3/8	3/8	3/8	1/2
$p_2$	0.6681786379	11/16	5/8	7/8	7/8	1
$u_2$	0.4619397662	15/32	7/16	1/2	1/2	1/2
$p_3$	0.1989123673	3/16	3/16	3/16	3/16	1/4
$u_3$	0.1913417161	3/16	3/16	3/16	1/4	1/4
$p_4$	0.4142135623	13/32	7/16	7/16	7/16	1/2
$u_4$	0.7071067811	11/16	11/16	11/16	3/4	3/4
$p_5$	0.4142135623	13/32	3/8	3/8	3/8	1/2
Shifts	—	27	23	21	18	13
Adds	—	42	37	36	33	28
MSE	—	1.1E–5	8.5E–5	4.2E–4	5.8E–4	2.3E–3
$C_g(8)$ (dB)	—	8.8251	8.8220	8.8159	8.8033	8.7686
$C_g(4)$ (dB)	—	7.5697	7.5697	7.5566	7.5493	7.5485

### 1.8.5.5 Lifting for complex-coefficient transforms

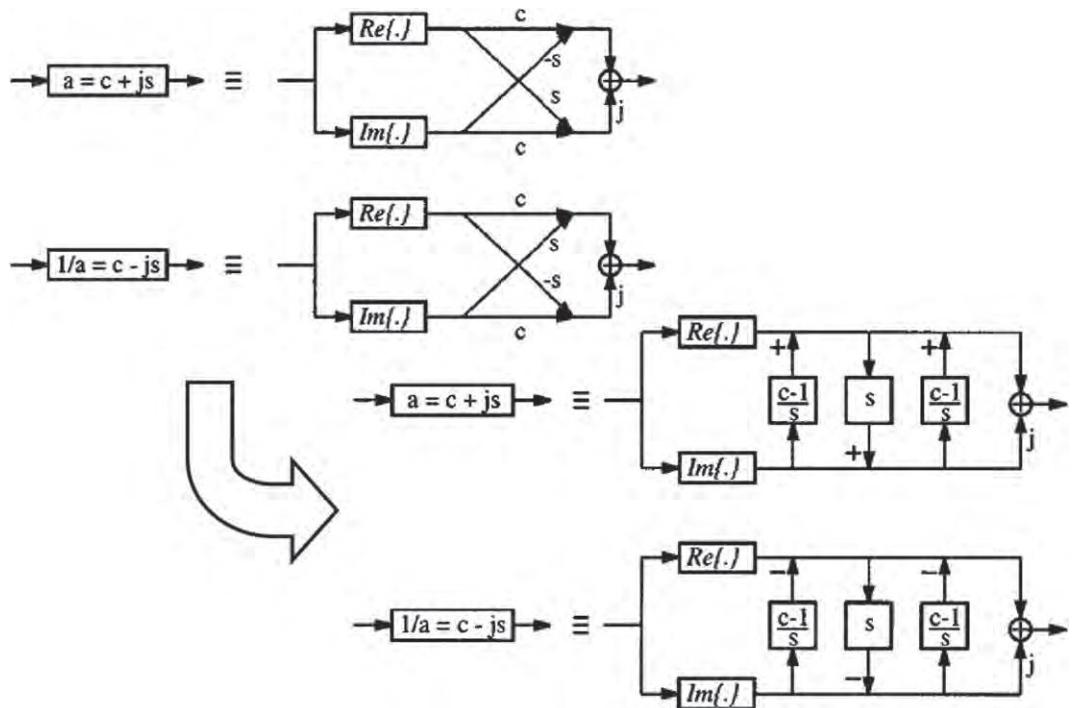
So far, we have focused on real-coefficient transforms since they are mostly employed in image/video coding applications. For complex-coefficient transforms such as the FFT, it turns out that lifting steps can be as easily applied as in the real-coefficient case. Figure 8.17 illustrates how to model a complex multiplication operation by a rotation angle and the resulting equivalent lifting representation [20,29]. Each lifting step in Figure 8.17 can then be approximated by a dyadic-rational value as previously shown. One particularly nice feature of the FFT is its regularity in factorized format: its implementation is recursive and all values of rotation angles involved are easily computed. Hence, complex-coefficient transforms such as the FFT can be easily constructed or approximated in multiplier-less fashion as well. Again, the lifting coefficients appearing in the structures can be quantized directly to obtain different resolutions (resulting in trade-off in computational cost) while preserving the exact reversibility property.

---

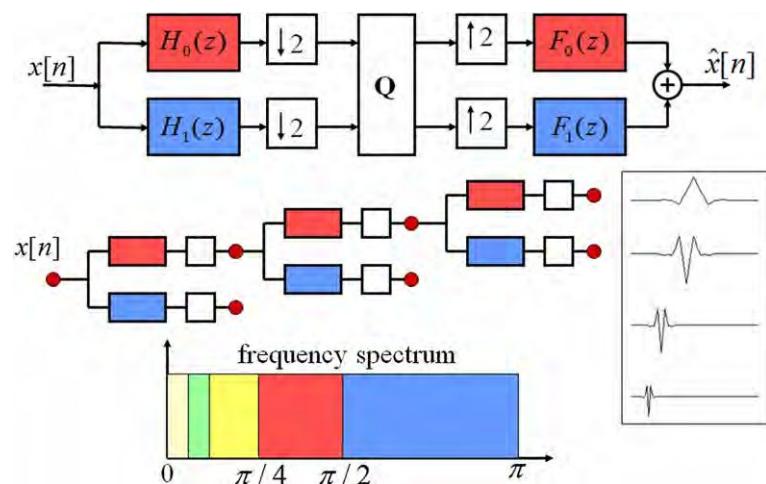
## 1.8.6 Wavelet filters design via spectral factorization

### 1.8.6.1 Wavelets and filter banks

One of the newest additions to the transform field is the wavelet transform which can be interpreted as an iteration of a two-channel filter bank with certain degrees of regularity on its low-pass output depicted in Figure 8.18. Part of the beauty and power of wavelets is their elegance: complicated tiling of the time-frequency plane can be easily achieved by merely iterating a two-channel decomposition. Furthermore, the wavelet representation of signals as a coarse approximation and detailed components at

**FIGURE 8.17**

Implementation of a complex multiplication operation via lifting steps.

**FIGURE 8.18**

The discrete wavelet transform as iteration of a special two-channel filter bank on its low-pass output.

different resolutions allows fast execution of many DSP applications such as image browsing, database retrieval, scalable multimedia delivery, etc.

The theory and design of two-channel filter banks as well as two-band wavelets have been studied extensively and we refer the readers to several excellent text books on the topic [5–8,30]. The simplest design procedure is the spectral factorization of a low-pass half-band filter with some regularity constraints (vanishing moments). Unlike all other transforms mentioned in this chapter, all of the degrees of freedom in wavelet design are allocated to regularity. In the scope of this chapter, we will only briefly cover this most successful wavelet filters design approach for completeness only.

### 1.8.6.2 Spectral factorization

It can be verified in a straightforward fashion that in order for the 2-channel filter bank in Figure 8.18 to have perfect reconstruction (i.e.,  $\hat{x}[n] = x[n - D]$ , where  $D$  is a certain time delay), the four filters involved have to satisfy the following two conditions

$$\text{Aliasing Cancellation: } F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0, \quad (8.31)$$

$$\text{Distortion Elimination: } F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-D}. \quad (8.32)$$

The following alternating-sign construction provides an elegant way to cancel aliasing and to reduce the design parameters involved by a factor of two

$$\begin{cases} F_0(z) = H_1(-z), \\ F_1(z) = -H_0(-z). \end{cases} \quad (8.33)$$

With the relationship between the analysis and synthesis filters set as in (8.33), the Distortion Elimination condition in (8.32) reduces to

$$F_0(z)H_0(z) - F_0(-z)H_0(-z) = 2z^{-D}. \quad (8.34)$$

If we define the product filter  $P_0(z)$  as  $P_0(z) \stackrel{\Delta}{=} F_0(z)H_0(z)$ , then it has to satisfy the following condition

$$P_0(z) - P_0(-z) = 2z^{-D}, \quad (8.35)$$

which is well-known in the signal processing community as the half-band condition (a filter that has one and only one odd power of  $z^{-1}$ ). Hence, the design of a 2-channel perfect-reconstruction filter bank with all four FIR filters can follow the standard procedure listed below:

- Design a low-pass half-band filter  $P_0(z)$ .
- Factor  $P_0(z)$  into  $H_0(z)$  and  $F_0(z)$ .
- Use the aliasing cancellation condition in (8.33) to obtain  $H_1(z)$  and  $F_1(z)$ .

The difference between any common filter pair designed from the approach above and a good wavelet filter pair is the degree of regularity or the level of smoothness of the resulting basis functions, called the scaling function and the wavelet function, after many levels of repeated iterations. As aforementioned, Daubechies decided to allocate all of the degrees of freedom in the design to the degree of regularity or the number of vanishing moments [30]. In other words, an additional constraint labeled *maxflat* is

imposed on top of the half-band condition in (8.35), resulting in the maxflat half-band filter defined as follows

$$P_0(z) = (1 + z^{-1})^{2P} Q_0(z). \quad (8.36)$$

This key filter has  $2P$  zeroes (roots) at  $z = -1$  or  $\omega = \pi$  and  $Q_0(z)$  is the polynomial of minimum order (which turns out to be  $2P - 2$ ) such that the half-band condition in (8.35) is met. The closed-form expression of this family of maxflat half-band filter can be found in [5, 6, 8, 30].

The key step in the design process obviously lies in the second spectral factorization procedure. There are multiple ways to split the roots  $\{z_i\}$  of  $P_0(z)$  and the desirable properties of the resulting filters  $\{H_0(z), H_1(z), F_0(z), F_1(z)\}$  dictate the constraints in the root assignment. Here are three popular desirable properties and their associated constraints in spectral factorization

- For the filters to have real coefficients, we need  $z_i$  and  $z_i^*$  to stay together: assign both to the same filter.
- To obtain orthogonal solutions, we need to separate  $z_i$  and  $z_i^{-1}$  to different filters: we must assign  $z_i$  to  $H_0(z)$  and  $z_i^{-1}$  to  $F_0(z)$  or vice versa.
- On the contrary, to obtain linear-phase solutions, we need  $z_i$  and  $z_i^{-1}$  to stay together: both are either assigned to  $H_0(z)$  or to  $F_0(z)$ .

Note that, from the second and third condition above, we cannot design a solution with both orthogonality and linear phase in the 2-channel case (except the trivial 2-tap Haar solution). When the number of channels increases ( $M > 2$ ), having both orthogonality and linear phase is actually quite easy to achieve—the type-II DCT in (8.7) and the WHT in (8.6) are two typical examples.

### 1.8.6.2.1 5-/3-tap symmetric and 4-tap orthogonal wavelet filters

Let's consider the following maxflat half-band filter with four roots at  $z = -1$  and two real roots at  $z^{-1} = 2 \pm \sqrt{3}$ :

$$\begin{aligned} P_0(z) &= \frac{1}{16} \left( -1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6} \right) \\ &= \frac{1}{16} \left( 1 + z^{-1} \right)^4 \left( -1 + 4z^{-1} - z^{-2} \right). \end{aligned} \quad (8.37)$$

Assigning two roots at  $z = -1$  to  $F_0(z)$  and the rest to  $H_0(z)$ , we obtain the following real-coefficient symmetric filter pair

$$\begin{cases} H_0(z) = -\frac{1}{8} + \frac{1}{4}z^{-1} + \frac{3}{4}z^{-2} + \frac{1}{4}z^{-3} - \frac{1}{8}z^{-4}, \\ H_1(z) = -\frac{1}{2} + z^{-1} - \frac{1}{2}z^{-2}. \end{cases} \quad (8.38)$$

This particular solution is first published in [31], so they are often referred to as the Le Gall 5/3 filters. Not only do they have dyadic coefficients; their lifting multiplier-less implementation also provides an efficient reversible integer-to-integer mapping as discuss later in Section 1.8.6.3. The discrete wavelet transform with these 5-/3-tap filters is the default transformation option for the lossless mode in the international image coding standard JPEG2000 [4].

On the other hand, if we assign the roots  $\{-1, -1, 2 - \sqrt{3}\}$  to  $H_0(z)$  while giving the roots  $\{-1, -1, 2 + \sqrt{3}\}$  to  $F_0(z)$ , we arrive at the celebrated compact-support orthogonal Daubechies D4 wavelet pair

$$\begin{cases} H_0(z) = \frac{1}{4\sqrt{2}} \left[ (1 + \sqrt{3}) + (3 + \sqrt{3})z^{-1} + (3 - \sqrt{3})z^{-2} + (1 - \sqrt{3})z^{-3} \right], \\ H_1(z) = \frac{1}{4\sqrt{2}} \left[ (1 - \sqrt{3}) - (3 - \sqrt{3})z^{-1} + (3 + \sqrt{3})z^{-2} - (1 + \sqrt{3})z^{-3} \right]. \end{cases} \quad (8.39)$$

With this particular maxflat half-band filter  $P_0(z)$ , there are only two possible orthogonal solutions. The pair in (8.39) above is called the minimum-phase solution. Switching the role of  $H_i(z)$  and  $H_i(z)$  yields the other maximum-phase solution. There are quite a few more real symmetric solutions. In fact, the 2-/6-tap and the 4-/4-tap pair are also quite popular in the image processing community.

#### 1.8.6.2.2 9-/7-tap symmetric and 8-tap orthogonal wavelet filters

Let's consider the higher-order symmetric maxflat half-band filter with eight roots at  $z = -1$ , two real roots, and four complex roots; the complex plane of  $P_0(z)$  is shown on the top part of Figure 8.19

$$P_0(z) = \frac{(1 + z^{-1})^8}{2048} \left( -5 + 40z^{-1} - 131z^{-2} + 208z^{-3} - 131z^{-4} + 40z^{-5} - 5z^{-6} \right).$$

There are exactly four orthogonal solutions from spectral factorization in this case: one minimum-phase, one maximum-phase, and two mixed-phase solutions (they are time-reversal of each other just like the minimum-phase is the time-reversed version of the maximum-phase) as illustrated in Figure 8.19.

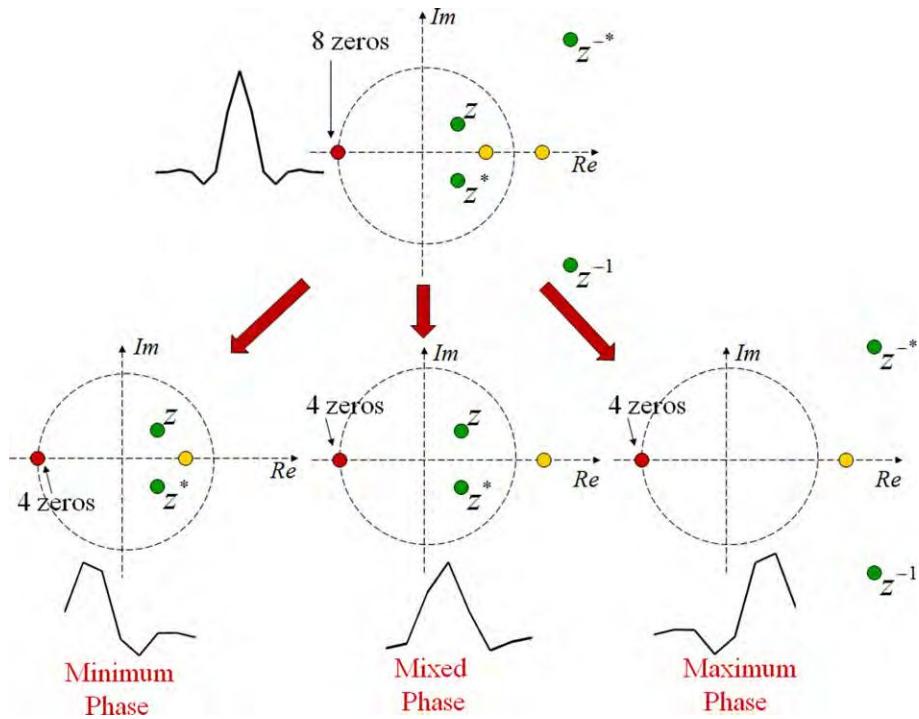
In the bi-orthogonal linear-phase case, there are many more non-trivial possibilities. The best solution in term of attaining the highest theoretical coding gain as well as the highest consistent coding performance in practice is the 9-/7-tap filter pair with  $H_0(z)$  having four roots at  $z = -1$  and four complex roots as depicted in Figure 8.20. This is often referred to as the Daubechies 9/7 wavelet filter pair [32], which is JPEG2000's default for floating-point high performance option [4]. Various interesting properties of these JPEG2000's wavelet filters have been explored in depth in [33].

It is worth noting that sacrificing a few vanishing moments (precisely, reducing the number of roots at  $z = -1$  from eight to six in  $P_0(z)$ ) opens up more flexibility in achieving other desirable properties such as dyadic coefficient. The following 9-/7-tap filter pair is called *binlet*—binary wavelet. The analysis bank now only has two vanishing moments (instead of four as in the Daubechies 9/7 JPEG2000 pair) whereas the synthesis bank still retains four vanishing moments since the vanishing moments—intimately related to the level of smoothness—are more critical in signal reconstruction than in signal decomposition. The 9-/7-tap binlet filter coefficients are tabulated below.

$$\begin{cases} h_0[n] = \frac{1}{64}[1 \ 0 \ -8 \ 16 \ 46 \ 16 \ -8 \ 0 \ 1], \\ h_1[n] = \frac{1}{16}[1 \ 0 \ -9 \ 16 \ -9 \ 0 \ 1]. \end{cases} \quad (8.40)$$

#### 1.8.6.3 Lifting and the wavelet transform

The lifting scheme is originally designed for the wavelet transform and two-channel FB decomposition [19]. Two-channel filter banks implemented with the lifting scheme save roughly half of the computational complexity due to the exploitation of the inter-subband relationship between coefficients. It has

**FIGURE 8.19**

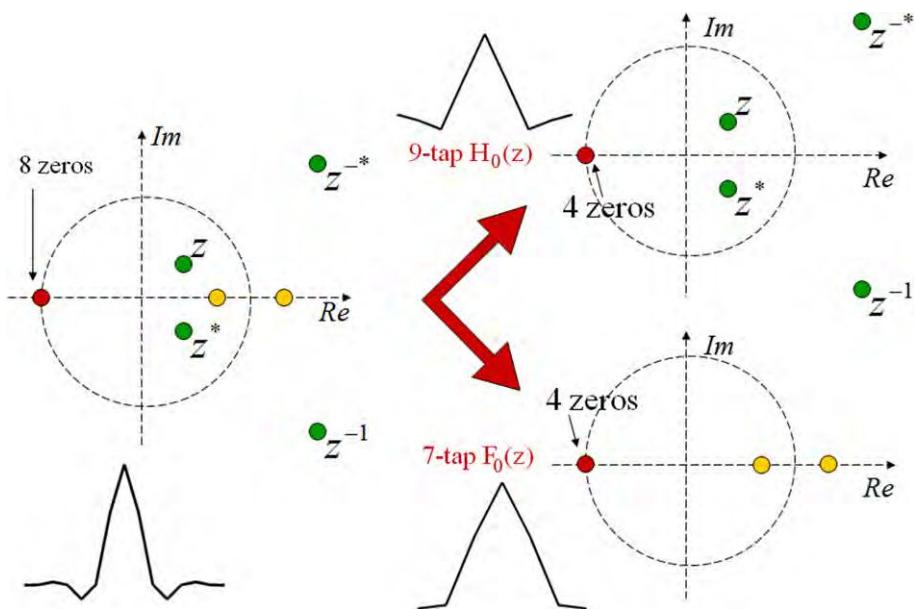
Orthogonal wavelet filter design via spectral factorization.

been proven that any  $2 \times 2$  polyphase matrix  $\mathbf{E}(z)$  with unity determinant can be factored into a cascade of prediction  $\mathbf{P}(z)$ , update  $\mathbf{U}(z)$  lifting steps, and a diagonal scaling matrix as shown in Figure 8.21 [34].

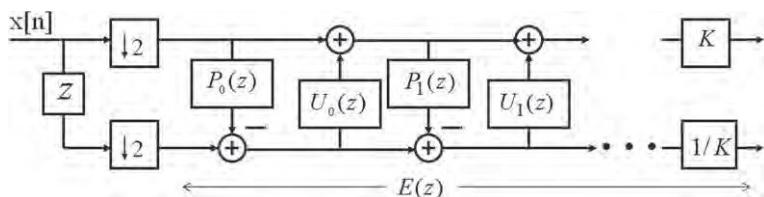
As previously mentioned, one of the most elegant as well as the most popular wavelet filter pair in practice is the Le Gall 5/3 [31] whose polyphase matrix can be constructed with 2 lifting steps:  $P(z) = \frac{1}{2}(1 + z^{-1})$  and  $U(z) = \frac{1}{4}(1 + z)$ . This transform depicted in Figure 8.22 leads to a multiplier-less implementation, and since it can map integers to integers with exact invertibility, it is naturally adopted by JPEG2000 for the lossless (or low cost) compression mode [4]. It has a very intuitive time-domain interpretation: each high-pass detail wavelet coefficient is simply the prediction error between the average of 2 neighboring even-indexed samples and the in-between odd-indexed sample. Figure 8.23 shows the lifting implementation of the irrational-coefficient 9/7 Daubechies wavelet filter pair [32,34] as previously discussed in Section 1.8.6.2.2 (see Figure 8.22).

## 1.8.7 Higher-order design approach via optimization

One major drawback with non-overlapped block transforms such as the DCT and its integer versions as described in the previous section is the existence of annoying blocking artifacts from discontinuities at

**FIGURE 8.20**

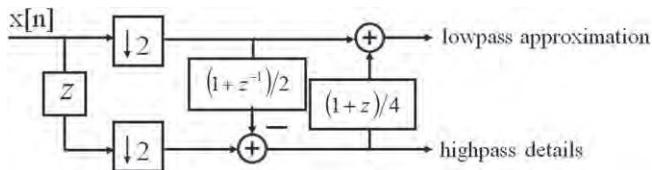
Linear-phase bi-orthogonal wavelet filter design via spectral factorization.

**FIGURE 8.21**

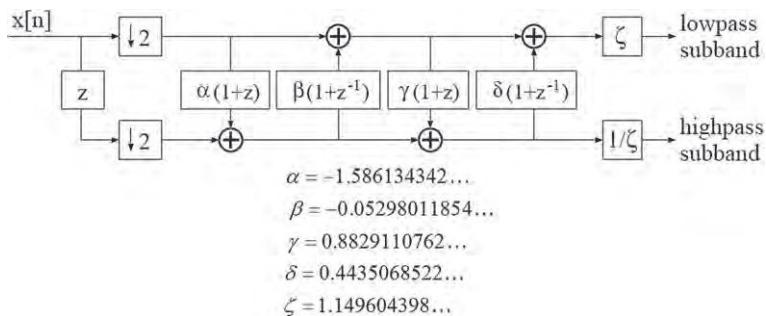
General wavelet construction and implementation with lifting steps.

block boundaries in low bit-rate applications. Transforms with overlapping basis functions such as the wavelet transform, the wavelet packet, and the lapped transform (LT) can elegantly solve the blocking problems, producing much more visually pleasant reconstructed images and video frames.

From a filter bank perspective, higher-order design lengthens the filters, leading to drastic improvement in frequency resolution and hence decorrelation efficiency while still retaining time or space resolution. As aforementioned, from a filter bank perspective, wavelet and lapped transform are simply higher-order systems whose polyphase matrices contain polynomials in  $z$  whereas all examples discussed so far are block transforms whose polyphase matrices are regular scalar matrices of zero order.

**FIGURE 8.22**

The 5/3-tap integer wavelet implementation in lifting.

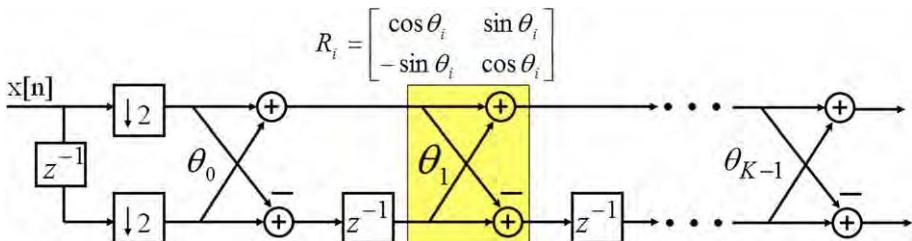
**FIGURE 8.23**

The 9/7-tap double-precision wavelet implementation in lifting.

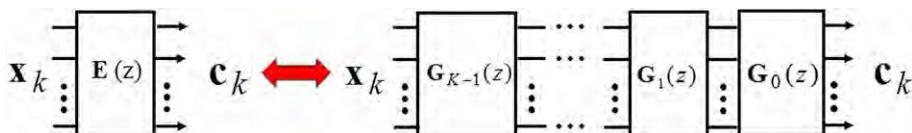
### 1.8.7.1 General modular construction

In the wavelet case where the basic building block is a 2-channel FB, the most general design approach is the spectral factorization of a low-pass half-band filter. The lifting scheme is used mainly for fast and efficient wavelet implementation. When the number of channels  $M$  increases, spectral factorization is not applicable and the most successful design approach is the modular construction from the lattice structure pioneered by Vaidyanathan [7] and Malvar [12]. This first filter bank design based on this philosophy is illustrated in Figure 8.24 where the each modular stage is a rotation angle (coupled with a delay element), propagating the orthogonality property. This simple structure has been proven to utilize the minimum number of delay elements in the resulting implementation since it shares the delays between the two channels. More importantly, the structure in Figure 8.24 covers *all* possible orthogonal solutions in the two channel case. Indeed, when  $K = 2$ , the particular selection of  $\{\theta_0 = 60^\circ, \theta_1 = -15^\circ\}$  generates the D4 orthogonal Daubechies wavelet filters in (8.39). It is a simple exercise for the reader to verify that the resulting polyphase matrix here always satisfy the paraunitary condition in (8.11), i.e.,  $\mathbf{E}^{-1}(z) = z^{-(K-1)} \mathbf{E}^T(z^{-1})$ .

Similarly, in the more general  $M$ -channel case, we can construct the high-order polyphase matrix  $\mathbf{E}(z)$  as in Figure 8.25 from a cascade of many low-order building blocks  $\mathbf{G}_i(z)$ , each propagating certain desirable properties such as symmetry, perfect reconstruction (or more elegantly, orthogonality), and zero DC leakage. The free parameters of the structure within each building block  $\mathbf{G}_i(z)$  are then further optimized to achieve other remaining desirable properties such as coding gain and stop-band attenuation

**FIGURE 8.24**

First 2-channel  $2K$ -tap filter bank design via the lattice structure propagating orthogonality.

**FIGURE 8.25**

Construction of a high-order polyphase matrix  $\mathbf{E}(z)$  from a cascade of low-order building blocks  $\mathbf{G}_i(z)$ .

as discussed in Section 1.8.3. This modular construction design approach yields numerous interesting filter bank solutions, which the reader can learn about in much greater details in two excellent textbooks [6, 7].

### 1.8.7.2 Higher-order design with pre-/post-processing operators

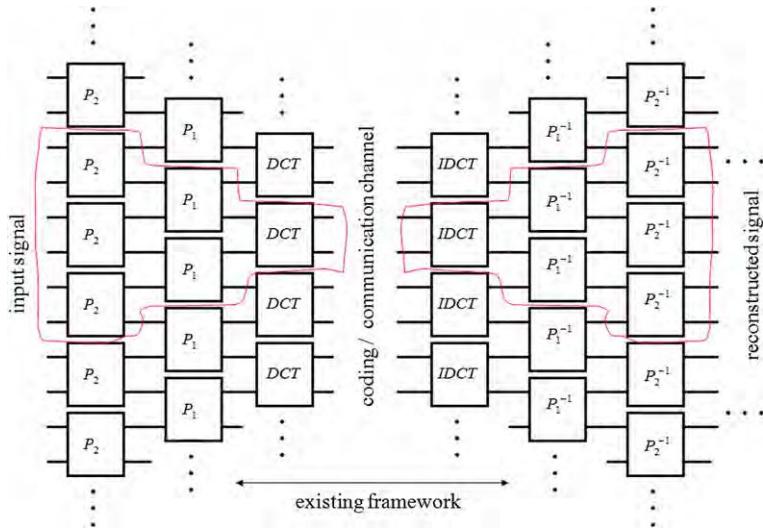
Within the more restrictive constraint of this chapter, let us present in depth a general structure that has an intuitive time-domain interpretation [35] as illustrated in Figure 8.26.

#### 1.8.7.2.1 A simple example

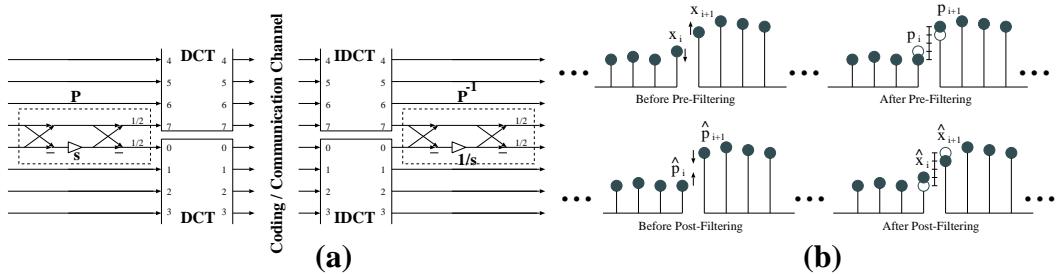
Several basic properties of this design approach are best explained through an almost trivial example—the addition of 2-point pre- and post-processing (or pre-/post-filtering) operators linking any two DCT blocks in traditional block coders such as JPEG [1] depicted in Figure 8.27a. This is a particular pre-processor in the general framework of Figure 8.26 and the post-processor has been chosen to be the exact inverse of the pre-processor’s.

Let  $\{x_i\}$ ,  $\{p_i\}$ ,  $\{\hat{p}_i\}$ , and  $\{\hat{x}_i\}$  be the set of the pre-processor’s input samples, the pre-processor’s output samples, the post-processor’s input samples, and the post-processor’s output samples, respectively. We examine the post-processor first since its operation is slightly more intuitive than the pre-processor’s.

Consider  $\hat{p}_i$  and  $\hat{p}_{i+1}$  in Figure 8.27b—two input samples of the post-processor at the boundary of two neighboring IDCT blocks—and their corresponding output samples,  $\hat{x}_i$  and  $\hat{x}_{i+1}$ . Define a crude measure of blocking artifact  $D$  as the absolute difference between two samples at the block boundary. Without any post-processing,  $D = |\hat{p}_i - \hat{p}_{i+1}|$ . It is intuitive that in order to reduce the blocking artifact

**FIGURE 8.26**

General transformation framework for block coding systems with pre- and post-processing operators.

**FIGURE 8.27**

Basic demonstration of the pre-/post-processing design. (a) Simple  $2 \times 2$  pre-/post-processing operator pair  $\{P, P^{-1}\}$  linking two DCT blocks. (b) Pre-processed and post-processed effects.

$D$ , we simply have to pull  $\hat{p}_i$  and  $\hat{p}_{i+1}$  closer (blurring out the discontinuity at boundary) depending on how far they are apart. One systematic method to achieve this objective is presented in Figure 8.27a where the scaling factor  $s$  is the single degree of freedom.

With  $P^{-1} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/s \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ , we can easily derive the following relationships

$$\hat{x}_i = \hat{p}_i + \frac{1/s - 1}{2}(\hat{p}_i - \hat{p}_{i+1}) \quad \text{and} \quad \hat{x}_{i+1} = \hat{p}_{i+1} - \frac{1/s - 1}{2}(\hat{p}_i - \hat{p}_{i+1}).$$

Hence, after post-processing,

$$\begin{aligned}\widehat{D} &= |\hat{x}_i - \hat{x}_{i+1}| = |(\hat{p}_i - \hat{p}_{i+1}) + (1/s - 1)(\hat{p}_i - \hat{p}_{i+1})| \\ &= |(\hat{p}_i - \hat{p}_{i+1})/s| = |1/s||\hat{p}_i - \hat{p}_{i+1}|.\end{aligned}\quad (8.41)$$

Choosing  $|1/s| < 1$  (or equivalently  $|s| > 1$ ), guarantees a decrease in blocking artifact. This post-processing effect is demonstrated in Figure 8.27b. For example, if we choose  $s = 2$ , then the post-processing operator partitions the distance  $D = |\hat{p}_i - \hat{p}_{i+1}|$  into four segments of equal length, moves  $\hat{x}_i$  up one segment length from  $\hat{p}_i$ , and moves  $\hat{x}_{i+1}$  down one segment length from  $\hat{p}_{i+1}$ . So, post-processing adaptively reduces the observed discrepancy  $D$  by a factor of two.

The pre-processor modifies the samples in the exact opposite direction. It attempts to “decorrelate” the boundary by lowering the smaller-value sample  $x_i$  while increase the larger-value  $x_{i+1}$  based on a percentage of their difference  $|x_i - x_{i+1}|$ . Again, the choice  $s > 1$  makes intuitive sense. If  $s < 1$ , samples are adjusted in the wrong direction. A good choice of  $s$  in the energy compaction sense for a smooth AR(1) signal model is the *Golden Ratio*  $\frac{1+\sqrt{5}}{2}$  or a close approximation such as  $\frac{8}{5}$  [35] (note that this choice also yields a multiplier-less post-processor).

### 1.8.7.2.2 More general solutions

The seemingly trivial example above actually provides a valuable lesson. When more than two samples are involved, the pre-processing operator should be designed as a *flattening* operator. It should attempt to make the input to the DCT as homogeneous as possible; thus, improving the overall energy compaction of the combined decomposition. This is quite consistent with most pre-processing schemes in practice: smoothing the input signal improves coding efficiency. However, in this framework, perfect reconstruction can be easily maintained (by choosing the post-processor as the inverse of the pre-processor). High-frequency signal components are never eliminated; they are only slightly shifted spatially. In other words, we take full advantage of the block-based framework by *carefully aligning high-frequency components at block boundaries*. Discontinuities between DCT blocks, i.e., actual high-frequency contents, do not affect coding performance whereas, within each block, data samples are smoothed out, enhancing the core block transform’s effectiveness in energy compaction.

The most general higher-order transform as shown in Figure 8.27 is equivalent to an  $M$ -channel critically-sampled filter banks with filters of length  $L = KM$  whose analysis polyphase matrix turns out to be

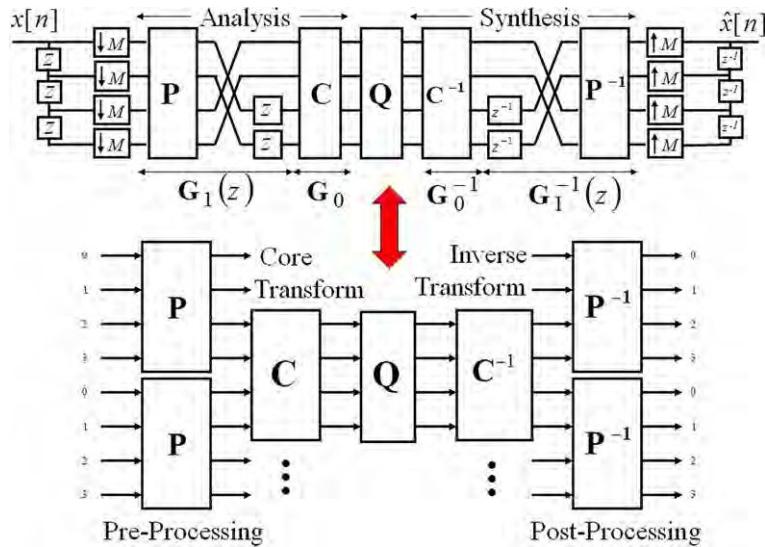
$$\mathbf{E}(z) = \mathbf{G}_0 \mathbf{G}_1(z) \mathbf{G}_1(z) \dots \mathbf{G}_{K-1}(z), \quad (8.42)$$

where each propagating component takes on the form  $\mathbf{G}_i(z) = \mathbf{\Lambda}(z) \mathbf{P}_i$  with

$$\mathbf{\Lambda}(z) \triangleq \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & z\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ z\mathbf{I} & \mathbf{0} \end{bmatrix} \quad (8.43)$$

being the permuted advance chain. The synthesis polyphase matrix is then

$$\mathbf{R}(z) = \mathbf{G}_{K-1}^{-1}(z) \mathbf{G}_{K-2}^{-1}(z) \dots \mathbf{G}_0^{-1}. \quad (8.44)$$

**FIGURE 8.28**

Construction of an  $M$ -channel filter bank from two  $M \times M$  matrices  $\mathbf{P}$  and  $\mathbf{C}$  (top) and the equivalent time-domain interpretation of pre- and post-processing (bottom).

The more detailed polyphase representation is shown in Figure 8.28 for the case of  $M = 4$  and  $K = 1$ . By choosing the inverse  $\mathbf{G}_i^{-1}(z) = \mathbf{P}_i^{-1} \mathbf{\Lambda}^{-1}(z)$ , we ensure that the resulting filter bank or decomposition has perfect reconstruction. Enforcing the tighter constraint  $\mathbf{P}^{-1} = \mathbf{P}^T$  leads to a paraunitary system, i.e., an orthogonal mapping.

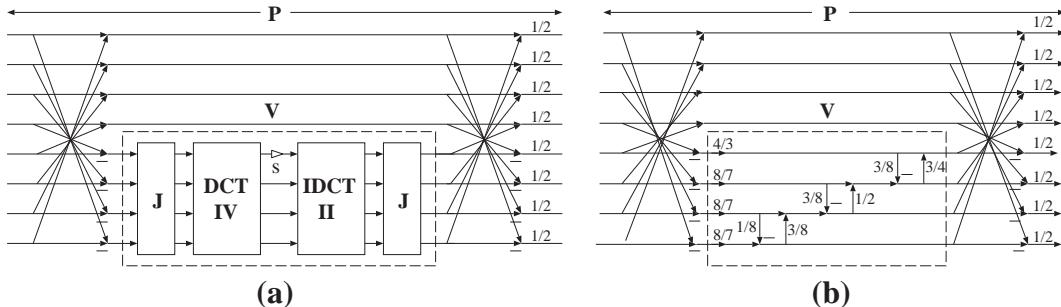
To obtain linear-phase basis functions, further structural constraint has to be imposed on the basic building block  $\mathbf{P}$ . A general closed form  $M$ -point pre-processing operator  $\mathbf{P}$  that works well with DCT of any size  $M$ , generating a linear-phase mapping, is presented in [35]. This general pre-processing block operator is shown in Figure 8.29a and it should be placed squarely between two adjacent DCT blocks as illustrated in Figure 8.28. The linear operator consists of two stages of butterflies and a matrix  $\mathbf{V}$  between them:

$$\mathbf{P} = \frac{1}{2} \begin{bmatrix} \mathbf{I} & \mathbf{J} \\ \mathbf{J} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{J} \\ \mathbf{J} & -\mathbf{I} \end{bmatrix}, \quad (8.45)$$

where the reader is reminded that  $\mathbf{I}$ ,  $\mathbf{J}$ , and  $\mathbf{0}$  are the  $\frac{M}{2} \times \frac{M}{2}$  identity matrix, the reversal matrix, and the null matrix, respectively. The matrix  $\mathbf{V}$ , controlling pre- and post-processing behavior, can be further chosen as

$$\mathbf{V} = \mathbf{J} \mathbf{C}_{M/2}^{II^T} \mathbf{S} \mathbf{C}_{M/2}^{IV} \mathbf{J}, \quad (8.46)$$

where  $\mathbf{C}_{M/2}^{II}$  is the  $\frac{M}{2}$ -point type-II DCT matrix in (8.7),  $\mathbf{C}_{M/2}^{IV}$  is the  $\frac{M}{2}$ -point type-IV DCT matrix in (8.8), and  $\mathbf{S}$  is a diagonal matrix that introduces bi-orthogonality. One example of  $\mathbf{S}$  is  $\text{diag}\{\frac{8}{5}, 1, \dots, 1\}$ . The structure produces orthogonal pre-processing linear operator if  $\mathbf{S}$  is chosen to be the identity matrix.

**FIGURE 8.29**

Pre-processors for the DCT. (a) General closed form  $M$ -point pre-filter. (b) Fast 8-point rational-coefficient pre-processing operator.

**FIGURE 8.30**

Pre-processing's block-wise flattening effect with  $8 \times 8$  block size. From left to right: original image; after 2-point, 4-point, 6-point, and 8-point pre-processing, respectively.

In general, it can be straightforwardly proven that any orthogonal matrix  $\mathbf{V}$  would yield a linear-phase orthogonal transform. For the particular case of 8-point pre/post-processing, a fast rational-coefficient approximation of  $\mathbf{P}$  is shown in Figure 8.29b.

It is interesting to note the flattening property of the pre-processing scheme as demonstrated in an image processing example shown in Figure 8.30 where pre-processing is carried out in 2D separably. When the size of the pre-processing operator grows, the pre-processed image (which will then be the input of the traditional block DCT decomposition) becomes more blocky since each  $8 \times 8$  block becomes smoother, less correlated with its neighbors, and more high-frequency components are shifted to the block boundary. The closed-form pre-processor  $\mathbf{P}$  of (8.45) depicted in Figure 8.29a is used to generate this example. At the decoder side, the post-processor—if chosen as the inverse of its counterpart—serves as a *smoothening interpolating operator* placed between block boundaries, mitigating blocking artifacts.

### 1.8.7.2.3 HD photo or JPEG-XR transform

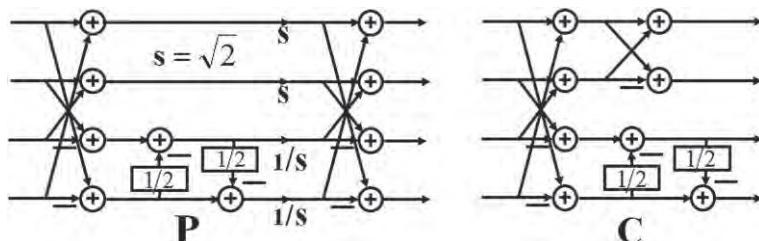
Both concepts of multiplier-less transform design via the lifting scheme as described in Section 1.8.5.4 and pre-/post-processing of Section 1.8.7.2 are on full display in JPEG-XR [27,28]. The still-image compression algorithm/file format JPEG-XR has been widely distributed along with Windows Vista and Windows 7 operating systems and was officially approved as an ISO/IEC International standard in 2010. The entire transformation stage in JPEG-XR (formerly known as Microsoft HD-Photo)—including a base integer transform called Photo Core Transform (PCT) C and an additional pre-processing operator called Photo Overlap Transform (POT) C—is just one particular solution of the much general framework illustrated in Figures 8.26 and 8.28.

Both operators are constructed from cascades of hardware-friendly dyadic-rational lifting steps. In fact, the choice of the two dyadic lifting steps  $\frac{1}{2}$  in the core transform PCT comes from Configuration C5 as shown previously in Table 8.1. In Figure 8.31, two pairs of scaling factors  $\sqrt{2}$  and  $\frac{1}{\sqrt{2}}$  are shown merely to illustrate the conceptual design. The actual implementation in JPEG-XR utilizes dyadic-rational lifting steps to approximate each pair of irrational scaling. Therefore, the entire transformation stage (including both PCT and POT) in JPEG-XR is constructed from a cascade of only dyadic-rational lifting steps [27,28].

### 1.8.7.3 Adaptive decomposition design

Here, by adaptivity, we refer to time-varying adaptable operators that the encoder or the decoder can select on-the-fly based on the local statistical behavior of the input. Adaptivity can lead to significant coding improvements if the amount of side information is cleverly avoided or kept to a minimum. It is clear that long basis functions associated with large operator size are best for representing smooth signal regions. On the other hand, transient signals, abrupt discontinuities, and strong edges are best captured with short high-time-resolution basis. Based on this observation, various adaptive operators are developed to provide a powerful decomposition framework with strong time-frequency localization. Several options are under consideration:

- varying the support of pre- and post-processing operator at each block boundary;
- varying the transform block size;



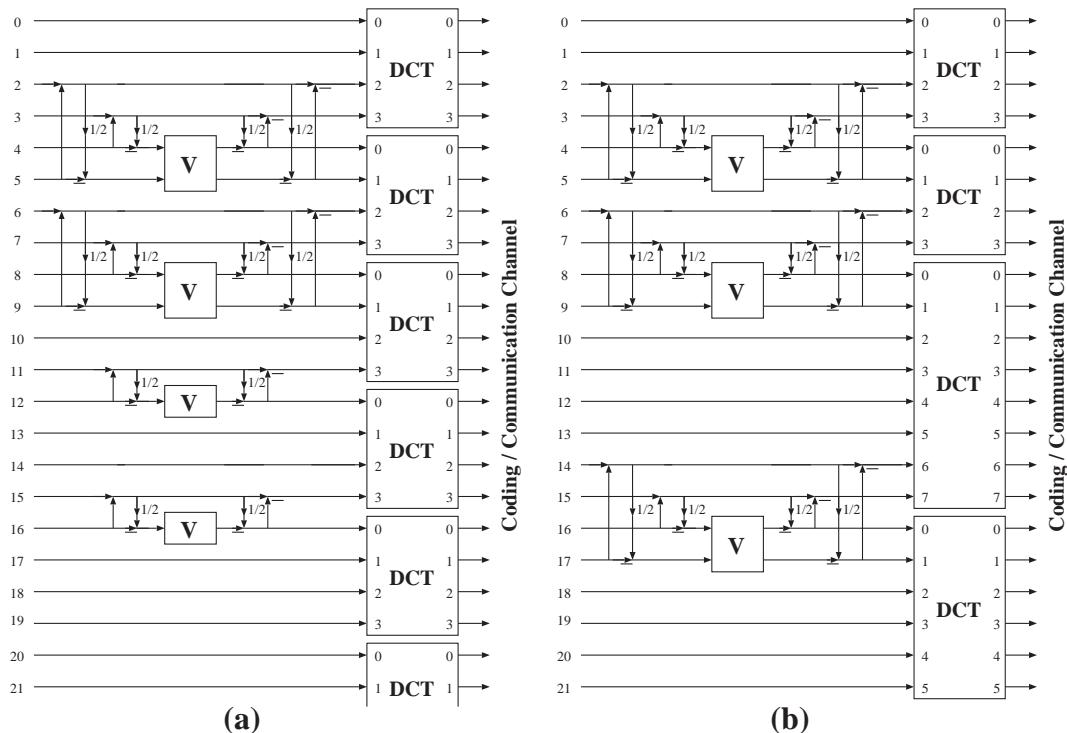
**FIGURE 8.31**

The photo overlap transform **P** (left) and the photo core transform **C** (right) in HD photo or JPEG-XR transformation stage.

- a combination of both of the above;
- allowing every operator's parameters to be time-varying.

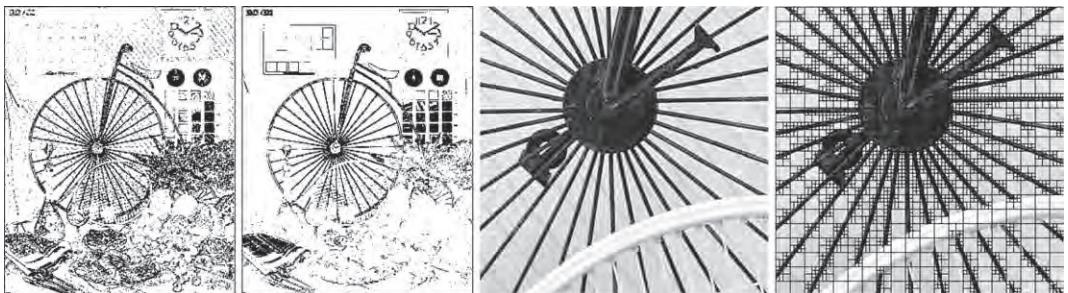
#### 1.8.7.3.1 Adaptive pre-/post-processing support

Within the pre-/post-processing framework in Section 1.8.7.2, based on the energy of the transform coefficients generated, we can decide to turn on or off pre/post-processing. It is just as easy to vary the size of pre-processing support dynamically. This adaptive-borrowing signal decomposition is illustrated in Figure 8.32a where the block size is fixed to  $M = 4$  while the pre-processing operator can be chosen among: no processing,  $2 \times 2$  processing, or  $4 \times 4$  processing. In other words, from top to bottom, we are switching from a  $4 \times 8$  to a  $4 \times 7$  to a  $4 \times 6$  to a  $4 \times 5$  and finally to a  $4 \times 4$  linear mapping. With the transform set to the DCT of a fixed block size (e.g., 8) and the size of the pre-/post-processing operator can be chosen from the set  $\{0, 2, 4, 8\}$ , then the side information for each block boundary is only 2 bits. The side information can be lowered with Huffman or context-based adaptive arithmetic coding.



**FIGURE 8.32**

Examples of adaptive time-varying decomposition. (a) Adaptive pre-processing with fixed DCT block size. (b) Adaptive DCT block size with fixed pre-processing operators.



**FIGURE 8.33**

Visual demonstration of adaptive decomposition algorithms. From left to right: pre-/post-processing mode selection for maximum local coding efficiency via exhaustive search; pre-/post-processing mode selection for fast overflow/underflow criterion; portion of the original *Bike* image; optimal block-size selection based on maximum coding efficiency criterion.

A visual illustration of adaptive pre-/post-processing is depicted in the left-most binary image in Figure 8.33. Here, dark pixels indicate blocks which the coder select the minimum-size  $2 \times 2$  operator whereas light pixels indicate blocks where the maximum-size  $8 \times 8$  pre-processing operator is preferred. The decision is based on the minimum bit-length needed to encode each particular data block and we have employed an exhaustive search [36]. It is clear that pre- and post-processing *should be avoided* at or near strong edges of the input image.

#### 1.8.7.3.2 Adaptive transform block size

Another adaptive decomposition scheme is to employ variable block sizes. In slowly-changing homogeneous part of the signal, a large data block is desirable (higher frequency resolution is needed). On the other hand, in fast-changing transient part of the signal, it is more advantageous to switch to a small block size (higher time resolution is needed). Such a signal-adaptive switching scheme has proven to be very effective in practice. For instance, MPEG-4's Advanced Audio Coder switches back-and-forth between a 256-point high-time-resolution short window and a 2048-point high-frequency-resolution long window to avoid pre-echo and to improve coding efficiency [37]. A variable-block-size decomposition scheme as depicted in Figure 8.32b can be employed where the pre-processing operator is fixed whereas the DCT block size is allowed to vary. The side information, just like in the adaptive-processing case, can be kept manageable as well with a well chosen set of block sizes, e.g.,  $\{4, 8, 16, 32\}$  for image/video applications.

A demonstration of the adaptive block transform algorithm is shown in the right of Figure 8.33. An expensive exhaustive search algorithm is set up to generate this example: pre- and post-processing is fixed to the maximum size allowable based on the size of any two neighboring DCT blocks; block size selection is restricted to  $\{4, 8, 16\}$ ; and the criterion is again the minimum amount of bits required to encode each local  $16 \times 16$  image region. This example vividly confirms that optimally adaptive decomposition algorithm has a tendency to latch onto strong local image features [36].

In the more general case, both adaptive pre-processing with different sizes, with multiple stages, and adaptive variable block size can be combined. This decomposition scheme generates a large library of basis functions that the encoder can choose from depending on the input signal behavior. How to make the right decision quickly and how to minimize the amount of side information involved are two important open research problems.

#### 1.8.7.4 Modulated design

One drawback with the cascading structural approach in (8.42) and illustrated in Figure 8.26 is that the optimization becomes problematic when the application demands or desires a larger number of channels  $M$  and/or a longer filter length  $L = KM$ . In speech and audio coding applications, for example, frequency resolution of the transform is often critical and the number  $M$  of frequency bins (or channels) is typically in the hundreds or even thousands [12,38].

Given that most of the cost function in Section 1.8.3.1 is not convex, the huge number of free parameters in the matrices  $\{\mathbf{G}_0, \mathbf{P}_i\}$ —from  $2K \binom{M/2}{2}$  in the linear-phase orthogonal case to  $K \binom{M}{2}$  in the orthogonal case to  $KM^2$  in the most general case [35]—poses a major challenge to any optimization approach. One particular elegant design that can efficiently deal with this obstacle is the cosine-modulated approach where a low-pass prototype filter is designed and the rest of the filters are obtained from cosine or sine modulation as illustrated in Figure 8.34. Many different cosine-modulation approaches have been developed and the most significant difference among them is the low-pass prototype choice and the phase of the cosine/sine sequence [6,7,12].

For the case  $K = 1$  ( $M \times 2M$  transform), the modulated design with the type-IV DCT in (8.8) is often referred to as the Modified Discrete Cosine Transform (MDCT) [39] or the Modulated Lapped Transform (MLT) [40]. The transform basis functions (or the filter coefficients) are given as

$$h_k[n] = \sqrt{\frac{2}{M}} h[n] \cos \left[ \left( n + \frac{M+1}{2} \right) \left( k + \frac{1}{2} \right) \frac{\pi}{M} \right] \quad (8.47)$$

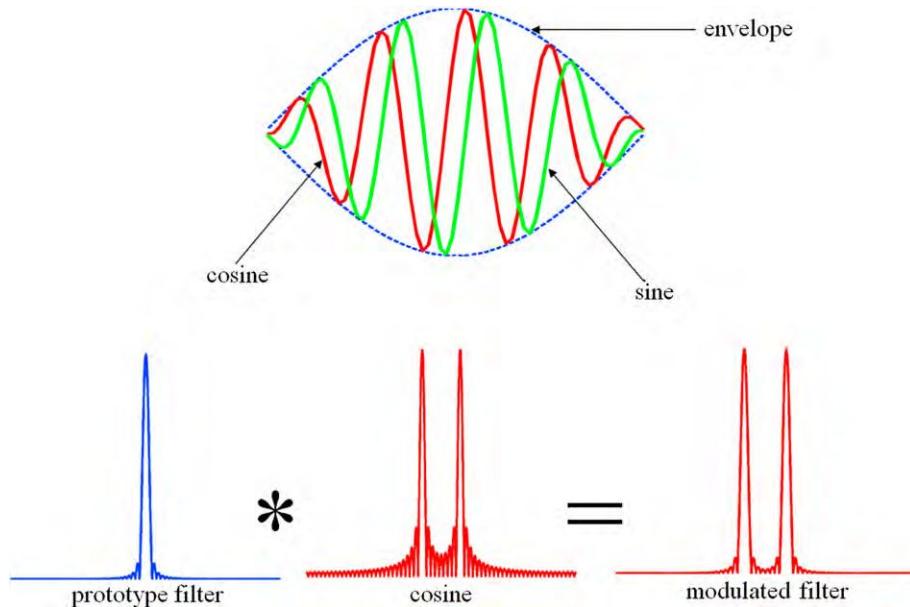
for  $0 \leq k \leq M - 1$ ,  $0 \leq n \leq 2M - 1$  and  $h[n]$  can be thought of as a symmetric window modulating the cosine sequence or the impulse response of a low-pass prototype (with cut-off frequency at  $\frac{\pi}{2M}$ ). It has been shown that as far as perfect reconstruction is concerned, the choice of the prototype window  $h[n]$  is rather flexible: all it has to satisfy is the following two constraints

$$\begin{cases} h[n] = h[2M - 1 - n], \\ h^2[n] + h^2[M - 1 - n] = 1. \end{cases} \quad (8.48)$$

A nice choice with a closed-form expression for the prototype filter  $h[n]$  is

$$h[n] = -\sin \left[ \left( n + \frac{1}{2} \right) \frac{\pi}{2M} \right], \quad (8.49)$$

which the reader can easily verify that it does not only satisfy both conditions in (8.48) but it also produces a smooth symmetric low-pass filter.

**FIGURE 8.34**

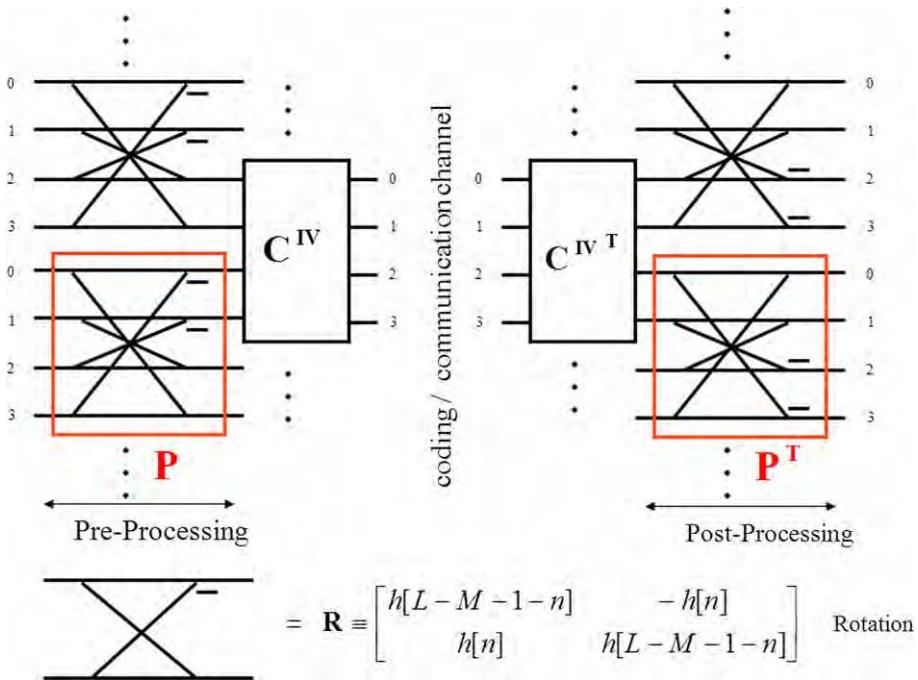
Construction via cosine/sine modulation.

It is interesting to note that the MDCT/MLT as defined in (8.47) and (8.49) also fits the pre- and post-processing framework in Figure 8.28. In this interpretation, the prototype window is completely governed by the pre-processing operator  $\mathbf{P}$  while the orthogonal core transform  $\mathbf{C}^{IV}$  takes care of the cosine-modulation part. The particular window choice in (8.49) leads to a series of  $\frac{M}{2}$  pairwise rotation angles, which yields an overall orthogonal transform. Unfortunately, we did have to give up the linear-phase property. However, for speech and audio applications, that is not a critical property. The MDCT/MLT as depicted in Figure 8.35 still has a very efficient implementation. Finally, as discussed in Section 1.8.5.2, the MDCT can easily be approximated with dyadic lifting steps, rendering a multiplier-less transformation with exact integer-to-integer mapping [41].

Both MP3 and MPEG-2 AAC employs the choice of window in (8.49). However, this is not the only choice. In fact, the open-source Vorbis codec selects a slightly different sine window

$$h[n] = \sin \left\{ \frac{\pi}{2} \sin^2 \left[ \left( n + \frac{1}{2} \right) \frac{\pi}{2M} \right] \right\}. \quad (8.50)$$

MP3 utilizes a hybrid adaptive decomposition approach: the MDCT/MLT is applied to the output of a 32-channel 512-tap near-perfect-reconstruction filter bank. MP3's hybrid filter bank adapts to the signal characteristics via the block switching technique similarly to the adaptive mechanism shown in Figure 8.32: the codec adaptively selects either a  $6 \times 12$  MDCT (labeled short block for higher time resolution)

**FIGURE 8.35**

Modulated lapped transform or MDCT implementation with pre-/post-processing operators.

or a  $18 \times 36$  MDCT (labeled long block for higher frequency resolution). On the other hand, MPEG-4 AAC dynamically switches between a  $128 \times 256$  MDCT (short blocks) and a  $1024 \times 2048$  MDCT (long blocks) [38]. The long block's frequency resolution is rather high, offering improved coding efficiency for stationary signals whereas the shorter blocks provide optimized coding capability for transient signals. The transform choice in MP3 and AAC clearly indicates the importance of adaptivity in state-of-the-art compression systems.

Finally, if the application requires even higher frequency resolution, the reader should consider the extended version of the MDCT/MLT, namely the extended lapped transform (ELT) [12, 42] whose basis functions are

$$h_k[n] = h[n] \cos \left\{ \left[ k + \frac{1}{2} \right] \left[ \left( n - \frac{L-1}{2} \right) \frac{\pi}{M} + (N+1) \frac{\pi}{2} \right] \right\} \quad (8.51)$$

for  $0 \leq k \leq M-1$ ,  $0 \leq n \leq L-1$ , and  $h[n]$  is again the impulse response of the prototype low-pass window. A major advantage of the ELT is the existence of various fast implementation algorithms which still rely on the DCT type-IV as the modulation engine but employ more layers of pre-processing operators (similarly to the overall framework shown in Figure 8.26).

---

## 1.8.8 Conclusion

In this chapter, we first briefly review the deep and rich history of transform design and then present a general systematic approach for modern transform design: the structural construction of high-order transforms from low-order basic building blocks: butterflies, lifting steps, and possibly scaling factors. We concentrate on the integer or dyadic-rational coefficient case which yields powerful integer invertible transforms. Numerous key examples in mainstream audio, image, and video applications are also presented.

---

## References

- [1] W.B. Pennebaker, J.L. Mitchell, *JPEG: Still Image Compression Standard*, Van Nostrand Reinhold, New York, NY, 1993.
- [2] J.L. Mitchell, D. LeGall, C. Fogg, *MPEG Video Compression Standard*, Chapman & Hall, New York, NY, 1996.
- [3] ITU Telecom. Standardization Sector of ITU, *Video coding for low bit rate communication*, ITU-T Recommendation H.263, March 1996.
- [4] D.S. Taubman, M.W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, Kluwer Academic Publishers, Norwell, MA, 2002.
- [5] S.G. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA, 1998.
- [6] G. Strang, T.Q. Nguyen, *Wavelets and Filter Banks*, second ed., Wellesley-Cambridge Press, Boston, 1998.
- [7] P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [8] M. Vetterli, J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.
- [9] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Comput.* 19 (1965) 297–301.
- [10] N. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, *IEEE Trans. Comput.* C23 (1974) 90–93.
- [11] K.R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, New York, NY, 1990.
- [12] H.S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Boston, MA, 1992.
- [13] I.E.G. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley and Sons, 2003.
- [14] H. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, Low-complexity transform and quantization in H.264/AVC, *IEEE Trans. Circuits Syst. Video Technol.* 13 (2003) 598–603.
- [15] W.K. Cham, Development of integer cosine transforms by the principle of dyadic symmetry, *IEE Proc.* 136 (1989) 276–282.
- [16] W.K. Cham, Y.T. Chan, An order-16 integer cosine transform, *IEEE Trans. Signal Process.* 39 (1991) 1205–1208.
- [17] K.T. Lo, W.-K. Cham, Development of simple orthogonal transforms for image compression, *IEE Proc. Vis. Image Signal Process.* 142 (1995) 22–26.
- [18] I.D. Tun, S.U. Lee, On the fixed-point-error analysis of several fast DCT algorithms, *IEEE Trans. Circuits Syst. Video Technol.* 3 (1993) 27–41.
- [19] W. Sweldens, The lifting scheme: a custom-design construction of bi-orthogonal wavelets, *Appl. Comput. Harmon. Anal.* 3 (1997) 185–200.

- [20] A.A.M.L. Bruekens, A.W.M. vanden Enden, New networks for perfect inversion and perfect reconstruction, *IEEE J. Sel. Areas Commun.* 10 (1992) 130–137.
- [21] L. Liu, T.D. Tran, P. Topiwala, From 16-bit to high-accuracy idct approximation: fruits of single architecture affiliation, in: *Proceedings of SPIE Applications of Digital Image Processing XXX*, August 2007.
- [22] H.S. Malvar, G.J. Sullivan, YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range, *JVT ISO/IEC MPEG and ITU-T VCEG*, Document No. JVT-I014r3, July 2003.
- [23] P. Topiwala, C.Tu, New Invertible Integer Color Transforms Based on Lifting Steps and Coding of 4:4:4 Video, *JVT ISO/IEC MPEG and ITU-T VCEG*, Document No. JVT-I014r8, July 2003.
- [24] K.R. Rao, P. Yip, V. Britanak, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 2007.
- [25] J. Liang, T.D. Tran, P. Topiwala, A 16-Bit Architecture for h.26l, Treating DCT Transforms and Quantization, *ITU-T Q.6/SG16*, Document No. VCEG-M16, June 2001.
- [26] J. Liang, T.D. Tran, Fast multiplier-less approximations of the DCT with the lifting scheme, *IEEE Trans. Signal Process.* 49 (2001) 3032–3044.
- [27] S. Srinivasan, C.Tu, S.L. Regunathan, G.J. Sullivan, Hd photo: a new image coding technology for digitalphotography, in: *Proceedings of SPIE Applications of Digital Image Processing XXX*, vol. 6696, August 2007.
- [28] C. Tu, S.Srinivasan, G.J. Sullivan, S.Regunathan, H.S. Malvar, Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in jpeg xr/hd photo, in: *Proceedings of SPIE Applications of Digital Image Processing XXXI*, vol. 7073, August 2008.
- [29] S. Oraintara, Y.J. Chen, T. Nguyen, Integer fast Fourier transform, *IEEE Trans. Signal Process.* 50 (2002) 607–618.
- [30] I. Daubechies, Ten lectures on wavelets, in: *Siam CBMS-NSF Regional Conference Series in Applied Mathematics*, Philadelphia, PA, 1992.
- [31] D. LeGall and A.Tabatabai, Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1988, pp. 761–765.
- [32] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, Image coding using wavelet transform, *IEEE Trans. Image Process.* 1 (1992) 205–220.
- [33] M. Under, T. Blu, Mathematical properties of the JPEG2000 wavelet filters, *IEEE Trans. Image Process.* 12 (2003) 1080–1090.
- [34] I. Daubechies, W.Sweldens, Factoring wavelet transforms into lifting steps, *J. Fourier Anal. Appl.* 4 (3) (1998) 247–269.
- [35] T.D. Tran, J. Liang, C. Tu, Lapped transform via time-domain pre- and post-filtering, *IEEE Trans. Signal Process.* 51 (2003) 1557–1571.
- [36] W. Dai, L.Liu, T.D. Tran, Adaptive block-based image coding with pre-/post-filtering, in: *Proceedings of the Data Compression Conference*, March 2005, pp. 73–82.
- [37] J.D. Johnston, S.R. Quackenbush, J. Herre, B. Grill, Review of MPEG-4 general audio coding, in: *Multimedia Systems, Standards, and Networks*, 2000, pp. 131–155.
- [38] A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding*, John Wiley and Sons, Hoboken, NJ, 2007.
- [39] J.P. Princen, A.W. Johnson, A.B. Bradley, Subband/transform coding using filter bank designs based on time domain aliasing cancellation, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 1987, pp. 2161–2164.

- [40] H.S. Malvar, Lapped transforms for efficient transform/subband coding, *IEEE Trans. Acoust. Speech Signal Process.* 38 (1990) 969–978.
- [41] Y. Yokotani, R. Geiger, G. Schuller, S. Oraintara, K.R. Rao, Lossless audio coding using the IntMDCT and rounding error shaping, *IEEE Trans. Audio Speech Lang. Process.* 14 (2006) 2201–2211.
- [42] H.S. Malvar, Extended lapped transforms: properties, applications and fast algorithms, *IEEE Trans. Signal Process.* 40 (1992) 2703–2714.

# Discrete Multi-Scale Transforms in Signal Processing

# 9

**Yufang Bao\*** and **Hamid Krim†**

\**Department of Mathematics and Computer Science/Center of Defense and Homeland Security (CDHS),  
UNC Fayetteville State University, Fayetteville, NC, USA*

†*Electrical and Computer Engineering Department, North Carolina State University, Raleigh, NC, USA*

## 1.09.1 Introduction

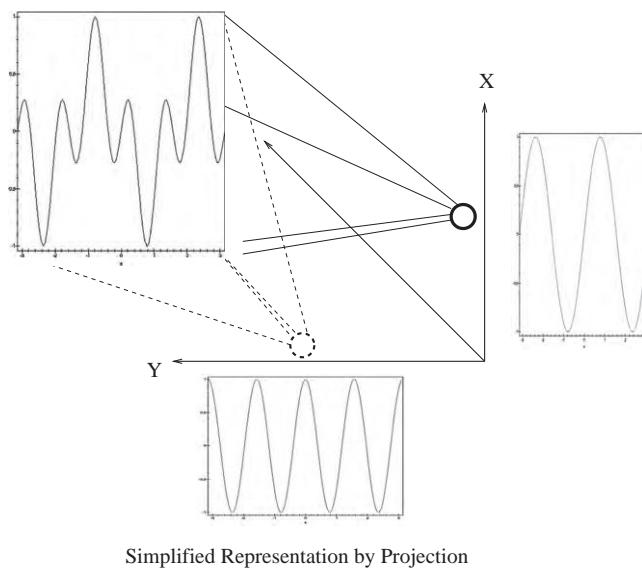
### 1.09.1.1 A overview of multiscale transforms

One's first and most natural reflex when presented with an unfamiliar object is to carefully look it over, and hold it up to the light to inspect its different facets in the hope of recognizing it, or of at least relating any of its aspects to a more familiar and well known entity. This almost innate strategy pervades all science and engineering disciplines.

Physical phenomena (e.g., earth vibrations) are monitored and observed by way of measurements in a form of temporal and/or spatial data sequences. Analyzing such data is tantamount to extracting information which is useful to further understanding of the underlying process (e.g., frequency and amplitude of vibrations may be an indicator for an imminent earthquake). Visual or manual analysis of typically massive amounts of acquired data (e.g., in remote sensing) are impractical, making one resort to adapted mathematical tools and analysis techniques to better cope with potential intricacies and complex structure of the data. Among these tools, figure a variety of functional transforms (e.g., Fourier Transform) which in many cases, may facilitate and simplify an analytical track of a problem, and frequently (and just as importantly) provide an alternative view of, and a better insight into, the problem (this, in some sense, is analogous to exploring and inspecting data under a “different light”). An illustration of such a “simplification” is shown in Figure 9.1, where a rather intricate signal  $x(t)$  shown in the leftmost figure may be displayed/viewed in a different space as two elementary tones. In Figure 9.2, a real bird chirp is similarly displayed as a fairly rich signal which, when considered in an appropriate space, is reduced and “summarized” to a few “atoms” in the Time-Frequency representation (TF). Transformed signals may formally be viewed as *convenient* representations in a different domain which is itself described by a set of vectors/functions  $\{\phi_i(t)\}_{i=\{1,2,\dots,N\}}$ . A contribution of a signal  $x(t)$  along a direction “ $\phi_i(t)$ ” (its projection) is given by the following inner product

$$C_i(x) = \langle x(t), \phi_i(t) \rangle = \int_{-\infty}^{\infty} x(t) \phi_i(t) dt, \quad (9.1)$$

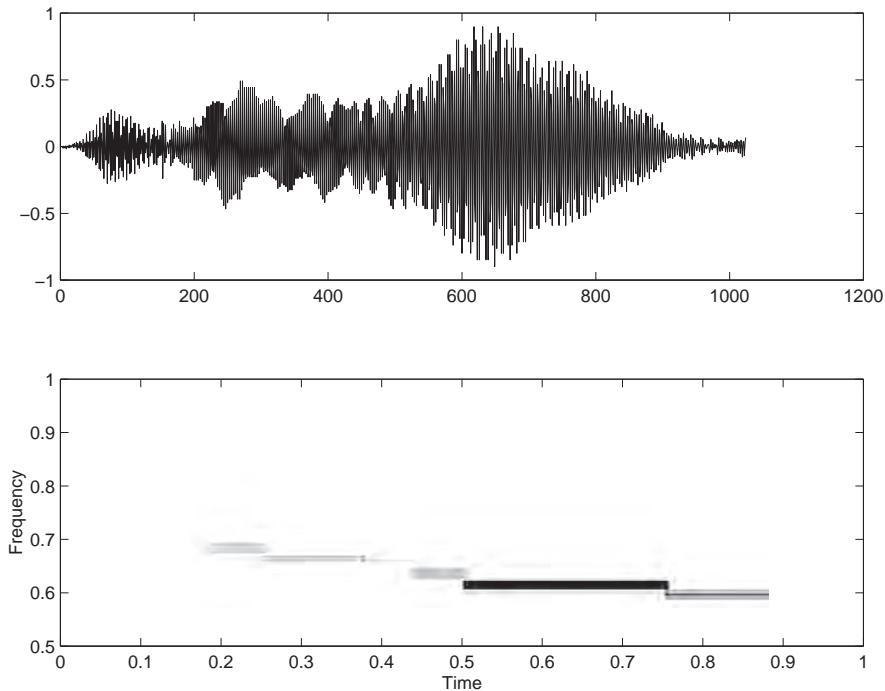
where the compact notation “ $\langle \cdot, \cdot \rangle$ ” for an inner product is used. The choice of functions in the set is usually intimately tied to the nature of information that we are seeking to extract from the signal of

**FIGURE 9.1**

A canonical function-based projection to simplify a representation.

interest. A Fourier function for example, is specified by a complex exponential “ $e^{j\omega t}$ ,” and reflects the harmonic nature of a signal, and provides it with a simple and an intuitively appealing interpretation as a “tone.” Its spectral (frequency) representation is a Dirac impulse which is well localized in frequency. A Fourier function is hence well adapted to represent a class of signals with a fairly well localized “spectrum,” and is highly inadequate for a transient signal which, in contrast, tends to be temporally well localized. This thus calls for a different class of analyzing functions, namely those with good temporal compactness. The wavelet transform, by virtue of its mathematical properties, indeed provides such an analysis framework which in many ways is complementary to that of the Fourier Transform.

The success of classical wavelets has motived the active research in the multiscale analysis field where multidisciplinary researchers pursue the best multiscale basis/frame in order to efficiently approximate a function that has singularities across scales. Although the classical wavelets can reach the optimal approximation for one-dimensional (1-D) functions, it has problems in generating sparse coefficients to approximate a two-dimensional (2-D) function with discontinuities. The classical wavelets locate singularities of a 2-D function in isotropic and multiscale representations. The Fourier frequency plane is partitioned with scaled square windows that are only in the horizontal/vertical directions. The band pass wavelet atoms, in general, are conceptually equivalent to applying differential operators (such as calculating the gradient difference) to an image to measure the regularities in localized areas along the horizontal/vertical directions. This isotropic measurement lacks flexibility, and therefore, although the classical wavelets can identify discontinuities efficiently, they can not efficiently detect the direction oriented curves in an image even at fine scales. Even though some improvements has been introduced to adopt anisotropic dilation in different directions, the problems still persist.

**FIGURE 9.2**

Bird Chirp with a simplified representation in an appropriate basis.

A 2-D signal, visually represented as an image, typically contains spatially distributed geometrical shapes that can be characterized by edges that signal the transients from one object to other objects. The edges are the discontinuities in an image intensity function along curves, contours, or lines that are directional oriented. Scientists have acknowledged [1–3] the instinct of a human eye in recognizing objects in a multiscale manner. The eye can quickly zoom into the most important isotropic or anisotropic features. Researchers in mathematics and engineering fields have extended this vision natural to the new era anisotropic wavelets. These new transformations have allowed researchers to selectively capture edges, which are important features in image processing.

The recently developed curvelets [4–7], contourlets [8,9], and shearlets [10–13] aim to mathematically animate the eye's function in order to catch the anisotropic curve features. We generally refer them to the anisotropic wavelets. The beauty with the anisotropic wavelets is their flexibility in partitioning the Fourier plane into a collection of atoms with varied scales and orientations. One advantage with the anisotropic wavelets is that, when a two dimensional function is projected into these wedge shaped atoms, the energy of singularities can be confined to only a sparse number of atoms, resulting only a small amount of significant coefficients across scales. They are more efficient tools than the classical wavelets in capturing the direction oriented curve features. Another advantage with the anisotropic wavelets is that many nice features of the classical wavelets are retained under the anisotropic wavelets. Some essential results are parallel to the classical wavelets, for example, the exact reconstruction and

energy conservation formula. This has not only made the anisotropic wavelets more attractive tools, but also allows fast development of the anisotropic wavelets using the classical wavelets as guidelines.

The goal of this chapter being of a tutorial nature, we cannot help but provide a somewhat high level exposition of the theoretical frameworks of the classical wavelets and the anisotropic wavelets, while our focus will be to provide a working knowledge of the tools as well as their potential for applications in information sciences in general. Some of the MATLAB code for the pictures used in this paper can be downloaded in [14]. Before we get into introduction of the various wavelets, we will introduce some classical Fourier transform related topics in this remaining section.

### 1.09.1.2 Signal representation in functional bases

As noted above, a proper selection of a set of analyzing functions heavily impacts the resulting representation of an observed signal in the dual space, and hence the resulting insight as well.

When considering finite energy signals (these are also said to be  $L^2(\mathcal{R}^d)$ ), i.e.,

$$\int_{\mathcal{R}^d} |x(t)|^2 dt < \infty, \quad (9.2)$$

a convenient functional vector space is that which is endowed with a norm inducing inner product, namely a Hilbert space, which will also be our assumed space throughout.

**Definition 1.** A vector space endowed with an inner product, which in turn induces a norm, and such that every sequence of functions  $x_n(t)$  whose elements are asymptotically close is convergent (this convergence is in the Cauchy sense whose technical details are deliberately left out to help the flow of the chapter), is called a Hilbert space.

**Remark.** An  $L^2(\mathcal{R}^d)$ ,  $d = 1, 2$  space is a Hilbert space.

**Definition 2.** Given  $\alpha > 0$ , a function  $x(t)$  is said to be Lipschitz- $\alpha$  at  $t_0 \in \mathcal{R}^d$ , if there exists a positive constant  $K$  and a polynomial  $p_{n,t_0}$  with degree  $n = [\alpha]$  such that, for all  $t$  in a neighborhood of  $t_0$ , we have

$$|x(t) - p_{n,t_0}(t)| \leq K|t - t_0|^\alpha. \quad (9.3)$$

It is shown that function  $x(t)$  is necessarily  $m$  times differentiable at point  $t_0$  if  $x(t)$  is uniformly Lipschitz with  $\alpha > m$  in a neighborhood of  $t_0$ . Further, if  $x(t)$  is Lipschitz  $\alpha$  with  $\alpha < 1$  at  $t_0$ , then  $f$  is not differentiable at  $t_0$ .

**Definition 3.** A set of vectors or functions  $\{\phi_i(t)\}$ ,  $i = 1, \dots$ , which are linearly independent and which span a vector space, is called a *basis*. If in addition these elements are orthonormal, then the basis is said to be an orthonormal basis.

Among the desirable properties of a selected basis in such a space are *adaptivity* and a *capacity* to preserve and reflect some key signal characteristics (e.g., signal smoothness). Fourier bases have in the past been at the center of all science and engineering applications. They have in addition, provided an efficient framework for analyzing periodic as well as non-periodic signals with dominant modes.

### 1.09.1.3 The continuous Fourier transform

The Fourier transform is a powerful tool in understanding features of signals/images through a basis. A Fourier Transform essentially measures the spectral content of a signal  $x(t)$  across all frequencies “ $\omega$ ,” view as row vector, and is defined as the decomposition of the signal over a basis  $\{e^{-j\omega t}\}$  as.

**Definition 4.** For a signal  $x(t), t \in \mathbb{R}^d$  viewed as a column vector, that is an absolutely integrable function, its Fourier transform is defined as

$$X(\omega) = \int_{\mathbb{R}^d} x(t) e^{-j\omega t} dt. \quad (9.4)$$

The formula is also referred to as the Fourier integral. The signal  $x(t)$  can be recovered by its inverse Fourier transform that is defined as

$$x(t) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} X(\omega) e^{j\omega t} d\omega. \quad (9.5)$$

As an orthonormal transform, the FT enjoys a number of interesting properties [15]. To simplify, we are constrained to 1-D function  $f(t)$  in the following until we get to the sections of anisotropic wavelets. If we use  $FT(f)(\omega) = F(\omega)$  to represent the Fourier transform of  $f(t)$ , we can list some of its important properties as follows,

- *Linearity:*

$$FT(f + g)(\omega) = F(\omega) + G(\omega).$$

- *Convolution:*

$$FT(f \star g)(\omega) = F(\omega)G(\omega).$$

- *Multiplication:*

$$FT(fg)(\omega) = F(\omega) \star G(\omega).$$

- *Shifting:*

$$FT(x(t - s))(\omega) = e^{-js\omega} X(\omega).$$

- *Scaling:*

$$FT(x(at))(\omega) = |a|^{-1} X(a^{-1}\omega).$$

- *Differentiation:*

$$FT\left(\frac{\partial^n x(t)}{\partial t^n}\right)(\omega) = (j\omega)^n X(\omega).$$

Throughout, and unless otherwise specified, the  $\star$  symbol between two functions usually indicates a convolution between the two functions, and as a superscript denotes a complex conjugate. Furthermore, a useful property of 1-D the Fourier transform is its energy preservation in the dual space (transform domain),

$$\int_{\mathbb{R}} |x(t)|^2 dt = \int_{\mathbb{R}} |X(\omega)|^2 d\omega.$$

This is also true for signals in  $\mathbb{R}^d$ , and is referred to as the Plancherel/Parseval property [15].

### 1.09.1.4 Fourier series

If a canonical function of a selected basis is  $\phi(t) = e^{j\omega t}$ , where  $\omega \in \mathbb{R}$ , then we speak of a Fourier basis which yields a Fourier Series (FS) for periodic signals and a Fourier Transform (FT) for aperiodic signals.

The Fourier Transform is defined for a broad class of signals [16], and may also be specialized to derive the Fourier Series (FS) of a periodic signal. This is easily achieved by noting that a periodic signal  $\tilde{x}(t)$  may be evaluated over a period  $T$ , which in turn leads to,

$$\tilde{x}(t) = x(t + T) = \sum_{n=-\infty}^{\infty} \alpha_n^x e^{jn\omega_0 t}, \quad (9.6)$$

with  $\omega_0 = 2\pi/T$  and  $\alpha_n^x = 1/T \int_0^T x(t) e^{-jn\omega_0 t} dt$ .

In applications, however,  $x(t)$  is measured and sampled at discrete times, requiring that the aforementioned transform be extended to obtain a spectral representation which closely approximates the theoretical truth and which remains just as informative. Towards that end, we proceed to define a Discrete Time Fourier Transform (DTFT) as

$$X(e^{j\omega}) = \sum_{i=-\infty}^{\infty} x(i) e^{-j\omega i}. \quad (9.7)$$

This expression may also be extended for finite observation (finite dimensional) signals via the Fast Fourier Transform [15]. While these transforms have been, and remain crucial in many applications, they show limitations in problems requiring a good “time-frequency” localization as often encountered in transient analysis. This may easily be understood by reinterpreting Eq. (9.7) as a weighted transform where each of the time samples is equally weighted in the summation for  $X(e^{j\omega})$ . Prior to introducing alternatives to uniform weighting in the Fourier space, we first discuss the practical implementation and exploitation of the DTFT.

### 1.09.1.5 The discrete Fourier transform

In application, the Discrete Fourier Transform is the computational surrogate for the continuous Fourier Transform and a natural extension of the DTFT. It results in a periodic sequence of numbers that is mapped to discretely sampled values of an analog signal (viewed as a time function). The sampling is performed over a  $N$ -long window which is extended periodically. The discrete Fourier transform (DFT) also views a finite length signal,  $x(n)$ , as a periodic signal, and is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N} kn}.$$

The original signal may be recovered by applying an inverse transform

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi}{N} kn}.$$

The Discrete Fourier transform decomposes a signal into an orthonormal basis. The energy is preserved by the Plancherel formula described earlier and written as

$$\|f\|^2 = \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2.$$

The discrete Fourier transform of a two-dimensional signal (image) is correspondingly defined as

$$X(k_1, k_2) = \sum_{n_1, n_2=0}^{N-1} x(n_1, n_2) e^{-j2\pi(k_1 n_1 + k_2 n_2)/N}.$$

The original signal may also be recovered by applying an inverse transform

$$x(n_1, n_2) = \frac{1}{N^2} \sum_{k_1, k_2=0}^{N-1} X(k_1, k_2) e^{j2\pi(k_1 n_1 + k_2 n_2)/N}.$$

Since the DFT is based on the sampling taken over a certain length, the frequency content of the original function may not be adequately represented by the DFT when the sampling rate is insufficiently low. This will cause the high frequency content to fold back into the lower frequency content and yield aliasing.

To overcome some of the limitations of the uniform weighting of a function of a classical FT, Gabor [17] first proposed to use a different weighting window leading to the so-called windowed FT which served well in many practical applications [18] and is discussed next.

### 1.09.1.6 Windowed Fourier transform

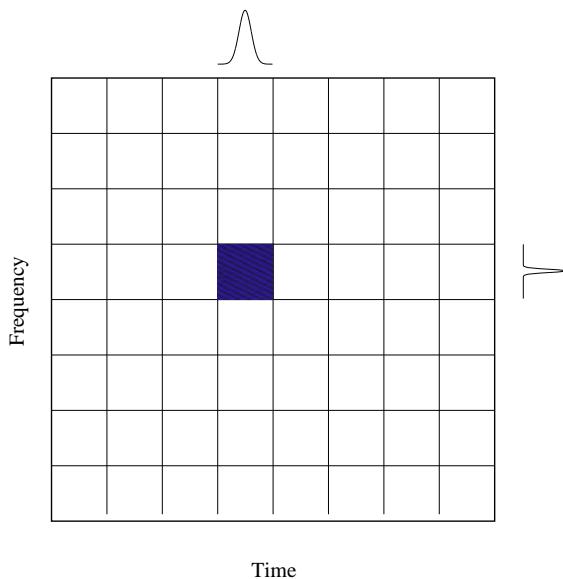
As just mentioned, when our goal is to analyze very local features, such as those present in transient signals for instance, it then makes sense to introduce a focusing window as follows:

$$W_{\mu, \omega}(t) = e^{j\omega t} W(t - \mu),$$

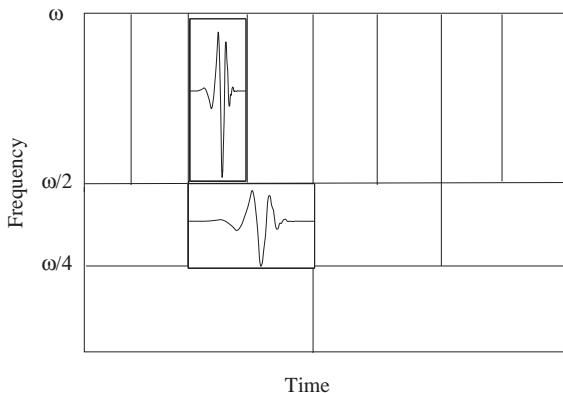
where  $\|W_{(\mu, \omega)}\| = 1 \forall (\mu, \omega) \in \mathbb{R}^2$  and where  $W(\cdot)$  is typically a smooth function with a compact support. This yields the following parameterized transform:

$$\hat{X}_W(\omega, \mu) = \langle x(t), W_{\mu, \omega}(t) \rangle. \quad (9.8)$$

The selection of a proper window is problem-dependent and is ultimately resolved by the desired spectro-temporal trade-off which is itself constrained by the Heisenberg uncertainty principle [18, 19]. From the T-F distribution perspective, and as discussed by Gabor [17] and displayed in Figure 9.3, the Gaussian window may be shown to have minimal temporal as well as spectral support of any other function. It hence represents the best compromise between the temporal and spectral resolution. Its numerical implementation entails a discretization of the modulation and translation parameters, and results in a uniform partitioning of the T-F plane as illustrated in Figure 9.4. Different windows result in various T-F distributions of elementary atoms, favoring either temporal or spectral resolution as may be seen for the different windows in Figure 9.5. While representing an optimal time-frequency compromise, the uniform coverage of the T-F plane by the Gabor transform falls short of adequately resolving a signal whose components are spectrally far apart. This may easily and convincingly be illustrated by the study case in Figure 9.6, where we note the number of cycles which may be enumerated within a window of fixed time width. It is readily seen that while the selected window (shown grid) may be adequate for one fixed frequency component, it is inadequate for another lower frequency component. An analysis of a spectrum exhibiting such a time-varying behavior is ideally performed by way of a frequency-dependent

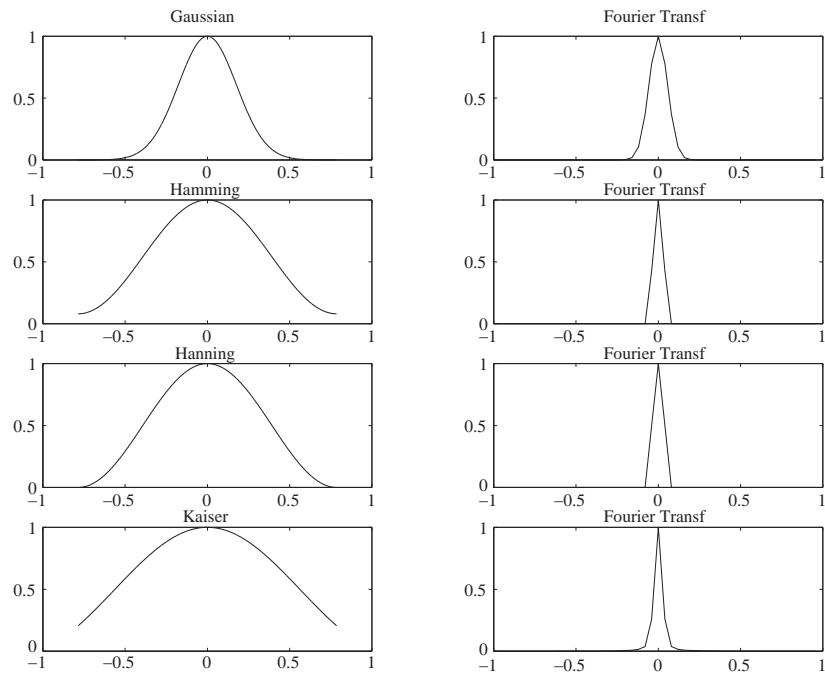
**FIGURE 9.3**

A Gaussian waveform results in a uniform analysis in the time-frequency plane.

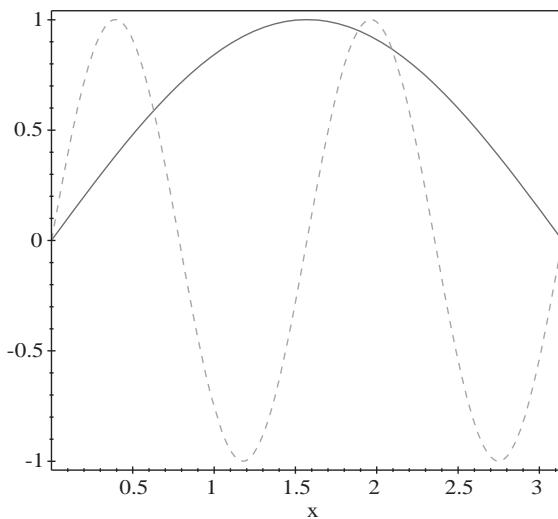
**FIGURE 9.4**

A time frequency tiling of dyadic wavelet basis by a proper sub-sampling of a wavelet frame.

time window in different designs as we elaborate in the next several sections. The wavelet transform described next, offers a highly adaptive window which is of compact support, and which, by virtue of its dilations and translations covers different spectral bands at all instants of time. The anisotropic wavelets using different windows, such as the curvelet transform, the contourlet transform, and the shearlet transform, will be introduced following the discussion of wavelet transform and wavelet decomposition.

**FIGURE 9.5**

Trade-off resulting from windows.

**FIGURE 9.6**

Time windows and frequency trade-off.

## 1.09.2 Wavelets: a multiscale analysis tool

This section is organized as follows: In Section 1.09.2.1, we define a multiscale analysis based on a wavelet basis and elaborate on their properties and their implications, as well as on their applications. In Section 1.09.2.2, we discuss a specific estimation technique which is believed to be sufficiently generic and general to be useful in a number of different applications of information sciences in general, and of signal processing and communications in particular.

### 1.09.2.1 Wavelet transform

Much like the FT, the WT is based on an elementary function, which is well localized in time and frequency. In addition to a compactness property, this function has to satisfy a set of properties to be admissible as a wavelet. The first fundamental property is stated next,

**Definition 5.** [18,20,21] A wavelet is a finite energy function  $\psi(\cdot)$  (i.e.,  $\psi(\cdot) \in L^2(\mathcal{R})$ ) with zero mean,

$$\int_{-\infty}^{\infty} \psi(t) dt = 0. \quad (9.9)$$

Commonly normalized so that  $\|\psi\| = \int |\psi|^2 dt = 1$ , it also constitutes a fundamental building block in the construction of functions (atoms) spanning the time-scale plane by way of dilation and translation parameters. We hence write,

$$\psi_{\mu,\xi}(t) = \frac{1}{\sqrt{\xi}} \psi\left(\frac{t - \mu}{\xi}\right),$$

where the scaling factor  $\xi^{\frac{1}{2}}$  ensures an energy invariance of  $\psi_{\mu,\xi}(t)$  over all dilations “ $\xi$ ”  $\in \mathcal{R}^+$  and translations “ $\mu$ ”  $\in \mathcal{R}$ . With such a function in hand, and towards mitigating the limitation of a windowed FT, we proceed to define a Wavelet Transform (WT) with such a capacity. A scale-dependent window with good time localization properties as shown in Figure 9.7, yields a transform for  $x(t)$  given by

$$\mathcal{W}_x(\mu, \xi) = \int_{-\infty}^{\infty} x(t) \frac{1}{\sqrt{\xi}} \psi^*\left(\frac{t - \mu}{\xi}\right) dt, \quad (9.10)$$

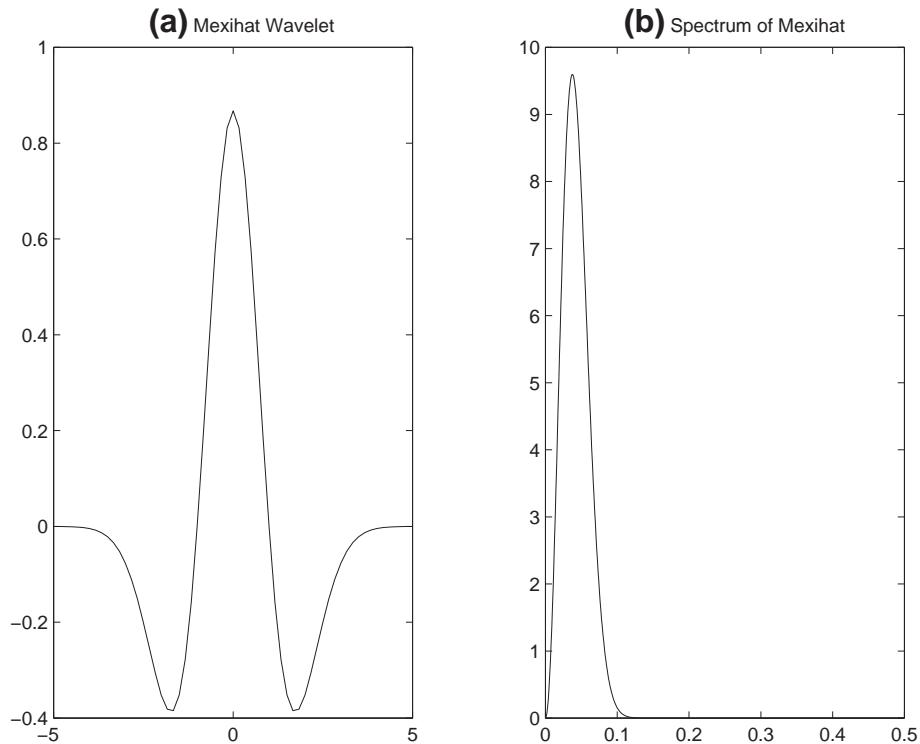
where “\*” denotes complex conjugate. This is of course in contrast to the Gabor transform whose window width remains constant throughout. A time-frequency plot of a continuous wavelet transform is shown in Figure 9.8 for a corresponding  $x(t)$ .

### 1.09.2.2 Inverting the wavelet transform

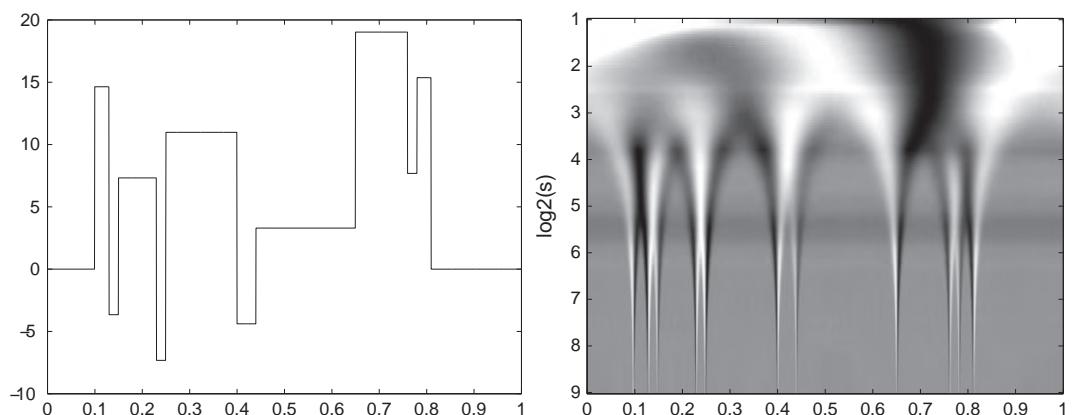
Similarly to the weighted FT, the WT is a redundant representation, which with a proper normalization factor, leads to a reconstruction formula,

$$x(t) = \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty \mathcal{W}_x(\mu, \xi) \frac{1}{\sqrt{\xi}} \psi\left(\frac{t - \mu}{\xi}\right) \frac{d\xi}{\xi^2} d\mu, \quad (9.11)$$

with  $C_\psi = \int_0^{+\infty} \frac{\hat{\Psi}^2(\omega)}{\omega} d\omega$ .

**FIGURE 9.7**

Admissible Mexican hat wavelet.

**FIGURE 9.8**

Continuous signal with corresponding wavelet transform with cones of influence around singularities.

While the direct and inverse WT have been successfully applied in a number of different problems [18], their computational cost due to their continuous/redundant nature is considered as a serious drawback. An immediate and natural question arises about a potential reduction in the computational complexity, and hence in the WT redundancy. This has clearly to be carefully carried out to guarantee a sufficient coverage of the T-F plane, and thereby ensure a proper reconstruction of a transformed signal. Upon discretizing the scale and dilation parameters, a dimension reduction relative to a continuous transform is achieved. The desired adaptivity of the window width with varying frequency naturally results by selecting a geometric variation of the dilation parameter  $\xi = \xi_0^m, m \in \mathbb{Z}$  (*set of all positive and negative integers*). To obtain a systematic and consistent coverage of the T-F plane, our choice of the translation parameter “ $\mu$ ” should be in congruence with the fact that at any scale “ $m$ ,” the coverage of the whole line  $\mathcal{R}$  (for instance) is complete, and the translation parameter be in step with the chosen wavelet  $\psi(t)$ , i.e.,  $\mu = n\mu_0\xi_0^m$  with  $n \in \mathbb{Z}$ . This hence gives the following scale and translation adaptive wavelet:

$$\psi(t)_{m,n} = \xi_0^{-m/2} \psi \left( \frac{t}{\xi_0^m} - n\mu_0 \right), \quad (9.12)$$

where the factor “ $\xi_0^{-m/2}$ ,” ensures a unit energy function. This reduction in dimensionality yields a redundant discrete wavelet transform endowed with a structure which lends itself to an iterative and fast inversion/reconstruction of a transformed signal  $x(t)$ .

**Definition 6.** The set of resulting wavelet coefficients  $\{\langle x(t), \psi_{mn}(t) \rangle\}_{(m,n) \in \mathbb{Z}^2}$  completely characterizes  $x(t)$ , and hence leads to a stable reconstruction [18, 19] if  $\forall x(t) \in L^2(\mathcal{R})$  (or finite energy signal) the following condition on the energy holds,

$$A \|x(t)\|^2 \leq \sum_{m,n} |\langle x(t), \psi_{mn}(t) \rangle|^2 \leq B \|x(t)\|^2. \quad (9.13)$$

Such a set of functions  $\{\psi_{mn}(t)\}_{(m,n) \in \mathbb{Z}^2}$  then constitutes a frame.

The energy inequality condition intuitively suggests that the redundancy should be controlled to avoid instabilities in the reconstruction (i.e., too much redundancy makes it more likely that any perturbation of coefficients will yield an unstable/inconsistent reconstruction). If the frame bounds “ $A$ ” and “ $B$ ” are equal, the corresponding frame is said to be tight, and if furthermore  $A = B = 1$ , it is an orthonormal basis. Note, however, that for any  $A \neq 1$  a frame is not a basis. In some cases, the frame bounds specify a redundancy ratio which may help guide one in inverting a frame representation of a signal, since a unique inverse does not exist. An efficient computation of an inverse (for reconstruction), or more precisely of a pseudo-inverse, may only be obtained by a judicious manipulation of the reconstruction formula [18, 19]. This is, in a finite dimensional setting, similar to a linear system of equations with a rank deficient matrix whose column space has an orthogonal complement (the union of the two subspaces yields all of the Hilbert space), and hence whose inversion is ill conditioned. The size of this space is determined by the order of the deficiency (number of linearly dependent columns), and explains the potential for instability in a frame projection-based signal reconstruction [18]. This type of “effective rank” utilization is encountered in numerical linear algebra applications (e.g., signal subspace methods [22], model order identification, etc.). The close connection between the redundancy of a frame and the rank deficiency of a matrix, suggests that solutions available in linear algebra [23] may provide

insight in solving our frame-based reconstruction problem. A well known iterative solution to a linear system and based on a gradient search solution is described in [24], (and in fact coincides with a popular iterative solution to the frame algorithm for reconstructing a signal) for solving

$$\mathbf{f} = \mathbf{L}^{-1}\mathbf{b},$$

where “ $\mathbf{L}$ ” is a matrix operating on  $\mathbf{f}$ , and  $\mathbf{b}$  is the data vector. Starting with an initial guess  $\mathbf{f}_0$  and iterating on it leads to

$$\mathbf{f}_n = \mathbf{f}_{n-1} + \alpha(\mathbf{b} - \mathbf{L}\mathbf{f}_{n-1}). \quad (9.14)$$

When  $\alpha$  is appropriately selected, the latter iteration may be shown to yield [18]

$$\lim_{n \rightarrow +\infty} \mathbf{f}_n = \mathbf{f}. \quad (9.15)$$

Numerous other good solutions have also been proposed in the literature and their detailed descriptions are given in [18, 19, 25, 26].

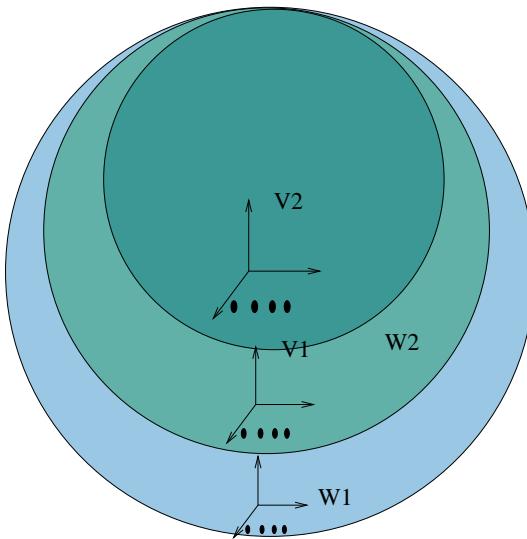
### 1.09.2.3 Wavelets and multiresolution analysis

While a frame representation of a signal is of lower dimension than that of its continuous counterpart, a complete elimination of redundancy is only possible by orthonormalizing a basis. A proper construction of such a basis ensures orthogonality within scales as well as across scales. The existence of such a basis with a dyadic scale progression was first shown, and an explicit construction was given by Meyer [27] and Daubechies et al. [19]. The connection to subband coding discovered by Mallat [28] resulted in a numerically stable and efficient implementation which helped propel wavelet analysis at the forefront of computational sciences (see Ref. [29] for a comprehensive historical as well technical development [19] which also led to Daubechies celebrated orthonormal wavelet bases).

To help maintain a smooth flow of this chapter, and achieve the goal of endowing an advanced undergraduate with a working knowledge of the Multiresolution Analysis (MRA) framework, we give a high level introduction, albeit comprehensive, and defer most of the technical details to the sources in the bibliography. For example, the trade-off in time-frequency resolution advocated earlier, lies at the heart of the often technical wavelet design and construction. The balance between the spectral and temporal decay which constitutes one of the key design criteria, has led to a wealth of new functional bases, and this upon the introduction of the now classical multiresolution analysis framework [18, 30]. The pursuit of a more refined analysis arsenal resulted in the recently introduced nonlinear multiresolution analysis in [31–33].

#### 1.09.2.3.1 Multiresolution analysis

The Multiresolution Analysis theory (MRA) developed by Mallat [34] and Meyer [27], may be viewed as a functional analytic approach to subband signal analysis, which had previously been introduced in and applied to engineering problems [35, 36]. The clever connection between an orthonormal wavelet decomposition of a signal and its filtering by a bank of corresponding filters led to an axiomatic formalization of the theory and subsequent equivalence. This as a result, opened up a new wide avenue of research in the development and construction of new functional bases as well as filter banks [21, 37–39]. The construction of a telescopic set of nested approximation and detail subspaces  $\{V_j\}_{j \in \mathbb{Z}}$  and  $\{W_j\}_{j \in \mathbb{Z}}$

**FIGURE 9.9**

Hierarchy of wavelet bases.

each endowed with an orthonormal basis, as shown in Figure 9.9, is a key step of the analysis. An inter and intra scale orthogonality of the wavelet functions, as noted above, is preserved, with the inter-scale orthogonality expressed as

$$W_i \perp W_j \quad \forall i \neq j. \quad (9.16)$$

By replacing the discrete parameter wavelet  $\psi_{ij}(t)$  where  $(i, j) \in \mathbb{Z}^2$  (*set of all positive and negative 2-tuple integers*) respectively denote the translation and the scale parameters, in Eq. (9.10) (i.e.,  $\mu = 2^j i$ ,  $\xi = 2^j$ ), we obtain the orthonormal wavelet coefficients as [18],

$$c_j^i(x) = \langle x(t), \psi_{ij}(t) \rangle. \quad (9.17)$$

The orthogonal complementarity of the scaling subspace and that of the residuals/details amount to synthesizing the higher resolution subspace

$$V_i \oplus W_i = V_{i-1}.$$

Iterating this property, leads to a reconstruction of the original space where the observed signal lies, and which in practice, is taken to be  $V_0$ . That is, the observed signal at its first and finest resolution (this may be viewed as implicitly accepting the samples of a given signal as the coefficients in an approximation space with a scaling function corresponding to that which the subsequent analysis is based on). The dyadic scale progression has been thoroughly investigated, and its wide acceptance/popularity is due to its tight connection with subband coding, whose practical implementation is fast and simple. Other nondyadic developments have also been explored (see for e.g., [21]).

The qualitative characteristics of the MRA we have thus far discussed, may be succinctly stated as follows,

**Definition 7.** [18] A sequence  $\{V_i\}_{i \in \mathcal{Z}}$  of closed subspaces of  $L^2(\mathcal{R})$  is a multiresolution approximation if the following properties hold:

- $\forall (j, k) \in \mathcal{Z}^2, x(t) \in V_j \Leftrightarrow x(t - 2^j k) \in V_j$ ,
- $\forall j \in \mathcal{Z}, V_{j+1} \subset V_j$ ,
- $\forall j \in \mathcal{Z}, x(t) \in V_j \Leftrightarrow x\left(\frac{t}{2}\right) \in V_{j+1}$ ,
- $\lim_{j \rightarrow -\infty} V_j = \overline{\bigcap_{j=-\infty}^{\infty} V_j} = \{0\}$ ,
- $\lim_{j \rightarrow -\infty} \overline{\bigcup_{j=-\infty}^{\infty} V_j} = L^2(\mathcal{R})$ ,
- There exists a function  $\phi(t)$  such that  $\{\phi(t - n)\}_{n \in \mathcal{Z}}$  is a Riesz basis of  $V_0$ , where the “overbar” denotes closure of the space.

### 1.09.2.3.2 Properties of wavelets

A wavelet analysis of a signal assumes a judiciously preselected wavelet, and hence a prior knowledge about the signal itself. As stated earlier, the properties of an analyzing wavelet have a direct impact on the resulting multiscale signal representation. Carrying out a useful and meaningful analysis is hence facilitated by a good understanding of some fundamental wavelet properties.

### 1.09.2.3.3 Vanishing moments

Recall that one of the fundamental admissibility conditions of a wavelet is that its first moment be zero. This is intuitively understood in the sense that a wavelet focuses on residuals or oscillating features of a signal. This property may in fact be further exploited by constructing a wavelet with an arbitrary number of vanishing moments. We say that a wavelet has  $n$  vanishing moments if

$$\int \psi(t)t^i dt = 0, \quad i = \{0, 1, \dots, n-1\}. \quad (9.18)$$

Reflecting a bit on the properties of a Fourier Transform of a function [15], it is easy to note that the number of zero moments of a wavelet reflects the behavior of its Fourier Transform around zero. This property is also useful in applications like compression, where it is highly desirable to maximize the number of small/negligible coefficients, and only preserve a minimal number of large coefficients. The associated cost with increasing the number of vanishing moments is that of an increased support size for the wavelet, hence that of the corresponding filter [18, 19], and hence of some of its localizing potential.

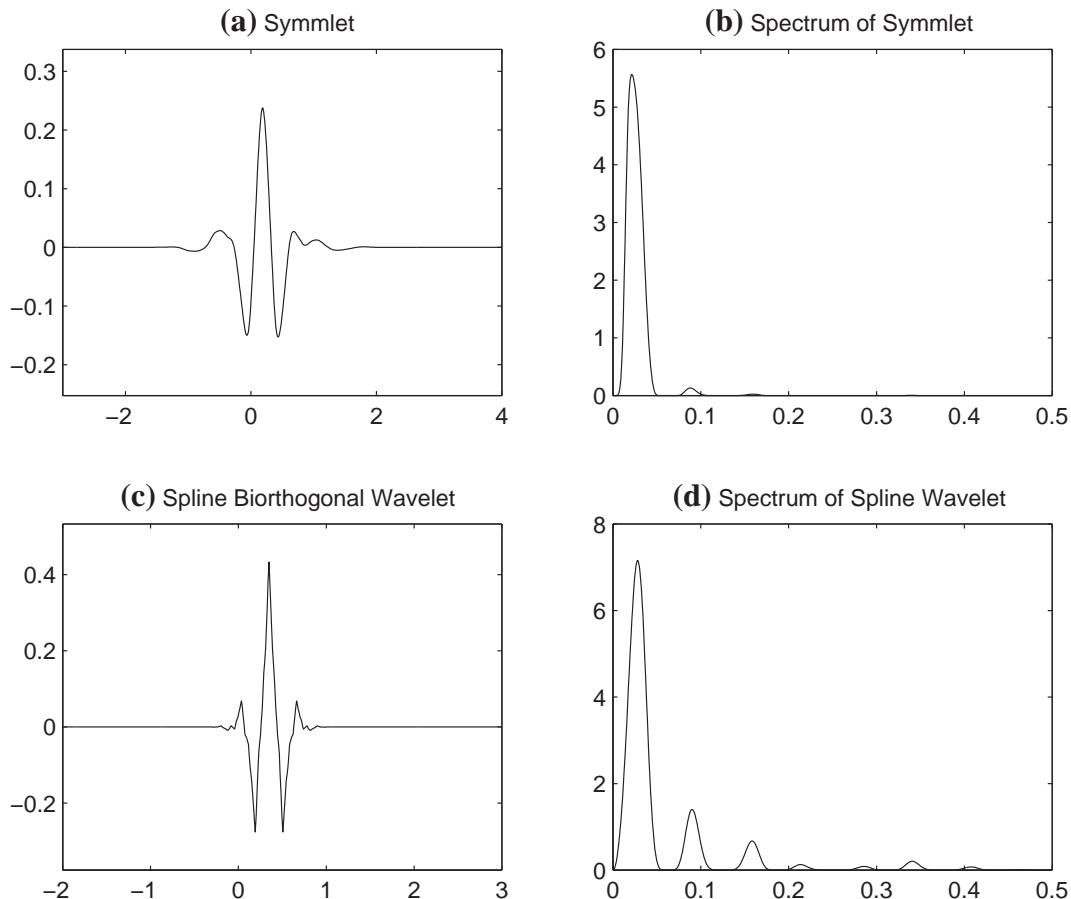
### 1.09.2.3.4 Regularity/smoothness

The smoothness of a wavelet “ $\psi(t)$ ” is important for an accurate and parsimonious signal approximation. For a large class of wavelets (those relevant to applications), the smoothness (or regularity) property of a wavelet, which may also be measured by its degree of differentiability ( $d^\alpha \psi(t)/dt^\alpha$ ) or equivalently by its Lipschitzity “ $\gamma$ ,” is also reflected by its number of vanishing moments [18]. The larger the number of vanishing moments, the smoother the function. In applications, such as image coding, a smooth analyzing wavelet is useful for not only compressing the image, but for controlling the visual distortion due to errors as well. The associated cost (i.e., some trade-off is in order) is again a size increase in the

wavelet support, which may in turn make it more difficult to capture local features, such as important transient phenomena.

#### 1.09.2.3.5 Wavelet symmetry

At the exception of a Haar wavelet, compactly supported real wavelets are asymmetric around their center point. A symmetric wavelet clearly corresponds to a symmetric filter which is characterized by a linear phase. The symmetry property is important for certain applications where symmetric features are crucial (e.g., symmetric error in image coding is better perceived). In many applications, however, it is generally viewed as a property secondary to those described above. When such a property is desired, truly symmetric biorthogonal wavelets have been proposed and constructed [19, 40], with the slight disadvantage of using different analysis and synthesis mother wavelets. In Figure 9.10, we show some



**FIGURE 9.10**

Biorthogonal wavelets preserve symmetry and symlets nearly do.

illustrative cases of symmetric wavelets. Other nearly symmetric functions (referred to as symlets) have also been proposed, and a detailed discussion of the pros and cons of each, is deferred to Daubechies [19].

#### 1.09.2.3.6 A filter bank view: implementation

As noted earlier, the connection between a MRA of a signal and its processing by a filter bank was not only of intellectual interest, but was of breakthrough proportion for general applications as well. It indeed provided a highly efficient numerical implementation for a theoretically very powerful methodology. Such a connection is most easily established by invoking the nestedness property of MR subspaces. Specifically, it implies that if  $\phi(2^{-j}t) \in V_j$  and  $V_j \subset V_{j-1}$ , and we can hence write,

$$\frac{1}{2^{j/2}}\phi(2^{-j}t) = \frac{1}{2^{\frac{j-1}{2}}} \sum_{k=-\infty}^{\infty} h(k)\phi(2^{-j+1}t - k), \quad (9.19)$$

where  $h(k) = \langle \phi(2^{-j}t), \phi(2^{-j+1}t - k) \rangle$ , i.e., the expansion coefficient at time shift  $k$ . By taking the FT of Eq. (9.19), we obtain

$$\Phi(2^j\omega) = \frac{1}{2^{1/2}} H(2^{j-1}\omega) \Phi(2^{j-1}\omega), \quad (9.20)$$

which when iterated through scales leads to

$$\Phi(\omega) = \prod_{p=-\infty}^{\infty} \frac{h(2^{-p}\omega)}{\sqrt{2}} \Phi(0). \quad (9.21)$$

The complementarity of scaling and detail subspaces noted earlier stipulates that any function in subspace  $W_j$  may also be expressed in terms of  $V_{j-1} = \text{Span}\{\phi_{j-1,k}(t)\}_{(j,k) \in \mathcal{Z}^2}$ , or

$$\frac{1}{2^{j/2}}\psi(2^{-j}t) = \sum_{i=-\infty}^{\infty} \frac{1}{2^{(j-1)/2}} g(i)\phi(2^{-j+1}t - i). \quad (9.22)$$

In the Fourier domain, this leads to

$$\Psi(2^j\omega) = \frac{1}{\sqrt{2}} G(2^{j-1}\omega) \Phi(2^{j-1}\omega), \quad (9.23)$$

whose iteration also leads to an expression of the wavelet FT in terms of the transfer function  $G(\omega)$ , as previously given for  $\Phi(\omega)$  in Eq. (9.21). In light of these equations it is clear that the filters  $\{G(\omega), H(\omega)\}$  may be used to compute functions at successive scales. This in turn implies that the coefficients of any function  $x(t)$  may be similarly obtained as they are merely the result of an inner product of the signal of interest  $x(t)$  with a basis function,

$$\begin{aligned} \langle x(t), \psi_{ij}(t) \rangle &= \sum \langle x(t), g(k) \frac{1}{2^{(j-1)/2}} \phi(2^{-j+1}(t - 2^{-j}i) - k) \rangle \\ &= m_k g(k) \mathcal{A}_{i-k}^{j+1}(x). \end{aligned} \quad (9.24)$$

To complete the construction of the filter pair  $\{H(\cdot), G(\cdot)\}$ , the properties between the approximation subspaces ( $\{V_j\}_{j \in \mathcal{Z}}$ ) and detail subspaces ( $\{W_j\}_{j \in \mathcal{Z}}$ ) are exploited to derive the design criteria for

the discrete filters. Specifically, the same scale orthogonality property between the scaling and detail subspaces in

$$\sum_{k \in \mathcal{Z}} \Phi(2^j \omega + 2k\pi) \Psi(2^j \omega + 2k\pi) = 0, \quad \forall j, \quad (9.25)$$

is the first property that the resulting filters should satisfy. For the sake of illustration, fix  $j = 1$  and use

$$\sqrt{2}\Phi(2\omega) = H(\omega)\Phi(\omega). \quad (9.26)$$

Using Eq. (9.26) together with the orthonormality property of  $j$ th scale basis functions  $\{\Phi_{jk}(t)\}$  [18], i.e.,  $\sum |\Phi(\omega + 2k\pi)|^2 = 1$ , where we separate even and odd terms, yields the first property of one of the so-called conjugate mirror filters,

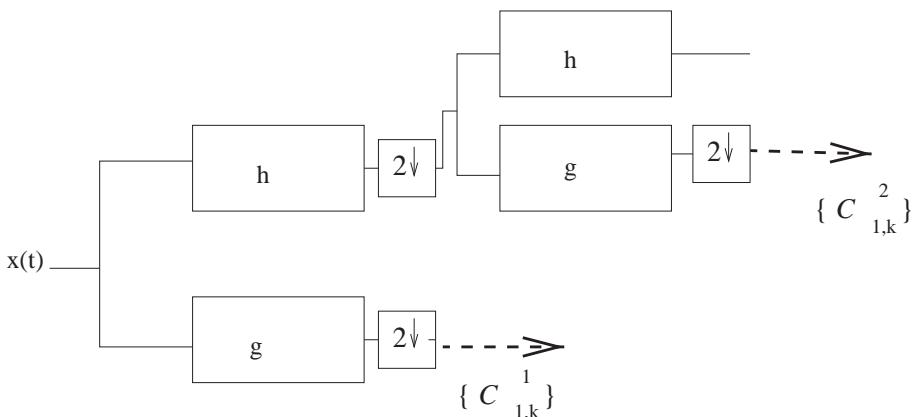
$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 1. \quad (9.27)$$

Using the non-overlapping property expressed in Eq. (9.25), together with the evaluations of  $\Psi(2\omega)$  and  $\Phi(2\omega)$ , and making a similar argument as in the preceding equation, yield the second property of conjugate mirror filters,

$$H(\omega)G^*(\omega) + H(\omega + \pi)G^*(\omega + \pi) = 0. \quad (9.28)$$

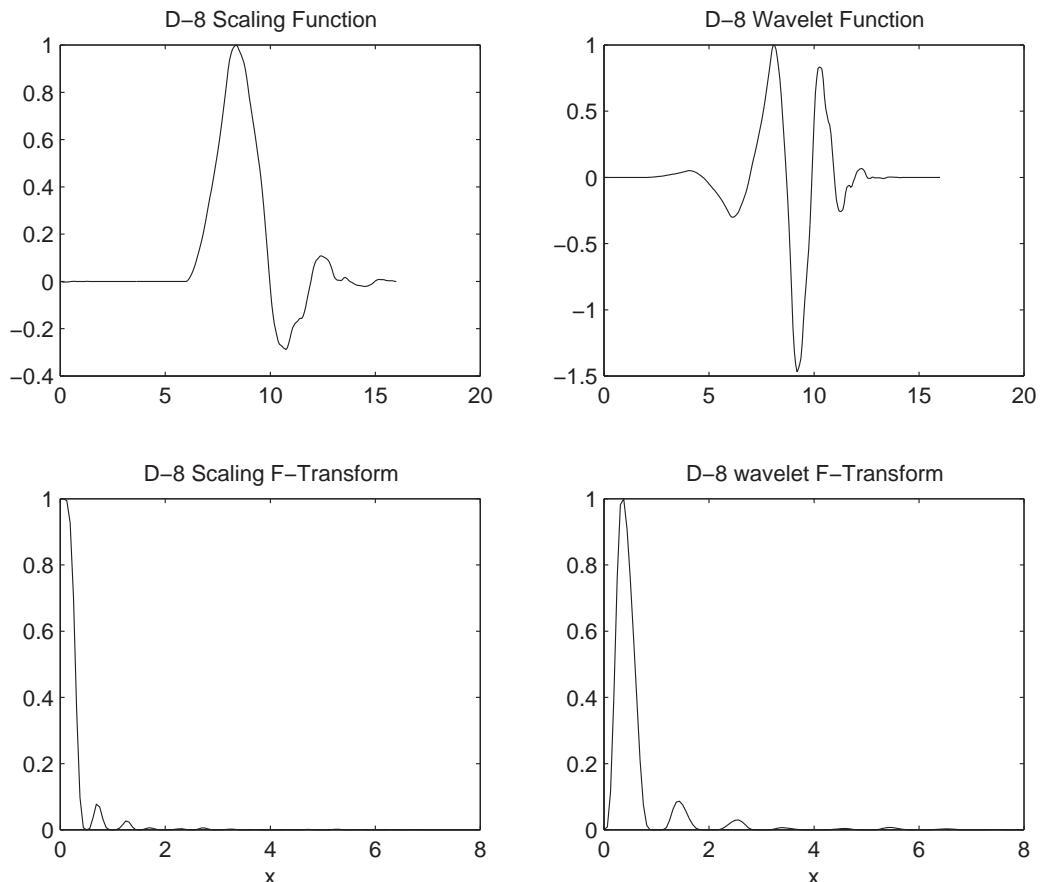
The combined structure of the two filters is referred to as “a conjugate mirror filter bank,” and their respective impulse responses completely specify the corresponding wavelets (see numerous references, e.g., [20] for additional technical details). Note that the literature in the MR studies tends to follow one of two possible strategies:

- A more functional analytic approach, which is mostly followed by applied mathematicians/mathematicians and scientists [18, 19, 30] (we could not possibly do justice to the numerous good texts now available, the author cites what he is most familiar with).
- A more filtering oriented approach widely popular among engineers and some applied mathematicians [21, 38, 41], see Figures 9.11 and 9.12.



**FIGURE 9.11**

Filter bank implementation of a wavelet decomposition.

**FIGURE 9.12**

A Daubechies-8 function and its spectral properties.

#### 1.09.2.3.7 Refining a wavelet basis: wavelet packet and local cosine bases

A selected analysis wavelet is not necessarily well adapted to any observed signal. This is particularly the case when the signal is time varying and has a rich spectral structure.

One approach is to then partition the signal into homogeneous spectral segments and by the same token find it an adapted basis. This is tantamount to further refining the wavelet basis, and resulting in what is referred to as a wavelet packet basis. A similar adapted partitioning may be carried out in the time domain by way of an orthogonal local trigonometric basis (sine or cosine). The two formulations are very similar, and the solution to the search for an adapted basis in both cases is resolved in precisely the same way. In the interest of space, we focus in the following development on only wavelet packets.

### 1.09.2.3.8 Wavelet packets

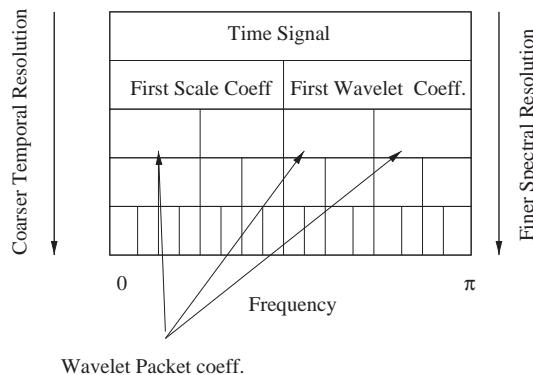
We maintained above that a selection of an analysis wavelet function should be carried out in function of the signal of interest. This of course assumes that we have some prior knowledge about the signal at hand. While plausible in some cases, this assumption is very unlikely in most practical cases. Yet it is highly desirable to select a basis which might still lead to an adapted representation of an apriority “unknown” signal. Coifman and Meyer [42] proposed to refine the standard wavelet decomposition. Intuitively, they proposed to further partition the spectral region of the details (i.e., wavelet coefficients) in addition to partitioning the coarse/scaling portion as ordinarily performed for a wavelet representation. This as shown in Figure 9.13 yields an overcomplete representation of a signal, in other words a dictionary of bases with a tree structure. The binary nature of the tree as further discussed below, affords a very efficient search for a basis which is best adapted to a given signal in the set.

Formally, we accomplish a partition refinement of say a subspace  $U_j$  by way of a new wavelet basis which includes both its approximation and its details, as shown in Figure 9.13. This construction due to Coifman and Meyer [42] may be simply understood as one’s ability to find a basis for all the subspaces  $\{V_j\}_{j \in \mathbb{Z}}$  and  $\{W_j\}_{j \in \mathbb{Z}}$  by iterating the nestedness property. This then amounts to expressing two new functions  $\tilde{\psi}_j^0(t)$  and  $\tilde{\psi}_j^1(t)$  in terms of  $\{\tilde{\psi}_{j-1,k}\}_{(j,k) \in \mathbb{Z}^2}$ , an orthonormal basis of a generic subspace  $U_{j-1}$  (which in our case may be either  $V_{j-1}$  or  $W_{j-1}$ ),

$$\tilde{\psi}_j^0(t) = \sum_{k=-\infty}^{\infty} h(k) \tilde{\psi}_{j-1}(t - 2^{j-1}k), \quad (9.29)$$

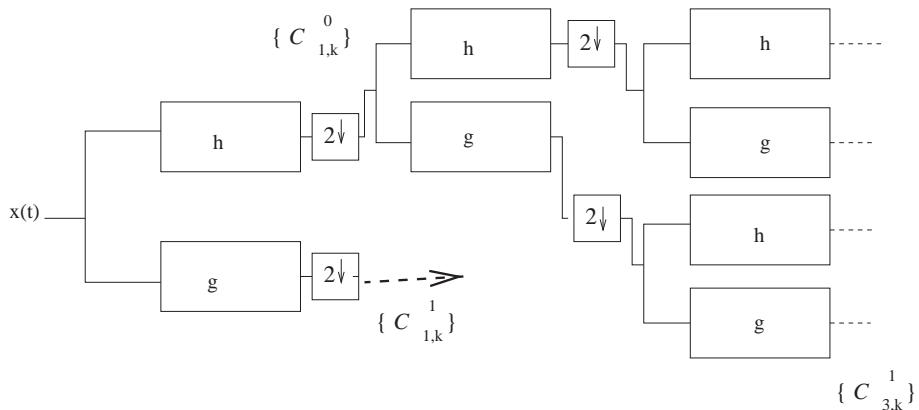
$$\tilde{\psi}_j^1(t) = \sum_{k=-\infty}^{\infty} g(k) \tilde{\psi}_{j-1}(t - 2^{j-1}k), \quad (9.30)$$

where  $h(\cdot)$  and  $g(\cdot)$  are the impulse responses of the filters in a corresponding filter bank, and the combined family  $\{\tilde{\psi}_j^0(t - 2^j k), \tilde{\psi}_j^1(t - 2^j k)\}_{(j,k) \in \mathbb{Z}^2}$  is an orthonormal basis of  $U_j$ . This, as illustrated in Figure 9.13, is graphically represented by a binary tree where, at each of the nodes reside the



**FIGURE 9.13**

Wavelet packet bases structure.

**FIGURE 9.14**

Wavelet packet filter bank realization.

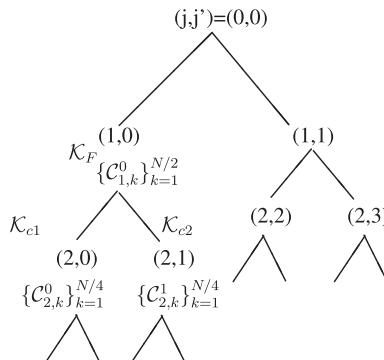
corresponding wavelet packet coefficients. The implementation of a wavelet packet decomposition is a straightforward extension of that of a wavelet decomposition, and consists of an iteration of both  $H(\cdot)$  and  $G(\cdot)$  bands (see Figure 9.14) to naturally lead to an overcomplete representation. This is reflected on the tree by the fact that, at the exception of the root and bottom nodes which only have respectively two children nodes and one ancestor node, each node bears two children nodes and one ancestor node.

#### 1.09.2.3.9 Basis search

To identify the best adapted basis in an overcomplete signal representation, as just noted above, we first construct a criterion which when optimized, will reflect desired properties intrinsic to the signal being analyzed. The earliest proposed criterion applied to a wavelet packet basis search is the so-called entropy criterion [43]. Unlike Shannon's information theoretic criterion, this is additive and makes use of the coefficients residing at each node of the tree in lieu of computed probabilities. The presence of more complex features in a signal necessitates such an adapted basis to ultimately achieve an ideally more parsimonious/succinct representation. As pointed out earlier, when searching for a wavelet packet or local cosine best basis, we typically have a dictionary  $\mathcal{D}$  of possible bases with a binary tree structure. Each node  $(j, j')$  (where  $j \in \{0, \dots, J\}$  represents the depth and  $j' \in \{0, \dots, 2^j - 1\}$  represents the branches on the  $j$ th level) of the tree then corresponds to a given orthonormal basis  $\mathcal{B}_{j,j'}$  of a vector subspace of  $\ell^2(\{1, \dots, N\})$  ( $\ell^2(\{1, \dots, N\})$  is a Hilbert space of finite energy sequences). Since a particular partition  $p \in \mathcal{P}$  of  $[0, 1]$  is composed of intervals  $I_{j,j'} = [2^{-j}j', 2^{-j}(j'+1)[$ , an orthonormal basis of  $\ell^2(\{1, \dots, N\})$  is given by  $\mathcal{B}^p = \bigcup_{\{(j,j')|I_{j,j'} \in \mathcal{P}\}} \mathcal{B}_{j,j'}$ . By taking advantage of the property

$$\text{Span}\{\mathcal{B}_{j,j'}\} = \text{Span}\{\mathcal{B}_{j+1,2j'}\} \overset{\perp}{\oplus} \text{Span}\{\mathcal{B}_{j+1,2j'+1}\}, \quad (9.31)$$

where  $\oplus$  denotes a subspace direct sum, we associate to each node a cost  $\mathcal{C}(\cdot)$ . We can then perform a bottom-up comparison of children versus parent costs (this in effect will eliminate the redundant/inadequate leaves of the tree) and ultimately prune the tree.

**FIGURE 9.15**

Tree pruning in search for a best-basis.

Our goal is to then choose the basis which leads to the *best* approximation of  $\{x[t]\}$  among a collection of orthonormal bases  $\{\mathcal{B}^p = \{\Psi x_i p\}_{1 \leq i \leq N} | p \in \mathcal{P}\}$ , where the subscript  $x_i$  emphasizes that it is adapted to  $\{x[t]\}$ . Trees of wavelet packet bases studied by Coifman and Wickerhauser [44] are constructed by quadrature mirror filter banks and comprise functions that are well-localized in time and frequency. This family of orthonormal bases partitions the frequency axis into intervals of different sizes, with each set corresponding to a specific wavelet packet basis. Another family of orthonormal bases studied by Malvar [45], and Coifman and Meyer [42], can be constructed with a tree of windowed cosine functions, and correspond to a division of the time axis into intervals of dyadically varying sizes.

For a discrete signal of size  $N$  (e.g., the size of the WP tableau shown in Figure 9.13), one can show that a tree of wavelet packet bases or local cosine bases has  $P = N(1 + \log_2 N)$  distinct vectors but includes more than  $2^{N/2}$  different orthogonal bases. One can also show that the signal expansion in these bases is computed with algorithms that require  $O(N \log_2 N)$  operations. Coifman and Wickerhauser [43] proposed that for any signal  $\{x[m]\}$  and an appropriate functional  $\mathcal{K}(\cdot)$ , one finds the best basis  $\mathcal{B}^{p_0}$  by minimizing an “additive” cost function

$$\text{Cost}(\mathbf{x}, \mathcal{B}^p) = \sum_{i=1}^N \mathcal{K}(|\langle \mathbf{x}, \Psi x_i p \rangle|^2) \quad (9.32)$$

over all bases. As a result, the basis which results from minimizing this cost function corresponds to the “best” representation of the observed signal. The resulting pruned tree of Figure 9.15 bears the coefficients at the remaining leaves/nodes.

#### 1.09.2.4 MR applications in estimation problems

The computational efficiency of a wavelet decomposition together with all of its properties have triggered unprecedented interest in their application in the area of information sciences (see e.g., [46–51]). Specific applications have ranged from compression [52–55] to signal/image modeling [56–59], and from signal/image enhancement to communications [60–67]. The literature in statistical applications as a whole has seen an explosive growth in recent years, and in the interest of space, we will focus

our discussion on a somewhat broader perspective which may in essence, be usefully reinterpreted in a number of different instances.

#### 1.09.2.4.1 Signal estimation/denoising and modeling

Denoising may be heuristically interpreted as a quest for parsimony of a representation of a signal. Wavelets as described above, have a great capacity for energy compaction, particularly at or near singularities. Given a particular wavelet, its corresponding basis as noted above, is not universally optimal for all signals and particularly not for a noisy one; this difficulty may be lifted by adapting the representation to the signal in a “best” way possible and according to some criterion. The first of the two possible ways, is to pick an optimal basis in a wavelet packet dictionary and discard the negligible coefficients [68]. The second which focuses on reconstruction of a signal in noise, and which we discuss here, accounts for the underlying noise statistics in the multiscale domain to separate the “mostly” signal part from the “mostly” noise part [69–71]. We opt here to discuss a more general setting which assumes unknown noise statistics and where a signal reconstruction is still sought in some optimal way, as we elaborate below. This approach is particularly appealing in that it may be reduced to the setting in earlier developments.

#### 1.09.2.4.2 Problem statement

Consider an additive noise model

$$x(t) = s(t) + n(t), \quad (9.33)$$

where  $s(t)$  is an unknown but deterministic signal corrupted by a zero-mean noise process  $n(t)$ , and  $x(t)$  is the observed, i.e., noisy, signal. The objective is to recover the signal  $\{s(t)\}$  based on the observations  $\{x(t)\}$ .

The underlying signal is modeled with an orthonormal basis representation,

$$s(t) = \sum_i C_i^s \psi_i(t),$$

and similarly the noise is represented as

$$n(t) = \sum_i C_i^n \psi_i(t).$$

By linearity, the observed signal can also be represented in the same fashion, its coefficients given by

$$C_i^x = C_i^s + C_i^n.$$

A key assumption we make is that for certain values of  $i$ ,  $C_i^s = 0$ ; in other words, the corresponding observation coefficients  $C_i^x$  represent “pure noise,” rather than signal corrupted by noise. As shown by Krim and Pesquet [72], this is a reasonable assumption in view of the spectral and structural differences between the underlying signal  $s(t)$  and the noise  $n(t)$  across scales. Given this assumption, wavelet-based denoising consists of determining which wavelet coefficients represent primarily signal, and which mostly capture noise. The goal is to then localize and isolate the “mostly signal” coefficients. This may be achieved by defining an information measure as a function of the wavelet coefficients. It identifies the “useful” coefficients as those whose inclusion improves the data explanation. One such measure is Rissanen’s information-theoretic approach (or Minimum Description Length (MDL)) [73]. In other words, the MDL criterion is utilized for resolving the tradeoff between model complexity (each

retained coefficient increases the number of model parameters) and goodness-of-fit (each truncated coefficient decreases the fit between the received—i.e., noisy—signal and its reconstruction).

#### 1.09.2.4.3 The coding length criterion

Wavelet thresholding is essentially an order estimation problem, one of balancing model accuracy against overfitting, and one of capturing as much of the “signal” as possible, while leaving out as much of the “noise” as possible. One approach to this estimation problem is to account for any prior knowledge available on the signal of interest, which usually is of probabilistic nature. This leads to a Bayesian estimation approach as developed in [74, 75]. While generally more complex, it does provide a regularization capacity which is much needed in low Signal to Noise Ratio environments.

A parsimony-driven strategy which we expound on here, addresses the problem of modeling in general, and that of compression in particular. It provides in addition a fairly general and interesting framework where the signal is assumed deterministic and unknown, and results in some intuitively sensible and mathematically tractable techniques [68]. Specifically, it brings together Rissanen’s work on stochastic complexity and coding length [73, 76], and Huber’s work on minimax statistical robustness [77, 78].

Following Rissanen, we seek the data representation that results in the shortest encoding of both observations and complexity constraints. As a departure from the commonly assumed Gaussian likelihood, we rather assume that the noise distribution  $f$  of our observed sequence is a (possibly) scaled version of an unknown member of the family of  $\varepsilon$ -contaminated normal distributions,

$$\mathcal{P}_\varepsilon = \{(1 - \varepsilon)\Phi + \varepsilon G : G \in \mathcal{F}\},$$

where  $\Phi$  is the standard normal distribution,  $\mathcal{F}$  is the set of all suitably smooth distribution functions, and  $\varepsilon \in (0, 1)$  is the known fraction of contamination (this is no loss of generality, since  $\varepsilon$  may always be estimated if unknown). Note that this study straightforwardly reduces to the additive Gaussian noise case, by setting the mixture parameter  $\varepsilon = 0$ , and is in that sense more general.

For fixed model order, the expectation of the MDL criterion is the entropy, plus a penalty term which is independent of both the distribution and the functional form of the estimator. In accordance with the minimax principle, we seek the least favorable noise distribution and evaluate the MDL criterion for that distribution. In other words, we solve a minimax problem where the entropy is maximized over all distributions in  $\mathcal{P}_\varepsilon$ , and the description length is minimized over all estimators in  $\mathcal{S}$ . The saddle-point (provided its existence) yields a minimax robust version of MDL, which we call the Minimax Description Length (MMDL) criterion.

In [68], it is shown that the least favorable distribution in  $\mathcal{P}_\varepsilon$ , which also maximizes the entropy, is one which is Gaussian in the center and Laplacian (“double exponential”) in the tails, and switches from one to the other at a point whose value depends on the fraction of contamination  $\varepsilon$ .

**Proposition 1.** *The distribution  $f_H \in \mathcal{P}_\varepsilon$  that minimizes the negentropy is*

$$f_H(c) = \begin{cases} (1 - \varepsilon)\phi_\sigma(a)e^{\frac{1}{\sigma^2}(ac+a^2)} & c \leq -a, \\ (1 - \varepsilon)\phi_\sigma(c) & -a \leq c \leq a, \\ (1 - \varepsilon)\phi_\sigma(a)e^{\frac{1}{\sigma^2}(-ac+a^2)} & a \leq c, \end{cases} \quad (9.34)$$

where  $\phi_\sigma$  is the normal density with mean zero and variance  $\sigma^2$ , and  $a$  is related to  $\varepsilon$  by the equation

$$2 \left( \frac{\phi_\sigma(a)}{a/\sigma^2} - \Phi_\sigma(-a) \right) = \frac{\varepsilon}{1-\varepsilon}. \quad (9.35)$$

#### 1.09.2.4.4 Coding for worst case noise

Let the set of wavelet coefficients obtained from the observed signal be denoted by  $\mathcal{C}^N = \{C_1^x, C_2^x, \dots, C_N^x\}$  as a time series without regards to the scale, and where the superscript indicates the corresponding process. Let exactly  $K$  of these coefficients contain signal information, while the remainder only contain noise. If necessary, we re-index these coefficients so that

$$C_i^x = \begin{cases} C_i^s + C_i^n & i = 1, 2, \dots, K, \\ C_i^n & \text{otherwise.} \end{cases} \quad (9.36)$$

By assumption, the set of noise coefficients  $\{C_i^n\}$  is a sample of independent, identically distributed random variates drawn from Huber's distribution  $f_H$ . It follows, by Eq. (9.36), that the observed coefficients  $C_i^x$  obey the distribution  $f_H(c - C_i^s)$  when  $i = 1, 2, \dots, K$ , and  $f_H(c)$  otherwise. Thus, the likelihood function is given by

$$\ell(\mathcal{C}^N; K) = \prod_{i \leq K} f_H(C_i^x - C_i^s) \prod_{i > K} f_H(C_i^x).$$

Since  $f_H$  is symmetric and unimodal with a maximum at the origin, the above expression is maximized (with respect to the signal coefficient estimates  $\{\widehat{C}_i^s\}$ ) by setting

$$\widehat{C}_i^s = C_i^x$$

for  $i = 1, 2, \dots, K$ . It follows that the maximized likelihood (given  $K$ ) is

$$\ell^*(\mathcal{C}^N; K) = \prod_{i \leq K} f_H(0) \prod_{i > K} f_H(C_i^x).$$

Thus, the problem is reduced to choosing the optimal value of  $K$ , in the sense of minimizing the MDL criterion,

$$\mathcal{L}(\mathcal{C}^N; K) = -\log \ell^*(\mathcal{C}^N; K) + K \log N = -\sum_{i \leq K} \log f_H(0) - \sum_{i > K} \log f_H(C_i^x) + K \log N. \quad (9.37)$$

Neglecting terms independent of  $K$ , this is equivalent to minimizing

$$\tilde{\mathcal{L}}(\mathcal{C}^N; K) = \frac{1}{2\sigma^2} \sum_{i > K} \eta(C_i^x) + K \log N,$$

where

$$\eta(c) = \begin{cases} c^2 & \text{if } |c| < a, \\ a|c| - a^2 & \text{otherwise} \end{cases}$$

is proportional to the exponent in Huber's distribution  $f_H$ . This can simply be achieved by a thresholding scheme [68].

**Proposition 2.**

1. When  $\log N > \frac{a^2}{2\sigma^2}$ , the coefficient  $|C_i^x|$  is truncated if

$$|C_i^x| < \frac{a}{2} + \frac{\sigma^2}{a} \log N.$$

2. When  $\log N \leq \frac{a^2}{2\sigma^2}$ , the coefficient  $|C_i^x|$  is truncated if

$$|C_i^x| < \sigma \sqrt{2 \log N}.$$

**Remarks.** More ample details may be found in [68].

When  $\sigma^2 \rightarrow 0$ , the thresholding scheme reduces to Case 2, and  $C_i^x$  is never truncated; since this represents the no-noise case, it is reasonable that all coefficients should be retained in the reconstruction. On the other hand, for large  $\sigma^2$ , the thresholding scheme reduces to Case 1, which is more conservative. For  $\sigma^2 \rightarrow \infty$ , the signal-to-noise ratio becomes zero and the best one can do is to estimate the signal as identically zero.

Similarly, when  $a \rightarrow \infty$ , the noise distribution becomes purely Gaussian, and the thresholding scheme reduces to Case 2, as expected. The resulting threshold of this particular noise case coincides with the results of [69, 70] and is qualitatively similar to that derived in [71]. On the other hand, when  $a \rightarrow 0$ , the noise distribution becomes purely Laplacian, and the thresholding scheme reduces to Case 1.

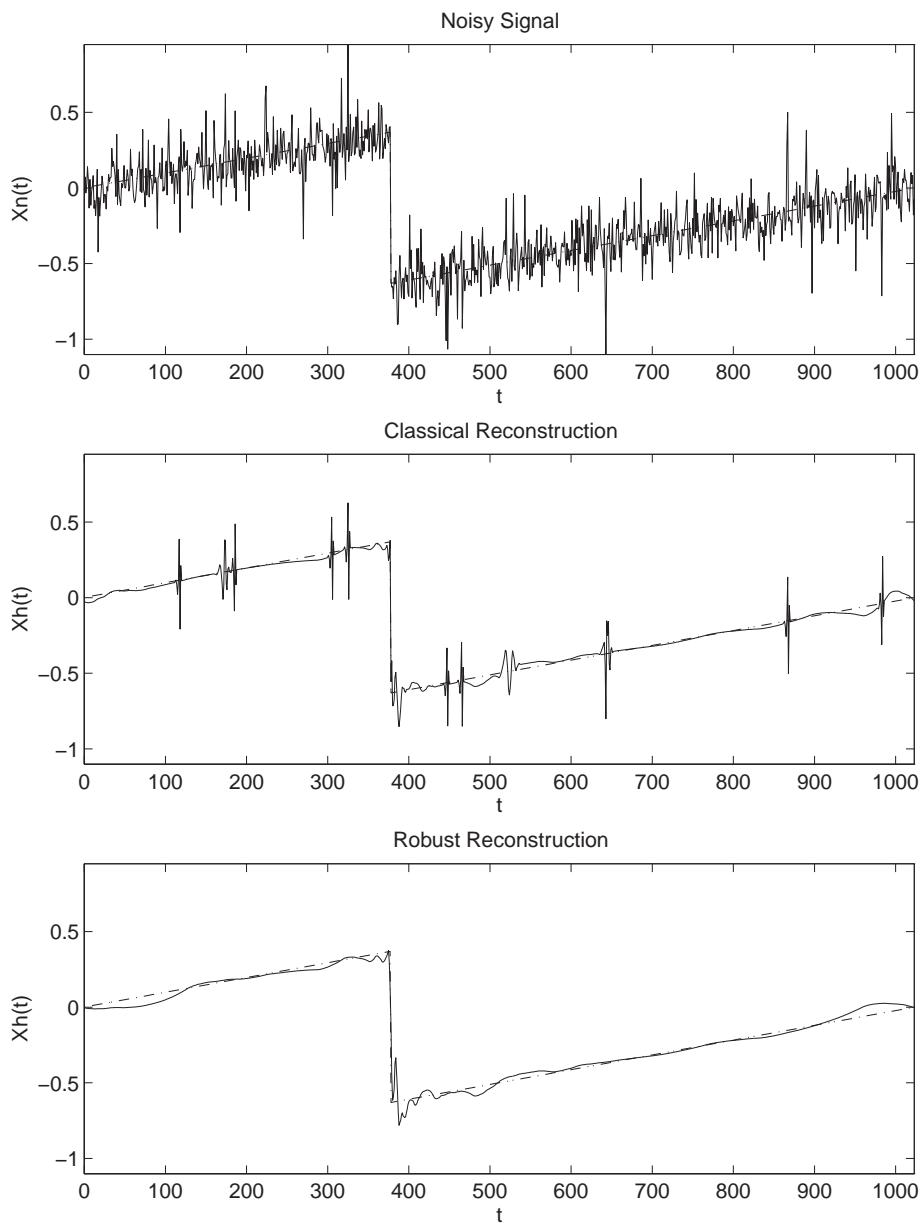
Finally, when  $N \rightarrow 1$ , the thresholding scheme reduces to Case 2, suggesting that outliers are unlikely to occur in a small sample, and it is hence more reasonable to assume purely Gaussian noise. On the other hand, for large  $N$ , the thresholding scheme reduces to Case 1, since outliers are highly likely to occur in a large sample.

It is important to distinguish the minimax error result obtained in [69] which was achieved over a signal smoothness class, from those discussed here and derived in [68] which are obtained over a family of noise distributions.

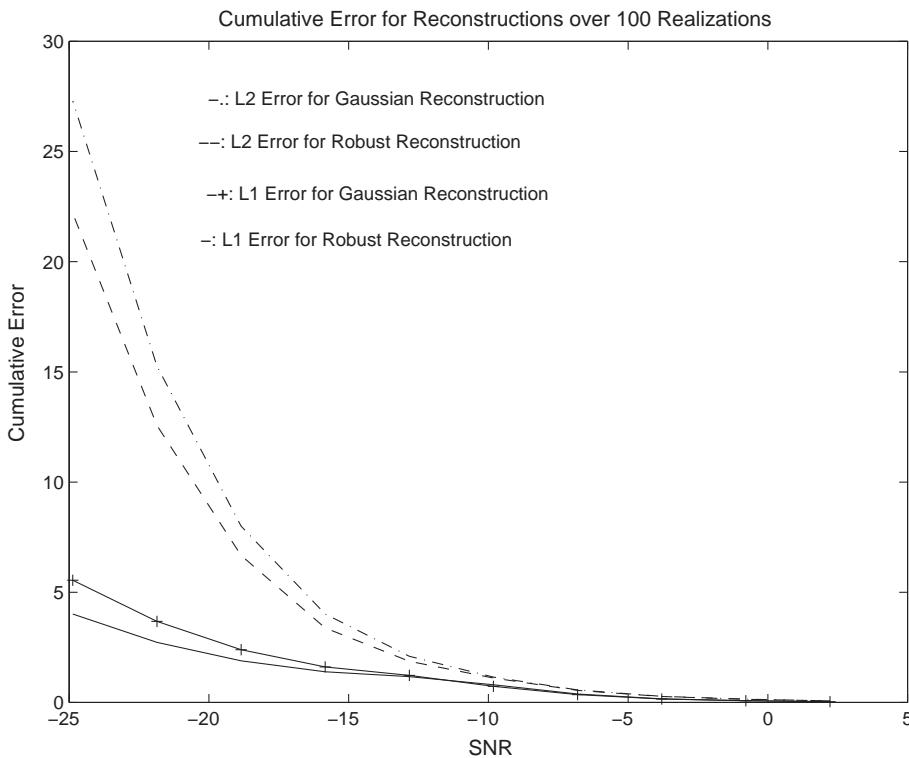
#### 1.09.2.4.5 Numerical experiments

In the examples that follow, we demonstrate the performance of the robust thresholding procedure described above, and compare it with that of the thresholding scheme based upon the assumption of normally distributed noise.

**Example I.** Using WAVELET (available from the Stanford Statistics Department, courtesy of D. L. Donoho and I. M. Johnstone), we synthesized a broken ramp signal of length  $N = 1024$ . This signal admits an efficient representation in a wavelet basis, i.e., one with very few non-zero coefficients. The noise was additive and i.i.d., obeying a  $N(0, \sigma^2)$  distribution contaminated by a fraction  $\varepsilon = 10\%$  of white Gaussian noise with distribution  $N(0, 9\sigma^2)$ . The overall Signal-to-Noise Ratio (SNR) was maintained at 10 dB (see Figure 9.16, top).

**FIGURE 9.16**

Noisy ramp signal, and its Gaussian and robust reconstructions.

**FIGURE 9.17**

$L_1$  and  $L_2$  error performance versus SNR, for the Gaussian and robust estimators.

We implemented two estimators, the first one of which is based on a purely Gaussian noise assumption (i.e.,  $\varepsilon = 0$ ), and where the thresholding scheme due to [72, 79] was used. The second was the MMDL robust estimator described above. The reconstructions based on each estimator appear in Figure 9.16. As may easily be observed, and in contrast to the MMDL technique, the Gaussian assumption induces a high susceptibility to outliers.

Monte-Carlo simulations were carried out to evaluate the reconstruction performance over a range of SNRs. At each value of the SNR, 100 experiments were conducted, and the cumulative reconstruction error is displayed in Figure 9.17. The robust estimator uniformly outperforms the *classic* estimator in both  $L_1$  and  $L_2$  errors over a wide range of SNRs. Furthermore, the performances of the Gaussian and robust estimators become indistinguishable at high SNRs, i.e., small noise variance, showing that robustness does not come at the cost of reduced efficiency.

#### Bounding the reconstruction error.

Although the robust estimator-based reconstruction error is much improved it is still potentially unbounded. As further discussed in [68], and due to the compactness of wavelets, unbounded noise will still result in unbounded reconstruction error, a property that may be considered undesirable.

This problem may be circumvented by making the assumption that the signal has bounded energy; in that case, one of at least two alternatives is possible:

1. In practice, the signal is known to be bounded, and prior knowledge of the physical properties of the signal may be used to determine the  $\|\cdot\|_\infty$  of the sequence of signal wavelet coefficients  $\{C_i^s\}$ . This information may be used to truncate observed coefficients  $\{C_i^x\}$  not only below, as discussed earlier, but also above.
2. In the absence of such prior knowledge, it may still be possible to bound the reconstruction error through an adaptive supremum–secondary thresholding scheme based upon some representation criterion, e.g., entropy.

The first of these approaches uses the following modified thresholding: let  $\alpha > 0$  be an upper bound on the magnitude of the signal coefficients; then,

$$\tilde{C}_i^s = \begin{cases} 0 & \text{if } |C_i^x| \leq \frac{a}{2} + \frac{\sigma^2}{a} \log N, \\ \hat{C}_i^s = C_i^x & \text{if } \frac{a}{2} + \frac{\sigma^2}{a} \log K \leq |C_i^x| \leq \alpha, \\ \alpha \operatorname{sgn}(C_i^x) & \text{if } \alpha \leq |C_i^x| \end{cases}$$

provided that  $\log N > a^2/2\sigma^2$  and  $\alpha > (a/2) + (\sigma^2/a) \log N$ . The graph shows that although the robust estimator's reconstruction error initially grows more slowly than that of the Gaussian estimator, the two errors soon converge as the variance of the outliers grows. The reconstruction error for the bounded-error estimator, however, levels off past a certain magnitude of the outlier, as expected.

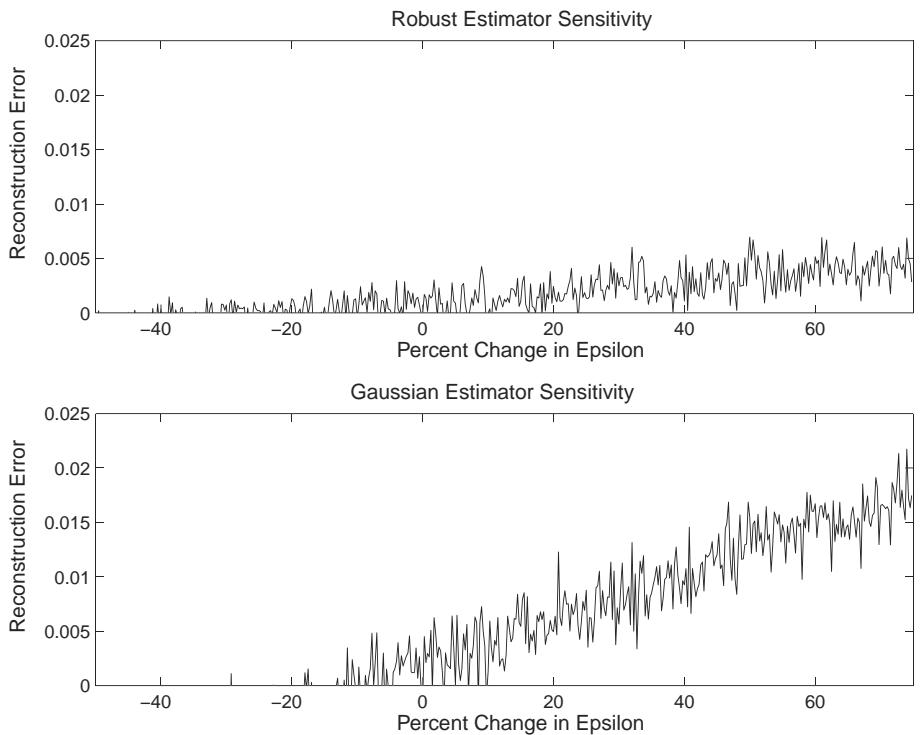
*Sensitivity analysis for the fraction of contamination.*

Although such crucial assumptions as the normality of the noise or exact knowledge of its variance  $\sigma^2$  usually go unremarked, it is often thought that Huber-like approaches are limited on account of the assumption of known  $\varepsilon$ . We demonstrate the resilience and robustness of the approach by studying the sensitivity of the estimator to changes in the assumed value of  $\varepsilon$ .

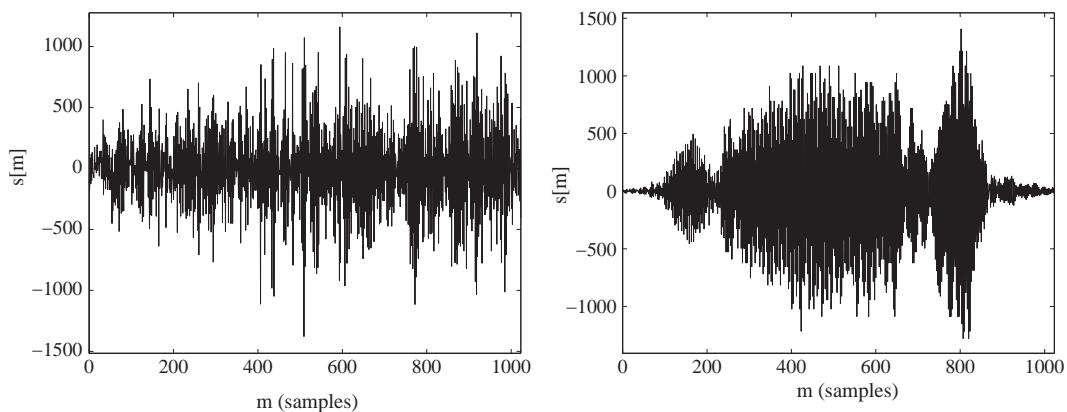
Figure 9.18 shows the total reconstruction error as a function of variation in the true fraction of contamination  $\varepsilon$ . In other words, an abscissa of “0” corresponds to an assumed fraction of contamination equal to the true fraction; larger abscissas correspond to outliers of larger magnitude than assumed by the robust estimator, and vice versa. Clearly, the Gaussian estimator assumes zero contamination throughout. The figure shows that the reconstruction error for the Gaussian estimator grows very rapidly as the true fraction of contamination increases, whereas that of the robust estimator is nearly flat over a broad range. This should not come as a surprise: outliers are, by definition, rare events, and for a localized procedure such as wavelet expansion, the precise frequency of outliers is much less important than their existence at all.

**Example II.** The example above assumed a fixed wavelet basis. As discussed above, however, highly non-stationary signals can be represented most efficiently by using an adaptive basis that automatically chooses resolutions as the behavior of the signal varies over time. Using an  $L^2$  error criterion we search for the best basis of a chirp and show the reconstruction in Figure 9.19 (see [80] for more details).

The separability property of most carefully designed wavelets, and their inability to achieve fine tuned directional analysis has led to seeking alternative wavelets which are more sensitive to arbitrary curves

**FIGURE 9.18**

Error stability vs. variation in the mixture parameter  $\varepsilon$ .

**FIGURE 9.19**

A noisy chirp reconstructed from its best basis and denoised as shown in the figure on the right.

and image edges. These include the introduction of curvelets, contourlets (an efficient implementation of curvelets) and to shearlets, which are respectively discussed in more detail, in the next three sections.

### 1.09.3 Curvelets and their applications

In this section, we introduce the 2D curvelet transform developed by Candès and Donoho around 2000 [4]. Initially, they proposed the ridgelet transform in Cartesian Fourier plane [81] in order to locate segmented curve features in an image. Soon, they discovered that multi-scale partition of the Fourier plane using polar coordinate parameters can provide much simpler structures. This led to their development of the current continuous curvelet transform. This is closely related to Smith's parabolic scaling based transform using the Fourier Integral Operators [82]. The convenience in using polar coordinates in defining the continuous curvelet transform indeed has a cost in implementing it in the digital world. Candès, Donoho, and their collaborators have also introduced the discretized curvelet transform, in which, the discrete Fourier transform involving polar coordinates has been resampled using Cartesian coordinates [83].

It is worth to point out that the varying numbers of wedge-shaped atoms are used across scales when the anisotropic curvelets are used to partition the Fourier plane. This is very different from the classical wavelets, in which only a fixed number of isotropic squares are used. In 2-D spatial plane, the curvelet atoms effective impact is similar to applying differential operators to an image along selected directions to measure the regularities in localized areas that can be contained inside rotated rectangles. Large coefficients can be obtained when a differential operator is applied across the discontinuities along a curve, and thus can be used to characterize salient features in an image.

To start, we first review the conversion between polar coordinates and Cartesian coordinates. In the Fourier frequency plane, a point  $\Omega = (\Omega_x, \Omega_y)$  with Cartesian coordinates, can be converted to polar coordinates  $(r, \sigma)$  by

$$r = \pm|\Omega| = \pm\sqrt{\Omega_x^2 + \Omega_y^2}, \quad \sigma = \arctan(\Omega_y / \Omega_x),$$

assume we confine the angle  $\sigma$  to  $(-\pi/2, \pi/2)$  (the angle between the polar axis and the ray from the pole to the point). Meanwhile, the polar coordinates  $(r, \sigma)$  can be converted to the Cartesian coordinates  $(\Omega_x, \Omega_y)$ , see Figure 9.20, by

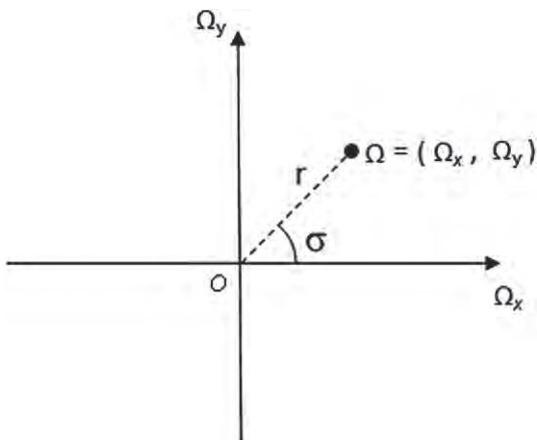
$$\Omega_x = r \cos \sigma, \quad \Omega_y = r \sin \sigma. \quad (9.38)$$

Let  $R_\theta$  be the operator that rotates a point  $\Omega$  in the counter clockwise (ccw) direction around the origin by an angle of  $\theta$  radians, the end point in polar coordinates will be  $R_\theta(\Omega) = (r, \sigma + \theta)$ . In Cartesian coordinates, the operator can be written as a matrix

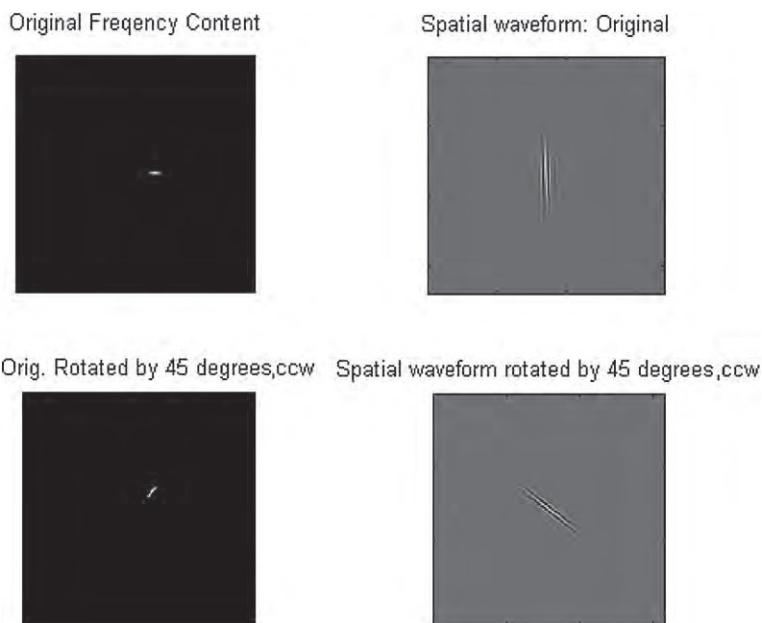
$$R_\theta = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix},$$

and the end point of the rotation,  $R_\theta(\Omega) = R_\theta \cdot \Omega$ , is the multiplication of the rotating matrix and the vector  $\Omega$  ( $\Omega$  can be viewed as a column vector).

The following proposition states that, when the input of a function is rotated by  $\theta$  radians in the ccw direction, its Fourier transform is also rotated by  $\theta$  radians in the Fourier plane, see Figure 9.21.

**FIGURE 9.20**

The polar and rectangle coordinates.

**FIGURE 9.21**

First row, left, the selected Fourier content of an image, right, its corresponding spatial waveform. Second row, left, the Fourier content is rotate ccw by 45°, right, its corresponding spatial waveform is also rotated ccw by 45°.

**Proposition 3.** Let the Fourier transform of  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$  be  $F(\Omega)$ , we have

$$F(R_\theta(\Omega)) = \int_{\mathcal{R}^2} f(R_\theta(\mathbf{x})) d\mathbf{x}. \quad (9.39)$$

### 1.09.3.1 The continuous curvelet transform

The continuous curvelet transform is defined as the inner product of a function with curvelet atoms at various dilating scales, rotating angles, and translating locations. It calculates the energy of the function correlated to each curvelet atom so that the frequency content of the function can be examined separately in each localized subband. At each scale, the curvelet atoms are generated from rotating a smooth 2-D window with compact support in its Frequency domain, and shifting its waveform in spatial domain to various locations. This window is called the mother curvelet at scale  $a$ , see Figure 9.22.

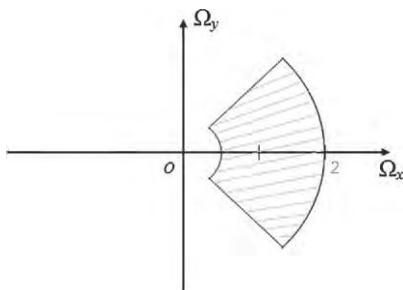
#### 1.09.3.1.1 The mother curvelets

At scale  $a$ , the Fourier transform of a mother curvelet is written as  $\Psi_a(\Omega_x, \Omega_y)$  in Cartesian coordinates, or  $\Psi_a(r, \sigma)$  in the polar coordinates. It is defined using two 1-D windows, along the radial and angular directions, and is given by

$$\Psi_a(r, \sigma) = a^{3/4} \Psi_1(ar) \Psi_2\left(\frac{\sigma}{\sqrt{a}}\right), \quad 0 < a \leq 1, \quad (9.40)$$

where the radial (or amplitude) window,  $\Psi_1(r)$ , is a smooth, bandpass, and non-negative real-valued function that has its support  $r \in [1/2, 2]$ . This interval is chosen for the convenience of using dyadic intervals for discretization. The angular (or phase) window,  $\Psi_2(\sigma)$ , is a smooth, lowpass, and non-negative real-valued even function that has its support  $\sigma \in [-1, 1]$ .

At scale  $a$ , the mother curvelet  $\Psi_a(r, \sigma)$  is constructed by scaling the two window functions,  $\Psi_1$  and  $\Psi_2$ , in two different ways. The support of this mother curvelet is a single polar wedge with an area of  $Const. \cdot a^{-3/2}$ , and is located inside  $\{(r, \sigma) : \frac{1}{2a} < r < \frac{2}{a}, |\sigma| < \sqrt{a}\}$ . It is symmetric with respect to the polar axis. The length of the wedge in the polar axis direction is at the same order as  $a^{-1}$ , and the larger arc length in the angular direction is at the same order as  $a^{-1/2}$ , see Figure 9.22. Approximately,



**FIGURE 9.22**

A mother curvelet window  $\Psi_a(\Omega)$  at scale  $a = 1$ .

the wedge has  $\text{width} \approx \text{length}^2$ . It can be observed that the mother curvelets at two different scales  $a, a'$ , with  $a \neq a'$ , have no scaling relation between them, so they differ slightly at various scales.

#### 1.09.3.1.2 The amplitude and angular windows

The choice of the amplitude and angular windows not only decides the properties of the mother curvelets, but also decides the properties of the curvelet atoms. The above construction shows that the curvelet windows have compact supports in the Fourier plane because both  $\Psi_1$  and  $\Psi_2$  have compact supports. Further, the following Proposition 4 [84] implies that their corresponding spatial waveforms  $\psi_1(x), \psi_2(t) \in C^\infty$ . Hence the mother curvelets are  $C^\infty$  in the spatial plane.

**Proposition 4.** *A function  $\psi(x)$  is bounded and  $n$  times continuously differentiable with bounded derivatives if its Fourier transform  $\Psi(\Omega_x)$  satisfies*

$$\int_{-\infty}^{+\infty} |\Psi(\Omega_x)|(1 + |\Omega_x|^n) d\Omega_x < +\infty. \quad (9.41)$$

The compact support of a mother curvelet in the Fourier plane indicates that the spatial waveform of the mother curvelet does not have a compact support in the spatial plane. Its waveform spreads out and is known as the side robe effects. To allow the curvelets to focus on localized areas in the spatial domain for detecting the singularities, the windows  $\Psi_1$  and  $\Psi_2$  should be chosen as smooth as possible in the Fourier plane so that their spatial waveforms decay rapidly. The rapid decay of a function is defined as, there exists a constant  $M_n$  such that, the following is true for all integer  $n > 0$  for  $f(\mathbf{x})$ ,

$$|f(\mathbf{x})| \leq \frac{M_n}{(1 + |\mathbf{x}|^n)}, \quad \forall \mathbf{x}. \quad (9.42)$$

Proposition 4 is also true for the inverse Fourier transform. If the spatial waveform of a mother curvelet decays fast, then it indicates that Fourier windows  $\Psi_1(r)$  and  $\Psi_2(\sigma)$  should be  $C^n$ . Therefore, the windows  $\Psi_1$  and  $\Psi_2$  should be smooth so they be  $n$  times continuously differentiable. In the literature, the windows are often required to be  $C^\infty$  to ensure the spatial functions decay rapidly.

At scale  $a$ , the mother waveform oscillates along the horizontal direction with most of its support located inside an ellipsoid shaped area along a ridge in the vertical direction. The ridge is defined as the (line) locations where the amplitude of a function reaches its maximum amplitude. This ellipsoid shape becomes “needle-like” shape when the scale  $a$  gets smaller. Namely, the area becomes longer and narrower.

Finally, the amplitude and angular windows  $\Psi_1$  and  $\Psi_2$  should satisfy the following admissibility conditions:

$$\int_0^\infty \frac{\Psi_1^2(r)}{r} dr = C_{\Psi_1}, \quad (9.43)$$

$$\int_{-1}^1 \Psi_2^2(\sigma) d\sigma = C_{\Psi_2}. \quad (9.44)$$

The admissibility conditions in Eqs. (9.43) and (9.44) can be normalized such that  $C_{\Psi_1} = 1$ , and  $C_{\Psi_2} = 1$ . Equation (9.43) indicates that  $\Psi_1(0) = 0$ , namely,  $\Psi_1(r)$  is the transfer function of a bandpass filter.

A continuously differentiable Fourier transfer function,  $\Psi_1$ , that satisfies  $\Psi_1(0) = 0$ , is sufficient to ensure that  $\Psi_1$  satisfies the admissibility condition in Eq. (9.43) [84]. The selection of the windows for constructing the curvelets will be further discussed in detail in the discrete case.

### 1.09.3.1.3 The curvelet atoms

**Definition 8.** A curvelet atom is referred to an element in the set  $\{\psi_{a,\theta,b}(x), x \in \mathcal{R}^2, a \in (0, 1], \theta \in [0, 2\pi), b \in \mathcal{R}^2\}$ . It can be defined as the inverse Fourier transform of a function,  $\Psi_{a,\theta,b}(\Omega)$ , as following

$$\Psi_{a,\theta,b}(\Omega) = \exp(-j \langle b, \Omega \rangle) \Psi_a(R_\theta \Omega), \quad (9.45)$$

where  $\Psi_a(\Omega)$  is a mother curvelet. The polar coordinate format of  $\Psi_a(R_\theta \Omega)$  is equal to  $\Psi_a(r, \sigma + \theta)$ .

Specifically, at scale  $a$ , the curvelet atom at  $\theta = 0, b = (0, 0)$  (we usually use  $\mathbf{0} = (0, 0)$ ) is the mother curvelet because

$$\Psi_{a,0,\mathbf{0}}(\Omega) = \Psi_a(\Omega). \quad (9.46)$$

The spatial waveform of the curvelet atom,  $\psi_{a,\theta,b}(x)$ , takes a complex value, and can be written in terms of a mother curvelet as,

$$\psi_{a,\theta,b}(x) = \psi_{a,0,\mathbf{0}}(x)(R_\theta(x - b)), \quad a \in (0, 1], \theta \in [0, 2\pi), b \in \mathcal{R}^2. \quad (9.47)$$

Figure 9.21 shows two curvelet atoms (in both Fourier and spatial planes) at different orientations. Although complex valued curvelet atoms are capable of separating the amplitude and the phase information contained in a 2-D function, in many situations the real-valued curvelet atoms can be built by modifying the mother curvelet windows, namely, replacing the radial window  $\Psi_1(r)$  with  $\Psi_{R,1}(r) = \Psi_1(r) + \Psi_1(-r)$ , or replacing the angular window  $\Psi_2(\sigma)$  with  $\Psi_{R,2}(\sigma) = \Psi_2(\sigma) + \Psi_2(\sigma + \pi)$ . It results in mother curvelets supported on two polar wedges that are symmetric with respect to the  $x, y$  axes in the rectangle coordinates. Figure 9.23 shows the frequency support (the gray area) of a mother curvelet that generates complex-valued curvelet atoms in spatial domain, while Figure 9.24 shows the frequency support of a mother curvelet that generates real-valued curvelet atoms.

The curvelet atoms generated by the mother curvelets dissect the Fourier plane into localized subbands so that the Fourier plane is packed redundantly with all the dilated and rotated curvelet atoms. Note that shifting in the spatial plane has no effect on the amplitude of the content in the Fourier plane.

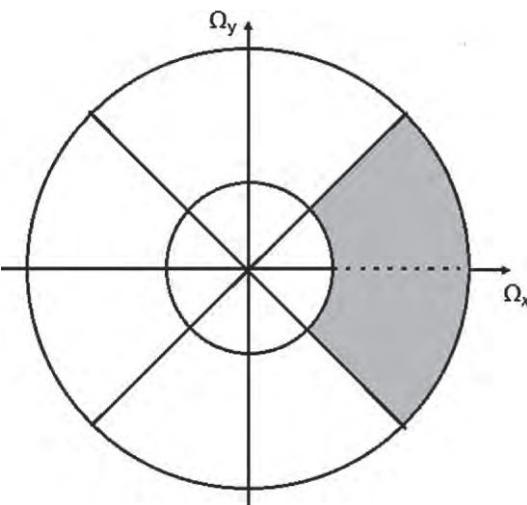
### 1.09.3.1.4 The definition of continuous curvelet transform

**Definition 9.** Let  $f(x) \in L^2(\mathcal{R}^2)$ , and let  $\{\psi_{a,\theta,b}(x)\}_{a \in (0, 1], \theta \in [0, 2\pi), b \in \mathcal{R}^2}$  be the families of curvelet atoms defined above. The continuous curvelet transform, see Figure 9.25, is defined as

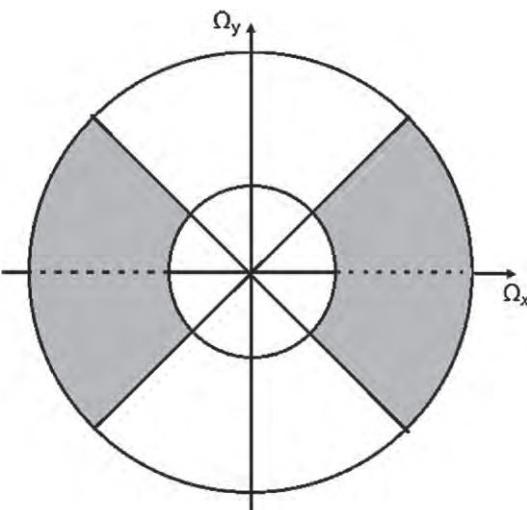
$$C_f(a, \theta, b) = \langle f, \psi_{a,\theta,b} \rangle = \int_{\mathcal{R}^2} f(x) \psi_{a,\theta,b}^*(x) dx, \quad (9.48)$$

where  $\{C_f(a, \theta, b), a \in (0, 1], \theta \in [0, 2\pi), b \in \mathcal{R}^2\}$  are called the curvelet coefficients of the function  $f(x)$ .

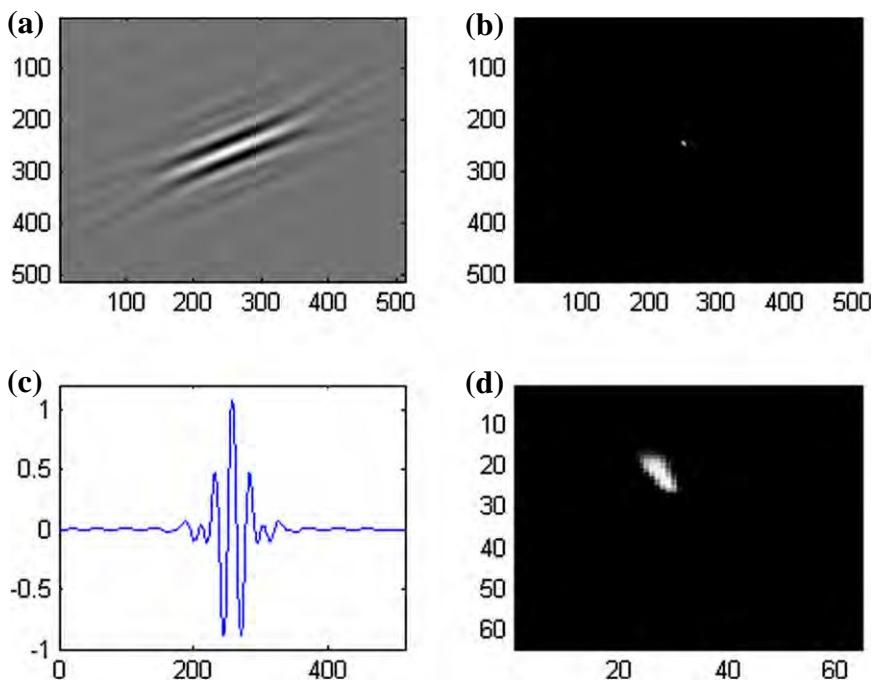
Note that here we mentioned the families of curvelet atoms. This is because each mother curvelet at scale  $a$  generates a family of curvelet atoms. The curvelet transform projects a 2-D function into the

**FIGURE 9.23**

Shown here is a mother curvelet that generates a family of complex valued curvelets. The mother curvelet has its frequency content supported in a single wedge, represented by the gray area.

**FIGURE 9.24**

Shown here is a mother curvelet that generates a family of real-valued curvelets. The mother curvelet has its frequency content supported in a pair of symmetric wedge, represented by the gray area.

**FIGURE 9.25**

(a) The real part of a wavelet atom in the spatial space. (b) The Fourier transform of the wavelet atom shown. (c) The graph of oscillatory function along  $\theta$  direction taken from the real part of the wavelet atom. In this picture, the horizontal direction is the pixel location along  $\theta$ , the vertical direction is the function's value. (d) The Fourier transform from (b) is zoomed to the center  $64 \times 64$  pixels for a better view of the content. The pixels size is shown in (a, b, d).

families of curvelet atoms to receive a series of decomposed coefficients. Each coefficient reflects the correlation between the function and a curvelet atom focusing inside a localized area. Typically, the curvelet coefficients are calculated in the Fourier plane using the following result, which is the Parseval's formula.

**Proposition 5.** *The curvelet transform can be calculated in the Fourier plane as,*

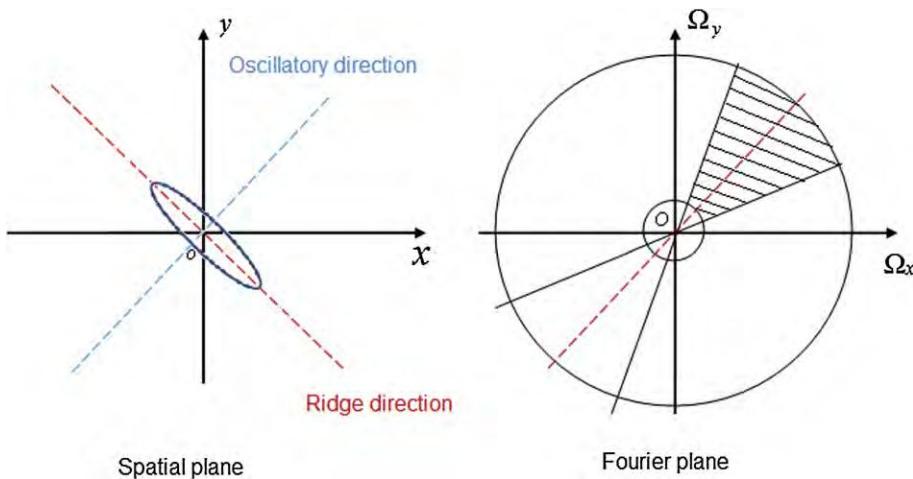
$$C_f(a, \theta, \mathbf{b}) = \langle F, \Psi_{a,\theta,\mathbf{b}} \rangle = \frac{1}{(2\pi)^2} \int_{\mathcal{R}^2} F(\Omega_x, \Omega_y) \Psi_{a,\theta,\mathbf{b}}^*(\Omega_x, \Omega_y) d\Omega_x d\Omega_y, \quad (9.49)$$

for  $a \in (0, 1]$ ,  $\theta \in [0, 2\pi)$ ,  $\mathbf{b} \in \mathcal{R}^2$ .

#### 1.09.3.1.5 The properties of the curvelet transform

The vanishing moments and the oscillatory behavior

In the classical wavelet theory, a wavelet with  $n$  vanishing moments has been interpreted as a multiscale differential operator of order  $n$ . A similar concept can be extended to the curvelets in parallel. A mother

**FIGURE 9.26**

A curvelet atom in its spatial plane and Fourier plane.

curvelet  $\psi_{a,0,\mathbf{0}}$  is said to have  $q$  vanishing moments if its waveform in spatial plane satisfies

$$\int_{-\infty}^{\infty} \psi_{a,0,\mathbf{0}}(x, y) x^n dx = 0, \quad \text{for all } 0 < n \leq q. \quad (9.50)$$

The curvelet windows we selected can have infinitely many vanishing moments as Eq. (9.50) is satisfied for any  $q$ .

The curvelet window that has infinitely many vanishing moments ensures that the coefficients of a smooth image function tend to zero when a mother curvelet is centered at  $\mathbf{x}$ . In the spatial plane, a rotated curvelet atom with oscillatory direction at  $\theta$  radians, if shifted to the spatial location  $\mathbf{b} \in \mathbb{R}^2$ , plays the role of an angular differential operator in a localized area centered at  $\mathbf{b}$ , to measure the regularity along the orientation, see Figure 9.26. This area is referred to as the ellipse support area. If an image function satisfies a localized Lipschitz regularity along the radial direction at location  $\mathbf{x}$ , then a polynomial can be used to approximate the image function in a localized area. The aforementioned infinitely many vanishing moments of the curvelet windows will therefore lead to many curvelet coefficients tending to zero.

### The decay rate of the curvelet coefficients

As we mentioned above, the requirement for infinitely many vanishing moments ensures that the curvelets decay rapidly in the spatial domain. A curvelet coefficient therefore measures the correlation between a function and a curvelet atom focusing in its ellipse support area over the spatial support. The resulting coefficients may be classified into the following three cases [7]:

1. The magnitude of a curvelet coefficient tends to zero if the ellipse support of the curvelet atom does not intersect with a singularity of the function.
2. The magnitude of a curvelet coefficient tends to zero if the ellipse support of the curvelet atom intersects with a singularity, but its ridge direction is not oriented at the tangential direction of a curve.

3. The magnitude of a curvelet coefficient decays slowly if the ellipse support of the curvelet atom intersects with a discontinuity, and it is ridge is oriented along the tangential direction of the curve where the singularities are located. More precisely, the curvelet oscillates along the normal direction of the curve to measure the regularities in a local area centered at  $\mathbf{b}$ .

#### 1.09.3.1.6 The reproducing formula

Similarly to the classical wavelets, the admissibility conditions, Eqs. (9.43) and (9.44), ensure that all the curvelet atoms cover the Fourier plane completely, so the curvelet transform is a redundant representation of a function that can be perfectly reconstructed.

**Theorem 1.** *If the Fourier Transform of a highpass function  $f \in L^2(\mathcal{R}^2)$  vanishes (equal to 0) for  $|\Omega| < 2/a_0$ , this function can be perfectly reconstructed from the curvelet coefficients according to the following Calderon-like reproducing formula:*

$$f(\mathbf{x}) = \int_0^{2\pi} \int_0^{a_0} \int_{\mathcal{R}^2} C_f(a, \theta, \mathbf{b}) \psi_{a, \theta, \mathbf{b}}(\mathbf{x}) \frac{d\mathbf{b}}{a^{1/2}} \frac{da}{a^{3/2}} \frac{d\theta}{a}. \quad (9.51)$$

The proof of this theorem is deferred to [7].

This reproducing formula can be extended to reconstruct any function  $f \in L^2(\mathcal{R}^2)$  that includes both low frequency and high frequency components. To do so, we can introduce a scaling function,  $\phi(\mathbf{x})$ , the so-called father curvelet, through its Fourier Transform,  $\Phi(\Omega)$ , given by,

$$\Phi^2(\Omega) = 1 - \Psi^2(\Omega), \quad \text{where } \Psi^2(\Omega) = \int_0^{a_0|\Omega|} |\Psi_1(r)|^2 \frac{dr}{r}. \quad (9.52)$$

The function  $\Phi(\Omega)$  can be viewed as an aggregation of curvelets at scales greater than  $a_0$ , written as

$$\Phi(\Omega) = \int_{a_0}^{+\infty} \frac{|\Psi_1(r|\Omega)|^2}{r} dr = \int_{a_0|\Omega|}^{+\infty} \frac{|\Psi_1(r)|^2}{r} dr \quad (9.53)$$

and may further be written as

$$\Phi(\Omega) = \begin{cases} 1 & |\Omega| \leq 1/(2a_0), \\ (1 - \Psi^2(\Omega))^{1/2} & 1/(2a_0) < |\Omega| < 2/a_0, \\ 0 & |\Omega| \geq 2/a_0. \end{cases}$$

Note that  $a_0$  is limited to  $0 < a_0 \leq 1$ , it is typically set to  $a_0 = 1$  as in the following.

The high frequency content in  $\Phi(\Omega)$  equaling zero indicates that the father curvelet is the impulse response of a lowpass frequency filter. Shifting the father curvelet  $\phi(\mathbf{x})$  builds the shifting atoms  $\phi_b(\mathbf{x}) = \phi(\mathbf{x} - \mathbf{b})$ ,  $\mathbf{b} \in \mathcal{R}^2$ . It can be seen that the father curvelet is isotropic and is not scaled. It only produces an atom that is shifted to spatial location  $\mathbf{b}$ .

**Theorem 2.** *Any  $f \in L^2(\mathcal{R}^2)$  can be reproduced as*

$$f(\mathbf{x}) = \int_{\mathcal{R}^2} \langle \phi_b, f \rangle \phi_b(\mathbf{x}) d\mathbf{b} + \int_0^1 \int_0^{2\pi} \int_{\mathcal{R}^2} C_f(a, \theta, \mathbf{b}) \psi_{a, \theta, \mathbf{b}}(\mathbf{x}) \frac{d\mathbf{b}}{a^{3/2}} \frac{da}{a} \frac{d\theta}{a^{1/2}} \quad (9.54)$$

and it satisfies the Plancherel's formula

$$\|f(\mathbf{x})\|_2^2 = \int_{\mathcal{R}^2} \langle \phi_b, f \rangle^2 d\mathbf{b} + \int_0^1 \int_0^{2\pi} \int_{\mathcal{R}^2} |C_f(a, \theta, \mathbf{b})|^2 \frac{d\mathbf{b}}{a^{3/2}} \frac{da}{a} \frac{d\theta}{a^{1/2}}. \quad (9.55)$$

The term  $\frac{d\mathbf{b}}{a^{3/2}} \frac{da}{a} \frac{d\theta}{a^{1/2}}$  may be viewed as the reference measure in the parameter space of  $(a, \theta, \mathbf{b})$ . It suggests that the curvelets use anisotropic measuring of the unit cell in the space of  $(a, \theta, \mathbf{b})$ . This theorem ensures that both the curvelets and the classical wavelets can be decomposed into a low frequency component and a combination of detail components. The polar wedge shaped curvelet windows in the frequency domain outperform the classical wavelet in 2-D case, as it allows one to study 2-D functions by measuring the energy inside the windows oriented along selective orientations to efficiently identify the regularities.

### 1.09.3.2 The discrete curvelet transform

The continuous curvelets can be sampled along its scale, rotation, and translation parameters to form a pyramid of discrete curvelets for numerical computation. A tight frame can be adopted for efficiency so that a 2-D function may be projected onto the discrete curvelet frame with the least number of large coefficients. Here only one tight frame is introduced. First we introduce the window functions to be used in generating the curvelets.

#### 1.09.3.2.1 The selection of windows

As we mentioned earlier, the mother curvelets are primarily defined by two window functions,  $\Psi_1$  and  $\Psi_2$ . A good choice is the Meyer wavelet windows in the Fourier domain. Figure 9.27 shows the highpass and lowpass windows that are orthogonal and overlapping. The lowpass window in the 1-D Fourier domain may be used as the angular window,  $\Psi_2(\sigma)$ , and is defined as

$$\Psi_2(\sigma) = \begin{cases} 1 & |\sigma| \leq 1/3, \\ \cos\left(\frac{\pi}{2}\beta(3|\sigma| - 1)\right) & 1/3 < |\sigma| \leq 2/3, \\ 0 & |\sigma| > 2/3. \end{cases} \quad (9.56)$$

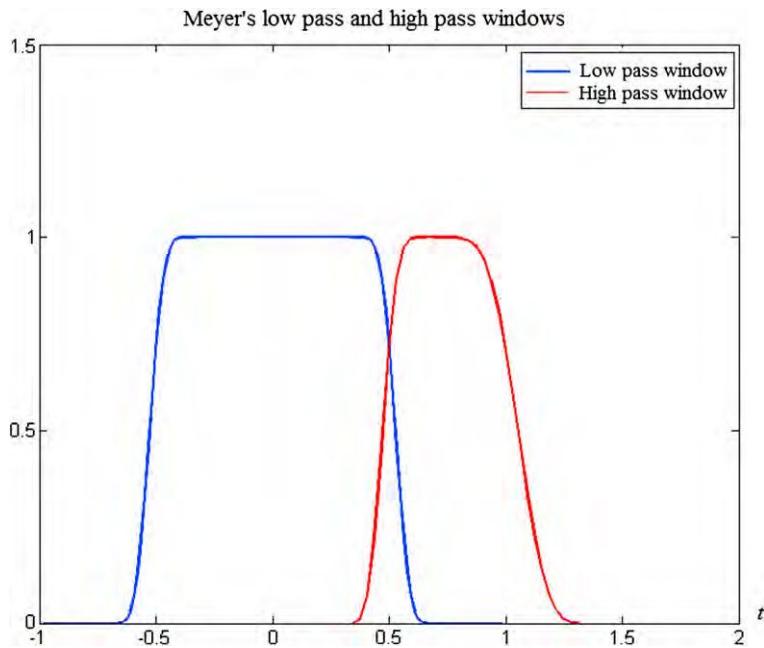
The highpass window in the 1-D Fourier domain may also be used as the radial window  $\Psi_1(r)$  and is defined as

$$\Psi_1(r) = \begin{cases} 0 & |r| \leq 2/3, \text{ and } |r| \geq 8/3, \\ \sin\left(\frac{\pi}{2}\beta\left(\frac{3|r|}{2} - 1\right)\right) & 2/3 < |r| \leq 4/3, \\ \cos\left(\frac{\pi}{2}\beta\left(\frac{3|r|}{4} - 1\right)\right) & 4/3 < |r| < 8/3, \end{cases}$$

where  $\beta(x)$  is any smooth function that defines the transition bands. It is defined on  $[0, 1]$  and satisfies

$$\beta(x) + \beta(1 - x) = 1, \quad x \in \mathcal{R}. \quad (9.57)$$

The following admissibility conditions ensure that the discretized curvelet atoms generated from these windows can completely cover the Fourier plane.



**FIGURE 9.27**

The Meyer's wavelet windows: lowpass  $\Psi_2(t)$ , in blue color, and highpass  $\Psi_1(t)$ , in red color, only its component in the positive axis is shown here. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.)

**Proposition 6.** *On a discrete scale, the two windows,  $\Psi_1$  and  $\Psi_2$  selected above, satisfy the following admissibility conditions,*

$$\sum_{k=-\infty}^{\infty} |\Psi_1(2^{-k}r)|^2 = 1, \quad r > 0, \quad (9.58)$$

$$\sum_{l=-\infty}^{\infty} |\Psi_2(\sigma - l)|^2 = 1, \quad \sigma \in \mathcal{R}. \quad (9.59)$$

The proof of the proposition only needs to be shown in the overlapped regions between two adjacent windows; for example, for  $\Psi_1(r)$ , when  $r \in (2/3, 3/4)$ , we have

$$\begin{aligned} \sum_{k=-\infty}^{\infty} |\Psi_1(2^{-k}r)|^2 &= |\Psi_1(r)|^2 + |\Psi_1(2r)|^2 = \sin^2\left(\frac{\pi}{2}\beta\left(\frac{3|r|}{2} - 1\right)\right) \\ &\quad + \cos^2\left(\frac{\pi}{2}\beta\left(\frac{3|2r|}{4} - 1\right)\right) = 1. \end{aligned} \quad (9.60)$$

Refer to [85] for details of the proof for a different set of window functions. Note that Eq. (9.59) is the same as Eq. (9.44), while Eq. (9.58) is similar to Eq. (9.43) for the continuous case when  $C_{\Psi_1} = \ln 2$ . The selected windows can be scaled to fit  $\text{supp}\{\Psi_1(r)\} = [1/2, 2]$ , and  $\text{supp} \Psi_2(\sigma) = [-1, 1]$ .

The discretized windows are used in the following to build the discrete curvelet atoms. Meanwhile, the coarse scale father curvelet window can be defined as the lowpass window  $\Phi(r)$  in the radial direction which satisfies

$$|\Phi(r)|^2 + \sum_{k \geq 0} |\Psi_1(2^{-k}r)|^2 = 1. \quad (9.61)$$

The choice of  $\beta(x)$  determines the transition band of the Meyer window. The definition of the  $\beta(x)$ , for example, can be chosen as

$$\beta(x) = x^4(35 - 84x + 70x^2 - 20x^3), \quad (9.62)$$

the resulting windows  $\Psi_1$  and  $\Psi_2$  are  $C^3$ . **Or**

$$\beta(x) = x^3(5 - 5x + x^2), \quad (9.63)$$

the resulting windows  $\Psi_1$  and  $\Psi_2$  are  $C^2$ . **Or**

$$\beta(x) = x^2(3 - 2x), \quad (9.64)$$

the resulting windows  $\Psi_1$  and  $\Psi_2$  are  $C^1$ . The 1-D window  $\Psi_1$  has infinitely many vanishing moments in the parameter  $r$  because  $\Psi_1(r) = 0$  for  $|r| < 2/3$ . The mother curvelets constructed from the windows  $\Psi_1$  and  $\Psi_2$  should have infinitely many moments too [83, 85].

### 1.09.3.2.2 Constructing a tight frame of curvelet atoms

Now we are to construct a tight Parseval curvelet frame by sampling the continuous curvelets. With a tight frame, the whole Fourier plane can be covered with discrete curvelet atoms with minimum overlap, and the following equation is satisfied,

$$\sum_{n \in \Gamma} |\langle f, \psi_n \rangle|^2 = C \|f\|_2^2. \quad (9.65)$$

The three parameters  $(a, \theta, b)$  will be sampled in order to obtain a tight frame. First, at each scale  $k$ , the radial scale is sampled on a dyadic interval at  $a_k = 2^{-k}$ ,  $k \geq 0$ , resulting in the selection of the mother curvelet,  $\psi_{k,0,0}$ , defined as the inverse Fourier transform of  $\Psi_{k,0,0}(r, \sigma)$ .  $\Psi_{k,0,0}(r, \sigma)$  is defined using the polar coordinates as,

$$\Psi_{k,0,0}(r, \sigma) = \begin{cases} \Psi_1(2^{-k} \cdot r) \Psi_2\left(\frac{2^{(k+2)/2}\sigma}{\pi}\right) 2^{-3k/4}, & k \text{ is an even integer,} \\ \Psi_1(2^{-k} \cdot r) \Psi_2\left(\frac{2^{(k+1)/2}\sigma}{\pi}\right) 2^{-3k/4}, & k \text{ is an odd integer.} \end{cases}$$

Second, the rotation angular parameter is sampled at  $\theta_{k,l}$ , given as

$$\theta_{k,l} = \begin{cases} \pi l 2^{-(k+2)/2}, & k \text{ is an even integer, } 0 \leq l \leq 2^{\frac{(k+2)}{2}+1} - 1, \\ \pi l 2^{-(k+1)/2}, & k \text{ is an odd integer, } 0 \leq l \leq 2^{\frac{(k+1)}{2}+1} - 1. \end{cases}$$

For example, at  $k = 0$  and  $k = 1$ ,  $\theta_{k,l} = \frac{\pi}{2}l$ ,  $l = 0, 1, 2, 3$ ; at  $k = 2$  and  $k = 3$ ,  $\theta_{k,l} = \frac{\pi}{4}l$ ,  $l = 0, 1, 2, \dots, 7$ . It can be deduced that

$$\sum_{l=0}^{2^{\frac{(k+2)}{2}+1}-1} \left| \Psi_2 \left( \frac{2^{(k+2)/2}(\sigma + \theta_{k,l})}{\pi} \right) \right|^2 = 1, \quad \text{for } k \text{ is an even integer} \quad (9.66)$$

and

$$\sum_{l=0}^{2^{\frac{(k+1)}{2}+1}-1} \left| \Psi_2 \left( \frac{2^{(k+1)/2}(\sigma + \theta_{k,l})}{\pi} \right) \right|^2 = 1, \quad \text{for } k \text{ is an odd integer.} \quad (9.67)$$

This results in the discrete curvelet atoms,

$$\psi_{k,l,b}(\mathbf{x}) = \psi_{k,0,0}(R_{\theta_{k,l}}(\mathbf{x} - \mathbf{b})) \quad (9.68)$$

Finally, the parameter  $\mathbf{b}$  is sampled at

$$\mathbf{b}_s^{k,l} = R_{\theta_{k,l}}^{-1}(s_1 2^{-k}, s_2 2^{-k/2}),$$

where the  $\mathbf{b}_s^{k,l}$  is the grids defined by rotating the dyadic points by an angle  $\theta_{k,l}$ . The discrete curvelet atoms are given as,

$$\psi_{k,l,s}(\mathbf{x}) = \psi_{k,0,0}(R_{\theta_{k,l}}(\mathbf{x} - \mathbf{b}_s^{k,l})), \quad s = (s_1, s_2). \quad (9.69)$$

This curvelet frame consists of interleaved curvelet atoms, separately selected from the samplings of two adjacent scales, the even and odd scales. Under this tight frame, the discrete curvelet transform coefficients are given by

$$Cf_{k,l,s} = \langle f, \psi_{k,l,s} \rangle. \quad (9.70)$$

The coefficients are calculated in the Fourier plane using the following formula based on the mother curvelet:

$$Cf_{k,l,s} = \frac{1}{(2\pi)^2} \int_{\mathcal{R}^2} F(\Omega) \Psi_k^*(R_{\theta_{k,l}}(\Omega)) e^{j\langle \mathbf{b}_s^{k,l}, \Omega \rangle} d\Omega. \quad (9.71)$$

**Theorem 3.** For a highpass function  $f \in L^2(\mathcal{R}^2)$ , Eqs. (9.58), (9.66), (9.67) guarantee that a reproducing formula can be obtained by

$$f = \sum_k \sum_l \sum_{s=(s_1,s_2) \in \mathcal{R}^2} Cf_{k,l,s} \psi_{k,l,s}, \quad (9.72)$$

and the energy conservation is maintained.

$$\|f\|_2^2 = \sum_k \sum_l \sum_{s=(s_1,s_2) \in \mathcal{R}^2} |Cf_{k,l,s}|^2. \quad (9.73)$$

Meanwhile, if we add the father curvelet  $\phi_b(x)$  into the frame, calculate the father curvelet coefficients from the inner product between functions  $\phi_b(x)$  and  $f$ , and put both with the curvelet coefficients, and we still write them as  $Cf_{k,l,s}$ , then, any function in  $L^2(\mathcal{R}^2)$  can be reconstructed from the integrated families of the curvelet atoms, and resulting in the same formulas of Eqs. (9.72) and (9.73).

Under this curvelet tight frame, Candès and Donoho [5] have proved the following theorem, which estimates the approximation error under the curvelets coefficients. It shows a faster convergence rate than that using the classical wavelets in 2-D.

**Theorem 4.** *Let  $f(x)$  be a twice differentiable function in  $\mathcal{R}^2$  except at discontinuities along a  $C^2$  curve. Let  $f_N^C$  be the  $N$ -term approximation of  $f$  reconstructed from using the  $n$  largest curvelet coefficients obtained by simple thresholding. The approximation error follows:*

$$\left\| f - f_N^C \right\|_2^2 \leq CN^{-2}(\log N)^3.$$

The proof can be found in [5].

#### 1.09.3.2.3 Implementation procedure for discrete curvelet transform

The discrete curvelet transform is implemented using a lowpass filter  $P_0$ , together with a set of passband filters  $\Delta_1, \Delta_2, \dots, \Delta_s$ . This bank of filters partition the whole frequency plane into  $s+1$  decompositions that are concentrated, and are located near certain frequencies in a sequence of dyadic intervals. Indeed, in the digital curvelet implementation, this bank of filters are generated using concentric squares instead of concentric circles.

To generate the concentric squares, see Figure 9.28, a lowpass filter at scale  $k$  is first generated using a smooth Meyer window,  $\Phi(r)$ , following Eq. (9.56), as

$$\Phi_k(\Omega_x, \Omega_y) = \Phi(2^{-k}\Omega_x)\Phi(2^{-k}\Omega_y). \quad (9.74)$$

The  $P_0$  refers to the window when  $k = 0$ . Then the bandpass filters  $\Delta_k$  corresponding to bandpass window functions,  $\Psi_{1,k}(\Omega_x, \Omega_y)$ , are defined as

$$\Psi_{1,k}(\Omega_x, \Omega_y) = \sqrt{\Phi_{k+1}^2(\Omega_x, \Omega_y) - \Phi_k^2(\Omega_x, \Omega_y)}, \quad k \geq 0.$$

The angular window  $\Psi_{2,k}(\Omega_x, \Omega_y)$  in the Cartesian plane is also defined using the function  $\Psi_2(\sigma)$  in Eq. (9.56),

$$\Psi_{2,k}(\Omega_x, \Omega_y) = \begin{cases} \Psi_2\left(\frac{2^{k/2}\Omega_y}{\Omega_x}\right) & k \text{ is an even integer,} \\ \Psi_2\left(\frac{2^{(k-1)/2}\Omega_y}{\Omega_x}\right) & k \text{ is an odd integer.} \end{cases}$$

The mother curvelet at scale  $k$  can be obtained in the Fourier domain as,

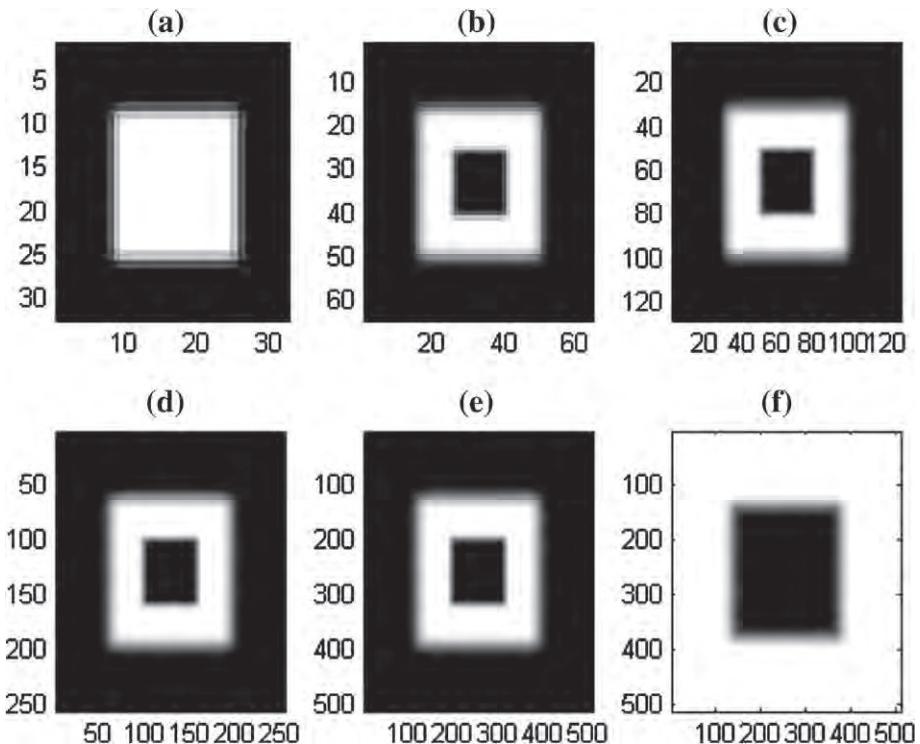
$$\Psi_k(\Omega) = 2^{-3k/4}\Psi_{1,k}(\Omega)\Psi_{2,k}(\Omega), \quad \Omega = (\Omega_x, \Omega_y).$$

The curvelet atoms at  $b = 0$  can be obtained in the Fourier domain as,

$$\Psi_{k,l}(\Omega) = \Psi_k(S_{\theta_{k,l}}^{-1}\Omega) = 2^{-3k/4}\Psi_{1,k}(\Omega)\Psi_{2,k}\left(S_{\theta_{k,l}}^{-1}(\Omega)\right), \quad \Omega = (\Omega_x, \Omega_y),$$

where  $S_{\theta_{k,l}}$ , is the shear matrix, with its inverse  $S_{\theta_{k,l}}^{-1}$ . Both matrices are given as

$$S_{\theta_{k,l}} = \begin{pmatrix} 1 & 0 \\ -\tan(\theta_{k,l}) & 1 \end{pmatrix} \quad \text{and} \quad S_{\theta_{k,l}}^{-1} = \begin{pmatrix} 1 & 0 \\ \tan(\theta_{k,l}) & 1 \end{pmatrix}.$$

**FIGURE 9.28**

Shown here are windows at several scales, namely the window at (a) The low frequency level. (b)  $k = 1$  level. (c)  $k = 2$  level. (d)  $k = 3$  level. (e)  $k = 4$  level. (f) The remaining high frequency at  $k = 4$  level. Note that number of pixels in  $x, y$  direction is shown for each image because of the different size of the window at different scale.

The selection of  $\theta_{k,l}$  is defined through

$$\tan(\theta_{k,l}) = \begin{cases} l \cdot 2^{-k/2} & k \text{ is an even integer, } l = -2^{k/2}, \dots, 2^{k/2} - 1, \\ l \cdot 2^{-(k-1)/2} & k \text{ is an odd integer, } l = -2^{(k-1)/2}, \dots, 2^{(k-1)/2} - 1. \end{cases}$$

The  $\Psi_{k,l}(\Omega_x, \Omega_y)$  indeed defines the Cartesian approximation of the curvelet atoms in the polar Fourier plane along the horizontal cone direction. Rotating this system by  $\pi/2$  defines the Cartesian approximation along the vertical cone direction. The two systems together define the Cartesian approximation of the curvelet atoms [83, 86, 87]. Using the discretized curvelet atoms, the Curvelab software [88] calculates the discrete curvelet coefficients using two different approaches, the unequispaced FFTs (USFFT), and the Wrapping methods. It is believed that the Wrapping method is simpler to understand. Here we summarize the two implementation procedures in the following.

### The unequispaced FFTs (USFFT) method

The USFFT implements the method that carries out the operation of shearing a curvelet atom to shearing  $F$  according to the following formula:

$$\begin{aligned} C_f(k, l, s) &= \frac{1}{(2\pi)^2} \int_{\mathcal{R}^2} F(\Omega) \Psi_k(S_\theta^{-1}\Omega) \exp\left(j \langle S_\theta^{-T} \mathbf{b}, \Omega \rangle\right) d\Omega_x d\Omega_y \\ &= \frac{1}{(2\pi)^2} \int_{\mathcal{R}^2} F(S_\theta \Omega) \Psi_k(\Omega) \exp(j \langle \mathbf{b}, \Omega \rangle) d\Omega_x d\Omega_y \end{aligned} \quad (9.75)$$

with  $\theta = \theta_{k,l}$ ,  $\mathbf{b} \simeq (s_1 2^{-k}, s_2 2^{-k/2})$ . Note that  $\Psi_k(\Omega)$  is a real valued function, therefore  $\Psi_k^*(\Omega) = \Psi_k(\Omega)$  is used in Eq. (9.75). The procedure for implementing the USFFT to calculate the discrete curvelet coefficients is summarized in the following, for more details, refer to [83, 86].

1. Apply a 2-D FFT to a function  $f(\mathbf{x})$ , and obtain its discrete Fourier samples  $F(n_1, n_2)$ ,  $0 < n_1, n_2 \leq N$ , where  $N$  is the sample size.
2. For each scale/angle pair  $(k, l)$ , resample (or interpolate)  $F(n_1, n_2)$  to obtain sheared sample values, written as  $F(n_1, n_2 - n_1 \tan \theta_{k,l})$  for  $(n_1, n_2) \in R_k$ . The  $R_k$  is a rectangle of  $Length = 2^k$ ,  $Width = 2^{k/2}$ , that is at the center of the bandpass content of the mother curvelet at scale  $k$ .
3. Multiply the resampled  $F$  with the mother curvelet window  $\Psi_{k,l}$ , we have  $Cf_{k,l}(n_1, n_2) = F(n_1, n_2 - n_1 \tan \theta_{k,l}) \Psi_{k,l}(n_1, n_2)$ .
4. Apply an inverse 2-D FFT to each  $Cf_{k,l}(n_1, n_2)$  obtained in the previous step to calculate the discrete curvelet coefficients  $C_f(k, l, s)$ .

In Curveletlab [88], the following package in MATLAB is used to decompose the original image into curvelet coefficients:

```
% is_real      Type of the transform
%           0: complex-valued curvelets
%           1: real-valued curvelets
Curveletcoef = fdct_usfft (X,is_real);
```

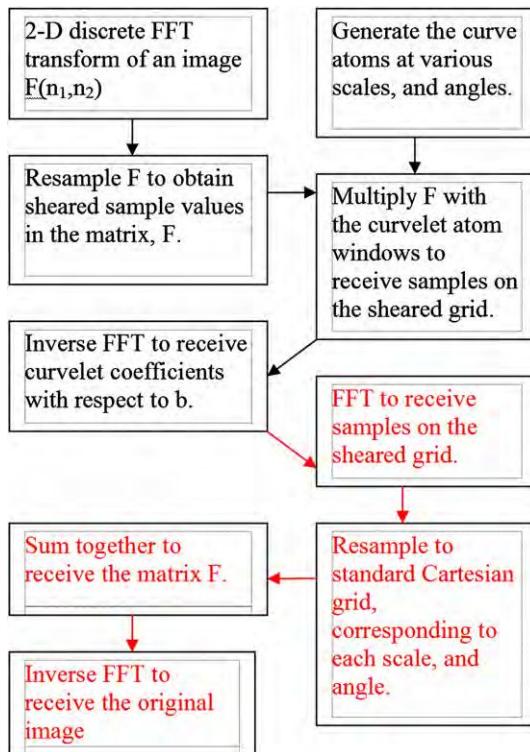
The inverse curvelet transform is simply computed by “reversing” all the operations of the direct transform, with some adjustments:

1. For each scale/angle pair  $(k, l)$ , perform a (properly normalized) 2-D FFT of each curvelet coefficient array  $C_f(k, l, s)$ , to obtain  $(Cf)_{k,l}(n_1, n_2)$  in  $\Psi_{k,l}$ .
2. The  $(Cf)_{k,l}(n_1, n_2)$  should be viewed as samples on the sheared grid. Resample them back to the standard Cartesian grid.
3. Integrate  $(Cf)_{k,l}(n_1, n_2)$  from all scales and angles together. This recovers  $F(n_1, n_2)$ .
4. Take a 2-D inverse FFT to obtain  $f(x, y)$ .

Figure 9.29 provides the diagram for implementing the USFFT curvelet transform and its inverse transform.

In Curveletlab, the following package in MATLAB is used to reconstruct the original image from the curvelet coefficients with syntax as:

```
Y = ifdct_usfft(C,is_real);
```

**FIGURE 9.29**

The USFFT curvelet transform diagram (in black color) and its inverse transform (in red color). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.)

The computational cost for the forward and inverse transform requires about  $O(n^2 \log n)$  operations, and  $O(n^2)$  memory storage.

### The wrapping method

The Wrapping is implemented by using the following formula to calculate the curvelet coefficients:

$$C_f(k, l, s) = \frac{1}{(2\pi)^2} \int_{\mathcal{R}^2} F(\Omega) \Psi_k(S_\theta^{-1}(\Omega)) \exp(j \langle \mathbf{b}, \Omega \rangle) d\Omega_x d\Omega_y.$$

Note that the  $S_\theta^{-1}\mathbf{b}$  in Eq. (9.75) is replaced by  $\mathbf{b}$  that takes its value on the rectangle grid. In this method, the curvelet atoms,  $\Psi_{k,l} = \Psi_k(S_{\theta_{k,l}}^{-1}(\Omega))$ , should be obtained by applying the shear operation  $S_{\theta_{k,l}}^{-1}$  to the rectangle  $R_k$  as in the USFFT method. The resulting window should be a trapezoidal shaped region.

The following is the procedure for implementing the Wrapping technique to calculate the discrete curvelet coefficients,

1. Calculate the 2-D FFT of an image  $f(x)$  to obtain the Fourier samples  $F(n_1, n_2)$ .

2. For  $(k, l)$ , decompose  $F(n_1, n_2)$  into each curvelet window by calculating  $F(n_1, n_2)\Psi_{k,l}(n_1, n_2)$ .
3. Perform wrapping in this stage, namely, wrap this product around the origin in the Fourier plane. This is done by relabeling the samples.
4. Apply the inverse 2-D FFT to the wrapped  $F(n_1, n_2)$  to calculate the discrete curvelet coefficients  $C_f(k, l, s)$ .

In Curveletlab, the following package is used to calculate the curvelet coefficients using the wrapping method with syntax as:

```
Curveletcoefwrp = fdct_wrapping(X, 0).
```

The inverse curvelet transform is computed as

1. For each  $(k, l)$ , perform a properly normalized 2-D FFT of each curvelet coefficient array  $C_f(k, l, s)$ , to obtain  $(Cf)_{k,l}(n_1, n_2)$  in  $\Psi_{k,l}$ .
2. For each  $(k, l)$ , multiply the array of the coefficients by the corresponding wrapped curvelet  $\Psi_{k,l}(n_1, n_2)$  which gives  $(|\Psi_{k,l}|^2 F)(n_1, n_2)$ .
3. Unwrap each array  $(|\Psi_{k,l}|^2 F)[n_1, n_2]$ . with  $\Psi_{k,l}(n_1, n_2)$  on the frequency grid and integrate them all to form  $F(n_1, n_2)$ .
4. Finally, take a 2-D inverse FFT of  $F(n_1, n_2)$  to obtain  $f(x, y)$ .

In Curveletlab, the following package is used to reconstruct the original image from the curvelet coefficients with syntax as:

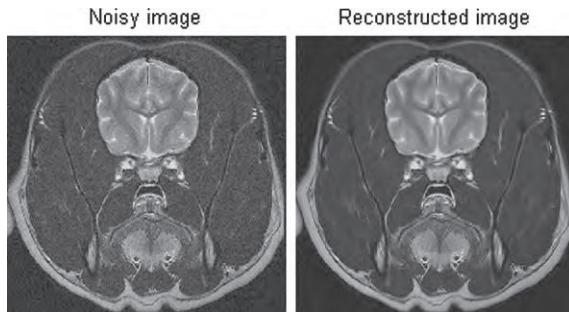
```
image = fdct_wrapping_dispcoef(Curveletcoefwrp).
```

In the wrapping approach, both the forward and inverse transforms are computed in  $O(n^2 \log n)$  operations, and require  $O(n^2)$  storage.

### 1.09.3.3 Applications

Researchers have applied the curvelet transform in solving PDEs and in numerous other applications including image processing, seismic exploration, fluid mechanics, and data compression [87, 89–100]. All applications take advantage of the flexibility of curvelets to selectively capture curve-based features. For illustration in this manuscript, we limit ourselves to the application of curvelets in image denoising.

The assumption about noisy signals in previous sections, can be similarly made for 2-D noisy images of interest herein. It is well known that a 2-D isotropic wavelet transform of an image exhibits a large number of significant coefficients at every fine scale, making wavelet denoising more complex as many more wavelet coefficients are needed for reconstructing edges while smoothing out the noise [68–70, 101]. The development of curvelet theory has shown that discrete curvelet coefficients provide near-optimal way to represent smooth objects with discontinuities along  $C^2$  continuous curves in an image. The curvelet-based denoising method is similar to that of the wavelet. It consists of determining which curvelet coefficients represent the transient features in an image. It is shown that the high amplitude curvelet coefficients usually indicate the position of edges, and the low amplitude curvelet coefficients usually capture noise present in the image. Therefore, the denoising is accomplished by applying a threshold to the curvelet coefficients.

**FIGURE 9.30**

Shown in the left is a noisy image, in the right is the image after applying the hard threshold denoising method.

A hard threshold estimator is defined as,

$$I_T(x) = \begin{cases} x & \text{if } |x| \geq T, \\ 0 & \text{if } |x| < T. \end{cases}$$

Applying the thresholds is equivalent to locally averaging the noise contribution present in the image around the local area the image is smooth over.

Here we assume the noise normal distribution,  $N(0, \sigma^2)$ , and denote by  $y(\mathbf{n})$  the discrete curvelet coefficients from a noisy image. The following hard-threshold is applied to estimate the curvelet coefficients,  $\tilde{y}(\mathbf{n})$ , of the denoised image as follows:

$$\tilde{y}(\mathbf{n}) = \begin{cases} y(\mathbf{n}) & \text{if } |y(\mathbf{n})|\sigma \geq T_a, \\ 0 & \text{if } |y(\mathbf{n})|\sigma < T_a. \end{cases}$$

The threshold  $T_a$  is chosen as scale depended, and can be estimated from the noisy image.

It is shown that the curvelet-based denoising displays higher sensitivity than the wavelet-based denoising, see Figure 9.30. In curvelet denoising, salient features in the noisy image can be clearly enhanced, while the noise can be selectively removed [102]. However, even though the pure discrete curvelet denoising has improved the image denoising method, it is interesting to see better denoising effects has been achieved by a hybrid approach [89] called total variational method (TV) that combines the discrete curvelet transform with the classical wavelet method. The TV method has shown a higher capability in exploiting curve-like features.

---

## 1.09.4 Contourlets and their applications

### 1.09.4.1 Contourlet transform

In this section, we introduce the contourlet transform, proposed by Do and Vetterli [9]. The contourlet transform is a discrete filter bank implementation that expands multi-dimensional signals into

multiresolution copies for analysis, approximation, and compression [9, 103]. Here we consider only two dimensional (2-D) signals-images. A filter bank technique uses multirate signal processing technique that has been successfully applied in many application areas [41, 104]. It is a natural and practical approach to efficiently compute a set of good basis functions for an image [105, 106]. The contourlet filter bank technique proceeds to decompose the frequency domain into angularly oriented directions from fine to coarse resolution. It implements similar features of the curvelet analysis so that certain frequency regions in coarse or detailed version can be focused on under a selected basis/frame. This advances the classical discrete filter bank technique that is used in the classical wavelet analysis. It allows the representation of a 2-D signal to be optimized using only a few significant coefficients corresponding to the geometrical features in an image, such as edges with continuous curves.

The multiresolution decomposition filter bank technique is represented by the Laplacian pyramid decomposition, originally developed by Burt and Adelson [107] for image coding. Its development has inspired the usage of discrete filter bank technique to decompose a 2-D image into images of multiresolution in discrete-time wavelet bases. This has further inspired the research for building the connection between the filter banks and classical wavelets led by Mallat [84] and Daubechies [20]. In turn, this has led to the development of the filter bank technique, which has played an active role in constructing various bases for classical wavelets. It has strengthened the connection between harmonic analysis and discrete signal processing. A filter bank with a directional oriented decomposition capacity was later proposed by Bamberger [108]. The recently proposed contourlet filter banks have integrated the Laplacian pyramid (LP) filter bank and the directional filter bank (DFB) for multiresolution anisotropic wavelet decomposition.

The contourlet transform is a tree-structured filter bank that cascades multiple LP and DFB filter banks iteratively. In general, the filter banks use maximally decimated filter banks similar to those in subband coding, and invoke the usage of multirate signal processing to reduce the computational complexity. Each filter bank in the structure can be divided into two stages: an analysis filter bank and a synthesis filter bank. An analysis filter bank decomposes a signal into different version sub-signals. Each subsignal is down-sampled to its frequency content concentrating in a segment in the Fourier plane. A synthesis filter bank up-samples and combines the output signals from the analysis filter bank into one signal. The basic structure of the contourlets is a two channel maximally decimated filter bank, referred to as a quadrature mirror filter (QMF) bank.

In synthesis, a perfect reconstruction is achieved when the output signal equals to the original signal, which is of interest for an optimal approximation of the original signal. This is useful in signal compression, as an individual image of reduced size is useful for transmission even though the total sampling size is increased. To avoid a redundant expansion of a signal, filters that implement orthonormal bases are sought. Filters that implement biorthogonal bases or tight frames are, however, necessary in some cases in order to relax some constraints.

A digital contourlet filter bank inevitably inherits the distortion problem in digital signal processing; to avoid this problem, a non downsampling version of the contourlets has been proposed [109], which is parallel to the non sub-sampled discrete wavelet transform (NSWT) with the à trous algorithm. Additionally, an M-band method that provided non-redundant DFB was proposed in [110].

In this section, we downplay the distortion problems with downsampling/upsampling to keep our focus on multiresolution analysis. As a polyphase decomposition is typically used to form a fast computing system for implementing filter banks [41, 106, 111], in the following, we start by introducing

some basic sampling operations in digital image processing. We then introduce the LP filter banks, and the DFB filter banks before we introduce the contourlets as the combination of the two type filter banks.

#### 1.09.4.1.1 The downsampling and upsampling operators

Downsampling is a decimation operator that reduces the original image size, while upsampling is the interpolation operator that increases the original image size. The two operations are the basic techniques that form the multirate digital system. In the 2-D digital image processing, downsampling is to generate multiple components of reduced size for the economy of analyzing an image, while upsampling is used to recover the parent image. In this section and the following,  $\mathbf{n} \in \mathcal{Z}^2$  is viewed as a column vector  $\mathbf{n} = (n_1, n_2)^T$  since matrix multiplication is involved. Here “ $T$ ” is referred to the transpose of a vector/matrix.

Both sampling operations involve a  $2 \times 2$  sampling matrix given by

$$\mathbf{M} = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}, \quad \text{where } m_1, m_2 \in \mathcal{Z}.$$

Given a 2-D image,  $x(\mathbf{n})$ ,  $\mathbf{n} = (n_1, n_2) \in \mathcal{Z}^2$ , with its  $z$ -transform  $X(z)$ , see appendix (A.1), an  $M$ -fold downsampling (decimator) takes  $x(\mathbf{n})$  as an input, and produces an output,  $x_d(\mathbf{n})$ , given by

$$x_d(\mathbf{n}) = x(\mathbf{M}\mathbf{n})$$

An  $M$ -fold upsampling (expander, or interpolator) takes  $x(\mathbf{n})$  as an input, and produces an output,  $x_u(\mathbf{n})$ , given by

$$x_u(\mathbf{n}) = \begin{cases} x(\mathbf{s}) & \text{if } \mathbf{n} = \mathbf{Ms}, \mathbf{s} \in \mathcal{Z}^2 \\ 0 & \text{otherwise.} \end{cases}$$

The  $z$ -transform of an  $M$ -fold downsampled signal,  $x_d(\mathbf{n})$ , is given as

$$X_d(z) = |\det(\mathbf{M})|^{-1} \sum_{\mathbf{k}=(k_1, k_2) \in \mathcal{S}} X\left(z_1^{1/m_1} e^{-j2\pi k_1/m_1}, z_2^{1/m_2} e^{-j2\pi k_2/m_2}\right).$$

The  $\mathcal{S}$  is the set of all integer points in  $\mathbf{Mt}$ ,  $t \in [0, 1) \times [0, 1)$ . The  $z$ -transform of an  $M$ -fold upsampled signal,  $x_u(\mathbf{n})$  is given as

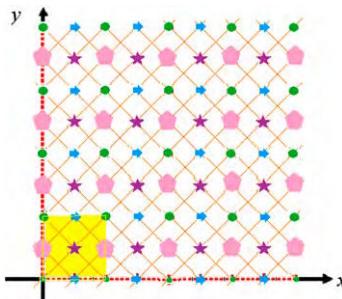
$$X_u(z) = X(z_1^{m_1}, z_2^{m_2}), \quad \text{with } z = (z_1, z_2).$$

The dyadic subsampling occurs in  $x$  or  $y$  direction when  $|\det(\mathbf{M})| = 2$ , where  $\det(\mathbf{M})$  is the determinant of a matrix  $\mathbf{M}$ . Dyadic subsampling in both  $x$  and  $y$  directions occurs when  $m_1 = 2$  and  $m_2 = 2$  in the diagonal matrix  $\mathbf{M}$ . The dyadic sampling is usually used in the digital filter bank design of a discrete wavelet transform or discrete anisotropic wavelet transform. Sampling operations that invoke non-diagonal matrix are to be introduced in the following section.

#### 1.09.4.1.2 The polyphase decomposition

The polyphase decomposition involves subsampling and upsampling operations which use a non-diagonal matrix,  $\mathbf{M}$ , of non-singular integer-value, given as

$$\mathbf{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}.$$

**FIGURE 9.31**

The resampling lattice generated by a dyadic sampling matrix.

This corresponds to a shearing (resampling) operation. The collection of all the subsampling vectors,  $\mathbf{M}\mathbf{n}$ , is called a lattice given by

$$LAT(\mathbf{M}) = \{\mathbf{m} : \mathbf{m} = \mathbf{M}\mathbf{n}, \mathbf{n} \in \mathbb{Z}^2\}.$$

The  $LAT(\mathbf{M})$  is indeed a vector space spanned by the linear combination of basis column vectors from  $\mathbf{M}$ , written as  $n_1\mathbf{M}_1 + n_2\mathbf{M}_2$ , with  $\mathbf{M}_1 = (m_{11}, m_{21})^T$ ,  $\mathbf{M}_2 = (m_{12}, m_{22})^T$ . In our context, a non-singular subsampling matrix  $\mathbf{M}$  refers to the matrices defined in the section of DFB.

Let  $\mathcal{S}$  be the set of points of  $\mathbf{M}\mathbf{t}$ ,  $\mathbf{t} \in [0, 1] \times [0, 1]$ , called the fundamental parallelepiped of matrix  $\mathbf{M}$ . It consists of all real-valued vectors in an area of the parallelogram controlled by  $\mathbf{M}_1$  and  $\mathbf{M}_2$ . An example of  $\mathcal{S}$  is shown in the shadowed square area in Figure 9.31. However, we are only interested in the integer valued points in  $\mathcal{S}$ . According to the division theorem in number theory [41], any integer vector can be written as  $\mathbf{M}\mathbf{n} + s$  with  $s = (s_1, s_2) \in \mathcal{S}$ , so there are  $D = |\det(\mathbf{M})|$  integer points in  $\mathcal{S}$ . We can label each integer vector  $s \in \mathcal{S}$  using index  $k = 0, 1, \dots, D - 1$ . Note that  $k$  cannot be a natural ordering like 1-D case. The sampling lattice of  $\mathbf{M}\mathbf{n} + s$  results from shifting  $LAT(\mathbf{M})$  to the location  $s \in \mathcal{S}$ , yielding the  $k$ th coset of  $LAT(\mathbf{M})$  generated by an integer valued points inside  $\mathcal{S}$ . Therefore, the original lattice of integer numbers in the coordinate system is the union of all the cosets.

For a 2-D image, the polyphase decomposition is the subsampling over the cosets defined above. The  $k$ th component sub image takes a value at the location vector in the  $k$ th coset associated with  $s$ , and is defined as

$$x^{(k,s)}(\mathbf{n}) = x(\mathbf{M}\mathbf{n} + s).$$

Denote by  $x_u^{(k,s)}(\mathbf{n})$  the upsampled version of the  $k$ th polyphase component,  $x^{(k,s)}(\mathbf{n})$ , with a shifting by  $s \in \mathcal{S}$ , defined as

$$x_u^{(k,s)}(\mathbf{n}) = \begin{cases} x^{(k,s)}(\mathbf{m}) & \text{if } \mathbf{n} = \mathbf{M}\mathbf{m} + s, \mathbf{s} \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

The parent image can then be reconstructed as the sum of  $x_u^{(k,s)}(\mathbf{n})$ , namely,

$$x(\mathbf{n}) = \sum_{k=0}^{D-1} x_u^{(k,s)}(\mathbf{n}).$$

Its  $z$ -transform can be written as

$$X(\mathbf{z}) = \sum_s z_1^{-s_1} z_2^{-s_2} X^{(k,s)}(\mathbf{z}^M), \quad \text{with } \mathbf{z} = (z_1, z_2), s = (s_1, s_2) \in \mathcal{S}.$$

Here, for a non-diagonal matrix  $M$ ,  $\mathbf{z}^M$  stands for  $(z_1^{m_{11}} z_2^{m_{21}}, z_1^{m_{12}} z_2^{m_{22}})$ , and  $X^{(k,s)}(\mathbf{z})$  is the  $\mathbf{z}$ -transform of  $x^{(k,s)}(\mathbf{n})$ , see [39].

**Definition 10.** To be consistent with the subsampling concepts, the polyphase representation of the  $z$ -transform,  $X(z)$ , of an image  $x(\mathbf{n})$  is defined as a vector,  $\mathbf{x}(z)$ , written as

$$\mathbf{x}(z) = (X_0(z), X_1(z), \dots, X_{D-1}(z)),$$

with  $X_k(z) = X^{(k,s)}(z)$ , for  $k = 0, 1, \dots, D - 1$ .

For example, a dyadic subsampling matrix  $M$  will have its cosets generated by points  $(i, j)$ ,  $i, j = 0, 1$  in  $S$ , and the  $z$ -transform of  $x(\mathbf{n})$  can be written as

$$X(z) = X_0(z_1^2, z_2^2) + z_1^{-1} X_1(z_1^2, z_2^2) + z_2^{-1} X_2(z_1^2, z_2^2) + z_1^{-1} z_2^{-1} X_3(z_1^2, z_2^2),$$

where  $X_k(z_1, z_2) = \sum_{n_1} \sum_{n_2} z_1^{-n_1} z_2^{-n_2} x(2n_1 + i, 2n_2 + j)$ ,  $k = 2 * j + i$ ,  $i, j = 0, 1$ . The polyphase representation can then be written as

$$\mathbf{x}(z) = (X_0(z_1, z_2), X_1(z_1, z_2), X_2(z_1, z_2), X_3(z_1, z_2)).$$

The use of a polyphase decomposition in the design of a finite input response (FIR) filter bank reduces the computational complexity in the realization of a filter bank, in which the polyphase representation is frequently used. Note that  $X(z)$  is a vector, the  $z$ -transfer function takes a scalar value, while the polyphase representation  $\mathbf{x}(z)$  takes vector value. We assume the signal is periodically expanded if it is of finite length.

A filter bank with a solo implementation of the polyphase decomposition is also termed as a lazy filter bank because of its simple reconstruction structure that involves only the basic operations, such as addition, delay, downsampling and upsampling. A iterated lazy filter bank leads to the so called lazy wavelet.

### 1.09.4.2 The Laplacian pyramid frames

Introduced by Burt and Adelson [8,107], the LP was initially aimed at representing data in a multiresolution setting from fine to coarse levels for data compression purpose. The analysis filter bank of LP iteratively decomposes an image into multiple sub images of coarse approximation, namely, the low resolution sub images with lowpass content, and detail images representing the difference between the lowpass filtered image and the input image. The synthesis filter bank integrates the outputs from the analysis filters into an estimate of the original image. The perfect reconstruction occurs when the estimate is a replica of the original image.

The analysis filter bank of the LP results in a sequence of bandpass images and a further upsampled low resolution image in the end level. The synthesis filters are used to recover the original image from this

sequence of sub-images. The double layered filter bank analyzes an image in various components of size-reduced images with their frequency contents limited to sub-bands. The pyramid design has inspired the connection of filter banks with the wavelet implementation and calculating linear expansions of multiresolution signals.

#### 1.09.4.2.1 The analysis filter banks and synthesis filter banks of the Laplacian pyramid

The analysis filter bank is shown in Figure 9.32. It computes the approximation and the detail image at one single level. Take  $x_0(\mathbf{n})$  as the input image, the analysis LP filter bank decomposes the image using a lowpass filter  $H$  with the impulse response  $\tilde{h}(\mathbf{n}) = h(-\mathbf{n})$ . The output from the lowpass filter  $H$  is down-sampled, resulting in  $x_1(\mathbf{n})$  that can be written as

$$x_1(\mathbf{n}) = \sum_{s \in \mathbb{Z}^2} x_0(s)h(\mathbf{M}\mathbf{n} - s) = \langle x, \tilde{h}(\cdot - \mathbf{M}\mathbf{n}) \rangle,$$

where  $\mathbf{M}$  is a sampling matrix which defines the downsampling. The  $z$ -transform of  $x_1(\mathbf{n})$  can be written in a polyphase representation as

$$X_1(z) = \mathbf{h}(z)x_0^T(z) \quad (9.76)$$

with

$$\mathbf{x}_0(z) = (X_{00}(z), X_{01}(z), \dots, X_{0(D-1)}(z)),$$

and

$$\mathbf{h}(z) = (H_0(z), H_1(z), \dots, H_{D-1}(z))$$

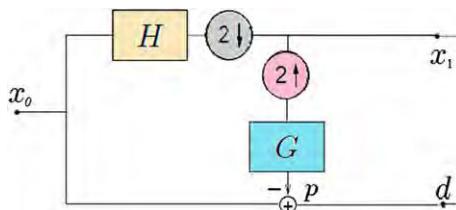
both are  $1 \times D$  polyphase representation vectors.

The detail image is the difference between the approximation image,  $x_1(\mathbf{n})$ , and the original image,  $x_0(\mathbf{n})$ . It is calculated as the difference between a different version of the approximation image,  $p(\mathbf{n})$ , and the original image,  $x_0(\mathbf{n})$ , written as

$$d(\mathbf{n}) = x_0(\mathbf{n}) - p(\mathbf{n}).$$

The lowpass image  $p(\mathbf{n})$  of the original size is obtained as the following,

$$p(\mathbf{n}) = \sum_{s \in \mathbb{Z}^2} x_1(s)g(\mathbf{n} - \mathbf{Ms}).$$



**FIGURE 9.32**

The LP filter bank in a single level.

To calculate  $p(\mathbf{n})$ , the already down-sampled image  $x_1(\mathbf{n})$  is upsampled, and filtered by another lowpass filter  $G$  with impulse response  $g(\mathbf{n})$ . The polyphase representation  $\mathbf{p}(z)$  can then be written as

$$\mathbf{p}^T(z) = (P_0(z), P_1(z), \dots, P_{D-1}(z))^T = \mathbf{g}^T(z)X_1(z), \quad (9.77)$$

where  $\mathbf{g}(z) = (G_0(z), G_1(z), \dots, G_{D-1}(z))$  is a  $1 \times D$  vector, and  $X_1(z)$  is a scalar function. Thus the polyphase representation of  $d(\mathbf{n})$ , can be calculated using Eqs. (9.76) and (9.77), and can be written as

$$\mathbf{d}^T(z) = (I - \mathbf{g}^T(z)\mathbf{h}(z))\mathbf{x}_0^T(z). \quad (9.78)$$

If we cascade Eqs. (9.78) and (9.76), we will have a vector equation

$$\begin{pmatrix} X_1(z) \\ \mathbf{d}^T(z) \end{pmatrix} = \begin{pmatrix} \mathbf{h}(z) \\ I - \mathbf{g}^T(z)\mathbf{h}(z) \end{pmatrix} \mathbf{x}_0^T(z) = \Phi(z) \mathbf{x}_0^T(z), \quad (9.79)$$

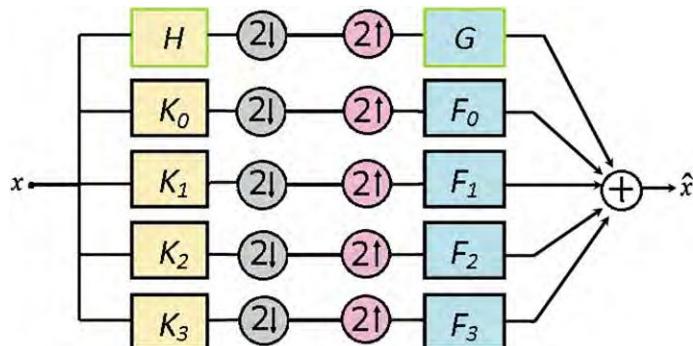
where  $\Phi(z)$  is a matrix of size  $(D + 1) \times D$ .

In the synthesis filter bank, a simple reconstruction of the original image can be computed by directly inverting the steps in the analysis filter. A more efficient reconstruction method is to search for the inverse operator under an orthogonal frame system to calculate the orthogonal projections that is defined as the pseudo inverse of matrix  $\Phi(z)$ . This approach provides the best approximation [112].

The LP filter bank can be realized through a polyphase filter bank structure, in which the lowpass filter  $H$  is determined by its transfer function  $H(z)$ , and the highpass filter  $K_i$ ,  $i = 0, 1, 2, \dots, D - 1$ , is determined by a transfer function whose polyphase representation is given as the  $i$ th row vector of the matrix  $\Phi(z) = (I - \mathbf{g}^T(z))\mathbf{h}(z)$ . Therefore the highpass filters are derived from the lowpass filter  $H$  and  $G$  in the filter bank described in Figure 9.32. Figure 9.33 shows the polyphase structure filter bank for dyadic subsampling which is to be used in the contourlet filter bank.

Based on the structure of a single level analysis filter bank shown in Figure 9.32, the procedure of multiresolution decomposition using the Laplacian pyramid (LP) filter bank is listed as follows:

- Input an image  $x_0(\mathbf{n})$  into the one level LP filter bank. The outputs are the low resolution upsampled image  $x_1(\mathbf{n})$  and a detailed difference image  $d_1(\mathbf{n})$ .



**FIGURE 9.33**

The LP filter implemented using the polyphase decomposition with dyadic subsampling.

- Input the  $x_1(\mathbf{n})$  into the analysis filters again. The outputs are a low resolution image  $x_2(\mathbf{n})$  that is further upsampled, and a detailed image  $d_2(\mathbf{n})$  that is the difference between  $x_1(\mathbf{n})$  and  $x_2(\mathbf{n})$ .
- Iterate the procedure  $K$  times.

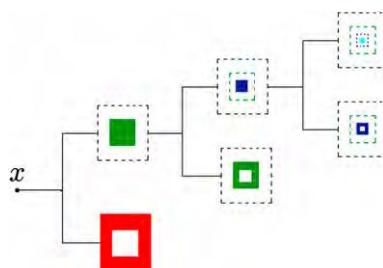
The output sequence  $\{x_k\}_{k=1,2,\dots,K}$  is called a Gaussian pyramid because the Gaussian lowpass filters are typically used. The filtering structure for generating the band pass sequence  $\{d_k\}_{k=1,2,\dots,K}$  is called the Laplacian pyramid. As the  $x_k$  is further decomposed, the final outputs are the sequence of detail images  $d_k$ , with only one low resolution image at the end level, thus yielding a filter bank is generally referred to as the Laplacian pyramid. It is a cascade of a sequence of filter banks for a multiresolution approximation of an original image.

#### 1.09.4.2.2 The wavelet frame associated with the Laplacian pyramid

A wavelet transform can be implemented by computing the fine to coarse multiresolution decomposition as shown in Figure 9.34. The wavelet frame [8] defined in Eq. (9.13) for the decomposition can be specified by using the filters  $H$  and  $G$  of the LP filter bank to determine the mother/father wavelets.

In the LP filter bank, a frame operator  $\widehat{K}_i$  can be defined as  $\widehat{K}_i(x, \phi_i) = \langle x, \phi_i \rangle$ , where each  $\phi_i$  is the  $(i+1)$ th row of the matrix  $\Phi(z)$  in Eq. (9.79). It characterizes the filter  $K_i$  in the polyphase filter bank. This shows that the Laplacian pyramid, with bounded output for any bounded input, provides a frame expansion in  $L^2(\mathcal{Z}^2)$  [9]. If the frame vector  $\{\phi_i\}$  is normalized with  $\|\phi_i\| = 1$  and we have  $A = B = 1$ , then  $\{\phi_i\}$  is an orthonormal basis in Eq. (9.13).

A non-orthonormal frame induces a redundant representation of an image, for which the reconstruction has to use a redundant frame operator that involves a dual frame of  $\widehat{K}_i$ , which performs the inverse/pseudo inverse of the matrix  $\Phi$ . It generates instability that affects the quality of reconstructing the original image. A perfect reconstruction occurs when the original image is replicated. In numerical analysis, the research interests have been focused on searching for orthonormal bases for optimal approximations. With an orthogonal basis, when a signal  $x(\mathbf{n})$  is projected onto a subspace  $V$ , given as  $P_V(x)$ , it generates the minimum approximation error. In some case, due to the requirement of the basis functions, we have to relax the requirement for an orthonormal basis and replace it with an orthogonal basis, a biorthogonal basis, or even a tight frame for multiresolution analysis. This is used to inject some flexibilities, such as, to minimize the redundancy of the expansion.



**FIGURE 9.34**

An LP filter bank that decomposes the frequency content of an image into three levels is shown.

The Laplacian pyramid uses a pair of biorthogonal filters with a sampling matrix  $\mathbf{M}$ , and satisfies

$$\langle \bar{h}(\cdot - \mathbf{M}\mathbf{v}), g(\cdot - \mathbf{M}\mathbf{l}) \rangle = \delta(\mathbf{v} - \mathbf{l}).$$

The filters become orthogonal when they satisfy

$$h(\mathbf{n}) = g(-\mathbf{n}) \quad \text{and} \quad \langle g(\cdot), g(\cdot - \mathbf{M}\mathbf{n}) \rangle = \delta(\mathbf{n}).$$

The orthogonal requirement on the filters  $H$  and  $G$  is equivalent to

$$H^*(z) = G(z) \quad \text{and} \quad G^*(z)G(z) = 1.$$

A perfect reconstruction can then be achieved, thus providing a simple structure that connects the LP filter bank with the classical wavelets [84] for multiresolution analysis. A similar implementation involving a directional decomposition that further connects the anisotropic curvelets with the discrete filter banks is discussed next.

### 1.09.4.3 The directional filter banks

Introduced by Bamberger and Smith [108] in 1992, a directional filter bank (DFB) is a maximally decimated QMF bank that partitions the 2-D Fourier plane with  $2^r$  directional oriented wedge-shaped sub-bands at each level  $r$ . A perfect reconstruction of the original image may be obtained. The implementation introduced here is using a polyphase like tree structured FIR filter bank [9, 113]. This DFB has a simplified structure and uses fan filters and resampling operators with a dyadic downsampling in each level.

#### 1.09.4.3.1 The resampling operators

The design of a directional filter bank requires resampling operators controlled by sampling matrices, such as shearing operators. Here we first introduce the resampling matrices.

**Definition 11.** A quincunx sampling matrix is defined as one of the following two matrices

$$\mathbf{Q}_0 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{Q}_1 = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}.$$

**Definition 12.** An integer matrix  $\mathbf{M}$  is said to be a unimodular matrix if  $|det(\mathbf{M})| = 1$ . Here, we refer to the following four matrices  $\mathbf{R}_i$ ,  $i = 0, 1, 2, 3$ ,

$$\mathbf{R}_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{R}_1 = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{R}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{R}_3 = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}.$$

The two quincunx matrices can be further decomposed as

$$\mathbf{Q}_0 = \mathbf{R}_1 \mathbf{D}_0 \mathbf{R}_2 = \mathbf{R}_2 \mathbf{D}_1 \mathbf{R}_1 \quad \text{and} \quad \mathbf{Q}_1 = \mathbf{R}_0 \mathbf{D}_0 \mathbf{R}_3 = \mathbf{R}_3 \mathbf{D}_1 \mathbf{R}_0,$$

where the two diagonal matrices  $\mathbf{D}_0$  and  $\mathbf{D}_1$  are the dyadic subsampling matrix along  $x$ , or  $y$  directions, and are given as

$$\mathbf{D}_0 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{D}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

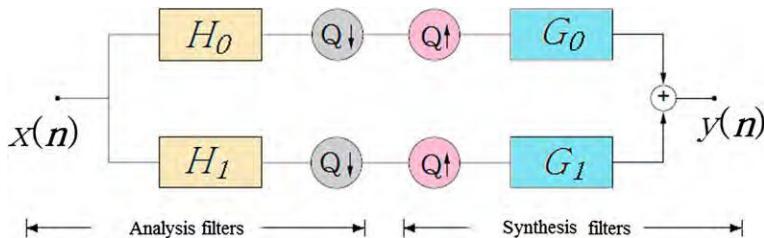
A filter that uses  $Q_0$  or  $Q_1$  carries out a downsampling operation that sheers an image along the orientation at  $45^\circ$  or  $-45^\circ$ . In the DFB,  $Q_0$  and  $Q_1$  are typically cascaded together to result in dyadic down-sampling in both  $x$  and  $y$  directions. This is because  $Q_0 Q_1 = Q_1 Q_0 = 2I$ , where  $I$  is the unit matrix.

#### 1.09.4.3.2 The design of the quincunx filter banks

A quincunx filter bank (QFB) [113, 114] is a two-channel QMF as shown in Figure 9.35. It incorporates the analysis and synthesis filters with the decimators and expanders being the quincunx resampling matrices. A QFB with fan filters used in the analysis part is a typical example. The fan filters are non-separable filters. In Figure 9.36 is shown the support of one fan filter which can be designed starting from a 1-D filter [41].

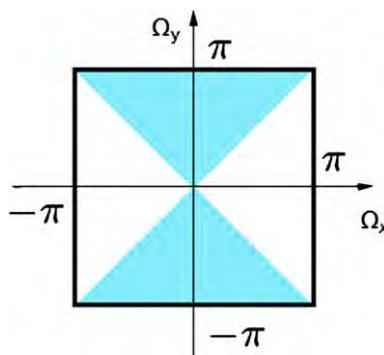
The single level structure analysis filter bank of QFB decomposes the frequency plane of a 2-D image into two wedge shaped cones. The horizontal cone uses  $H_0$ , a horizontal pass fan filter. The vertical cone uses  $H_1$ , a vertical pass fan filter. The synthesis filters, denoted by  $G_0$  and  $G_1$ , perfectly reconstruct the original image when the two dual filter frames satisfy the following conditions,

$$H_0(\omega)G_0(\omega) + H_1(\omega)G_1(\omega) = 2,$$



**FIGURE 9.35**

The quincunx filter bank with resampling matrix  $Q$ .



**FIGURE 9.36**

The support of a fan filter in the Fourier plane. The shaded area is the bandpass area.

$$H_0(\omega + \pi)G_0(\omega) + H_1(\omega + \pi)G_1(\omega) = 0.$$

If the synthesis filters are the reversed versions of the analysis filters then the frame generated by the QFB is an orthogonal frame. This simplifies the design of the synthesis filters. When the analysis and synthesis filters are restricted to be finite impulse response (FIR), the perfect reconstruction conditions imply that the synthesis filters are specified by the analysis filters (up to a shift and scale factor  $i$ ) as the following,

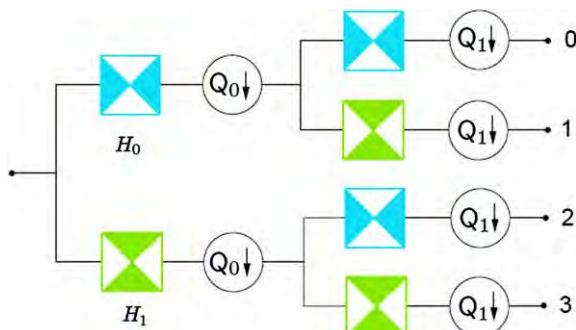
$$G_0(z) = z^i H_1(-z), \quad G_1(z) = -z^i H_0(z).$$

#### 1.09.4.3.3 The design of the directional filter banks

A QFB is the essential QMF bank in a tree structured DFB filter bank. An iterative DFB decomposes the frequency space of a 2-D image into  $2^r$  small angularly oriented wedges at the  $r$ th level. The DFB is a cascade of multiple QFBs, together with selective resampling matrices to determine the shearing angle of the segmented wedge in Fourier plane. The iterative DFB can be viewed as proceeding in multiple stages. According to the Nobel identity shown in Figure 9.38a, the overall output in two adjacent stages can be viewed as applying a decimation  $M = Q_0 Q_1$  using the overall analysis filter  $H_1(z)H_2(z^{M^T})$ . Figure 9.38b shows the overall effect of a DFB in two stages.

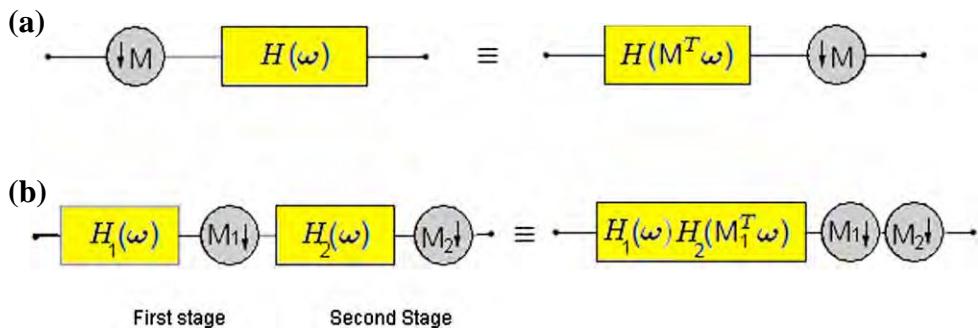
The DFB decompositions in multiple levels are carried out in the following procedures:

- At the first level decomposition, the DFB uses a pair of fan filters with supports complementing each other. The output is resampled using  $Q_1$ . The two fan filters split the Fourier plane into two cone shaped regions: one along the vertical direction, and another one along the horizontal direction. The output is downsampled by two in the  $x$  direction.
- At the second level, the same pair of fan filters are used, however, the output image is re-sampled using  $Q_1$ , so that the four sub images correspond to the four directional sub-bands shown in Figure 9.39. The filter structure for generating these four images is shown in Figure 9.37. The outputs at this level are down-sampled by 2 in both  $x$  and  $y$  directions.
- At the level  $r \geq 3$ , the DFB partitions the Fourier plane into  $2^r$  wedges. Figure 9.40 shows the partition at the level  $r = 3$ . This is realized by attaching to the QFB at the previous level with one of

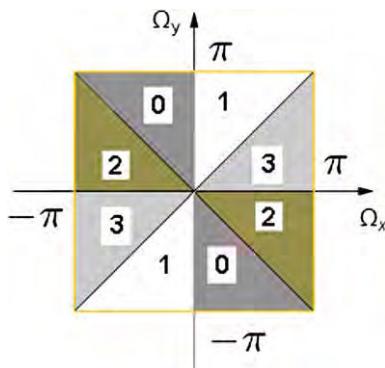


**FIGURE 9.37**

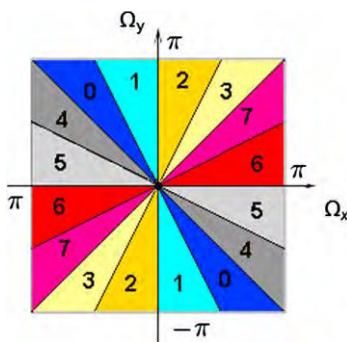
The partition of the Fourier plane using DFB at the first two levels.

**FIGURE 9.38**

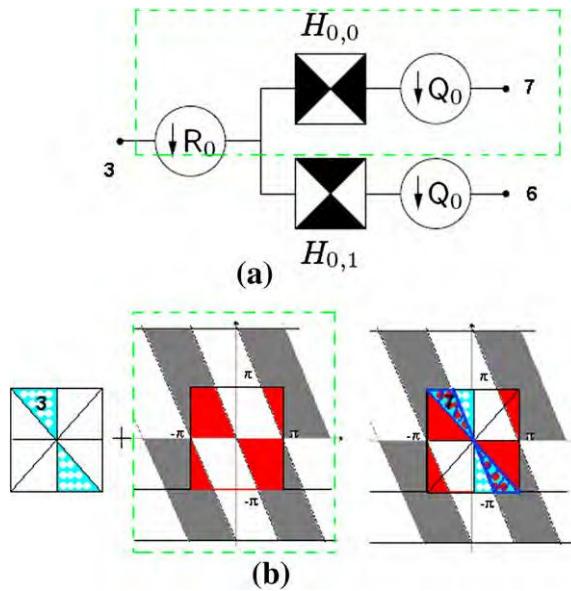
The effect of Noble identity.

**FIGURE 9.39**

The four directional bands output from the first two level partitions.

**FIGURE 9.40**

The eight directional bands output from a DFB at the level  $r = 3$ .

**FIGURE 9.41**

(a) A resampling operator is attaching to a QFB to implement the DFB, (b) Its effects on partitioning the Fourier plane: the 3rd region in the four partitions is used as the input, the output is the wedge indicated in the last picture.

the four uni-modular resampling operations,  $R_i$ ,  $i = 0, 1, 2, 3$ , before and after the analysis filters. Figure 9.41 shows the filter bank structure for generating the 0th wedge in Fourier frequency plane at the 3rd level.

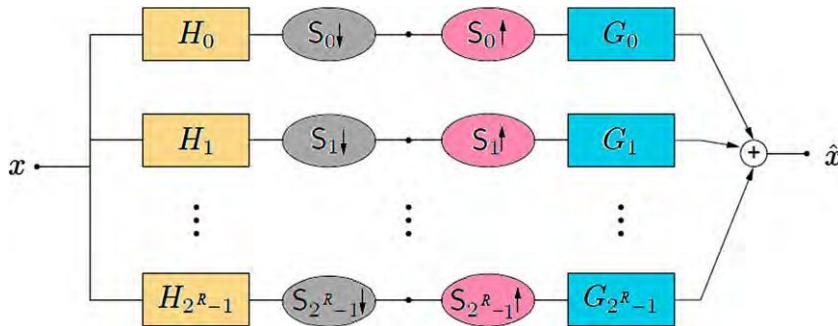
To ensure that each channel of the filters uses a diagonal sampling matrix, a back-sampling operation [115] is applied so that, at each level  $r$ , the  $v$ th channel sub-band can be equivalently viewed as being filtered by an analysis filter  $H_v$ , for  $v = 0, 1, \dots, 2^r - 1$ . The diagonal sampling operator is represented by the sampling matrix

$$S_v^{(r)} = \begin{cases} \text{diag}(2^{r-1}, 2) & 0 \leq v < 2^{r-1}, \\ \text{diag}(2, 2^{r-1}) & 2^{v-1} \leq v < 2^r. \end{cases}$$

The DFB is a perfect reconstruction filter bank if and only if the QFB used at each level implements perfect reconstruction.

#### 1.09.4.3.4 Constructing the basis for the directional filter bank

As shown in Figure 9.42, in the DFB at the  $r$ th level, every analysis filter  $H_v$  with impulse response  $h_v$ , is followed by a diagonal sampling  $S_v^{(r)}$ ,  $v = 0, 1, \dots, 2^r - 1$ . The outputs are  $2^r$  wedges in the Fourier plane. Each wedge is an angularly oriented segment of the original image's Fourier content. Correspondingly, the original image can be reconstructed from applying the diagonal sampling matrix

**FIGURE 9.42**

The analysis and synthesis filters in a DFB that partitions the Fourier plane into  $2^R$  wedges.

$S_v^{(r)}$  to the outputs at the  $r$ th level, followed by applying the synthesis filters  $\{G_v\}_{v=0,1,\dots,2^r-1}$ , with impulse response  $\{g_v(\mathbf{n})\}_{v=0,1,\dots,2^r-1}$ . We also write  $g_v(\mathbf{n}), h_v(\mathbf{n})$  as  $g_v^{(r)}(\mathbf{n}), h_v^{(r)}(\mathbf{n})$  for the  $r$ th level.

Each filter pair, with  $H_v$  and  $G_v$  in the filter bank as the analysis and synthesis filters, is related to the angular wedge region  $U_v$ ,  $v = 0, 1, \dots, 2^r - 1$ . The filters generate a pair of dual frames of  $l^2(\mathbb{Z}^2)$ . Therefore, a basis of  $l^2(\mathbb{Z}^2)$  can be obtained by translating the impulse responses of synthesis filters,  $g_v(\mathbf{n})$ , as  $\left\{ g_{v,\mathbf{m}}^{(r)}(\mathbf{n}) = g_v^{(r)}(\mathbf{n} - S_v^{(r)}\mathbf{m}) : 0 \leq v < 2^r, \mathbf{m} \in \mathbb{Z}^2 \right\}$ . Under this basis, an image  $x(\mathbf{n}) \in l^2(\mathbb{Z}^2)$  can be uniquely reconstructed as

$$x(\mathbf{n}) = \sum_{v=0}^{2^r-1} \sum_{\mathbf{m} \in \mathbb{Z}^2} c_v(\mathbf{m}) g_v^{(r)}(\mathbf{n} - S_v^{(r)}\mathbf{m}),$$

where  $c_v(\mathbf{m}) = \langle x(\cdot), h_v^{(r)}(S_v^{(r)}\mathbf{m} - \cdot) \rangle$ ,  $v = 0, 1, \dots, 2^r - 1$  are the coefficients with respect to a basis  $h_v(S_v^{(r)}\mathbf{m} - \cdot)$  of  $l^2(\mathbb{Z}^2)$  at the  $r$ th DFB decomposition level. The two dual bases  $l^2(\mathbb{Z}^2)$  satisfy the following conditions when they are biorthogonal frames:

$$\langle g_{v'}^{(r)}(\cdot - S_{v'}^{(r)}\mathbf{m}'), h_v^{(r)}(S_v^{(r)}\mathbf{m} - \cdot) \rangle = \delta(v - v')\delta(\mathbf{m} - \mathbf{m}'),$$

with the Dirac function given as

$$\delta(a - a') = \begin{cases} 0, & \text{when } a \neq a', \\ 1, & \text{when } a = a'. \end{cases}$$

An orthogonal basis of  $l^2(\mathbb{Z}^2)$  is obtained when  $h_v^{(r)}(\mathbf{n}) = g_v^{(r)}(-\mathbf{n})$ , that is, when the synthesis filters of the DFB are time-reverses of the analysis filters.

The following are two extreme cases of DFBs. In the first case, the filter system generates a basis of the  $l^2(\mathbb{Z}^2)$  whose elements have compact supports that do not overlap each other in the spatial domain. The second case shows a filter system that generates a basis of  $l^2(\mathbb{Z}^2)$  whose elements have compact supports that do not overlap each other in the Fourier domain.

*Case 1:* In this case, the analysis DFB implements a polyphase decomposition with the  $v$ th polyphase component  $x_v(\mathbf{n}) = x(p_v^T + S_v^{(r)}\mathbf{n})$ , where

$$\mathbf{p}_v = \begin{cases} (v, v) & \text{if } 0 \leq v < 2^{r-1}, \\ (v+1-2^{r-1}, v-2^{r-1}) & \text{if } 2^{r-1} \leq v < 2^r. \end{cases}$$

The polyphase decomposition is realized when each filter  $G_v$  takes the  $z$ -transform as  $G_v(z) = z^{-\mathbf{p}_v}$ , where the basis  $g_v^{(r)}(\mathbf{n} - S_v^{(r)}\mathbf{m}) = \delta(\mathbf{n} - p_v^T - S_v^{(r)}\mathbf{m})$  extends a standard orthonormal basis of  $L^2(\mathcal{Z}^2)$ .

*Case 2:* Another orthonormal basis of  $L^2(\mathcal{Z}^2)$  is obtained using a DFB with ideal fan filters at each level, resulting in the synthesis filters  $G_v$  with Fourier transform given by

$$G_v(\boldsymbol{\omega}) = 2^{v/2} \delta_{R_v}(\boldsymbol{\omega}) \exp \left\{ j \left\langle \boldsymbol{\omega}, \mathbf{p}_v^T \right\rangle \right\}.$$

These two cases, however, will have non-compact support in their dual spaces. Namely, in the first case, the Fourier content of each element in the basis is supported in the whole Fourier plane, while in the second case, each element in the basis is a sinc function with support in the whole spatial domain.

In application, a DFB realization that generates a basis of compact support in the Fourier plane and fast decay in the spatial plane works better to separate a signal into sub-signals that are localized in various frequency regions. In the contourlets [9], a biorthogonal filter pair “9–7” is usually used.

#### 1.09.4.4 The contourlet filter bank

Now we are ready to address the contourlet filter bank, which is a tree structured double layered filter bank that combines the LP and DFB to decompose an original image iteratively. Figure 9.43 shows the multiscale decomposition result generated by the contourlet filter bank. The implementation uses a combination of LP and DFB filter banks.

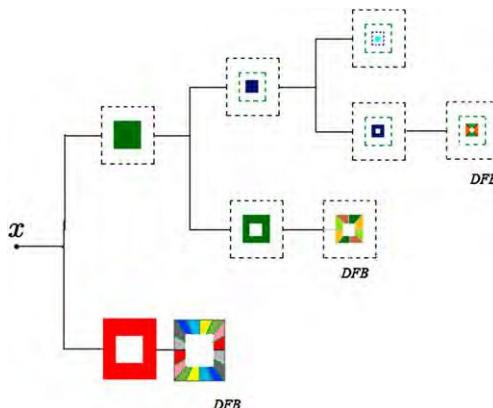
##### 1.09.4.4.1 The wavelet frames of Laplacian filter banks

The LP filters in the contourlet filter bank are octave-band filters: The image is decomposed into a lowpass frequency content, and a bandpass content. The lowpass filtered image is an approximation of the high resolution image with a dyadic downsampling applied in both directions. The bandpass filtered image is the detail difference between the original image and the approximation image. This decomposition proceeds iteratively. At level  $k$ , the original input image is decomposed into several subimages that are focused on sub-bands with a corona support as shown in Figure 9.34, where the Fourier plane of an image is decomposed into three levels.

The LP filter provides a frame expansion in  $L^2(\mathcal{R}^2)$ . Under the multiresolution analysis (MRA), the LP filter bank is associated with a wavelet frame represented by nested subspaces  $\{V_k\}_{k \in \mathcal{Z}}$  (refer to Section 1.09.2) that satisfies

$$V_k \subset V_{k-1}, \quad \text{for } \forall k \in \mathcal{Z},$$

$$\text{Closure} \left( \bigcup_{k=-\infty}^{\infty} V_k \right) = L^2(\mathcal{R}^2).$$

**FIGURE 9.43**

The contourlet filter bank partitions the Fourier plane into wedges for three levels.

The lowpass filter  $G$  in Figure 9.32 defines a wavelet frame of  $L^2(\mathcal{R}^2)$ . The filter structure uniquely defines the function  $\phi(t)$  that satisfies the following equation (called an orthogonal scaling function),

$$\phi(t) = D^{\frac{1}{2}} \sum_{\mathbf{n} \in \mathbb{Z}^2} g(\mathbf{n}) \phi(\mathbf{M}t - \mathbf{n}).$$

Recall that, in general,  $D = |\det(\mathbf{M})|$ , where  $\mathbf{M}$  is a diagonal matrix that corresponds to subsampling. In our LP filter bank introduced here, the subsampling is constrained to a dyadic subsampling only; therefore we have  $D = 4$ , and

$$\mathbf{M} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}.$$

The sub-space  $V_k$  is spanned by an orthogonal basis  $\{\phi_{k,\mathbf{n}}\}_{\mathbf{n} \in \mathbb{Z}^2}$  that is defined as

$$\phi_{k,\mathbf{n}}(t) = D^{-\frac{k}{2}} \phi(\mathbf{M}^{-k}t - \mathbf{n}).$$

Writing the orthogonal complement of  $V_k$  in  $V_{k-1}$  as  $W_k$  (see Section 1.09.2), we have

$$V_{k-1} = V_k \bigoplus W_k.$$

Now let  $\{f_i\}_{i=0,1,2,3}$  represent the impulse responses of the highpass synthesis filters  $\{F_i\}_{i=0,1,2,3}$  in the reconstruction part of the LP polyphase structured filter bank, see Figure 9.77. It generates the mother wavelet function  $\psi_i$  for each  $i = 0, 1, 2, 3$  as the following equation:

$$\psi_i(t) = D^{\frac{1}{2}} \sum_{s \in \mathbb{Z}^2} f_i(s) \phi(\mathbf{M}t - s), \quad t \in \mathcal{R}^2, \quad i = 0, 1, 2, 3.$$

A tight frame of the detail space  $W_k$  can be obtained as a family of dilated and translated functions,  $\psi_{k,i,n}(t)$ , given as

$$\psi_{k,i,n}(t) = D^{-\frac{k}{2}} \psi_i(\mathbf{M}^{-k} t - \mathbf{n}), \quad \text{for } k \in \mathcal{Z}, \mathbf{n} \in \mathcal{Z}^2, \quad t \in \mathcal{R}^2, \quad i = 0, 1, 2, 3. \quad (9.80)$$

Here,  $\psi_{k,i,n}(t)$  corresponds to the  $i$ th polyphase component of LP filter bank at level  $k$ . Under this set of tight frames, given an input image  $x(\mathbf{n})$ , the outputs of the LP filter bank at the  $k$ th level are the approximation wavelet coefficients  $c_k(\mathbf{n})$ , and the detail wavelet coefficients  $d_{k,i}(\mathbf{n})$ , they can be obtained from the following formulas based on the approximation coefficients at the previous level,

$$\begin{aligned} c_k(\mathbf{n}) &= \langle x, \phi_{k,n} \rangle = \sum_{s \in \mathcal{Z}^2} c_{k-1}(s) g(s - \mathbf{M}\mathbf{n}), \\ d_{k,i}(\mathbf{n}) &= \langle x, \psi_{k,i,n} \rangle = \sum_{s \in \mathcal{Z}^2} c_{k-1}(s) f_i(s - \mathbf{M}\mathbf{n}). \end{aligned}$$

**Theorem 5.** [112] It can be shown that  $L^2(\mathcal{R}^2)$  can be decomposed into detail spaces  $\{W_k, k \in \mathcal{Z}\}$  that are mutually orthogonal, and can be written as

$$L^2(\mathcal{R}^2) = \bigoplus_{k \in \mathcal{Z}} W_k.$$

At a scale level  $k$ , the family of functions  $\{\psi_{k,i,n}(t) : 0 \leq i < D, \mathbf{n} \in \mathcal{Z}^2\}$  is a tight frame of  $W_k$ . In addition, the entire family  $\{\psi_{k,i,n}(t) : 0 \leq i < D, k \in \mathcal{Z}, \mathbf{n} \in \mathcal{Z}^2\}$  forms a tight frame of  $L^2(\mathcal{R}^2)$ .

#### 1.09.4.4.2 The wavelet frames of directional filter banks

In a contourlet filter bank, the highpass content output from the LP filter bank is the input for the DFB filter bank. The DFB generates  $2^r$  directional oriented wedges that partition the Fourier plane. Typically, only the high frequency content is decomposed since it corresponds to salient features in an image. The directional oriented filter bank provides an anisotropic basis for an efficient decomposition.

As provided in Section 1.09.4.3.4, under the DFB, at level  $r$ , the family  $\{g_{v,n}^{(r)}(\mathbf{m}) = g_v^{(r)}(\mathbf{m} - \mathbf{S}_v^{(r)}\mathbf{n}) : 0 \leq v < 2^r, \mathbf{n} \in \mathcal{Z}^2\}$  is generated and is called a directional orthonormal basis of  $l^2(\mathcal{Z}^2)$ . Each  $g_{v,n}^{(r)}(\mathbf{m})$  is the impulse response of a directional filter. When  $v = 0, \dots, 2^{r-1} - 1$ , it corresponds to an angle in  $[-45^\circ, 45^\circ]$ , and when  $v = 2^{r-1}, \dots, 2^r - 1$ , it corresponds to an angle in  $[45^\circ, 135^\circ]$ .

This directional orthonormal basis  $\{g_{v,n}^{(r)}(\mathbf{m}) : 0 \leq v < 2^r, \mathbf{n} \in \mathcal{Z}^2\}$  is used to construct the frames of the further decomposed detail space. Since the detail subspaces  $W_k$  from the LP correspond to four components of the polyphase decomposition, the following function is introduced using Eq. (9.80),

$$\mu_{k,2n+s_i}(t) = \psi_{k,i,s}(t),$$

where  $s_0 = (0, 0)$ ,  $s_1 = (1, 0)$ ,  $s_2 = (0, 1)$ ,  $s_3 = (1, 1)$ . With a lattice of dyadic subsampling in both directions, the points  $s_i$ ,  $i = 0, 1, 2, 3$ , represent all the cosets with the dyadic subsampling, therefore, any integer vector  $s$  can be written as  $\mathbf{n} + s_i$ , and we have the family  $\mu_{k,s}(t)$  constituting a tight frame of  $W_k$  at level  $k$  in the LP decomposition.

Now define

$$\psi_{k,v,\mathbf{n}}^{(r)}(t) = \sum_{s \in \mathcal{Z}^2} g_v^{(r)}(s - S_v^{(r)}\mathbf{n}) \mu_{k,s}(t).$$

The set of functions,  $\{\psi_{k,v,\mathbf{n}}^{(r)}\}_{\mathbf{n} \in \mathcal{Z}^2}$ , constitutes a tight frame of the finer partition subspace of the detail space  $W_k$ , written as  $W_{k,v}^{(r)}$ , with  $v$  referring to the  $v$ th angular direction. The frame bounds are equal to 1 for each  $k = 1, \dots, K$ , and  $v = 0, \dots, 2^r - 1$ . Also note that the indexing  $v$  is referred to an angular direction in DFB, which is different from the indexing  $i$ , which refers to a polyphase component in the LP filter.

A directional subspace  $W_{k,v}^{(r)}$  is defined on a wedge contained in a rectangular grid that has support in a spatial domain as an oval area, similar to what is shown in Figure 9.26. These subspaces are orthogonal with

$$W_{k,v}^{(r)} = W_{k-1,2v}^{(r+1)} \bigoplus W_{k-1,2v+1}^{(r+1)},$$

and

$$W_k = \bigoplus_{v=0}^{2^r-1} W_{k,v}^{(r)}.$$

Further, combining  $\psi_{k,v,\mathbf{n}}^{(r)}(t)$  together with  $\phi_{k_0,\mathbf{n}}$  for  $k \leq k_0$ ,  $0 \leq v < 2^r$ ,  $\mathbf{n} \in \mathcal{Z}^2$ , we obtain a tight frame of  $l^2(\mathcal{R}^2)$ , as in the following Theorem [114],

**Theorem 6.** *The  $l^2(\mathcal{R}^2)$  can be decomposed into mutually orthogonal subspaces as*

$$l^2(\mathcal{R}^2) = V_{k_0} \bigoplus \left( \bigoplus_{k \leq k_0} W_k \right).$$

*For a sequence of finite positive integers  $k \leq k_0$ , the family*

$$\{\phi_{k_0,\mathbf{n}}(t), \psi_{k,v,\mathbf{n}}^{(r)}(t) : k \leq k_0, 0 \leq v < 2^r, \mathbf{n} \in \mathcal{Z}^2\}$$

*is a tight frame of  $L^2(\mathcal{R}^2)$ .*

**Theorem 7.**  *$l^2(\mathcal{R}^2)$  can be decomposed into mutually orthogonal subspaces as*

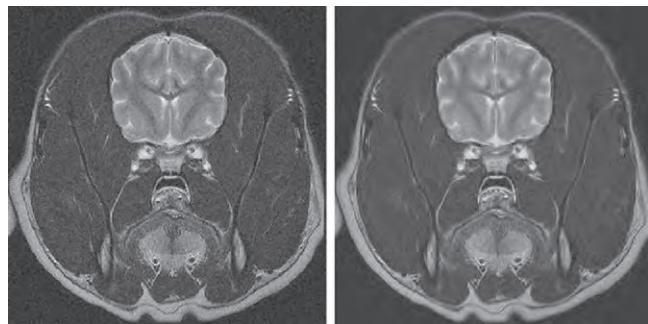
$$l^2(\mathcal{R}^2) = \bigoplus_{k \in \mathcal{Z}} W_k.$$

*The family*

$$\left\{ \psi_{k,v,\mathbf{n}}^{(r)}(t) : k \in \mathcal{Z}, 0 \leq v \leq 2^r - 1, \mathbf{n} \in \mathcal{Z}^2 \right\}$$

*is a directional wavelet tight frame of  $L^2(\mathcal{R}^2)$ . In each case, the frame bounds are equal to 1.*

This combination of frames from both the LP and DFB filter banks yields frequency partitions similar to those achieved by curvelets. The support of the LP is reduced to one fourth of its original size while



**FIGURE 9.44**

Shown in the left is a noisy image, in the right is the image after applying the hard threshold denoising method using the contourlets.

the number of directions of the DFB is doubled. Hence, it connects the directional anisotropic wavelets with the digital filter bank system.

In Contourlet MATLAB program [116], the syntax that is used for decomposition is:

```
coeffs = pdfbdec(double(im), pfilter, dfilter, nlevels);
```

The syntax that is used for reconstruction the original image from the contourlet coefficients is:

```
imrec = pdfbrec(coeffs, pfilter, dfilter);
```

Since the contourlets can be viewed as a filter bank implementation of the curvelets, here we only show in Figure 9.44 an image that demonstrates the denoising outcome using the contourlets.

## 1.09.5 Shearlets and their applications

The shearlet transform is an alternative anisotropic multiresolution system proposed by Guo, Labate and their colleagues [11–13, 117–124]. It was inspired by the composite wavelet theory, which takes advantage of geometric affine transforms, such as translation and dilation (or resampling) in a multi-dimensional Cartesian coordinate system (our focus is on 2-D). In the spatial plane, the affine transforms generate a frame of functional elements (also called building blocks) that are oscillatory waveforms across various scales, locations, and orientations.

Similar to the curvelet transform, a shearlet transform provides a Parseval (normalized tight) frame equipped with well localized functions at various scales, locations and directions. When a 2-D function is projected into this system, sparse significant coefficients, which correspond to discontinuities occur along  $C^2$  piecewise smooth curves, can be used to optimally approximate 2-D functions. It overcomes the problem with the classical 2-D wavelet transform, in which a larger amount of coefficients are needed in order to optimally represent a 2-D function with discontinuities.

Meanwhile, the shearlet transform has its advantages over the curvelet transform. One advantage with the continuous shearlet transform is that, it can be defined directly in a rectangular coordinate system using an affine transform. It avoids the usage of polar coordinates (as curvelets) and hence avoid the computationally complex numerical conversion from polar coordinates to rectangular coordinates. The second advantage with the shearlet transform is that, it exhibits a better capability at distinguishing different types of corner (junction) points. This can be used to separate corner (junction) points from the regular edge points. In addition, shearlet transform can be mathematically related to group theoretic methods, and can hence be interpreted as a natural building blocks in measuring the smoothness of a 2-D function. It can also be related to the so-called co-orbit space theory [125].

In this subsection, we first describe the composite wavelets [10, 126] before we introduce the shearlets, since the shearlets be viewed as a special class of the composite wavelets. Applications of the shearlet transform in image processing enjoys some interesting properties in applications in edge detection, feature extraction, and denoising. We will also discuss more recent applications at the end of this section.

### 1.09.5.1 The composite wavelets

The composite wavelet system provides a general framework with angularly oriented oscillatory spatial waveforms that can be used for representing geometrical features of curves. Since it heavily relies on the affine transform, we proceed by first introducing the affine transform system,  $\mathcal{H}$ , given as the following:

**Definition 13.** Given a function  $\psi(\mathbf{x}) \in L^2(\mathbb{R}^2)$ , a continuous affine system,  $\mathcal{H}$ , generated by  $\psi(\mathbf{x})$  is defined as

$$\mathcal{H} = \left\{ \psi_{\mathbf{G}, \mathbf{b}}(\mathbf{x}) = |\det \mathbf{G}|^{1/2} \psi(\mathbf{G}(\mathbf{x} - \mathbf{b})) : \mathbf{b}, \mathbf{x} \in \mathbb{R}^2, \mathbf{G} \in \mathcal{G} \right\}, \quad (9.81)$$

where  $\mathcal{G}$  is a set of invertible  $2 \times 2$  matrices.

All elements in this affine system,  $\mathcal{H}$ , are called composite wavelets when this affine system forms a tight (Parseval) frame of  $L^2(\mathbb{R}^2)$ , namely, for any  $f \in L^2(\mathbb{R}^2)$ , the following equation holds:

$$\sum_{\mathbf{b}, \mathbf{G}} |\langle f, \psi_{\mathbf{G}, \mathbf{b}} \rangle|^2 = \|f\|^2, \quad (9.82)$$

where  $\mathbf{G}$  is an invertible  $2 \times 2$  matrix in the set  $\mathcal{G}$ .

Under this general framwork of composite wavelets, various wavelet systems can be introduced as special cases for decomposing an image. The 2-D isotropic continuous (the classical) wavelet transform is a special example. It can be obtained when  $\mathcal{G}$ , the set of matrices in Eq. (9.81), takes the form  $\mathcal{G} = \{c\mathbf{I} : c > 0\}$ , here  $\mathbf{I}$  is the unit matrix. Furthermore, the separable classical wavelets can also be obtained when we further use a 2-D separable function  $\psi(\mathbf{x})$ , given as  $\psi(x, y) = \psi_1(x)\psi_2(y)$ . The most exciting case of composite wavelets is the shearlet transform system obtained from using the shearing and translating operations. A shearlet transform system defines a new class of wavelets that produces a set of efficient anisotropic components that can flexibly be adapted to represent the geometrical shapes.

### 1.09.5.2 The shearlet atoms in the spatial plane

The shearlet atoms are defined by selecting the matrix set  $\mathcal{G}$  in Eq. (9.81), in the affine system,  $\mathcal{H}$ , as  $\mathcal{G} = \{\mathbf{M}_{as} : a > 0, s \in \mathcal{R}\}$ , with  $\mathbf{M}_{as}$  given as

$$\mathbf{M}_{as} = \begin{pmatrix} a & -\sqrt{as} \\ 0 & \sqrt{a} \end{pmatrix}. \quad (9.83)$$

It is useful to write the matrix  $\mathbf{M}_{as} = \mathbf{B}_s \mathbf{A}_a$ , where

$$\mathbf{A}_a = \begin{pmatrix} a & 0 \\ 0 & \sqrt{a} \end{pmatrix} \quad \text{and} \quad \mathbf{B}_s = \begin{pmatrix} 1 & -s \\ 0 & 1 \end{pmatrix}.$$

Note that the inverse matrix  $\mathbf{M}_{as}^{-1}$  can be written as  $\mathbf{A}_a^{-1} \mathbf{B}_s^{-1}$ , and is given as

$$\mathbf{M}_{as}^{-1} = \begin{pmatrix} a^{-1} & sa^{-1} \\ 0 & a^{-1/2} \end{pmatrix}. \quad (9.84)$$

Indeed, the matrix  $\mathbf{M}_{as}$  is the composition of two affine operations in a plane: a parabolic scaling,  $\mathbf{A}_a$ , followed by a shearing,  $\mathbf{B}_s$ . The shearlets are defined on the basis of the operator,  $D_{\mathbf{M}_{as}}$ , defined by the Matrix  $\mathbf{M}_{as}$ , as

$$D_{\mathbf{M}_{as}} \psi(\mathbf{x}) = |\det(\mathbf{M}_{as})|^{-\frac{1}{2}} \psi(\mathbf{M}_{as}^{-1} \mathbf{x}),$$

and the translation operator  $T_b$ , defined as

$$T_b \psi(\mathbf{x}) = (\mathbf{x} - \mathbf{b}),$$

yielding the shearlet atoms defined as

$$\psi_{a,s,b}(\mathbf{x}) = T_b D_{\mathbf{M}_{as}} \psi(\mathbf{x}). \quad (9.85)$$

**Definition 14.** The family of shearlet atoms in the spatial plane is defined as the collection of all atoms in the affine system generated by  $\psi(\mathbf{x}) \in L^2(\mathcal{R}^2)$ , given by

$$\psi_{a,s,b}(\mathbf{x}) = |\det(\mathbf{M}_{as})|^{-\frac{1}{2}} \psi(\mathbf{M}_{as}^{-1}(\mathbf{x} - \mathbf{b})), \quad a > 0, \quad s \in \mathcal{R}, \quad \mathbf{x}, \mathbf{b} \in \mathcal{R}^2. \quad (9.86)$$

The function  $\psi(\mathbf{x}) \in L^2(\mathcal{R}^2)$  is called the mother shearlet window.

Note that, because  $\det(\mathbf{M}_{as}) = a^{3/2}$ , Eq. (9.86) can also be written as

$$\psi_{a,s,b}(\mathbf{x}) = a^{-\frac{3}{4}} \psi(\mathbf{M}_{as}^{-1}(\mathbf{x} - \mathbf{b})). \quad (9.87)$$

In the literature about shearlets, the matrix  $\mathbf{M}_{as}$  can also be chosen as

$$\mathbf{M}_{as} = \begin{pmatrix} a & \sqrt{as} \\ 0 & \sqrt{a} \end{pmatrix}, \quad \text{with} \quad \mathbf{M}_{as}^{-1} = \begin{pmatrix} a^{-1} & -sa^{-1} \\ 0 & a^{-1/2} \end{pmatrix}.$$

In this paper, we stick with the matrix  $\mathbf{M}_{as}$ , given by (9.83), for Eq. (9.86) to achieve the shearing effects. Hence, the inverse matrix,  $\mathbf{M}_{as}^{-1}$ , is used in its Fourier domain definition; however, one can alternatively use  $\mathbf{M}_{as}^{-1}$  in the spatial domain, and use  $\mathbf{M}_{as}$  in the Fourier plane [119].

### 1.09.5.3 The shearlet atoms in the frequency domain

The mother shearlet window defines the properties of the shearlet atoms. Every shearlet atom is a result of an affine transform of the mother shearlet window. Hence, we first discuss the choice of the mother shearlet window in order to quickly address the geometrical features of an image using a shearlet transform.

#### 1.09.5.3.1 The mother shearlet window

The rule of thumb for selecting a mother shearlet window in a shearlet transform, is to generate shearlet atoms that are able to focus on localized frequency content to represent geometrical features. Under this shearlet system, the energy of a function  $f(\mathbf{x})$  will be mostly focused on a sparse number of coefficients, when the function is decomposed. In general, the mother shearlet window is chosen as a  $C^\infty$  function with a compact support in the Fourier plane, so that it decays rapidly in the spatial domain. The mother shearlet window function,  $\psi(x, y)$ , is defined by its Fourier transform,  $\Psi(\Omega)$ ,  $\Omega = (\Omega_x, \Omega_y)$ , as

$$\Psi(\Omega) = \Psi_1(\Omega_x)\Psi_2\left(\frac{\Omega_y}{\Omega_x}\right),$$

where  $\Psi_1(\Omega_x)$  and  $\Psi_2(\Omega_y)$  satisfy the following two conditions:

- $\Psi_1(\Omega_x)$  is a 1-D highpass odd function that satisfies the regular 1-D admissibility condition,

$$\int_0^\infty |\Psi_1(a\Omega_x)|^2 \frac{da}{a} = 1. \quad (9.88)$$

- $\Psi_2(\Omega_y)$  is a 1-D positive lowpass even function that satisfies,

$$\int_{\mathcal{R}} |\Psi_2^2(\Omega_y)| d\Omega_y = 1. \quad (9.89)$$

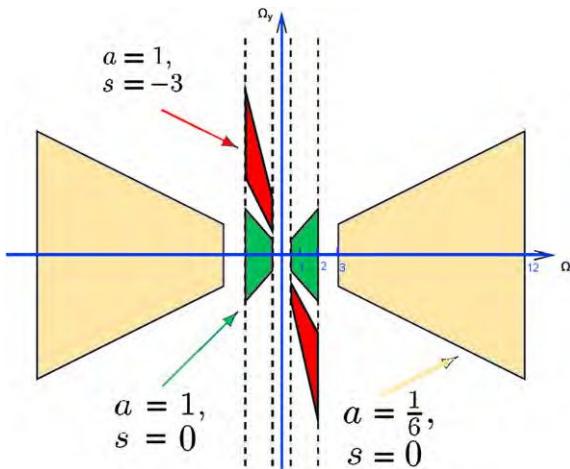
A good choice for the two 1-D windows are the Meyer's wavelet windows defined in Section 1.09.3.

#### 1.09.5.3.2 The Fourier content of shearlet atoms

Given the mother shearlet window defined above, the Fourier transform of the shearlet atoms defined in Eq. (9.86) can be written as

$$\begin{aligned} \Psi_{a,s,\mathbf{b}}(\Omega) &= a^{\frac{3}{4}} e^{-j2\pi\langle\Omega, \mathbf{b}\rangle} \Psi\left(\mathbf{M}_{as}^T \Omega\right) \\ &= a^{\frac{3}{4}} e^{-j2\pi\langle\Omega, \mathbf{b}\rangle} \Psi_1(a\Omega_x) \Psi_2\left(\frac{\sqrt{a}(\Omega_y - s\Omega_x)}{a\Omega_x}\right) \\ &= a^{\frac{3}{4}} e^{-j2\pi\langle\Omega, \mathbf{b}\rangle} \Psi_1(a\Omega_x) \Psi_2\left(a^{-\frac{1}{2}}\left(\frac{\Omega_y}{\Omega_x} - s\right)\right), \quad a > 0, \quad s \in \mathcal{R}, \quad \mathbf{b} \in \mathcal{R}^2. \end{aligned} \quad (9.90)$$

In addition to requiring that the functions  $\Psi_1$  and  $\Psi_2$  satisfy the conditions in Eqs. (9.88) and (9.89), we also assume  $\text{supp}\{\Psi_1\} \subset [-2, -\frac{1}{2}] \cup [\frac{1}{2}, 2]$  and  $\text{supp}\{\Psi_2\} \subset [-1, 1]$ . The frequency content of each shearlet atom, which is yielded from the mother shearlet window, is supported on a pair of trapezoids that are symmetric with respect to the origin point. They are centered between the two lines,  $\Omega_y = m\Omega_x$ , with slopes  $m = s \pm \sqrt{a}$ . We refer to  $s$  as the orientation of this pair of trapezoids. Figure 9.45 shows

**FIGURE 9.45**

The frequency supports of some shearlet atoms in the Fourier plane,  $a$  is the scale,  $s$  is the orientation of a pair of trapezoids.

some examples of the trapezoid pairs. Each supporting region may be written as

$$\left\{ (\Omega_x, \Omega_y) \in \mathcal{R}^2 : \Omega_x \in \left[ -\frac{2}{a}, -\frac{1}{2a} \right] \cup \left[ \frac{1}{2a}, \frac{2}{a} \right], -\sqrt{a} + s \leq \frac{\Omega_y}{\Omega_x} \leq \sqrt{a} + s \right\}.$$

So far, the mother shearlet window generates real-valued spatial shearlet atoms; however, the complex valued spatial shearlet atoms can also be obtained if the support of  $\Psi_1$  is confined to  $\left[ \frac{1}{2}, 2 \right]$ . In both cases, the mother shearlet reproduces the shearlet atoms that are angularly oriented waveforms in the spatial domain, with its shape controlled by the shearing parameter  $s$  and scaling parameter  $a$ , see Figure 9.46.

#### 1.09.5.4 The continuous shearlet transform

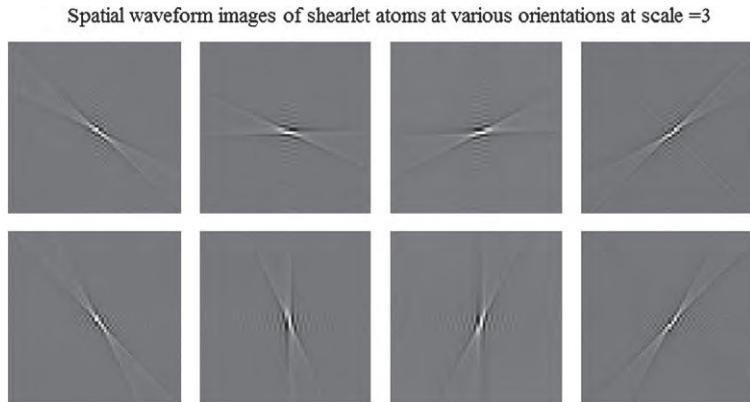
**Definition 15.** Let  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$ , and let  $\{\psi_{a,s,b}(\mathbf{x}) : a > 0, s \in \mathcal{R}, \mathbf{b} \in \mathcal{R}^2\}$  be the family of shearlet atoms defined in Eq. (9.86), the continuous shearlet transform is defined as

$$\gamma_f(a, s, \mathbf{b}) = \langle f, \psi_{a,s,b} \rangle = \int_{\mathcal{R}^2} f(\mathbf{x}) \psi_{a,s,b}^*(\mathbf{x}) d\mathbf{x}, \quad (9.91)$$

where  $\{\gamma_f(a, s, \mathbf{b})\}_{a>0, s \in \mathcal{R}, \mathbf{b} \in \mathcal{R}^2}$  are called the shearlet coefficients of the function  $f(\mathbf{x})$ .

Note that the shearlet coefficients of the function  $f(\mathbf{x})$  can also be obtained in the Fourier domain as

$$\begin{aligned} \gamma_f(a, s, \mathbf{b}) &= \langle F, \Psi_{a,s,b} \rangle \\ &= \int_{\mathcal{R}^2} F(\boldsymbol{\Omega}) \Psi_{a,s,b}^*(\boldsymbol{\Omega}) d\boldsymbol{\Omega} \end{aligned}$$

**FIGURE 9.46**

The eight spatial waveform images of shearlet atoms correspond to different orientations at scale = 3 are shown here. Note that the images are zoomed to show only the center portions of the images.

$$= a^{3/4} \int_{\mathcal{R}^2} F(\Omega) \Psi_1(a\Omega_x) \Psi_2 \left( a^{-\frac{1}{2}} \left( \frac{\Omega_y}{\Omega_x} - s \right) \right) e^{j2\pi \langle \Omega, b \rangle} d\Omega. \quad (9.92)$$

It can be proved that the mother shearlet window function given in the previous section satisfies the following admissibility condition,

$$\int_{\mathcal{R}} \int_0^\infty |\Psi(M_{as}^T \Omega)|^2 a^{-3/2} da ds = 1 \quad \text{for a.e. } \Omega \in \mathcal{R}^2.$$

This admissibility condition shows that the family of shearlets generates a reproducing system, which yields the following theorem,

**Theorem 8.** *For a function  $f \in L^2(\mathcal{R}^2)$ , the exact reproduction can be obtained by the following generalized Calderón reproducing formula:*

$$f(\mathbf{x}) = \int_{\mathcal{R}^2} \int_{-\infty}^\infty \int_0^\infty \gamma_f(a, s, \mathbf{b}) \psi_{a,s,\mathbf{b}}(\mathbf{x}) \frac{da}{a^3} ds d\mathbf{b}. \quad (9.93)$$

In addition, we have

$$\|f(\mathbf{x})\|^2 = \int_{\mathcal{R}^2} \int_{-\infty}^\infty \int_0^\infty |\gamma_f(a, s, \mathbf{b})|^2 \frac{da}{a^3} ds d\mathbf{b}. \quad (9.94)$$

#### 1.09.5.4.1 Properties of the shearlet transform

A 2-D function  $f$  that is smooth away from discontinuities along a curve  $\Gamma$  can be modeled as an image in  $[0, 1]^2$  showing an object and a boundary, written as,  $O \cup \Gamma = \cup_{n=1}^L (O_n \cup C_n)$ , namely,  $L$  small

objects  $O_n, n = 1, 2, \dots, L$  form a connected open set  $O$  in the image with a boundary,  $\Gamma$ , connected by piecewise smooth curves of finite length, with finite number of corner (junction) points.

As  $a \rightarrow 0$ , a shearlet atom has a slim support in the spatial domain, see Figure 9.46. If this slim support is centered at a point on a edge curve and is aligned along the normal direction of the curve, it results in slow decay or large value shearlet coefficients.

The following conclusions demonstrate that the location and orientation of singularities can be well characterized by the decay rate of continuous shearlet coefficients, which can be used to further classify the geometrical features at singularity points [121, 122, 127].

- If  $\mathbf{b} \notin \Gamma$ , then

$$\lim_{a \rightarrow 0+} a^{-N} \gamma_f(a, s, \mathbf{b}) = 0, \quad \text{for } \forall s \in \mathcal{R}, \forall N \in \mathcal{N},$$

namely, the continuous shearlet coefficients satisfy

$$|\gamma_f(a, s, \mathbf{b})| \leq \text{Const.} a^N, \quad \forall N \in \mathcal{N}, \quad \text{when } a \rightarrow 0+.$$

- If  $\mathbf{b} \in \Gamma$  is a regular edge point, namely, the curve is at least  $C^2$  smooth near  $\mathbf{b}$ , but  $s$  is not at the normal direction of the curve at  $\mathbf{b}$  then we have

$$\lim_{a \rightarrow 0+} a^{-N} \gamma_f(a, s, \mathbf{b}) = 0,$$

namely,  $\gamma_f(a, s, \mathbf{b})$  decays rapidly to 0 as  $a \rightarrow 0$ .

- If  $\mathbf{b} \in \Gamma$  is a regular edge point, namely, the curve is at least  $C^2$  smooth near  $\mathbf{b}$ , and  $s$  is at the normal direction of the curve at  $\mathbf{b}$ ; that is,  $s$  is at the orientation that is perpendicular to  $\Gamma$  at  $\mathbf{b}$ , we have

$$\lim_{a \rightarrow 0+} a^{-3/4} \gamma_f(a, s, \mathbf{b}) = \text{Const.} f_b,$$

where  $f_b$  is the jump of a function  $f$  at  $\mathbf{b}$  along the normal direction.

In regard to the case that  $\mathbf{b} \in \Gamma$  is a corner (junction) point, the following results follow:

- If  $\mathbf{b} \in \Gamma$  is a corner (junction) point, that is, more than two curves merged together at  $\mathbf{b}$ , and the curves are not smooth at  $\mathbf{b}$ , and if  $s$  corresponds to one of the normal directions of the curves, we have

$$0 < \lim_{a \rightarrow 0+} a^{-3/4} \gamma_f(a, s, \mathbf{b}) < +\infty.$$

Namely,  $\gamma_f(a, s, \mathbf{b})$  decays as  $a^{3/4}$ , as  $a \rightarrow 0$ .

- If  $\mathbf{b} \in \Gamma$  is a corner (junction) point, but  $s$  does not correspond to any of the normal directions of the curves, the shearlet transform decays as  $a^{9/4}$  except in some special case. Note that, in the corner points, as  $a \rightarrow 0+$ , we will have  $\gamma_f(a, s, \mathbf{b})$  decays as  $a^{3/4}$  in two or more different normal directions at  $\mathbf{b}$  when two or more curves are merged together at  $\mathbf{b}$ .
- A spike-type singularity point has a different behavior for the decay of the shearlet coefficients. For example, for a Dirac delta function centered at  $\mathbf{b}_0$ , we have  $|\gamma_f(a, s, \mathbf{b})| \asymp a^{-3/4}$ , as  $a \rightarrow 0$  for all  $s$  when  $\mathbf{b} = \mathbf{b}_0$ . Therefore, the shearlet coefficients actually grow at fine scales for  $\mathbf{b} = \mathbf{b}_0$ . This is in contrast to the case that  $|\gamma_f(a, s, \mathbf{b})|$  rapidly decays to zero if  $\mathbf{b}$  is a regular edge point.

The Lipschitz regularity is commonly used to describe the local regularity of a function. In shearlet theory, it is shown [121] that the decay rate of the shearlet coefficients of a function  $f$  depends on the local regularity of  $f$ . This further says that amplitudes of the shearlet coefficients are useful to describe the regularity points of a curve when the curve can be measured by Lipschitz regularity. If inequality (9.3) holds for all  $\mathbf{b} \in O$  with constant  $K > 0$  that is independent of  $\mathbf{b}_0$ , then the function  $f$  is uniformly Lipschitz  $\alpha$  over an open set  $O$ .

The following result [121] provides us with a further insight of the shearlet coefficients. It takes advantage of the fact that the mother shearlet has an infinite number of vanishing moments, namely

$$\int x^k \psi(x, y) dx = 0, \quad \text{for } k = 0, 1, \dots$$

**Theorem 9.** *If function  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$  is Lipschitz- $\alpha$ ,  $\alpha > 0$  near a point  $\mathbf{b}_0$ , we will have*

$$|\gamma_f(a, s, \mathbf{b})| \leq \text{Const.} a^{\frac{1}{2}(\alpha + \frac{3}{2})} \left( 1 + |a^{-1/2}(\mathbf{b} - \mathbf{b}_0)| \right), \quad \text{for } a < 1.$$

#### 1.09.5.4.2 Shearlets in the horizontal and vertical directions

In this section, we explain that the family of shearlet atoms can be split into two families, one in the horizontal direction, another one in the vertical direction. The separation will ease the numerical implementation of the shearlet transform because the non-uniform angular covering of the frequency plane becomes a problem when the slope of a trapezoid is too steep, namely, when  $|s|$  takes a large value, see Figure 9.45.

The following proposition ensures that reproducing a function  $f \in L^2(\mathcal{R}^2)$  using shearlets can be separated into reconstruction under three systems: two (horizontal and vertical) shearlet systems, plus a coarse scale isotropic system. First, we introduce the horizontal shearlet family, which can be defined by restricting the value of  $a$  and  $s$  to  $0 < a < 1$ ,  $-2 \leq s \leq 2$ . The shearlets are focused on the horizontal direction zone  $C^{(h)}$  defined as

$$C^{(h)} = \left\{ (\Omega_x, \Omega_y) \in \mathcal{R}^2 : |\Omega_x| \geq 2 \text{ and } \left| \frac{\Omega_y}{\Omega_x} \right| \leq 1 \right\}.$$

**Proposition 7.** *Any function  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$  that has its support in the horizontal zone  $C^{(h)}$  can be exactly reconstructed by the following reproducing formula:*

$$f(\mathbf{x}) = \int_{\mathcal{R}^2} \int_{-2}^2 \int_0^1 \gamma_f(a, s, \mathbf{b}) \psi_{a,s,\mathbf{b}}(\mathbf{x}) \frac{da}{a^3} ds d\mathbf{b}, \quad (9.95)$$

assume the mother shearlet  $\psi(\mathbf{x})$  satisfies the conditions given in Section 1.09.5.3.1, and the shearlet atoms,  $\psi_{a,s,\mathbf{b}}(\mathbf{x})$ , are defined as in Eq. (9.86).

The proof of this proposition can be found in [120]. The overall requirement for  $s$  is to indeed satisfy  $-\sqrt{a} - 1 < s < \sqrt{a} + 1$ . This proposition can be extended to show that any function  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$  with support in the vertical zone defined as,

$$C^{(v)} = \left\{ (\Omega_x, \Omega_y) \in \mathcal{R}^2 : |\Omega_x| \geq 2 \text{ and } \left| \frac{\Omega_y}{\Omega_x} \right| > 1 \right\},$$

can be exactly reproduced using the shearlet system defined by Eq. (9.86); however, the matrix  $\mathbf{M}_{as}$  should be replaced by  $\mathbf{M}_{as}^T$ , and the mother shearlet window be replaced by  $\psi^{(v)}$  with its Fourier transform,  $\Psi^{(v)}(\Omega)$ , being defined as

$$\Psi^{(v)}(\Omega) = \Psi_1(\Omega_y)\Psi_2\left(\frac{\Omega_x}{\Omega_y}\right).$$

To distinguish the mother shearlet functions for the vertical/horizontal zones, we write the one used in generating the horizontal shearlets as  $\psi^{(h)}(\mathbf{x})$  ( $\psi^{(v)}(\mathbf{x})$  respectively). In addition, we can also define an isotropic function  $\phi(\mathbf{x})$  as a  $C^\infty$  window function in the spatial plane  $\mathcal{R}^2$ , its Fourier transform is defined as  $\Phi(\Omega) = 1$  for  $\Omega \in [-1/2, 1/2]^2$ ,  $\Phi(\Omega) = 0$  for  $\Omega \notin [-2, 2]^2$ . This function,  $\Phi$ , can be called the scaling function, or the father shearlet function.

Now, any function  $f(\mathbf{x}) \in L^2(\mathcal{R}^2)$  can be reconstructed using these three systems introduced above as

$$\begin{aligned} f(\mathbf{x}) &= \int_{\mathcal{R}^2} \langle f(\cdot), \phi(\cdot - \mathbf{b}) \rangle \phi(\mathbf{x} - \mathbf{b}) d\mathbf{b} \\ &\quad + \int_{\mathcal{R}^2} \int_{-2}^2 \int_0^1 \gamma_f^{(h)}(a, s, \mathbf{b}) \psi_{a,s,\mathbf{b}}^{(h)}(\mathbf{x}) \frac{da}{a^3} d\mathbf{b} ds \\ &\quad + \int_{\mathcal{R}^2} \int_{-2}^2 \int_0^1 \gamma_f^{(v)}(a, s, \mathbf{b}) \psi_{a,s,\mathbf{b}}^{(v)}(\mathbf{x}) \frac{da}{a^3} d\mathbf{b} ds. \end{aligned} \quad (9.96)$$

Typically, in finding the salient features of an image, only the high frequency content of a function in  $L^2(\mathcal{R}^2)$  matters, and it corresponds to the curvelet atoms in the horizontal and vertical cones, written as  $\gamma_f^{(h)}(a, s, \mathbf{b})$  and  $\gamma_f^{(v)}(a, s, \mathbf{b})$ . In general, we refer to the following two horizontal/vertical cones after we separate the lowpass content from the bandpass content.

1. The horizontal cone,

$$\mathcal{D}^{(h)} = \left\{ (\Omega_x, \Omega_y) : \left| \frac{\Omega_y}{\Omega_x} \right| \leq 1 \right\}.$$

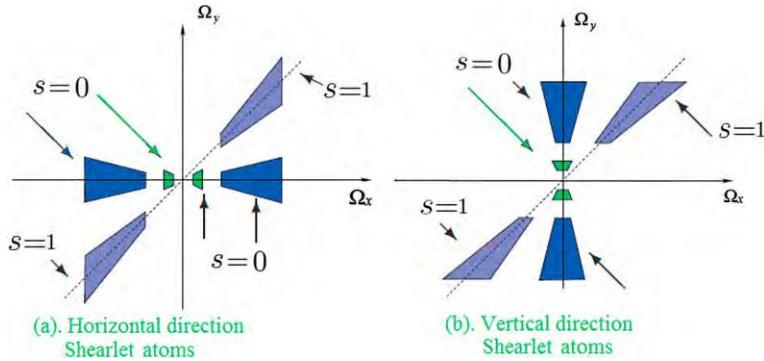
2. The vertical cone,

$$\mathcal{D}^{(v)} = \left\{ (\Omega_x, \Omega_y) : \left| \frac{\Omega_x}{\Omega_y} \right| \geq 1 \right\}.$$

Separating shearlet atoms into two families that cover the horizontal or vertical directions has made it easier for numerical implementation of the shearlet transform. Indeed, each “vertical” direction shearlet atom can be obtained by rotating the corresponding “horizontal” shearlet atom by  $\frac{\pi}{2}$ . Figure 9.47 shows some examples of horizontal and vertical shearlet atoms for  $s = 0, s = 1$  and some  $a$ . In the following, we focus on the numerical implementation of shearlets in these two directions.

### 1.09.5.5 The discrete shearlet atoms

Several discretization methods may be chosen to generate the discrete shearlet atoms along the horizontal cone and the vertical cone directions. We discretize the continuous shearlet transform by sampling the

**FIGURE 9.47**

The frequency support of some horizontal and vertical shearlet atoms in the Fourier plane at two different scales.

scale, shear and translation parameters as follows [117]: the scale parameter is sampled at  $a = 2^{-2k}$ ,  $k \geq 0$ ; the shearing parameter is sampled as  $s_{k,l} = l2^{-k}$ ,  $-2^k \leq l < 2^k$ ; and the translation parameter is sampled at  $\mathbf{b} = A_2^k B_l \mathbf{m}$ , with  $\mathbf{m} \in \mathcal{Z}^2$ , and

$$A_2^k = \begin{pmatrix} 2^{-2k} & 0 \\ 0 & 2^{-k} \end{pmatrix} \quad \text{and} \quad \mathbf{B}_l = \begin{pmatrix} 1 & -l \\ 0 & 1 \end{pmatrix}.$$

After sampling, Eq. (9.87) is rewritten as

$$\psi_{k,l,\mathbf{m}}(\mathbf{x}) = 2^{3k/2} \psi \left( B_l^{-1} A_2^{-k} (\mathbf{x} - A_2^k B_l \mathbf{m}) \right). \quad (9.97)$$

The following equation leads to the discretization method that we introduce in this section,

$$\left\{ T_{A_2^k B_l \mathbf{b}} \right\} D_{A_2^k B_l} \psi(\mathbf{x}) = D_{A_2^k B_l} T_{\mathbf{b}} \psi(\mathbf{x}), \quad \mathbf{x} \in \mathcal{R}^2.$$

The discrete shearlet atoms in the horizontal direction are therefore given as the following set,

$$\left\{ \psi_{k,l,\mathbf{m}}^{(h)}(\mathbf{x}) = D_{A_2^k B_l} T_{\mathbf{b}} \psi^{(h)}(\mathbf{x}) : k \geq 0, -2^k \leq l < 2^k, \mathbf{m} \in \mathcal{Z}^2 \right\}.$$

The discrete shearlet atoms can be further written as,

$$\psi_{k,l,\mathbf{m}}^{(h)}(\mathbf{x}) = 2^{3k/2} \psi^{(h)} \left( \mathbf{B}_l^{-1} A_2^{-k} \mathbf{x} - \mathbf{m} \right), \quad (9.98)$$

where

$$\mathbf{B}_l^{-1} A_2^{-k} = \begin{pmatrix} 2^{2k} & l2^k \\ 0 & 2^k \end{pmatrix} \quad \text{and} \quad A_2^k \mathbf{B}_l = \begin{pmatrix} 2^{-2k} & -l2^{-2k} \\ 0 & 2^{-k} \end{pmatrix}.$$

The Fourier transform of  $\psi_{k,l,m}^{(h)}(\mathbf{x})$  can be written as

$$\begin{aligned}\Psi_{k,l,m}^{(h)}(\Omega) &= 2^{-3k/2} \Psi^{(h)} \left( \left( A_2^k \mathbf{B}_l \right)^T \Omega \right) e^{-j2\pi \langle A_2^k \mathbf{B}_l m, \Omega \rangle} \\ &= 2^{-3k/2} \Psi_1 \left( 2^{-2k} \Omega_x \right) \Psi_2 \left( 2^k \frac{\Omega_y}{\Omega_x} - l \right) e^{-j2\pi \langle A_2^k \mathbf{B}_l m, \Omega \rangle}.\end{aligned}\quad (9.99)$$

The mother shearlet function can be similarly chosen as in the continuous case. Recall that the function  $\Psi_1(\Omega_x)$ ,  $\Psi_2(\Omega_y)$  are  $C^\infty(\mathcal{R})$ , and they are well localized in the Fourier plane. The window  $\Psi_1(\Omega_x)$  is supported on  $[-2, -1/2] \cup [1/2, 2]$  in the Fourier plane, and it satisfies

$$\sum_{k=0}^{\infty} \left| \Psi_1 \left( 2^{-2k} \Omega_x \right) \right|^2 = 1, \text{ for } |\Omega_x| > \frac{1}{2}. \quad (9.100)$$

The window  $\Psi_2(\Omega_y)$  is supported on  $[-1, 1]$  in the Fourier plane, and it satisfies

$$|\Psi_2(\Omega_y - 1)|^2 + |\Psi_2(\Omega_y)|^2 + |\Psi_2(\Omega_y + 1)|^2 = 1, \text{ for } |\Omega_y| \leq 1, \quad (9.101)$$

which leads to the following equation,

$$\sum_{l=-2^k}^{2^k-1} \left| \Psi_2 \left( 2^k \Omega_y + l \right) \right|^2 = 1, \text{ for } |\Omega_y| \leq 1, \ k \geq 0. \quad (9.102)$$

The Fourier content of a discrete shearlet atom,  $\Psi_{k,l,m}^{(h)}$ , is supported on a pair of trapezoids oriented along a line with slope  $l2^{-k}$ , of size about  $2^{2k} \times 2^k$ , written as

$$W_{k,l} = \left\{ \Omega = (\Omega_x, \Omega_y) : \Omega_x \in [-2^{2k+1}, -2^{2k-1}] \cup [2^{2k-1}, 2^{2k+1}], \left| \frac{\Omega_y}{\Omega_x} - l2^{-k} \right| \leq 2^{-k} \right\}.$$

This system of discrete shearlet atoms defines a Parseval frame on the horizontal cone  $\mathcal{D}^{(h)}$ . An exact reproducing theorem can then be obtained and stated as,

**Theorem 10.** *Let the horizontal shearlet atoms be defined as in Eq. (9.98), using the shearlet window that satisfies Eqs. (9.100) and (9.102). This shearlet system is a Parseval frame for  $L^2(\mathcal{D}^{(h)}, 2)$ .*

The discrete shearlets on the vertical zone can be similarly defined as above. In addition, the lowpass isotropic scaling shearlets can be generated by the translations of a scaling function.

### 1.09.5.6 Numerical implementation of shearlet transform

A numerically efficient implementation of a shearlet transform was previously introduced in [119, 128] based on combining a Laplacian pyramid with appropriate shearing filters. The problem with the implementation is the large side lobe effects around significant edges. Recently, an improved implementation based on separately calculating the vertical and horizontal shearlets was proposed by Yi et al. [13, 123, 124] and will be introduced here.

For convenience, the new implementation reformulates the shearlet transform by introducing the following functions in the horizontal and vertical zones separately,

$$W_{kl}^{(h)}(\Omega) = 2^{k/2} \Psi_2 \left( 2^k \frac{\Omega_y}{\Omega_x} - l \right) \chi_{\mathcal{D}^{(h)}}(\Omega), \quad (9.103)$$

$$W_{kl}^{(v)}(\Omega) = 2^{k/2} \Psi_2 \left( 2^k \frac{\Omega_x}{\Omega_y} - l \right) \chi_{\mathcal{D}^{(v)}}(\Omega). \quad (9.104)$$

The function  $W_{kl}^{(d)}(\Omega)$  where  $d = v, h$  is a window function supported on a pair of trapezoids in the Fourier plane. It corresponds to a spatial function  $w_{kl}^{(d)}(n_1, n_2)$ . Now the Fourier content of the shearlets in Eq. (9.98) can be written as

$$\Psi_{k,l,m}^{(d)}(\Omega) = 2^{-2k} V^{(d)}(2^{-2k}\Omega) W_{kl}^{(d)}(\Omega) e^{-j2\pi\langle m, \Omega \rangle}, \quad d = v, h,$$

where  $V^{(d)}(\Omega_x, \Omega_y)$ ,  $d = v, h$  is defined as

$$V^{(h)}(\Omega_x, \Omega_y) = \Psi_1(\Omega_x) \quad \text{and} \quad V^{(v)}(\Omega_x, \Omega_y) = \Psi_1(\Omega_y).$$

The discrete shearlet transform of a discrete sampled function  $f(n_1, n_2)$  can be computed as

$$\begin{aligned} \gamma_f^{(d)}(k, l, m) &= \int_{\mathcal{R}^2} F(\Omega) \Psi_{k,l,m}^{(d*)}(\Omega) d\Omega \\ &= \int_{\mathcal{R}^2} 2^{-2k} F(\Omega) V^{(d*)}(2^{-2k}\Omega) W_{kl}^{(d*)}(\Omega) e^{j2\pi\langle m, \Omega \rangle} d\Omega \\ &= \int_{\mathcal{R}^2} V_k^{(d*)} f(\Omega) W_{kl}^{(d*)}(\Omega) e^{j2\pi\langle m, \Omega \rangle} d\Omega \quad \text{for } d = v, h. \end{aligned} \quad (9.105)$$

Note that here “\*” as a sup-script indicates complex conjugate, with

$$V_k^{(d)} f(\Omega) = a F(\Omega) V^{(d)}(a\Omega), \quad a = 2^{-2k}.$$

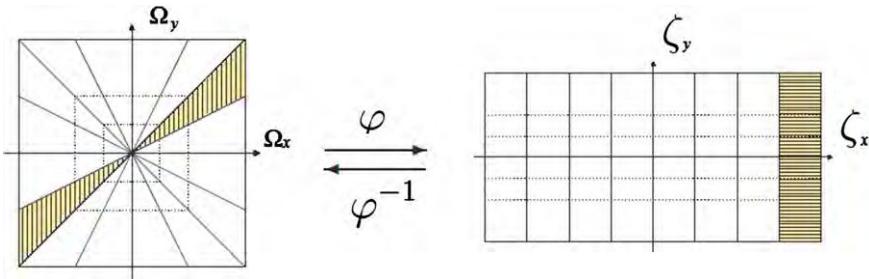
The  $V_k^{(d)} f(\Omega)$  is the output from the highpass filter in the horizontal/vertical zone. Equation (9.105) can be viewed as the inverse Fourier transform, which is equivalent to a convolution in the spatial plane, and can be written as

$$\gamma_f^{(d)}(k, l, m) = \left( v_k^{(d)} f * w_{kl}^{(d)} \right)(m).$$

Here  $v_k^{(d)} f$  is the inverse Fourier transform of the function  $V_k^{(d*)} f(\Omega)$ , and can be written as

$$v_k^{(d)} f(\mathbf{b}) = \int_{\mathcal{R}^2} V_k^{(d*)} f(\Omega) e^{j2\pi\langle \mathbf{b}, \Omega \rangle} d\Omega.$$

The value of  $v_k^{(d)} f(\mathbf{b})$  is obtained through an iterative calculation as in the algorithm introduced later. The value of  $w_{kl}^{(d)}(\mathbf{b})$  is the inverse Fourier transform of  $W_{kl}^{(d*)}(\Omega)$ , obtained through a mapping of

**FIGURE 9.48**

The mapping from a wedge in the Fourier plane to a rectangle in the Pseudo-polar coordinates according to Eq. (5.6).

the Fourier plane from the Cartesian grid to the pseudo-polar grid, which is defined using the following pseudo-polar coordinates  $(\zeta_x, \zeta_y) \in \mathbb{R}^2$ :

$$\begin{aligned} (\zeta_x, \zeta_y) &= \left( \Omega_x, \frac{\Omega_y}{\Omega_x} \right) \quad \text{if } (\Omega_x, \Omega_y) \in \mathcal{D}^{(v)}; \\ (\zeta_x, \zeta_y) &= \left( \Omega_y, \frac{\Omega_x}{\Omega_y} \right) \quad \text{if } (\Omega_x, \Omega_y) \in \mathcal{D}^{(h)}. \end{aligned}$$

Figure 9.48 shows the mapping from the Fourier plane to the pseudo-polar coordinates. Under this mapping, the following result is obtained,

$$\begin{aligned} V_a^{(d)} f(\zeta_x, \zeta_y) &= V_a^{(d)} f(\Omega_x, \Omega_y), \\ W^{(d)}(2^k(\zeta_y - l)) &= W_{kl}^{(d)}(\Omega_x, \Omega_y). \end{aligned}$$

Note that each directional window  $W_{kl}^{(d)}$  in the Fourier plane can be obtained by translating the window  $W^{(d)}$  in the pseudo-polar grid using the function  $\Psi_2$  in Eqs. (9.103) and (9.104), resulting in the directional localization supported in a wedge in the Fourier plane, as shown in Figure 9.49. This results in the spatial waveform  $w_{kl}^{(d)}(\mathbf{m})$  to be used in the following algorithm.

The digital implementation of the discrete shearlet transform in Eq. (9.105) can be realized through a filter bank as follows. Let  $H_k$  ( $G_k$ ) be the lowpass (highpass) filter of a wavelet transform with  $2k - 1$  zeros inserted between consecutive coefficients of a given 1-D filter  $H$  ( $G$  respectively). Define  $u * (H, G)$  to be the separable convolution of the rows and the columns of  $u$  with  $H$  and  $G$ , respectively.

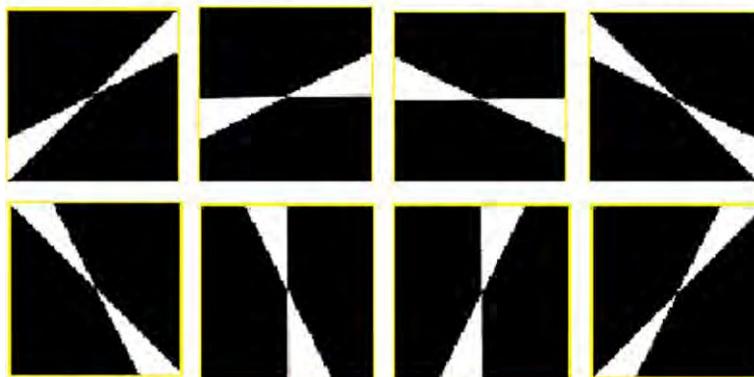
Notice that  $G$  is the wavelet filter corresponding to a highpass odd function  $\psi_1$ ,  $H$  is the filter corresponding to the coarse scale. The window  $W^{(d)}$  is related to an even lowpass function  $\psi_2$ . The  $\psi_1$  and  $\psi_2$  can be selected through the 1-D Meyer's wavelets.

The following is a cascade algorithm for implementing the discrete shearlet transform.

Let  $f \in l_2(\mathbb{Z}^2)$  be a 2-D image, define

$$\begin{aligned} S_0 f &= f(\mathbf{x}) \\ S_k f &= (S_{k-1} f) * (H_k, H_k), \quad k \geq 1. \end{aligned}$$

Images of the windows that partition the Fourier plane at various orientations

**FIGURE 9.49**

The  $2^3$  partition windows of the Fourier plane.

For  $d = v, h$ , the discrete shearlet transform can be calculated by the following

$$\gamma_f^{(d)}(k, l, \mathbf{m}) = \left( v_k^{(d)} f * w_{kl}^{(d)} \right) (\mathbf{m}), \quad \text{for } k \geq 0, -2^k \leq l \leq 2^k - 1, \mathbf{m} \in \mathcal{Z}^2, \quad (9.106)$$

where

$$v_k^{(h)} f = (S_k f) * (G_k, \delta), \quad v_k^{(v)} f = (S_k f) * (\delta, G_k).$$

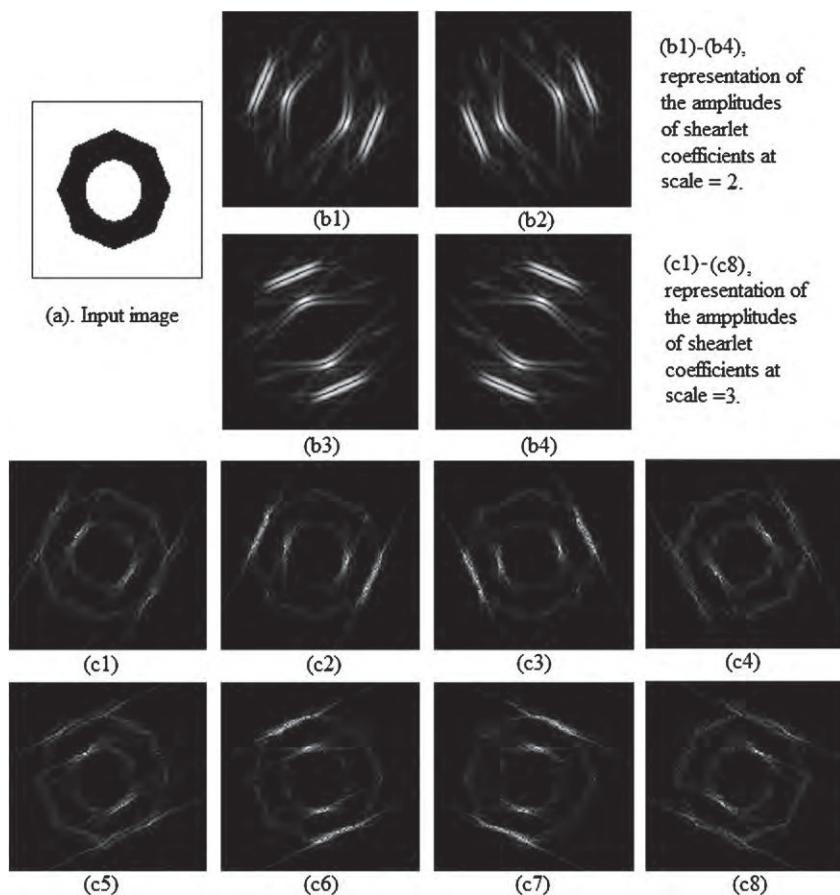
Figure 9.50 shows a representation of the discrete shearlet coefficients for different orientations at decomposition levels 2 and 3. For convenience, the vertical and horizontal shearlet coefficients are re-labeled according to the parameter  $l$  that represents the orientation,

$$\gamma_f(k, l, \mathbf{m}) = \begin{cases} \gamma_f^{(h)}(k, l - 1 - 2^k, \mathbf{m}) & 1 \leq l \leq 2^{k+1}, \\ \gamma_f^{(v)}(k, 3 * 2^k - 1 - l, \mathbf{m}) & 2^{k+1} < l \leq 2^{k+2}. \end{cases} \quad (9.107)$$

### 1.09.5.7 Applications

In the following, we are to separately introduce the applications of shearlet transform in processing edge related information, for example, estimation of edge orientation, edge detection, feature classification, and image denoising.

Edge related tasks, such as edge detection can be very difficult when noise, mixed with multiple edges, is presented in an image. The asymptotic decay property of the shearlet coefficients  $\gamma_f$  suggests that a shearlet transform can be a very efficient tool to determine the edges and their orientations inside an image  $f$ . A shearlet transform not only captures the singularities in a 2-D function, but also identifies the geometrical feature of curves where the singularities are located. This leads to the feature classification, namely, to determine whether a singularity point is a regular edge point or a irregular edge point, such as a corner and junction point.

**FIGURE 9.50**

(a) The image of a disk is shown. (b1-b4) The representations of shearlet coefficients of the disk image at multiple scales are shown for several values of the orientation index  $l$  for scale = 2. (c1-c8) The representations for scale = 3 are shown.

Another difficult task in image processing is denoising. Noise is the most prevalent as spike singularities are classified as isolated points that have significantly higher gradients than their neighbors. The fact that the shearlet transform coefficients on spike points actually grow at fine scales shows that a shearlet transform can be used to distinguish noise points from true edge points, and can therefore be an efficient tool for denoising.

#### 1.09.5.7.1 Estimation of edge orientation

Detecting edge orientation using a shearlet transform takes advantage of geometrical information contributions at various scales, directions, and locations. Multiple orientations may be found when there

are more than two curves merged at a location. Recently, Yi et al. [13, 123, 124] proposed to estimate the edge orientation using the following formula:

$$l(k, \mathbf{m}) = \arg \max_l |\gamma_f(k, l, \mathbf{m})|, \quad (9.108)$$

where  $\mathbf{m}$  is an edge point,  $k$  is a fine scale. The orientation angle,  $l(k, \mathbf{m})$ , can then be directly calculated. Note that this angle is a discrete value associated with  $l = 1, 2, 3, \dots, 2^k$  possible orientations. To reduce the quantization error, the values of the orientations will be interpolated using a parabolic function of  $l$  to model the continuous shearlet transform  $\gamma_f(k, l, \mathbf{m})$ , and the orientation angle of the edge is associated with the maximum value in the parabolic function. The following procedure describes the algorithm that uses shearlet transform to detect the orientation of an edge.

#### Shearlet Orientation Detection Algorithm:

- Compute  $l_1 = l(k, \mathbf{m})$  using Eq. (9.108); let  $l_0 = (l_1 - 1)(\text{mod } N)$ ,  $l_2 = (l_1 + 1)(\text{mod } N)$ , where  $N$  is the size of the set of indices  $l$ .
- $l_i$ ,  $i = 0, 1, 2$  are the slopes of three adjacent discretized directions. Let  $\theta_i$ ,  $i = 0, 1, 2$ , be the angles of orientations calculated from the  $l_i$ .
- Let  $S(\theta) = c_1\theta^2 + c_2\theta + c_3$ . When  $\theta = \theta_i$ ,  $S(\theta_i)$  is identified with  $\gamma_f(k, l_i, \mathbf{b})$ ,  $i = 0, 1, 2$ . Then  $c_1, c_2, c_3$  is obtained by fitting the system  $S(\theta_i) = c_1\theta_i^2 + c_2\theta_i + c_3$ . The detected angle,  $\theta_{\max} = -\frac{c_2}{2c_1}$ ,  $c_1 < 0$  is the value where the parabolic function  $S(\theta)$  achieves its maximum.

Figure 9.51 provides an error analysis of applying this algorithm to an image with a circle object.

#### 1.09.5.7.2 Feature classification

We present in this section a computationally efficient shearlet approach for classifying several different feature points in an image. This method has advantages over wavelet-based method, on account of the fact that the shearlet transform catches the directional features of an image.

In a typical image, there usually exist four classes of points, namely, junction and corner points; points near edges; points on smooth edges; points inside a smooth region. The different types of points may be classified by examining the behavior of the discrete shearlet transform coefficients according to the shearlet properties at these points.

First, at a fixed scale  $k_0$ , we define the energy function at each spatial point  $\mathbf{m}$  as

$$E_{k_0}(\mathbf{m}) = \sum_l |\gamma_f(k_0, l, \mathbf{m})|. \quad (9.109)$$

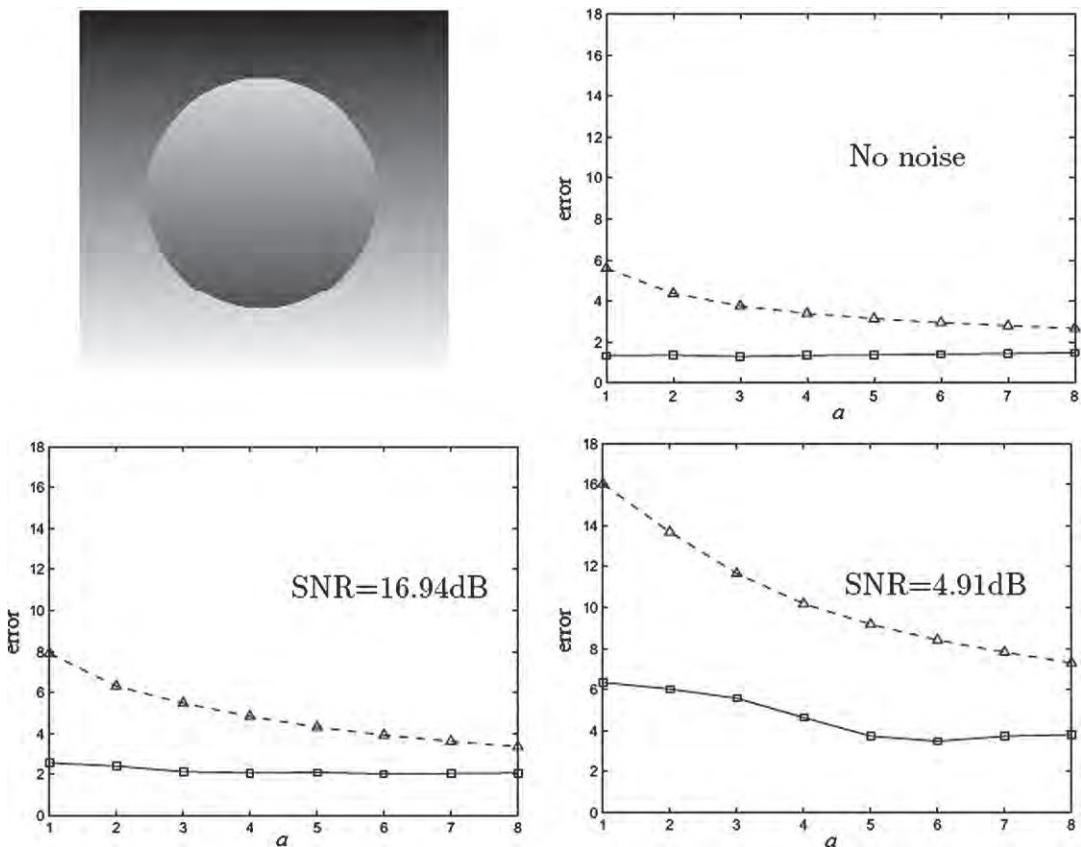
The points with large energy values  $E(\mathbf{m})$  are classified as boundary points.

Second, define the following function:

$$s_{\mathbf{m}_0}(l) = |\gamma_f(k_0, l, \mathbf{m}_0)|.$$

The function is tested on the boundary points, and the corners and junction points will be separated from the regular edge points by applying the following criteria:

- The point  $\mathbf{m}_0$  is a corner/junction point if more than one peak is presented in the function  $s_{\mathbf{m}_0}(l)$  at  $\mathbf{m}_0$ .
- The point  $\mathbf{m}_0$  is a regular edge point if the function  $s_{\mathbf{m}_0}(l)$  has only a single peak at  $\mathbf{m}_0$ .

**FIGURE 9.51**

Courtesy image [124]. Comparison of the average error in the estimation of edge orientation (expression (3.4.9)), for the disk image shown on the left, using the wavelet method (dashed line) versus the shearlet method (solid line), as a function of the scale  $a$ , for various SNRs (additive white Gaussian noise).

In order to implement this algorithm, for each boundary point  $\mathbf{m}$  at scale  $k_0$ , let  $L_{\mathbf{m}}$  be the set that collects all the orientations at which the local maxima of the amplitudes of the discrete shearlet coefficients occur along parameter  $l$ , namely,

$$L_{\mathbf{m}} = \{l : |\gamma_f(k_0, l, \mathbf{m})| > |\gamma_f(k_0, l + 1, \mathbf{m})| \text{ and } |\gamma_f(k_0, l, \mathbf{m})| > |\gamma_f(k_0, l - 1, \mathbf{m})|\}.$$

To verify the significance of the peaks in  $L_{\mathbf{m}}$  for each boundary point  $\mathbf{m}$ , in the following algorithm, we use normalized shearlet coefficients at the point  $\mathbf{m}$  defined as the following in order to find the local maximum points,

$$p_{\mathbf{m}}(l) = \begin{cases} \frac{|\gamma_f(k_0, l, \mathbf{m})|}{\sum_{l \in L_k} |\gamma_f(k_0, l, \mathbf{m})|}, & l \in L_k, \\ 0, & l \notin L_k. \end{cases} \quad (9.110)$$

Feature Classification Algorithm:

- Use the energy function  $E_{k_0}(\mathbf{m})$  defined in Eq. (9.109) to cluster the image points into three mutually exclusive sets  $I_1, I_2, I_3$ , the K-Mean clustering algorithm with Euclidean metric is used in the clustering algorithm.
- Reorder the index of sets  $I_i$ , namely, let

$$i_{\max} = \operatorname{argmax}_i \frac{1}{|I_i|} \sum_{\mathbf{m} \in I_i} E_{k_0}(\mathbf{m}), \text{ and } i_{\min} = \operatorname{argmin}_i \frac{1}{|I_i|} \sum_{\mathbf{m} \in I_i} E_{k_0}(\mathbf{m}).$$

The set  $I_{i_{\max}}$  is the set of boundary points, because the significant coefficients only occur at the boundary points. The set  $I_{i_{\min}}$  contains regular points inside a region because the coefficients rapidly decay to zero at these points. The set  $I_i$  between sets  $I_{i_{\min}}$  and  $I_{i_{\max}}$  contains the near edge points.

- For a point  $\mathbf{m} \in I_{i_{\max}}$ , sort the entries of the normalized coefficients  $p_{\mathbf{m}}(l)$  from the greatest to the smallest, and denote it as  $[\tilde{p}_{\mathbf{m}}(1), \dots, \tilde{p}_{\mathbf{m}}(N)]$ . Using again the K-Mean clustering algorithm on  $I_{i_{\max}}$ , this set can be further classified into one group of smooth edge points, and another group of corner and junction points according to the number of peaks at each boundary point.

Note that corner points can be further separated from junction points, and junction points can be separated into different classes corresponding to different geometric features. Figure 9.52 illustrates the result of applying the feature classification algorithm.

#### 1.09.5.7.3 Edge detection

Edge detection is a difficult task when noise is presented, and when several edges intersect each other and they are close together. The asymptotic decay rate of the continuous shearlet transform provides a better solution to precisely capture the geometry of edges, and it can be used to obtain both locations and orientations of edges in an image  $f$ . Here, we describe an effective edge detection scheme to precisely detect the shape of an edge using the shearlet transform that can be attributed to the anisotropic analytic and geometric properties of shearlets.

In this algorithm, the first step is to identify the edge candidates in an image  $f$ . They are selected by the shearlet transform modulus maxima at the  $k$ th scale level, that is, those points,  $\mathbf{m}$ , that are the local maxima of the function

$$M_k f(\mathbf{m})^2 = \sum_l |\gamma_f(k, l, \mathbf{m})|^2.$$

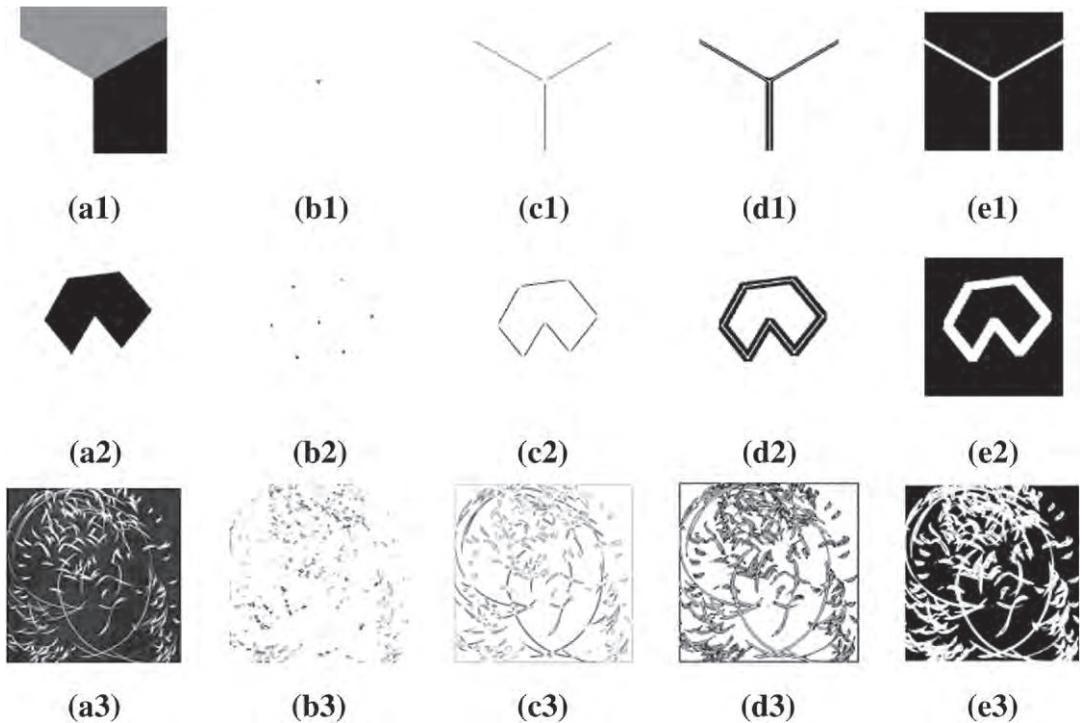
The second step is to confirm edge points by excluding the noise points. According to the property of shearlet transform, if  $\mathbf{m}$  is an edge point, the shearlet transform of  $f$  has the property that

$$|\gamma_f(k, l, \mathbf{m})| \sim C 2^{-\beta k}, \quad \beta > 0.$$

Note that points with  $\beta < 0$  will be classified as noise; therefore, the edge points can be identified as those points that have  $\beta > 0$ , and the task of edge detection is transferred to the task of determining the sign of  $\beta$ .

Theoretically, this can be estimated by computing the best linear fit to data

$$\{\log |\gamma_f(k, l, \mathbf{m})|\}_{k=1,2,\dots,L}.$$

**FIGURE 9.52**

Courtesy image [124]. (a1–a3) Test images. (b1–b3) Identification of corners and junctions. (c1–c3) Identification of smooth edge points. (d1–d3) Identification of points near the edges. (e1–e3) Identification of regular points (smooth regions).

The magnitude of the shearlet coefficients at each point is compared with the values of its neighbor's along the gradient direction (this is obtained from the orientation map of the shearlet decomposition). If the magnitude is smaller, the point is discarded; if it is the largest, it is kept. This yields a set of possible edge points. This is the so called non-maximal suppression routine. Further, this edge candidate set will be filtered by a window filter, and a smooth edge set can be selected by threshold. In practice, the following shearlet edge detection algorithm is implemented to simplify the computation complexity.

#### Shearlet Edge Detection Algorithm:

- Let  $f \in l_2(\mathbb{Z}^2)$  be a 2-D image, calculate  $\gamma_f^{(d)}(k, l, \mathbf{m})$  given as follows:

$$\gamma_f^{(d)}(k, l, \mathbf{m}) = \left( v_k^{(d)} f * w_{kl}^{(d)} \right) (\mathbf{m}), \quad \text{for } k \geq 0, -2^k \leq l \leq 2^k - 1, \mathbf{m} \in \mathbb{Z}^2, \quad (9.111)$$

where

$$S_0 f = f(\mathbf{x}),$$

$$\begin{aligned} S_k f &= (S_{k-1} f) * (H_k, H_k), \quad k \geq 1, \\ v_k^{(h)} f &= (S_k f) * (G_k, \delta), \\ v_k^{(v)} f &= (S_k f) * (\delta, G_k). \end{aligned}$$

- Define

$$\chi_l^{(d)}(\mathbf{m}) = \begin{cases} 1 & \text{if } \gamma_f^{(d)}(k, l, \mathbf{m}) > \gamma_f^{(d)}(k-1, l, \mathbf{m}) \\ 0 & \text{otherwise} \end{cases}, \quad d = v, h. \quad (9.112)$$

This defines a function to indicate the points that has negative Lipschitz regularity.

- Based on this indicator function, define

$$R_k^{(d)} f(\mathbf{m}) = \sum_l \gamma_f^{(d)}(k, l, \mathbf{m}) \chi_l^{(d)}(\mathbf{m}), \quad d = v, h.$$

- Modify  $v_k^{(d)} f$ ,  $d = v, h$ , according to the following formula:

$$v_k^{(d)} f = \begin{cases} v_k^{(d)} f + R_k^{(d)} f & \text{if } |v_k^{(d)} f| \leq |R_k^{(d)} f| \\ R_k^{(d)} f & \text{otherwise} \end{cases}, \quad d = v, h. \quad (9.113)$$

This formula modifies  $v_k^{(d)} f$  so that the value of  $v_k^{(d)} f$  is kept small (unchanged) for locations with positive Lipschitz regularity and the value of  $v_k^{(d)} f$  is increased over the scales for locations with negative Lipschitz regularity.

- After the discrete shearlet  $\gamma_f^{(d)}(k, l, \mathbf{m})$  is calculated using the updated  $v_k^{(d)} f$ . The edge at level  $k$  will be decided by checking the local maxima of the following function:

$$E_k(\mathbf{m}) = \left( \sum_l \gamma_f^{(v)}(k, l, \mathbf{m}) \right)^2 + \left( \sum_l \gamma_f^{(h)}(k, l, \mathbf{m}) \right)^2.$$

Figure 9.53 shows the results of edge detection, in which the Pratt's figure of merit (FOM) is used to measure the performance of an edge detector [129].

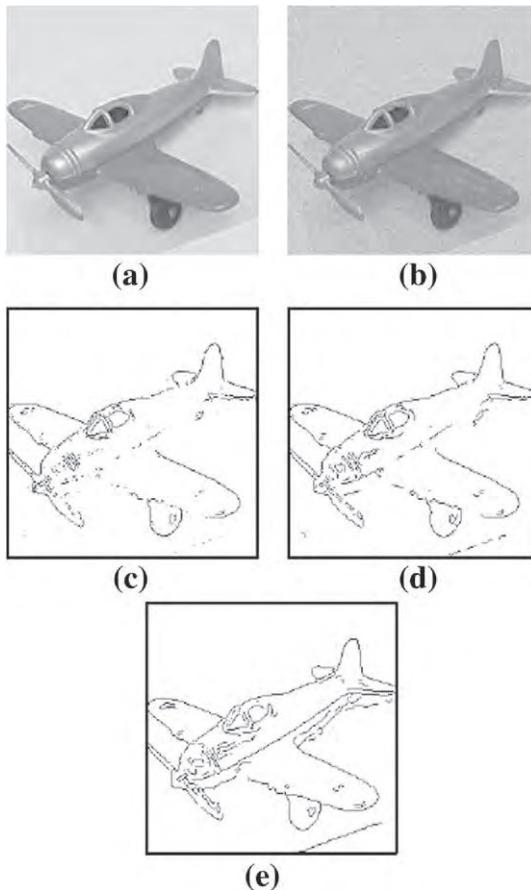
#### 1.09.5.7.4 Image denoising

The properties of shearlet coefficients at fine scales show that a 2-D function  $f$  can be reproduced using the following shearlet representation, see [130],

$$f(\mathbf{b}) = \sum_{k,l,\mathbf{m} \in M_1} \langle f, \psi_{k,l,\mathbf{m}} \rangle \psi_{k,l,\mathbf{m}}(\mathbf{b}) + \sum_{k,l,\mathbf{m} \in M_2} \langle f, \psi_{k,l,\mathbf{m}} \rangle \psi_{k,l,\mathbf{m}}(\mathbf{b}), \quad (9.114)$$

where  $M_1$  is the set of coefficients associated with the smooth regions of  $f$ , and  $M_2$  is the set of coefficients associated with the edges of  $f$ .

A simple approach to achieve image denoising is carried out by using a shrinkage approach to remove shearlet coefficients below a threshold,  $\tau$ . A denoised image using the discrete shearlet transform can

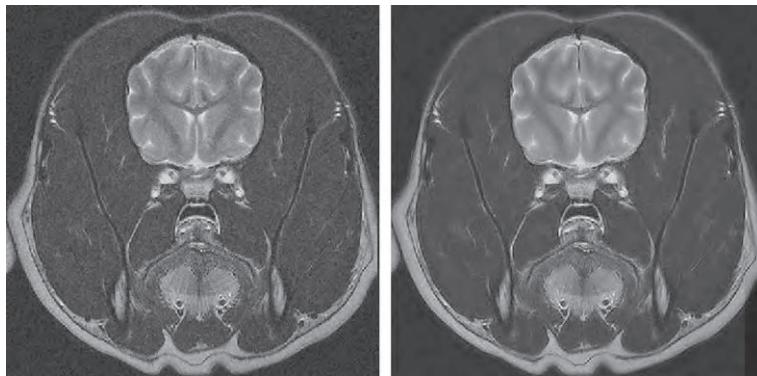
**FIGURE 9.53**

Courtesy image [124]. Results of edge detection methods. From (a) to (e): (a) original image, (b) noisy image (PSNR = 24.58 dB), (c) Sobel result (FOM = 0.54), (d) wavelet result (FOM = 0.84), and (e) shearlet result (FOM = 0.94).

be expressed as

$$\tilde{f}(\mathbf{b}) \approx \sum_{k,l,\mathbf{m} \in M_1} \langle f, \psi_{k,l,\mathbf{m}} \rangle \psi_{k,l,\mathbf{m}}(\mathbf{b}) + \sum_{k,l,\mathbf{m} \in M_2^c} \langle f, \psi_{k,l,\mathbf{m}} \rangle \psi_{k,l,\mathbf{m}}(\mathbf{b}), \quad (9.115)$$

where  $M_2^c$  is the set of shearlet coefficients whose values are greater than  $\tau$ . This approach is able to optimally capture directional features, and reach the desired denoising effect. Figure 9.54 shows a denoising result using the shrinkage method.

**FIGURE 9.54**

In this picture, the noisy brain image is shown to the left, and the filtered image using the shearlet denoising method, is shown to the right.

## A Appendix

### A.1 The $z$ -transform

Much like in the continuous time domain, we can cover the whole complex plane by using a Laplace Transform in contrast to the harmonic Fourier Transform along the imaginary axis, we may cover the whole interior of the unit circle in the discrete domain by invoking the so-called  $z$ -Transform. For purposes of direct relevance to the derivations of the various multi-scale transforms in this chapter, we define the  $z$ -transform in the 2-D plane. Given a 2-D discretized signal,  $x(\mathbf{n})$ ,  $\mathbf{n} \in \mathbb{Z}^2$ , its  $z$ -transform,  $X(z)$ ,  $z \in \mathbb{Z}^2$  is defined as

$$X(z) = \sum_{n \in \mathbb{Z}^2} x(\mathbf{n})z^{-\mathbf{n}} = \sum_{n \in \mathbb{Z}^2} x(\mathbf{n})z_1^{-n_1}z_2^{-n_2}, \text{ with } \mathbf{n} = (n_1, n_2), z = (z_1, z_2).$$

Note that the  $z$ -transform of an image  $x(\mathbf{n})$  is reduced to its discrete time Fourier series when we evaluate its  $z$ -transform function at  $z = e^{j\omega}$ .

Given a discrete 2-D image  $x(\mathbf{n})$  as an input to a 2-D linear discrete invariant system with an impulse response of  $h(\mathbf{n})$ , the resulting output is the following convolution sum:

$$y(\mathbf{n}) = x(\mathbf{n}) * h(\mathbf{n}) = \sum_{s \in \mathbb{Z}^2} x(s_1, s_2)h(n_1 - s_1, n_2 - s_2).$$

The  $z$ -transform of the output,  $y(\mathbf{n})$  of the system is similarly (to the continuous analog)  $Y(z) = H(z)X(z)$ , where  $z$ -transform of the impulse response  $h(\mathbf{n})$ , is written as  $H(z)$ , and called the transfer function of the filter system. The  $z$ -transform is a very widely used tool in the analysis and design of discrete filter systems.

*Relevant Theory:* Signal Processing Theory and Statistical Signal Processing

See this Volume, [Chapter 2](#) Continuous-Time Signals and Systems

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 7](#) Multirate Signal Processing

See this Volume, [Chapter 10](#) Frames

See [Vol. 3](#), [Chapter 4](#) Bayesian Computational Methods in Signal Processing

## References

- [1] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *J. Physiol.* 160 (1962) 106–154.
- [2] B. Olshausen, D. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, *Nature* 381 (1996) 607–609.
- [3] B. Olshausen, D. Field, Sparse coding of sensory inputs, *Curr. Opin. Neurobiol.* 14 (2004) 481–487.
- [4] E.J. Candès, D. Donoho, Curvelets – a surprisingly effective nonadaptive representation for objects with edges, in: A. Cohen, C. Rabut, L. Schumaker (Eds.), *Curves and Surface Fitting: Saint-Malo 1999*, Vanderbilt University Press, Nashville, 2000, pp. 105–120.
- [5] E.J. Candès, D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise  $c_2$  singularities, *Pure Appl. Math.* (2004) (vol. to appear, comm.).
- [6] E. Candès, D. Donoho, Continuous curvelet transform: I. Resolution of the wavefront set, *Appl. Comput. Harmon. Anal.* 19 (2003) 162–197.
- [7] E. Candès, D. Donoho, Continuous curvelet transform: II. Discretization and frames, *Appl. Comput. Harmon. Anal.* 19 (2003) 198–222.
- [8] M.N. Do, M. Vetterli, Contourlets, in: *Beyond Wavelets*, in: G.V. Welland (Ed.), Academic Press, 2003.
- [9] M.N. Do, M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image Process.* 14 (2005) 2091–2106.
- [10] K. Guo, D. Labate, W. Lim, G. Weiss, E. Wilson, Wavelets with composite dilations, *Electr. Res. Announc. AMS* 10 (2004) 78–87.
- [11] K. Guo, D. Labate, W. Lim, G. Weiss, E. Wilson, Wavelets with composite dilations and their MRA properties, *Appl. Comput. Harmon. Anal.* 20 (2006) 202–236.
- [12] D. Labate, W. Lim, G. Kutyniok, G. Weiss, Sparse multidimensional representation using shearlets, in: *SPIE Conf. Wavelets XI*, San Diego, USA, 2005, pp. 254–262.
- [13] S. Yi, D. Labate, G.R. Easley, H. Krim, A shearlet approach to edge analysis and detection, *IEEE Trans. Image Process.* 18 (2009) 929–941.
- [14] <<http://www.vissta.ncsu.edu/>>.
- [15] A. Papoulis, *Signal Analysis*, McGraw Hill, 1977.
- [16] E. Brigham, *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [17] D. Gabor, Theory of communication, *J. IEEE* 93 (1946) 429–457.
- [18] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, Boston, 1997.
- [19] I. Daubechies, S. Mallat, A. Willsky, Special edition on wavelets and applications, *IEEE Trans. IT* (1992).
- [20] I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NSF, SIAM, Philadelphia, 1992.
- [21] M. Vetterli, J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.

- [22] H. Krim, On the distribution of optimized multiscale representations, in: ICASSP, vol. 5, IEEE, Munich, Germany, 1997.
- [23] G.H. Golub, C.F. VanLoan, Matrix Computations, The Johns Hopkins University Press, Baltimore, Maryland, 1984.
- [24] D. Luenberger, Optimization by Vector Space Methods, John Wiley, New York, London, 1968.
- [25] K. Gröchenig, Acceleration of the frame algorithm, IEEE Trans. Signal Process. 41 (1993) 3331–3340.
- [26] A.B. Hamza, H. Krim, Relaxed minimax approach to image denoising, IEEE Trans. Signal Process. 49 (12) (2001) 3045–3054.
- [27] Y. Meyer, Ondelettes et opérateur, vol. 1, Hermann, Paris, 1990.
- [28] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-11 (1989) 674–693.
- [29] Y. Meyer, Wavelets and Applications, first ed., SIAM, Philadelphia, 1992.
- [30] Y. Meyer, Ondelettes et opérateur, vol. 1, Hermann, Paris, 1990.
- [31] F.J. Hampson, J.-C. Pesquet, A nonlinear decomposition with perfect reconstruction, 1998 (Preprint).
- [32] P.L. Combettes, J.-C. Pesquet, Convex multiresolution analysis, IEEE Trans. Pattern Anal. Machine Intell. 20 (1998) 1308–1318.
- [33] W. Sweldens, The lifting scheme: a construction of second generation wavelets, SIAM J. Math. Anal. 29(9) (1997) 511–546.
- [34] S. Mallat, Multiresolution approximation and wavelet orthonormal bases of  $L^2(\mathbb{R})$ , Trans. Amer. Math. Soc. 315 (1989) 69–87.
- [35] J. Woods, S. O’Neil, Sub-band coding of images, IEEE Trans. ASSP 34 (1986) 1278–1288.
- [36] M. Vetterli, Multi-dimensional subband coding: some theory and algorithms, Signal Process. 6 (1984) 97–112.
- [37] F. Meyer, R. Coifman, Brushlets: a tool for directional image analysis and image compression, Appl. Comput. Harmonic Anal. (1997) 147–187.
- [38] G. Strang, T. Nguyen, Wavelets and Filter Banks, first ed., Wellesley-Cambridge Press, Boston, 1996.
- [39] P.P. Vaidyanathan, Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial, Proc. IEEE, 78 (1990) 56–93.
- [40] A. Cohen, I. Daubechies, J.C. Feauveau, Biorthogonal bases of compactly supported wavelets, Comm. Pure Appl. Math. 45 (1992) 485–560.
- [41] P.P. Vaidyanathan, Multirate Systems and Filter Banks, Prentice Hall, New Jersey, 1992.
- [42] R. Coifman, Y. Meyer, Remarques sur l’analyse de Fourier à fenêtre, C.R. Acad. Sci. Série I 312 (1991) 259–261.
- [43] R.R. Coifman, M.V. Wickerhauser, Entropy-based algorithms for best basis selection, IEEE Trans. Inform. Theory IT-38 (1992) 713–718.
- [44] M.V. Wickerhauser, INRIA lectures on wavelet packet algorithms, in: Ondelettes et paquets d’ondelettes (Roquencourt), 17–21 June 1991, pp. 31–99.
- [45] H. Malvar, Lapped transforms for efficient transform subband coding, IEEE Trans. Acoust. Speech, Signal Process. ASSP-38, June 1990, pp. 969–978.
- [46] A. Aldroubi, E.M. Unser, Wavelets in Medicine and Biology, CRC Press, 1996.
- [47] A. Akansu, M.J.T. Smith, Subband and Wavelet Transforms, Kluwer, 1995.
- [48] A. Arneodo, F. Argoul, E. Bacry, J. Elezgaray, J. Muzy, Ondelettes, Multifractales et Turbulence, Diderot, Paris, France, 1995.
- [49] A. Antoniadis, E.G. Oppenheim, Wavelets and Statistics, Lecture Notes in Statistics, Springer Verlag, 1995.
- [50] P. Mueller, E.B. Vidakovic, Bayesian Inference in Wavelet-Based Models, Lecture Notes in Statistics, first ed., vol. LNS 141, Springer-Verlag, 1999.
- [51] B. Vidakovic, Statistical Modeling by Wavelets, John Wiley, New York, 1999.

- [52] K. Ramchandran, M. Vetterli, Best wavelet packet bases in a rate-distortion sense, *IEEE Trans. Image Process.* 2 (1993), 160–175.
- [53] J. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. Signal Process.* 41 (1993) 3445–3462.
- [54] P. Cosman, R. Gray, M. Vetterli, Vector quantization of image subbands:a survey, *IEEE Trans. Image Process.* 5 (1996) 202–225.
- [55] A. Kim, H. Krim, Hierarchical stochastic modeling of SAR imagery for segmentation/compression, *IEEE Trans. Signal Process.* 47 (1999) 458–468.
- [56] M. Basseville, A. Benveniste, K.C. Chou, S. Golden, R. Nikoukhah, A.S. Willsky, Modeling and estimation of multiresolution stochastic processes, *IEEE Trans. Information Theory*, IT-38 (1992) 766–784.
- [57] M. Luetgen, W. Karl, A. Willsky, Likelihood calculation for multiscale image models with applications in texture discrimination, *IEEE Trans. Image Process.* 4 (1995) 194–207.
- [58] E. Fabre, New fast smoothers for multiscale systems, *IEEE Trans. Signal Process.* 44 (1996) 1893–1911.
- [59] P. Fieguth, W. Karl, A. Willsky, C. Wunsch, Multiresolution optimal interpolation and statistical ananlysis of topex/poseidon satellite altimetry, *IEEE Trans. Geosci. Remote Sensing* 33 (1995) 280–292.
- [60] M.K. Tsatsanis, G.B. Giannakis, Principal component filter banks for optimal multiresolution analysis, *IEEE Trans. Signal Process.* 43 (1995) 1766–1777.
- [61] M.K. Tsatsanis, G.B. Giannakis, Optimal linear receivers for DS-CDMA systems: a signal processing approach, *IEEE Trans. Signal Process.* 44 (1996) 3044–3055.
- [62] A. Scaglione, G.B. Giannakis, S. Barbarossa, Redundant filterbank precoders and equalizers, parts I and II, *IEEE Trans. Signal Process.* 47 (1999) 1988–2022.
- [63] A. Scaglione, S. Barbarossa, G.B. Giannakis, Filterbank transceivers optimizing information rate in block transmissions over dispersive channels, *IEEE Trans. Information Theory* 45 (1999) 1019–1032.
- [64] R. Learned, H. Krim, B. Claus, A.S. Willsky, C. Karl, Wavelet-packet based multiple access communication, *Proc. SPIE* 2303 (1994) 246–259.
- [65] R. Learned, A. Willsky, D. Boroson, Low complexity optimal joint detection for oversaturated multiple access communications, *IEEE Trans. Signal Process.* 45 (1997) 113–123.
- [66] A. Lindsey, Wavelet packet modulation for orthogonally multiplexed communication, *IEEE Trans. Signal Process.* 45 (1997) 1336–1339.
- [67] K. Wong, J. Wu, T. Davidson, Q. Jin, Wavelet packet division multiplexing and wavelet packet design under timing error effects, *IEEE Trans. Signal Process.* 45 (1997) 2877–2886.
- [68] H. Krim, I. Schick, Minimax description length for signal denoising and optimized representation, *IEEE Trans. Information Theory*, pp. 809–908, April 1999. *IEEE Trans. on IT Special Issue*, Eds. H. Krim, W. Willinger, A. Iouditski and D. Tse.
- [69] D.L. Donoho, I.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika* 81 (1994) 425–455.
- [70] H. Krim, S. Mallat, D. Donoho, A. Willsky, Best basis algorithm for signal enhancement, in: *ICASSP '95*, IEEE, Detroit, MI, May 1995.
- [71] N. Saito, Local feature extraction and its applications using a library of bases, PhD Thesis, Yale University, December 1994.
- [72] H. Krim, J.-C. Pesquet, On the Statistics of Best Bases Criteria, vol. *Wavelets in Statistics of Lecture Notes in Statistics*, Springer-Verlag, July 1995, pp. 193–207.
- [73] J. Rissanen, Modeling by shortest data description, *Automatica* 14 (1978) 465–471.
- [74] B. Vidakovic, Nonlinear wavelet shrinkage with Bayes rules and Bayes, *J. Am. Statist. Assoc.* 93 (1998) 173–179.
- [75] D. Leporini, J.-C. Pesquet, H. Krim, *Best Basis Representation with Prior Statistical Models*, *Lecture Notes in Statistics*, Springer-Verlag, 1999 (Chapter 11).
- [76] J. Rissanen, Stochastic complexity and modeling, *Ann. Statist.* 14 (1986) 1080–1100.

- [77] P. Huber, Robust estimation of a location parameter, *Ann. Math. Stat.* 35 (1964) 1753–1758.
- [78] P. Huber, Théorie de l’inférence statistique robuste, tech. rep., Univ. de Montreal, Montreal, Quebec, Canada, 1969.
- [79] D. Donoho, I. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, *JASA* 90 (1995) 1200–1223.
- [80] H. Krim, D. Tucker, S. Mallat, D. Donoho, Near-optimal risk for best basis search, *IEEE Trans. Inform. Theory* (1999) 2225–2238.
- [81] E.J. Candès, D. Donoho, Ridgelets: a key to higher-dimensional intermittency?, *Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.* 357 (1999) 2495–2509.
- [82] H.F. Smith, A parametrix construction for wave equations with  $c1,1$  coefficients, *Ann. Inst. Fourier (Grenoble)*, 48 (1998) 797–835.
- [83] E.J. Candès, L. Demanet, D. Donoho, L. Ying, Fast discrete curvelet transforms, *Multiscale Modeling and Simulation* 5 (3) (2006), 861–899.
- [84] S. Mallat, *Adaptive Wavelet Collocation for Nonlinear A Wavelet Tour of Signal Processing*, Academic Press, San Diego, 1998.
- [85] J. Ma, G. Plonka, The curvelet transform, *IEEE Signal Process. Mag.* (2010) 118–133.
- [86] M. Fadili, J. Starck, Curvelets and Ridgelets, *Encyclopedia of Complexity and Systems Science*, Meyers, Robert (Ed.), vol. 3, Springer, New York, 2009, pp. 1718–1738.
- [87] F.J. Herrmann, P.P. Moghaddam, C.C. Stolk, Sparsity- and continuity-promoting seismic image recovery with curvelet frames, *Appl. Comput. Harmon. Anal.* 24 (2) (2008) 150–173.
- [88] [<http://www.curvelet.org/software.html>](http://www.curvelet.org/software.html).
- [89] E.J. Candès, F. Guo, New multiscale transforms, minimum total variation synthesis: applications to edge-preserving image reconstruction (Signal Processing, special issue on Image and Video Coding Beyond Standards) 82 (11) (2002) 1519–1543.
- [90] E.J. Candès, D.L. Donoho, Recovering edges in ill-posed inverse problems: optimality of curvelet frames, *Ann. Statist.* 30 (2002) 784–842.
- [91] E.J. Candès, L. Demanet, Curvelets and Fourier integral operators, *C.R. Math. Acad. Sci. Paris, Ser.* 336 (5) (2003) 395–398.
- [92] E.J. Candès, D.L. Donoho, Curvelets: new tools for limited-angle tomography, Technical Report, California Institute of Technology, 2004.
- [93] E. Candès, J. Romberg, T. Tao, Stable signal recovery from incomplete and inaccurate information, *Commun. Pure Appl. Math.* 59 (2006) 1207–1233.
- [94] E.J. Candès, L. Demanet, Curvelets and fast wave equation solvers, Technical report, California Institute of Technology, 2005.
- [95] J. Ma, A. Antoniadis, F.-X.L. Dimet, Curvelet-based snake for multiscale detection and tracking for geophysical fluids, *IEEE Trans. Geosci. Remote Sensing* 44 (12) (2006) 3626–3637.
- [96] J. Ma, Curvelets for surface characterization, *Appl. Phys. Lett.* 90 (2007) 054109.
- [97] J. Ma, Deblurring using singular integrals and curvelet shrinkage, *Phys. Lett. A.* 368 (2007) 245–250.
- [98] J.L. Starck, E.J. Candès, D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. Image Process.* 11–6 (2002) 670–684.
- [99] J.L Starck, M. Elad, D.L Donoho, Image decomposition via the combination of sparse representation and a variational approach, *IEEE Trans. Image Process.* 14(10) (2004) 1570–1582.
- [100] C. Zhang, L. Cheng, Z. Qiu, L. Cheng, Multipurpose watermarking based on multiscale curvelet transform, *IEEE Trans. Inform. Forensics Security* 3 (4) (2008) 611–619.
- [101] Y. Bao, H. Krim, Upon bridging scale-space to multiscale frame analysis, vol. 19, in: A. Petrosian, F.G. Meyer (Eds.), *Wavelets in Signal and Image Analysis: From Theory to Practice, Computational Imaging and Vision*, Springer, Netherlands (Chapter 6).

- [102] M. Elad, Why simple shrinkage is still relevant for redundant representations? *IEEE Trans. IT* 52 (2006) 5559–5569.
- [103] Y. Lu, M.N. Do, Multidimensional directional filter banks and surfacelets, *IEEE Trans. Image Process.* 16 (2007) 918–931.
- [104] H.S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Norwood, MA, 1992.
- [105] K.-O. Cheng, N.-F. Law, W.-C. Siu, Multiscale directional filter bank with applications to structured and random texture retrieval, *Pattern Recogn.* 40 (2007) 1182–1194.
- [106] M. Vetterli, J. Kovacević, *Wavelets and Subband Coding*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [107] P.J. Burt, E.H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Trans. Commun.* 31 (1983) 532–540.
- [108] R.H. Bamberger, M.J.T. Smith, A filter bank for the directional decomposition of images: Theory and design, *IEEE Trans. Signal Process.* 40 (1992) 882–893.
- [109] A.L. Cunha, J. Zhou, M.N. Do, The nonsubsampled contourlet transform: Theory, design, and applications, *IEEE Trans. Image Process.* 15 (2006) 3089–3101.
- [110] S. Durand, M-band filtering and nonredundant directional wavelets, *Appl. Comput. Harmon. Anal.* 22 (2006) 124–139.
- [111] S.K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, third ed., McGraw Hill, 2004.
- [112] M.N. Do, M. Vetterli, Framing pyramids, *IEEE Trans. Signal Process.* 51 (2003) 2329–2342.
- [113] S. Park, M.J.T. Smith, R.M. Mersereau, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image Process.* 48 (2005) 797–835.
- [114] M.N. Do, Directional multiresolution image representations, PhD Thesis, Swiss Federal Institute of Technology, Lausanne, November 2001.
- [115] S. Park, M.J.T. Smith, R.M. Mersereau, A new directional filterbank for image analysis and classification, *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* (1999) 1417–1420.
- [116] <<http://www.ifp.illinois.edu/~minhdo/software/>>.
- [117] K. Guo, G. Kutyniok, D. Labate, Sparse multidimensional representations using anisotropic dilation and shear operators, in: G. Chen, M. Lai (Eds.), *Wavelets and Splines*, Nashboro Press, Nashville, TN, 2006, pp. 189–201.
- [118] K. Guo, D. Labate, Optimally sparse multidimensional representation using shearlets, *SIAM J. Math. Anal.* (2007) 298–318.
- [119] G. Easley, D. Labate, W.-Q. Lim, Sparse directional image representations using the discrete shearlet transform, *Appl. Comput. Harmon. Anal.* 25 (2008) 25–46.
- [120] G. Kutyniok, D. Labate, Resolution of the wavefront set using continuous shearlets, *Trans. Am. Math. Soc.* 361 (2009) 2719–2754.
- [121] G. Easley, K. Guo, D. Labate, Analysis of singularities and edge detection using the shearlet transform, in: *Proceedings of SAMPTA’09*, 2009.
- [122] K. Guo, D. Labate, W. Lim, Edge analysis and identification using the continuous shearlet transform, *Appl. Comput. Harmon. Anal.* 27 (2009) 24–46.
- [123] S. Yi, D. Labate, G.R. Easley, H. Krim, Edge detection and processing using shearlets, *ICIP*, 2008, pp. 1148–1151.
- [124] S. Yi, *Shape Dynamic Analysis*, PhD Thesis, North Carolina State University, Dept. of EECS, Raleigh, NC, 2011.
- [125] S. Dahlke, G. Kutyniok, G. Steidl, G. Teschke, Shearlet coorbit spaces and associated banach frames, *Appl. Comput. Harmon. Anal.* 27 (2009) 195–214.
- [126] K. Guo, W. Lim, D. Labate, G. Weiss, E. Wilson, The theory of wavelets with composite dilations, in: C. Heil (Ed.), *Harmonic Analysis and Applications*, Birkhäuser, Boston, 2006, pp. 231–249.

- [127] G. Easley, D. Labate, F. Colonna, Shearlet based total variation for denoising, *IEEE Trans. Image Process.* 18 (2009) 260–268.
- [128] <<http://www.math.uh.edu/~dlabate/software.html>>.
- [129] W. Pratt, *Digital Image Processing*, Wiley Interscience Publications, San Diego, 1978.
- [130] K. Guo, D. Labate, Characterization and analysis of edges using the continuous shearlet transform, *SIAM Imaging Sci.* 2 (2009) 959–986 .

# Frames in Signal Processing

# 10

Lisandro Lovisolo\* and Eduardo A.B. da Silva†

\*Universidade do Estado do Rio de Janeiro (UERJ), Department of Electronics and Telecommunications (DETEL)/Program of Electronics Engineering (PEL), Lab of Signal Processing, Intelligent Applications and Communications (PROSAICO)

†Universidade Federal do Rio de Janeiro (UFRJ), Program of Electrical Engineering—COPPE/UFRJ, Department of Electronics—School of Engineering

## 1.10.1 Introduction

Frames were introduced by Duffin and Schaeffer [1] in 1952 for the study of non-harmonic Fourier series [2,3]. The central idea was to represent a signal by its projections on a sequence of elements  $\{e^{j\lambda_n t}\}_n$ ,  $n \in \mathbb{Z}$ , not restricting the  $\lambda_n$  to be multiples  $n\Omega$  of a fundamental frequency  $\Omega$  as in harmonic (traditional) Fourier series. One can readily see that the set  $\{e^{j\lambda_n t}\}_n$  is highly overcomplete, in the sense that, for example, in a space  $L^2(-T, T)$ , it may consist of a sequence of elements or functions that are greater in number than a basis for  $L^2(-T, T)$ , being the last given for example by harmonic Fourier series, among others.

As a side effect, overcompleteness may make it hard to find a representation of a function  $f(t) \in L^2(-T, T)$  using the set  $\{e^{j\lambda_n t}\}_n$ , in the form

$$f(t) = \sum_n c_n e^{j\lambda_n t}. \quad (10.1)$$

This is because, due to overcompleteness, the weights or coefficients  $c_n$  in Eq. (10.1) can not be computed by the simple approach of projecting  $f(t)$  into each element of  $\{e^{j\lambda_n t}\}_n$ . This would be equivalent to using  $c_n = \langle f(t), e^{j\lambda_n t} \rangle$ , where  $\langle f(t), g(t) \rangle$  represents the inner product  $\int f(t)g^*(t)dt$  and  $g^*(t)$  denotes the complex conjugate of  $g(t)$ . One should note that, when using an orthonormal Basis for representing a signal (as is the sequence of elements defining harmonic Fourier series), computing the inner product is the standard approach for finding the  $c_n$ .

Despite that difficulty, overcompleteness may bring some advantages as one may obtain more compact, robust or stable signal representations than using Bases [4–7]. This is why frames are being widely researched and employed in the last two decades in mathematics, statistics, computer science and engineering [4].

The basic idea leading to the definition of frames is the representation of signals from a given space using more “points” or coefficients than the minimum necessary—this is the whole concept behind overcompleteness. This idea is behind modern analog-to-digital converters (ADC) that sample signals at rates that are higher than the Nyquist rate with low resolution [8]. Since, the signal is sampled at

high rate, the requirement on the sample precision can be relaxed [6]. This is a different perspective as compared to sampling based on the Nyquist criterion, when in general one assumes that a good quantization precision is employed so that quantization errors can be neglected.

In addition to the above, there is the motivation of achieving nonuniform and/or irregular sampling (since sampling clock jitter is equivalent to irregular sampling). This has led to a large set of results and some applications of the “frame theory” [9]. These ideas were further extrapolated leading to the definition of “Rate of Innovation” of signals [10]. In this case one considers the number of degrees of freedom per unit of time of a class of signals; this is employed in [10] for sampling purposes. In some way, these concepts have lead to the so-called “Compressed Sensing” (CS) perspective [11].

Although frames provide the mathematical justification for several signal processing methods and are at the heart of the flow of ideas and achievements shortly and briefly described above, it took a long time until frames came definitely into play. The first book concerning the topic was written by Young in 1980 [2]. In [12] Daubechies, Grossman and Meyer realized that frames can somehow be understood in a way very similar to expansions using bases. This established the link among frames and wavelets [5, 13, 14], and new breath was given to the research on frames. Applications of the “frame theory” have broadly appeared. Frames have been an active area of research, allowing the production of a plethora of theoretical results and applications. Good material on frames can be found in [4, 5, 13], and a book dedicated to this topic is [6]. We try to show a few of the relevant results regarding frames in this chapter.

### 1.10.1.1 Notation

In the following we use  $\mathbf{x}$ , to denote a signal in a discrete space, for a finite dimensional discrete space the same notation is employed. The value of the  $n$ th coordinate or sample of  $\mathbf{x}$  is referred to by  $x[n]$ . A bold capital is used for matrices, as  $\mathbf{G}$ . Continuous functions or signals are explicitly denoted with reference to the independent variable, as in  $x(t)$ . When just a letter like  $x$  is employed it is because the definition or result applies in both cases—discrete or continuous spaces.  $\|x\|$  refers to  $\langle x, x \rangle$ . The inner product  $\langle f, g \rangle$  is to be understood as  $\langle f(t), g(t) \rangle = \int f(t)g^*(t)dt$  in continuous spaces, and  $\langle \mathbf{f}, \mathbf{g} \rangle = \sum_n f[n]g^*[n]$  in discrete spaces.

---

## 1.10.2 Basic concepts

Let us try to briefly introduce what are frames. A frame of a space  $\mathbb{H}$  is a set of elements that spans  $\mathbb{H}$ . Therefore, a frame  $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$  (a set of elements  $g_k$  which are indexed by  $k \in \mathcal{K}$ ), can be used to express any  $x \in \mathbb{H}$  by means of

$$\mathbf{x} = \sum_{k \in \mathcal{K}} c_k \mathbf{g}_k, \quad (10.2)$$

where the  $c_k$  are called the frame coefficients.

The only restriction that we have imposed on  $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$  for it being a frame is that it should span  $\mathbb{H}$ . Therefore,  $\mathcal{G}$  is in general overcomplete, meaning that the set of frame coefficients  $\{c_k\}_{k \in \mathcal{K}}$  that can express  $x$  by means of Eq. (10.2) is not unique, as the overcompleteness of  $\mathcal{G}$  means that its elements are not linearly independent.

From above one sees that there may exist different ways to compute the frame coefficients  $c_k$  used in Eq. (10.2) to synthesize  $x$ . We employ interchangeably both terms “expansion” as well as representation

since all the frame elements (generally in larger number than the space dimension) may a priori be employed to express the signal. Due to overcompleteness one usually employs more elements than the support or dimension of the signal space  $\mathbb{H}$  to represent the signal. This is equivalent to saying that the frame coefficients for a given signal may be in larger number than the coefficients employed in the canonical signal representation.

**Example 10.1.** Consider the set of elements  $\mathcal{G}$  in  $\mathbb{R}^2$ ,  $\mathcal{G} = \{[0 \ 1/2], [1/4 \ -1/4], [-1/2 \ 0]\}$ . Any vector  $\mathbf{x} = [x[0], x[1]]$  projected into these elements gives a set of coefficients:

$$\tilde{\mathbf{c}} = \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1/2 \\ 1/4 & -1/4 \\ -1/2 & 0 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} = \begin{bmatrix} x[1]/2 \\ (x[0] - x[1])/4 \\ -x[0]/2 \end{bmatrix}. \quad (10.3)$$

The original vector can be obtained from  $\tilde{\mathbf{c}}$  using, for example

$$\begin{bmatrix} x[0] \\ x[1] \end{bmatrix} = \tilde{\mathbf{G}}^T \tilde{\mathbf{c}} = \begin{bmatrix} 0 & 0 & -2 \\ 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix} \text{ or } \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} = \tilde{\mathbf{G}}^T \tilde{\mathbf{c}} = \begin{bmatrix} 0 & 0 & -2 \\ 0 & -4 & -2 \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \end{bmatrix}, \quad (10.4)$$

among several other possibilities for  $\tilde{\mathbf{G}}^T$ . That is, the  $\tilde{\mathbf{G}}^T$  that reconstructs  $\mathbf{x}$  from its projection on a set  $\mathcal{G}$  is not unique.

Consider that  $\mathbf{G}$  is the matrix constructed from the frame elements by stacking them as rows.  $\mathbf{G}$  is used to compute the coefficients vector  $\tilde{\mathbf{c}}$  using

$$\tilde{\mathbf{c}} = \mathbf{G}\mathbf{x}. \quad (10.5)$$

Note that in the current example  $\dim(\mathbf{x}) = 2 \times 1$ ,  $\dim(\mathbf{G}) = 3 \times 2$ ,  $\dim(\tilde{\mathbf{c}}) = 3 \times 1$  and  $\dim(\tilde{\mathbf{G}}^T) = 2 \times 3$ . If a frame for  $\mathbb{R}^N$  has  $K > N$  elements, one has that  $\dim(\mathbf{x}) = N \times 1$ ,  $\dim(\mathbf{G}) = K \times N$ ,  $\dim(\tilde{\mathbf{c}}) = K \times 1$  and  $\dim(\tilde{\mathbf{G}}^T) = N \times K$ . That is, one has that the matrix  $\tilde{\mathbf{G}}^T$  used to obtain  $\mathbf{x} = \tilde{\mathbf{G}}^T \tilde{\mathbf{c}}$  must be such that  $\tilde{\mathbf{G}}^T \mathbf{G} = \mathbf{I}_N$  ( $\mathbf{I}_N$  is the identity matrix of dimension equating the dimension of the vectors in  $\mathcal{G}$ , which is 2 in the present example).

Obviously, the roles of the matrices  $\mathbf{G}$  and  $\tilde{\mathbf{G}}$  above can be interchanged. Since  $\tilde{\mathbf{G}}^T \mathbf{G} = \mathbf{I}_N$  one has that  $\mathbf{G}^T \tilde{\mathbf{G}} = \mathbf{I}_N$ , and thus one can employ  $\tilde{\mathbf{G}}$  as a projection operator on  $\mathbf{x}$  for finding a set of coefficients  $\mathbf{c}$ , and reconstruct  $\mathbf{x}$  using

$$\mathbf{x} = \mathbf{G}\mathbf{c}. \quad (10.6)$$

Note that in this case one can make

$$\tilde{\mathbf{G}}^T = \begin{bmatrix} 1 & 2 & -1 \\ 1 & -2 & -1 \end{bmatrix}, \quad (10.7)$$

or define  $\tilde{\mathbf{G}}$  as in (10.4). □

As we have seen in the example above, one should note that the set  $\mathbf{c}$  is redundant in the sense that the vectors employed to compute the  $\{c_k\}_{k \in \mathcal{K}}$  (onto which  $\mathbf{x}$  is projected) may be linearly dependent. One advantage of this is that the wrong computation of some coefficients of the expansion or even their losses may still allow the recovery of the original signal from the remaining coefficients within

an acceptable error, as one has different possible signal reconstruction operators provided by the frame concept. These ideas have led to applications of frame expansions for signal sampling with low quantization requirements [4], and also to the development of transmission systems that are robust to erasures (when data is lost or erased) over a network, which corresponds to the deletion of the correspondent frame elements [15–17].

### 1.10.2.1 The dual frame

The above example links to the strategy for computing frame coefficients using the inverse or dual frame [4–7], that provides the dual reconstruction formulas

$$\mathbf{x} = \sum_{k \in \mathcal{K}} \langle \mathbf{x}, \mathbf{g}_k \rangle \tilde{\mathbf{g}}_k = \sum_{k \in \mathcal{K}} \langle \mathbf{x}, \tilde{\mathbf{g}}_k \rangle \mathbf{g}_k. \quad (10.8)$$

From Eq. (10.8) one may define  $\tilde{\mathcal{G}} = \{\tilde{\mathbf{g}}_k\}_{k \in \mathcal{K}}$  as the inverse or dual frame to the frame  $\mathcal{G} = \{\mathbf{g}_k\}_{k \in \mathcal{K}}$ . For a given signal  $\mathbf{x}$ , since  $\mathcal{G}$  is overcomplete,  $\tilde{\mathcal{G}}$  is, in general, not unique [6], as shown in Example 10.1.

Considering a frame expansion of a signal—the set of coefficients  $c_k$ , the frame expansion can be viewed as a measure of “how much” the signal “has” of each frame element  $\{\mathbf{g}_k\}_{k \in \mathcal{K}}$ . As it can be readily seen, this property can be used to infer or analyze signal characteristics. Since in an overcomplete frame its elements are linearly dependent, at least two of its elements are not orthogonal. This implies that characteristics that are different, but similar to one another, can be effectively observed by projecting the signal into different frame elements that are similar to these characteristics. If an orthogonal basis was used instead, such characteristics would not be as evident from the projections, since each projection would have at most a small amount of each characteristic. This way, the similar characteristic would be mixed among different projections, and would therefore not be as evident as in the case of a redundant frame.

The reasoning in the above paragraph has been largely employed to justify the use of the so-called Gabor frames, that are discussed in Section 1.10.5.4. Gabor frames are constructed by modulation and translation of a prototype function, providing thus a time-frequency analysis capability [4, 18–22]. Gabor frames employ sets of time and frequency translated versions of a predefined function to analyze or synthesize a signal. These frames are named after Gabor due to their origins in [23]. The main idea is to explore functions that have some desired energy concentration in time and frequency domain to analyze the content or information that is represented in a signal. Several tools have been developed for achieving this, using the decomposition of a signal into a Gabor frame [24–26]. As the example regarding the Gaborgram in Section 1.10.6.2 ahead shows, the inverse frame elements may not resemble the frame elements and some care must be taken when using frame expansions to capture specific signal features.

Although the elements of a frame that are used to express a signal are in general selected *a priori*, frame expansions allow for the use of elements with special properties. The Balian-Low theorem [13] shows that it is not possible to construct a basis with elements that are well localized in both time and frequency domains simultaneously. Frames do not impose uniqueness to the signal representation, therefore the definition of the frame elements is less restrictive than it is for basis elements. Hence, we can achieve, for a frame, a better localization of its elements simultaneously in time and frequency domain than for a basis, better dealing with the limitations imposed by the uncertainty principle [4, 19, 21].

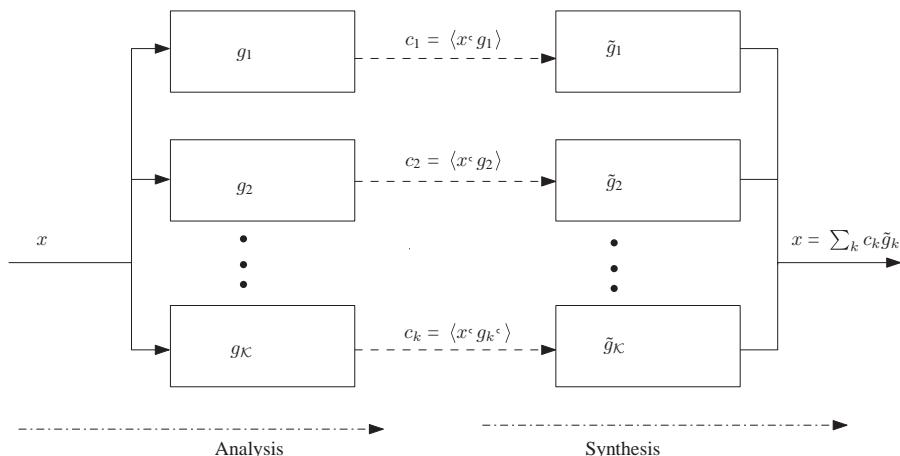
The property of frames discussed above has been largely employed by the so-called Windowed Fourier Analysis schemes [4,27]. In addition, since the frame coefficients may not be unique for a given signal, the set of frame coefficients can be altered or improved—considering the linear dependencies among frame elements—in order to highlight desired signal features. Moreover, frames can be designed for specific applications depending on the features that one desires to extract or analyze in the signal. Examples of such specific designs are ones using wavelets [4,5,7] and other time-frequency [4,28] or structurally/ geometrically oriented analysis techniques [29,30].

In general, the computational burden of obtaining signal representations into frames surpasses the computational demands of traditional expansions into basis, as, for example, the ones of the Fast Fourier Transform (FFT) [31]. However, as mentioned above, with frame expansions we can obtain a separation of signal structures that is much higher than the one that is possible using traditional approaches. Mallat [4] explores this in his proposed super-resolution techniques.

Due to the characteristics above, frame applications range from signal coding [4], signal sampling [1,4,6], signal analysis [4,5] and transient detection [24,25], to communication systems design [6,32]. Frames have also been related to the analysis and design of Filter Banks [33,34].

### 1.10.2.2 Signal analysis and synthesis using frames

A frame or its dual can be employed to analyze a signal and obtain the signal expansion, i.e., the frame coefficients, and to synthesize it from the coefficients obtained in the analysis process. This is shown in Figure 10.1. As it can be noticed, from Eq. (10.22), the roles of  $\{g_k\}_{k \in \mathcal{K}}$  and  $\{\tilde{g}_k\}_{k \in \mathcal{K}}$  can be interchanged in Figure 10.1, providing different perspectives for signal analysis.



**FIGURE 10.1**

Analysis and synthesis of a signal using a frame and its inverse—as noted in the text, the roles can be interchanged.

One notes that the “system model” of Figure 10.1 is very similar to the one employed in several signal processing tasks—for example in filter banks. This resemblance between frames and several signal processing tasks provides a large range of potential applications for frames. To highlight an important feature, the frame elements may be designed to highlight or analyze desired signal features that may be “overlapping” or linearly dependent.

In the scenario of filter banks, a relevant and special case is the one of a “perfect reconstruction filter bank” [31]. In this case the analysis and reconstruction filter banks correspond to a pair of dual frames. A more detailed analysis of the input-output mapping in this case is provided in Section 1.10.4.1 where we discuss frames of discrete spaces.

Some examples of common frame constructions for signal analysis purposes are Gabor and Wavelet frames. These provide expansion systems or representations that have different supports on the time-frequency plane. We will discuss them in Sections 1.10.5.4 and 1.10.5.5 respectively.

### 1.10.3 Relevant definitions

Now, that we have discussed some basic properties of frames, we are ready to present a more formal definition.

**Definition 10.1.** A sequence of elements  $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$  in a space  $\mathbb{H}$  is a *frame* for  $\mathbb{H}$  if there exist constants  $A$  and  $B$ ,  $0 < A < B < \infty$ , such that [4–7]

$$A\|x\|^2 \leq \sum_{k \in \mathcal{K}} |\langle x, g_k \rangle|^2 \leq B\|x\|^2, \quad \forall x \in \mathbb{H}. \quad (10.9)$$

- i. The numbers  $A$  and  $B$  are called the lower and upper *frame bounds*, respectively, and are not unique. The optimal lower frame bound is the supremum on all  $A$  and the optimal upper frame bound is the infimum on all  $B$  [6].
- ii. It is said that the frame is normalized [5] if  $\|g_k\| = 1$ ,  $\forall k \in \mathcal{K}$ . □

A commonly addressed problem, related to frames, is how to determine if a given sequence of elements  $\{g_k\}_{k \in \mathcal{K}}$  in  $\mathbb{H}$  is a frame of  $\mathbb{H}$ . This is often referred to as the frame characterization problem [6]. This characterization is commonly accomplished by the frame bounds. From Eq. (10.9), one notes that if for any signal one obtains  $A = 0$  then the signal is orthogonal to all the elements  $\{g_k\}_{k \in \mathcal{K}}$  and the signal can not be represented using these elements. That is the reason for the requirement that  $A > 0$ . Also, if for a given signal there is no upper bound in Eq. (10.9) then this means that the elements are too “dense” for that signal and the frame does not provide a stable representation.

Given a set of elements in a space  $\mathbb{H}$ , in order to verify if it is or is not a frame one may compute the frame bounds. It is in general much easier to find the upper frame bound  $B$  than the lower frame bound  $A$ .

**Example 10.2.** For instance, note that in a space  $\mathbb{R}^N$  any quantity  $M > N$  of finite norm vectors will always provide a  $B < \infty$ . However, for the lower bound defines if the set spans or not  $\mathbb{H}$  and in this case a more careful analysis is required. □

#### 1.10.3.1 The frame operator

Let us now define the frame operator and state the frame decomposition result.

**Definition 10.2.** Define the *frame operator*  $S\{\cdot\}$  as

$$S : \mathbb{H} \rightarrow \mathbb{H}, S\{x\} = \sum_{k \in \mathcal{K}} \langle x, g_k \rangle g_k. \quad (10.10)$$

From this definition some properties hold [6]:

- i. The frame operator is bounded, invertible, self-adjoint and positive. Therefore, it assumes an inverse  $S^{-1}\{\cdot\}$ . This is referred to as the *inverse frame operator*.

$$x = S\{S^{-1}\{x\}\} = \sum_{k \in \mathcal{K}} \langle S^{-1}\{x\}, g_k \rangle g_k. \quad (10.11)$$

- ii. If the lower and upper frame bound of  $\{g_k\}_{k \in \mathcal{K}}$  are, respectively  $A$  and  $B$ , then  $\{S^{-1}\{g_k\}\}_{k \in \mathcal{K}}$  is a frame with bounds  $B^{-1}$  and  $A^{-1}$ , and the frame operator for  $\{S^{-1}\{g_k\}\}_{k \in \mathcal{K}}$  is  $S^{-1}$ .  $\square$

From the definition of the frame operator one has the *frame decomposition* result that provides the reconstruction formulas [6]

$$x = \sum_{k \in \mathcal{K}} \langle x, g_k \rangle S^{-1}\{g_k\} = \sum_{k \in \mathcal{K}} \langle x, S^{-1}\{g_k\} \rangle g_k \quad (10.12)$$

Note that this provides a way to compute the  $\tilde{g}_k = S^{-1}\{g_k\}$  which are the elements of the inverse frame or the so-called canonical dual.

**Example 10.3.** Suppose that a normalized frame  $\{g_k\}_{k \in \mathcal{K}}$  is used to express the information of a signal  $x$ , to transmit or store this information and to reconstruct  $x$ . In this scenario, assume that one transmits/stores the frame coefficients

$$c_k = \langle x, S^{-1}\{g_k\} \rangle, \quad (10.13)$$

where we employ the inverse frame operator. From the definitions one has that

$$x = \sum_k c_k g_k. \quad (10.14)$$

However, due to quantization coefficients are corrupted by noise. In this case one must reconstruct  $\hat{x}$

$$\hat{x} = \sum_{k \in \mathcal{K}} (c_k + w_k) g_k = \sum_{k \in \mathcal{K}} (\langle x, S^{-1}\{g_k\} \rangle + w_k) g_k. \quad (10.15)$$

Obviously, due to the frame reconstruction one has that

$$\hat{x} = x + \sum_{k \in \mathcal{K}} w_k g_k. \quad (10.16)$$

The reconstructed signal is corrupted by noise.

One should note that in this scenario, we know that the  $c_k$  (obtained by means of Eq. (10.13) are the expansion of a signal over a frame). However, this may not be the case for the set of coefficients

$$\left\{ \langle x, S^{-1} \{g_k\} \rangle + w_k \right\}_{k \in \mathcal{K}} \quad (10.17)$$

used to reconstruct  $\hat{x}$  in Eq. (10.15). Assuming that  $w_k \in \ell^2(\mathbb{N})$ —i.e., the noise is bounded, that fact may be compensated by projecting  $\{\langle x, S^{-1} \{g_k\} \rangle + w_k\}_{k \in \mathcal{K}}$  using the operator [4,6]:

$$Q \{c_k\} = \left\{ \left\langle \sum_k c_k S^{-1} \{g_k\}, g_j \right\rangle \right\}_{j \in \mathcal{K}}. \quad (10.18)$$

Using this operator in the projected-quantized frame coefficients scenario above one has that (apply  $Q\{\cdot\}$  to Eq. (10.17))

$$Q \left\{ \langle x, S^{-1} \{g_k\} \rangle + w_k \right\}_{k \in \mathcal{K}} = \{c_k\}_{k \in \mathcal{K}} + Q \{w_k\}_{k \in \mathcal{K}}. \quad (10.19)$$

Using this, the signal can be reconstructed by means of

$$\hat{x} = \sum_{k \in \mathcal{K}} (c_k + Q \{w_k\}) g_k = x + \sum_{k \in \mathcal{K}} Q \{w_k\} g_k. \quad (10.20)$$

Supposing that the frame  $\{g_k\}_{k \in \mathcal{K}}$  is normalized, considering the quantization noise in Eq. (10.15) to be white and zero-mean, using  $E[\cdot]$  to denote the expected value, then in [4] it is shown that

$$E \left[ |Q \{w\}|^2 \right] \leq \frac{1}{A} E \left[ |w|^2 \right]. \quad (10.21)$$

That is, the resulting noise when reconstructing the signal by means of the processed coefficients as in Eq. (10.19) is reduced if  $A$  (the lower frame bound) is greater than one.  $\square$

### 1.10.3.2 The inverse frame

**Definition 10.3.** The *inverse frame*  $\tilde{\mathcal{G}} = \{\tilde{g}_k\}_{k \in \mathcal{K}}$  to a frame  $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$ , is the one that provides the reconstruction formulas

$$x = \sum_{k \in \mathcal{K}} \langle x, \tilde{g}_k \rangle g_k = \sum_{k \in \mathcal{K}} \langle x, g_k \rangle \tilde{g}_k. \quad (10.22)$$

$\square$

**Example 10.4.** Let us go back to Example 10.1, where we have seen two different possibilities for “inversion” of the frame given by the rows of the matrix

$$\mathbf{G} = \begin{bmatrix} 0 & 1/2 \\ 1/4 & -1/4 \\ -1/2 & 0 \end{bmatrix}. \quad (10.23)$$

However, one can now find its canonical dual, associated with the inverse frame operator. Using a pseudo-inversion algorithm [4] one finds the pseudo-inverse  $\mathbf{G}^+$  to  $\mathbf{G}(\mathbf{G}^+\mathbf{G} = \mathbf{I}_2)$ :

$$\mathbf{G}^+ = \begin{bmatrix} 1/3 & 2/3 & -5/3 \\ 5/3 & -2/3 & -1/3 \end{bmatrix}. \quad (10.24)$$

Note that, this is just another example of possible frame inversion, as it is shown in [4], since  $\mathbf{G}$  assumes an infinite number of left inverses.  $\square$

**Definition 10.4.** When  $A = B$  the frame is said to be *tight* [6] and

$$S^{-1}\{\cdot\} = S\{\cdot\}/A. \quad (10.25)$$

In addition:

- i. Frames for which  $A \approx B$  are said to be *snug* [34] and for these  $S^{-1}\{\cdot\} \approx S\{\cdot\}/A$ .  $\square$

**Definition 10.5.** A frame  $\mathcal{G} = \{g_k\}_{k \in \mathcal{K}}$  is said to be *normalized* when

$$\|g_k\| = c, \forall k \in \mathcal{K}, \quad c > 0 \text{ is a real constant.} \quad (10.26) \quad \square$$

### 1.10.3.3 Characterization of frames: basic property

Frame bounds provide limits for the energy scaling of signals when they are represented using the projections into the frame elements. Due to that, frames are often characterized in terms of their frame bounds. It is very common to define the *frame bounds ratio*  $A/B$  [13]. It is also possible to define the “tightness” of a frame from its frame bounds ratio: the closer that  $A$  and  $B$  are, then the tighter the frame is.

Daubechies [5] shows that the use of the frame operator (Eq. (10.10)) gives

$$S\{x\} = \frac{A+B}{2}(x - R\{x\}), \text{ i.e., } x = \frac{2}{A+B} \sum_{k \in \mathcal{K}} \langle x, g_k \rangle g_k + R\{x\}, \quad (10.27)$$

where  $R\{x\}$  can be understood as the error incurred in reconstructing  $x$  from the projections of  $x$  into the frame elements instead of into the inverse frame elements.

In addition, she also shows that

$$R\{x\} = I\{x\} - \frac{2}{A+B} S\{x\}, \quad (10.28)$$

where  $I\{x\}$  is the identity operator. Therefore one has that

$$-\frac{B-A}{B+A} I\{x\} \leq R\{x\} \leq \frac{B-A}{B+A} I\{x\} \quad \rightarrow \quad \|R\{x\}\| \leq \frac{B-A}{B+A} \|x\|. \quad (10.29)$$

Hence, one sees that if  $B/A$  is close to one, then the error incurred in the reconstruction by equating the inverse frame to the direct frame is small. This error gets smaller as  $A$  and  $B$  become closer.

Therefore, from an analysis-synthesis perspective if a frame is tight there is no need to find its inverse. Tight frames are self-dual [35], that is, the inverse frame to a tight frame is a scaled version of the frame itself.

For normalized frames, Daubechies calls  $\frac{A+B}{2}$  the frame redundancy. Several studies have been developed on normalized tight frames. For example [36] shows that these frames have some noise reduction property. That is, if a given reconstruction distortion is required one can use less bits to represent the frame coefficients in comparison to the precision required to represent basis coefficients. This is related to the acquisition-quantization problem we have discussed previously.

**Example 10.5.** Let  $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$  be an orthonormal basis of the three-dimensional space. Define the vectors:

$$\begin{aligned}\mathbf{g}_1 &= \frac{\sqrt{2}}{2} \mathbf{e}_1 + \frac{\sqrt{2}}{2} \mathbf{e}_2 & \mathbf{g}_2 &= \frac{\sqrt{2}}{2} \mathbf{e}_2 + \frac{\sqrt{2}}{2} \mathbf{e}_3 & \mathbf{g}_3 &= \frac{\sqrt{2}}{2} \mathbf{e}_1 + \frac{\sqrt{2}}{2} \mathbf{e}_3 \\ \mathbf{g}_4 &= \frac{1}{2} \mathbf{e}_1 + \frac{\sqrt{3}}{2} \mathbf{e}_2 & \mathbf{g}_5 &= \frac{1}{2} \mathbf{e}_2 + \frac{\sqrt{3}}{2} \mathbf{e}_3 & \mathbf{g}_6 &= \frac{\sqrt{3}}{2} \mathbf{e}_1 + \frac{1}{2} \mathbf{e}_3.\end{aligned}$$

For any  $\mathbf{x} \in \mathbb{R}^3$  we have that

$$\begin{aligned}\sum_k |\langle \mathbf{x}, \mathbf{g}_k \rangle|^2 &= \left| \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_1 \right\rangle + \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_2 \right\rangle \right|^2 + \left| \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_2 \right\rangle + \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_3 \right\rangle \right|^2 \\ &\quad + \left| \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_1 \right\rangle + \left\langle \mathbf{x}, \frac{\sqrt{2}}{2} \mathbf{e}_3 \right\rangle \right|^2 + \left| \left\langle \mathbf{x}, \frac{1}{2} \mathbf{e}_1 \right\rangle + \left\langle \mathbf{x}, \frac{\sqrt{3}}{2} \mathbf{e}_2 \right\rangle \right|^2 \\ &\quad + \left| \left\langle \mathbf{x}, \frac{1}{2} \mathbf{e}_2 \right\rangle + \left\langle \mathbf{x}, \frac{\sqrt{3}}{2} \mathbf{e}_3 \right\rangle \right|^2 + \left| \left\langle \mathbf{x}, \frac{\sqrt{3}}{2} \mathbf{e}_1 \right\rangle + \left\langle \mathbf{x}, \frac{1}{2} \mathbf{e}_3 \right\rangle \right|^2 \\ &= 2 |\langle \mathbf{x}, \mathbf{e}_1 \rangle + \langle \mathbf{x}, \mathbf{e}_2 \rangle + \langle \mathbf{x}, \mathbf{e}_3 \rangle|^2 \\ &= 2 \|\mathbf{x}\|.\end{aligned}$$

Therefore the six vectors define a tight frame with  $A = B = 2$ . This number can in some sense be understood as a measure of redundancy of the frame expansion [4].  $\square$

Tight frames turned out to be relevant, due to their operational simplicity and applications. For example tight frames were shown to be the most resistant to coefficients erasures [15]. Erasures means that when transiting information over a network one knows that the information is unreliable. If a frame is used to compute the information that is transmitted over the network then the reconstruction error under erasures is lower for a tight frame than for non tight ones. Tight frames can also provide sparse signal decompositions [37]. Due to these and other interesting properties a lot of attention has been given to the construction of tight frames [16, 37–40].

## 1.10.4 Some computational remarks

In general signal processing algorithms are implemented in digital processors. In this case, the signal is discrete and some simplifications take place.

### 1.10.4.1 Frames in discrete spaces

Consider the case of discrete spaces as the  $\ell^2(\mathbb{Z})$  (the space of squareable summable sequences). In general, in such spaces, signal analysis is accomplished by using sliding-windows. This is actually the case for Filter-Banks [31], using the structure depicted in Figure 10.1, the synthesis equation becomes [33,34]

$$\mathbf{x}[n] = \sum_{k=0}^{K-1} \sum_{m=-\infty}^{\infty} c_{k,m} \mathbf{g}_{k,m}[n]. \quad (10.30)$$

In the above equation each signal's sample  $\mathbf{x}[n], n \in \mathbb{Z}$ , is synthesized as a sum of the samples  $\mathbf{g}_{k,m}[n] = \mathbf{g}_k[n - mN], N \leq K$ ,  $K$  is the cardinality of the frame, the  $\mathbf{g}_{k,m}$  are translated versions of the  $\mathbf{g}_k$ , referring to the  $m$ th  $N$ -length signal block.

Similar concepts can be brought to frames in  $\ell^2(\mathbb{Z})$ , bearing in mind the correct usage of a frame and its dual. In this case, one should note that Eq. (10.30) must be understood as

$$\mathbf{x}[n] = \sum_{k=0}^{K-1} \sum_{m=-\infty}^{\infty} \langle \mathbf{x}, \tilde{\mathbf{g}}_{k,m} \rangle \mathbf{g}_{k,m}[n]. \quad (10.31)$$

That is, as discussed in Section 1.10.2.2, the frame and its dual should be employed in the two distinct tasks of analysis/decomposition and synthesis/recomposition of  $\mathbf{x}$ .

The previous definitions on frames considered any space  $\mathbb{H}$ . When considering  $\ell^2(\mathbb{Z})$ , the space in which digital signal processing applications exist (as well as in finite vector spaces—which are discussed in the next subsection) using finite support vectors may be handy, as in FIR Filters.

A signal  $\mathbf{x} \in \ell^2(\mathbb{Z})$  can be expressed using Eq. (10.30) if the family of functions [33]

$$\mathcal{G} = \{ \mathbf{g}_{k,m} : \mathbf{g}_{k,m}[n] = \mathbf{g}_k[n - mN], k = 0, 1, \dots, K-1, m \in \mathbb{Z} \} \quad (10.32)$$

is a frame for  $\ell^2(\mathbb{Z})$ . This result provides an important feature of frames for  $\ell^2(\mathbb{Z})$ .

**Definition 10.6.** A set of vectors  $\mathcal{G}$  as defined in Eq. (10.32) (block translated) is a *frame* of  $\ell^2(\mathbb{Z})$ , if there exist constants  $0 < A < B < \infty$  such that for any  $\mathbf{x} \in \ell^2(\mathbb{Z})$  one has

$$A \|\mathbf{x}\|^2 \leq \sum_{k=0}^{K-1} \sum_{m=-\infty}^{\infty} |\langle \mathbf{x}, \mathbf{g}_{k,m} \rangle|^2 \leq B \|\mathbf{x}\|^2. \quad (10.33)$$

□

One should note that if Eq. (10.33) holds for  $\mathcal{G}$  as in Eq. (10.32), then there exists another frame

$$\tilde{\mathcal{G}} = \{ \tilde{\mathbf{g}}_{k,m} : \tilde{\mathbf{g}}_{k,m}[n] = \tilde{\mathbf{g}}_k[n - mN], k = 0, 1, \dots, K-1, m \in \mathbb{Z} \} \quad (10.34)$$

that can be employed for obtaining the coefficients  $c_{k,m}$  in Eq. (10.31), Obviously, the roles of  $\mathcal{G}$  and  $\tilde{\mathcal{G}}$  can be interchanged, providing

$$\mathbf{x}[n] = \sum_{k=0}^{K-1} \sum_{m=-\infty}^{\infty} \langle \mathbf{x}, \mathbf{g}_{k,m} \rangle \tilde{\mathbf{g}}_{k,m}[n]. \quad (10.35)$$

As it can be noted, given  $\mathcal{G}$  as in Eq. (10.32), the  $\tilde{\mathcal{G}}$  as (10.34) satisfying Eqs. (10.31) and (10.35) is not unique.

#### 1.10.4.2 Finite vector spaces

In a finite vector space  $\mathbb{H}^N$  one in general restricts the frame to have  $K$  elements; and thus Eq. (10.9) becomes

$$A\|\mathbf{x}\|^2 \leq \sum_{k=1}^K |\langle \mathbf{x}, \mathbf{g}_k \rangle|^2 \leq B\|\mathbf{x}\|^2, \quad \mathbf{x} \in \mathbb{H}^N. \quad (10.36)$$

In this case, as motivated in previous examples some computational simplifications take place.

**Definition 10.7.** *Analysis and Synthesis operators:*

- i. Let the *synthesis operator* be

$$T\{\cdot\} : \mathbb{C}^K \rightarrow \mathbb{H}^N, \quad T\{c_k\}_{k=1}^K = \sum_{k=1}^K c_k \mathbf{g}_k, \quad (10.37)$$

where  $\mathbb{C}^K$  is a  $K$ -dimensional complex vector space.

- ii. Let the *analysis operator*  $T^*\{\cdot\}$  (adjunct operator of  $T\{\cdot\}$ ) be given by

$$T^*\{\cdot\} : \mathbb{H}^N \rightarrow \mathbb{C}^K, \quad T^*\{\mathbf{x}\} = \{\langle \mathbf{x}, \mathbf{g}_k \rangle\}_{k=1}^K. \quad (10.38)$$

- iii. The operator  $T\{\cdot\}$  synthesizes  $\mathbf{x}$  from the frame coefficients  $c_k$  that are obtained by the analysis operator of a dual frame given by

$$\tilde{T}^*\{\cdot\} : \mathbb{H}^N \rightarrow \mathbb{C}^K, \quad \tilde{T}^*\{\mathbf{x}\} = \{\langle \mathbf{x}, \tilde{\mathbf{g}}_k \rangle\}_{k=1}^K. \quad (10.39)$$

- iv. Using the analysis and synthesis operators the frame operator is then given by

$$S : \mathbb{H}^N \rightarrow \mathbb{H}^N, \quad S\{\mathbf{x}\} = T\{T^*\{\mathbf{x}\}\} = \sum_{k=1}^K \langle \mathbf{x}, \mathbf{g}_k \rangle \mathbf{g}_k. \quad (10.40)$$

□

In vector spaces the operators  $T\{\cdot\}$  and  $T^*\{\cdot\}$  can be interpreted as matrices [6], as in the examples provided before.

Assuming that  $\mathbf{x}$  is a column vector that is  $\dim(\mathbf{x}) = N \times 1$  and the coefficients are organized in a column vector  $\mathbf{c} (\dim(\mathbf{c}) = K \times 1)$  as well. One has that

$$\begin{aligned}\mathbf{x} & \quad \dim(\mathbf{x}) = N \times 1 \\ \mathbf{c} & \quad \dim(\mathbf{c}) = K \times 1 \\ \mathbf{g}_k & \quad \dim(\mathbf{g}_k) = N \times 1 \\ \mathbf{T} &= \begin{bmatrix} \mathbf{g}_1^T \\ \vdots \\ \mathbf{g}_K^T \end{bmatrix} \quad \dim(\mathbf{T}) = K \times N\end{aligned}\tag{10.41}$$

$$T\{\mathbf{c}\} = \mathbf{T}^T \mathbf{c} \quad \dim(\mathbf{T}^T) = N \times K\tag{10.42}$$

$$T^*\{\mathbf{x}\} = \mathbf{T} \mathbf{x} \quad \dim(\mathbf{T}^*) = K \times N\tag{10.43}$$

$$S\{\mathbf{x}\} = T\{\mathbf{T}\{\mathbf{x}\}\} = \mathbf{T}^T \mathbf{T} \mathbf{x} = \mathbf{S} \mathbf{x} \quad \dim(\mathbf{S}) = N \times N\tag{10.44}$$

$$S^{-1}\{\mathbf{x}\} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{x} = \mathbf{S}^{-1} \mathbf{x} \quad \dim(\mathbf{S}^{-1}) = N \times N.\tag{10.45}$$

The dual frame elements can be computed by means of

$$\tilde{\mathbf{g}}_k = S^{-1}\{\mathbf{g}_k\} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{g}_k,\tag{10.46}$$

and we have that

$$\tilde{T}\{\mathbf{c}\} = \tilde{\mathbf{T}}^T \mathbf{c} = \mathbf{T} (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{c} \quad \dim(\tilde{\mathbf{T}}^T) = N \times K\tag{10.47}$$

$$\tilde{T}^*\{\mathbf{x}\} = \tilde{\mathbf{T}}^* \mathbf{x} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T} \mathbf{x} \quad \dim(\tilde{\mathbf{T}}^*) = K \times N.\tag{10.48}$$

**Example 10.6.** As we have seen above  $\mathbf{S} = \mathbf{T}^T \mathbf{T}$ . Let  $\rho_i$  be the eigenvalues of  $\mathbf{S}$ , then the frame bounds are given by  $A = \min_i \rho_i$ , and  $B = \max_i \rho_i$  [6]. Thus, if  $\mathbf{T}^T \mathbf{T} = A \mathbf{I}_N$  ( $\mathbf{I}_N$  is the identity matrix of size  $N$ ) then the frame is tight ( $A = B$ ). Hence, for a tight frame  $\mathbf{S}^{-1} \mathbf{S} = \mathbf{I}_N$ .  $\square$

**Example 10.7.** The Gram matrix or Gramian of a sequence of vectors  $\{\mathbf{g}_k\}_{k \in \mathcal{K}}$  is the matrix  $G$ , whose elements are defined by  $g_{i,j} = \langle \mathbf{g}_i, \mathbf{g}_j \rangle_{i,j \in \mathcal{K}}$ . It has been employed to analyze different frames characteristics. For example, in [41], it is employed to investigate the symmetry properties of tight frames, providing tight frames design procedures. In [37] the Gramian is also employed for analyzing and designing frames with desired features.

We have seen the analysis, synthesis, and frame operators, and its inverse frames counterparts. As one may notice, a lot of the frame characteristics can be extracted from them. Now, define the Gram matrix  $G = \mathbf{T} \mathbf{T}^T$ . One readily notes that  $\dim(G) = K \times K$  and that each of its entries  $g_{i,j} = \langle \mathbf{g}_i, \mathbf{g}_j \rangle_{i,j \in \mathcal{K}}$ . This Gram matrix provides how similar are the frame elements one to another, therefore, it can be understood as an indicative of the connections among the different  $c_k$ .  $\square$

**Example 10.8.** In a vector space if the lower frame bound  $A$  were zero for a signal  $\mathbf{x}$  then the frame elements would all be orthogonal to  $\mathbf{x}$ , and the frame would not be capable of representing  $\mathbf{x}$ . In addition, note that if the frame has a finite set of elements, then, necessarily, due to the Cauchy-Schwartz inequality, the upper frame bound condition will be always satisfied.  $\square$

## 1.10.5 Construction of frames from a prototype signal

It is common to process signals in order to search for specific patterns in time, frequency or even scale. In this case, one often a priori specifies a set of desired behaviors or patterns to be searched in signals to be analyzed. These are specified as a set of pre-defined waveforms. If synthesis is not required the restrictions on the set are milder than in the case when synthesis is required. In the later case, one should guarantee the set of pre-defined functions or waveforms to be a frame. This is what we discuss now. We suppose a signal processing scenario where a given signal has to be analyzed and synthesized using a set of pre-defined waveforms that are derived from operations such as change of position in time (translation), change of frequency (modulation) and change of size (dilation) over a prototype signal. In this section we discuss some conditions that make a frame to be generated when these operations are applied to a prototype waveform.

### 1.10.5.1 Translation, modulation and dilation operators

There are several ways to construct frames from operations on a prototype signal, some of them are based on translations, modulations and dilations of a function  $g(t) \in L^2(\mathbb{R})$  (the space of square integrable functions) [6].

**Definition 10.8.** *Translation by  $a \in \mathbb{R}$ :*

$$T_a : L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R}), \quad T_a g(t) = g(t - a). \quad (10.49)$$

□

**Definition 10.9.** *Modulation by  $b \in \mathbb{R}$ :*

$$E_b : L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R}), \quad E_b g(t) = g(t)e^{2\pi jb t}. \quad (10.50)$$

□

**Definition 10.10.** *Dilation by  $c \in \mathbb{R} - \{0\}$ :*

$$D_c : L^2(\mathbb{R}) \rightarrow L^2(\mathbb{R}), \quad D_c g(t) = \frac{1}{\sqrt{|c|}} g\left(\frac{t}{c}\right). \quad (10.51)$$

□

### 1.10.5.2 Common frame constructions

The most common approaches to construct frames from a prototype are:

- A frame in  $L^2(\mathbb{R})$  constructed by translations of a given function  $g(t) \in L^2(\mathbb{R})$ , through

$$\{T_{na}g(t)\}_{n \in \mathbb{Z}}, \quad \text{with } a > 0, \quad (10.52)$$

is called a *frame of translates* [6];

- A *Weyl-Heisenberg* or *Gabor frame* is a frame in  $L^2(\mathbb{R})$  constructed from a fixed function  $g(t) \in L^2(\mathbb{R})$ , by means of

$$\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}, \quad \text{with } a, b > 0. \quad (10.53)$$

Such frame is also called a *Gabor System* or a *Windowed Fourier Frame* [4–6, 42] due to its connection to time-frequency analysis techniques;

- A frame constructed by dilations and translations of a prototype (mother) function  $g(t) \in L^2(\mathbb{R})$  by

$$\{T_{nac^j}D_{c^j}g(t)\}_{j,n \in \mathbb{Z}} = \left\{ c^{\frac{-j}{2}} g\left(c^{-j}t - na\right) \right\}_{j,n \in \mathbb{Z}}, \quad \text{with } c > 1, \text{ and } a > 0. \quad (10.54)$$

is called a *wavelet frame* [4–7].

In the following subsections we enumerate some conditions for obtaining frames in the above ways and some examples of their applications or relevance.

### 1.10.5.3 Frames of translates

Digital signal processing is heavily based on signal sampling, that is, the capability of sampling a function (whose independent variable may not be time) using regular intervals and reconstructing the signal from the samples [4, 6, 31, 43]. A sample corresponds to a value that in turn represents an evaluation/measurement of the function/signal. The operation of sampling can be modelled as the projection of the function/signal over a pre-defined element. Regular sampling corresponds to translating the element into different positions regularly sampled and taking a sample (measurement/projection) at the correspondent position. Although the following results employ time as the independent variable it could be any other.

**Definition 10.11.** A *frame of translates* is a frame for  $L^2(\mathbb{R})$  obtained through operations  $\{T_{na}g(t)\}_{n \in \mathbb{Z}}$ , on a fixed function  $g(t)$  with  $a > 0$ .  $\square$

Which conditions should hold on  $g(t)$  in order to  $\{T_{na}g(t)\}_{n \in \mathbb{Z}}$  being a frame? In this case it can be shown that [6]  $\{T_{na}g(t)\}_{n \in \mathbb{Z}}$  is a frame with bounds  $A$  and  $B$  if and only if

$$aA \leq G(\omega) \leq aB, \quad \omega \in [0, 1] - \mathcal{N}, \quad \mathcal{N} = \{\gamma \in [0, 1] : G(\omega) = 0\}. \quad (10.55)$$

Frames of translates are intimately related with sampling. The example below highlights their connection with Shannon Sampling Theorem [43].

**Example 10.9.** In this example one employs the Shannon Sampling Theorem [43, 44]. Let a band limited function of time  $s(t)$  be sampled using a train of impulses at a rate  $1/T$ . Assume that the signal bandwidth is  $W$ . If  $T < \frac{1}{2W}$ , one can reconstruct the signal using

$$s(t) = \sum_n s(nT) \frac{\sin[(t - nT)\frac{\pi}{T}]}{(t - nT)\frac{\pi}{T}}. \quad (10.56)$$

This is so because the infinite set of sinc pulses centered at  $nT$  provides a basis/frame of the  $W$  Hz band-limited space.

Define  $R$  as the “amount of over-sampling”

$$R = \frac{1}{2WT}, \quad \text{for } R \geq 1, \quad (10.57)$$

One notes that the Nyquist rate [31] implies  $2WT = 1$  and  $R \geq 1$  makes  $1/T \geq 2W$ . One has that [44]

$$s(t) = \frac{1}{R} \sum_n s(nT) \frac{\sin[(t - nT)\frac{\pi}{RT}]}{(t - nT)\frac{\pi}{RT}}. \quad (10.58)$$

Note that for  $R \geq 1$ , two sinc pulses centered at  $kT$  and  $lT$  ( $k, l \in \mathbb{Z}$ ) not just overlap, but are actually not orthogonal. Indeed, the set of  $T$  temporally-spaced sinc pulses is an overcomplete basis of the  $W$ -bandlimited signal space, in the case  $R \geq 1$ . That is, the sinc pulses centered at  $nT$  with  $R > 1$  are not linearly independent. However, the infinite set of sinc pulses above is shown to be a tight frame and  $R$  is interpreted as a redundancy factor for the frame [10,44]. One notes that for  $R = 1$  (10.58) equates (10.56) where the signal samples occur where the sinc pulses equate one (for  $t = nT$ ) or zero (other cases). However, as  $R$  increases the sinc pulse centered at  $nT$  gets wider and is not zero anymore for  $t - nT = kT$ ,  $k \in \mathbb{Z}^*$ .  $\square$

#### 1.10.5.4 Gabor frames

While the introduction of the frame concept remounts to 1952 [1], according to Christensen [6], the first mention of what are now called Gabor or Weyl-Heisenberg frames can be traced back even before that to the work of Gabor on communications [23] in 1946 and to the book of von Neumann on quantum mechanics originally published in 1932 [45].

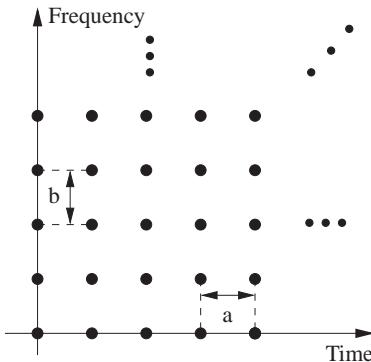
Let us simplify the idea of Gabor and simply state that its vision was to represent a signal  $s(t)$  by means of

$$s(t) = \sum_m \sum_n \tilde{c}_{m,n} E_{mb} T_{na} g(t). \quad (10.59)$$

The function  $g(t)$  is chosen to be a Gaussian due to its time-frequency concentration [19,21,46]. This representation can provide an analysis of the time frequency content around the points  $(na, mb)$  in the time-frequency plane. The resemblance between this idea and the frame definition sparks, and then a straightforward definition emerges.

**Definition 10.12.** A *Gabor frame* is a frame for  $L^2(\mathbb{R})$  obtained through operations  $\{E_{mb} T_{na} g(t)\}_{m,n \in \mathbb{Z}}$ , on a fixed function  $g(t)$  with  $a, b > 0$ .  $\square$

Now, one has the problem of guaranteeing that the set  $\{E_{mb} T_{na} g(t)\}_{n,m \in \mathbb{Z}}$  is capable of representing any signal  $s(t) \in L^2(\mathbb{R})$ , i.e., is the set  $\{E_{mb} T_{na} g(t)\}_{n,m \in \mathbb{Z}}$  a frame of  $L^2(\mathbb{R})$ ? In addition, how does one compute the set  $\{\tilde{c}_{m,n}\}_{n,m \in \mathbb{Z}^2}$  that allows the synthesis of a signal using Eq. (10.59)? The answer to the later question lies in the concept of inverse frame. But, we will start from the first question: for a given  $g(t)$ , when  $\{E_{mb} T_{na} g(t)\}_{m,n \in \mathbb{Z}}$  is a frame? The answer depends on a complicated interplay between  $g(t)$ ,  $a$  and  $b$ . For example in [47] a set of non-intuitive conditions was shown to hold on  $a$  and  $b$  to generate a Gabor System based on the characteristic function. In what follows, several results collected from the frame literature are presented, which provide either sufficient or necessary conditions for  $\{E_{mb} T_{na} g(t)\}_{m,n \in \mathbb{Z}}$  to constitute a frame.

**FIGURE 10.2**

Gabor system elements location in the time-frequency plane.

For  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  to compose a frame it is necessary that  $ab \leq 1$  [4–6]. That is, if  $ab > 1$  a frame will not be obtained; however, the assumption  $ab \leq 1$  does not guarantee the generation of a frame for any  $g(t)$ , see for example [47]. It should be observed that  $ab$  is a measure of the density of the frame in the time-frequency plane [4,5,19–21,42]; the smaller  $ab$  is the denser is the frame. Figure 10.2 illustrates this concept.

If  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  constitutes a frame then the frame bounds necessarily satisfy [6]

$$\forall t \in \mathbb{R}, \quad A \leq \frac{1}{b} \sum_n |g(t - na)|^2 \leq B \quad (10.60)$$

$$\forall \omega \in \mathbb{R}, \quad A \leq \frac{1}{b} \sum_k \left| \hat{g} \left( \omega - k \frac{b}{2\pi} \right) \right|^2 \leq B. \quad (10.61)$$

A well known sufficient condition for  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  to be a frame is presented in [14]. Let  $a, b > 0$ ,  $g(t) \in L^2(\mathbb{R})$ , denote  $g^*(t)$  the complex conjugate of  $g(t)$  and suppose that  $\exists A, B > 0$  such that

$$A \leq \sum_{n \in \mathbb{Z}} |g(t - na)|^2 \leq B \quad \forall t \in \mathbb{R} \text{ and} \quad (10.62)$$

$$\sum_{k \neq 0} \left\| \sum_{n \in \mathbb{Z}} T_{na}g(t) T_{na+\frac{k}{b}} g^*(t) \right\|_\infty < A, \quad (10.63)$$

then  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  is a Gabor frame for  $L^2(\mathbb{R})$ .

A more general sufficient condition for the generation of a frame  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  for  $a, b > 0$  given and  $g(t) \in L^2(\mathbb{R})$  is [6,48]: if

$$B := \frac{1}{b} \sup_{t \in [0,a]} \sum_{k \in \mathbb{Z}} \left| \sum_{n \in \mathbb{Z}} g(t-na) g^* \left( t-na - \frac{k}{b} \right) \right| < \infty, \text{ and} \quad (10.64)$$

$$A := \frac{1}{b} \inf_{t \in [0,a]} \left[ \sum_{n \in \mathbb{Z}} |g(t-na)|^2 - \sum_{k \neq 0} \left| \sum_{n \in \mathbb{Z}} g(t-na) g^* \left( t-na - \frac{k}{b} \right) \right|^2 \right] > 0 \quad (10.65)$$

then  $\{E_{mb}T_{na}g(t)\}_{m,n \in \mathbb{Z}}$  is a frame for  $L^2(\mathbb{R})$  with frame bounds  $A, B$ . Note that this result shows that if  $g(t)$  has a limited support  $[0, 1/x]$  then for any set  $ab \leq 1$  and  $b < x$  a Gabor frame is obtained.

Other conditions for the generation of Gabor frames exist (see for example [4–6,27]). In [49] an extension of the results in Eqs. (10.64) and (10.65) for irregularly sampled time-frequency parameters (when the set  $(an, bm)$  is replaced by any pair  $(a_{n,m}, b_{n,m}) \in [na, (n+1)a] \times [mb, (m+1)b]$ ) is provided.

In subSection 1.10.6.2, we discuss the use of such frames for time-frequency analysis by means of the Gaborgram and in Section 1.10.6.3 we present an example of “how to find” the inverse frame to a Gabor frame. In Section 1.10.6.4 a condition for constructing Gabor frames in discrete spaces from its continuous counterpart is stated.

### 1.10.5.5 Wavelet frames

Wavelet analysis asks if translated and scaled versions of function  $\psi(t)$  can be employed for representing a signal  $s(t) \in L(\mathbb{R})$ . In this sense, the signal is to be represented using functions like

$$\psi_{c,a}(t) = \frac{1}{\sqrt{c}} \psi \left( \frac{t-a}{c} \right). \quad (10.66)$$

If one thinks about the signal projections over different  $\psi_{c,a}(t)$  to represent the signal, the similarity with the Gabor “time-frequency” approach is unavoidable. One applies operations to a fixed prototype function and use the resulting modified versions of the prototype to express the signal. Discrete pairs  $(a, c)$  define how the prototype function has been modified and thus provide what signal characteristics are being analyzed.

If one employs a continuum of  $(a, c)$  pairs, one has the so-called *Continuous Wavelet Transform* [4,5,7,50]

$$W_\psi(a, c) = \langle f(t), \psi_{c,a}(t) \rangle = \int f(t) \frac{1}{\sqrt{c}} \psi \left( \frac{t-a}{c} \right) dt. \quad (10.67)$$

One can compute the inverse wavelet transform using

$$f(t) = K_\psi \int_0^\infty \int W_\psi(a, c) \frac{1}{\sqrt{c}} \psi \left( \frac{t-a}{c} \right) da \frac{dc}{c^2} \quad (10.68)$$

In the equations above, we have considered that  $\psi(t)$  is real and  $K_\psi$  is a constant that depends on  $\psi(t)$  [4, 51, 52]. It is worthy to say that, similarly, the Gabor frame can be linked to the so-called *Short Time Fourier Transform* [4, 21].

In the case of wavelet frames, we are concerned with the case of a discrete set of pairs  $(a, c)$ . In this sense in 1910 [53], Haar has shown that for the function

$$\psi(t) = \begin{cases} 1, & 0 \leq t < 1/2, \\ 1, & 1/2 \leq t < 1, \\ 0, & \text{otherwise} \end{cases} \quad (10.69)$$

the set of dilated and translated versions

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \quad j, k \in \mathbb{Z},$$

what means that  $\psi_{j,k}(t) = T_k D_{2^{-j}}\{f(t)\}$

(10.70)

is a basis for  $L^2(\mathbb{R})$ .

A more throughout study of such possibilities occurred in the beginning of the 1980s. Calderon in 1964 [51] and Grossman and Morlet in 1985 [52] introduced the *Wavelet Transform*. In continuation, the first *wavelet frame* construction appeared in 1986 in [12]. Great effort was dedicated to the study of wavelets due to their multi-resolution analysis [4, 5, 7, 13, 50]. A recent book [54] collects some relevant papers for wavelet analysis theory.

One should note that in Eqs. (10.66)–(10.70), the order in which the translation and dilation operation are applied differs from the one previously presented in Section 1.10.5.2. Therefore, for clarity, we now define what is meant by a Wavelet Frame.

**Definition 10.13.** For  $c > 1$ ,  $a > 0$  and  $g \in L^2(\mathbb{R})$ , a frame constructed by dilations and translations as

$$\{T_{nac^j}D_{c^j}g(t)\}_{j,n \in \mathbb{Z}} = \left\{c^{\frac{j}{2}}g(c^j t - na)\right\}_{j,n \in \mathbb{Z}} \quad (10.71)$$

is called a wavelet frame. □

In [5] both necessary and sufficient conditions are provided to construct wavelet frames. For example, if  $\{T_{nac^j}D_{c^j}g(t)\}_{j,n \in \mathbb{Z}} = \left\{c^{\frac{j}{2}}g(c^j t - na)\right\}_{j,n \in \mathbb{Z}}$  is a frame with frame bounds  $A$  and  $B$  then necessarily [55]

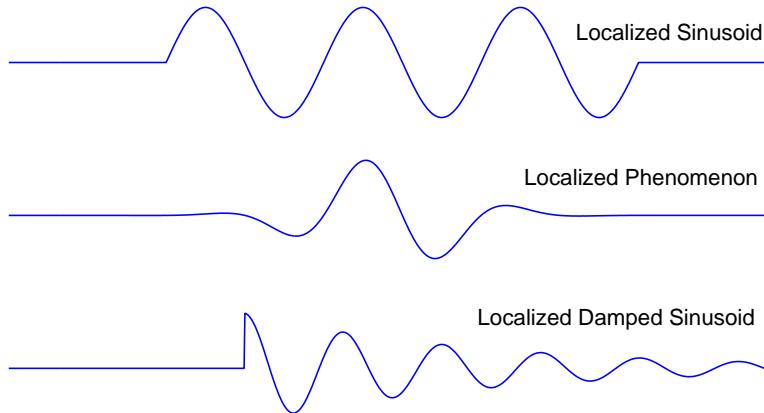
$$A \leq \frac{1}{a} \sum_{j \in \mathbb{Z}} \left| \hat{g}(c^j \omega) \right|^2 \leq B, \quad (10.72)$$

where  $\hat{g}(\omega)$  is the Fourier transform of  $g(t)$ .

A sufficient condition to generate a wavelet frame [6] is: suppose that  $c > 1$ ,  $a > 0$  and  $g(t) \in L^2(\mathbb{R})$  are given, if

$$B := \frac{1}{b} \sup_{|\omega| \in [1, c]} \sum_{j,n \in \mathbb{Z}} \left| \hat{g}(c^j \omega) \hat{g}(c^j \omega + \frac{n}{a}) \right| < \infty, \quad \text{and} \quad (10.73)$$

$$A := \frac{1}{b} \inf_{|\omega| \in [1, c]} \left[ \sum_{n \in \mathbb{Z}} \left| \hat{g}(c^j \omega) \right|^2 - \sum_{n \neq 0} \sum_{j \in \mathbb{Z}} \left| \hat{g}(c^j \omega) \hat{g}(c^j \omega + \frac{n}{a}) \right| \right] > 0 \quad (10.74)$$

**FIGURE 10.3**

Examples of fixed functions that can be used to construct frames.

then  $\{T_{nacj} D_{c^j} g(t)\}_{j,n \in \mathbb{Z}}$  is a frame for  $L^2(\mathbb{R})$  with bounds  $A, B$  given by the expressions above.

In [49] an extension of this condition for irregularly sampled time-scale parameters, when the set  $(an, c^j)$  is replaced by any pair  $(a_{n,j}, c_{n,j}) \in [c^j an, c^j a(n+1)] \times [c^j, c^{j+1}]$ , is provided.

### 1.10.5.6 Finite vector spaces

When considering finite vector spaces  $\mathbb{H}^N$  the simpler solution is to truncate or box-window the elements of the discrete space frame; however, this simple approach alters the frame bounds [56]. An alternative is to consider that the vector space is generated from an  $N$ -length section of an  $N$ -periodic  $l^2(\mathbb{Z})$  space, where the translation operator is a circular shift. The circular shift of a signal does not change the vector norm; this way the frame in  $\mathbb{H}^N$  has the same frame bounds as the frame in the  $N$ -periodic  $l^2(\mathbb{Z})$ .

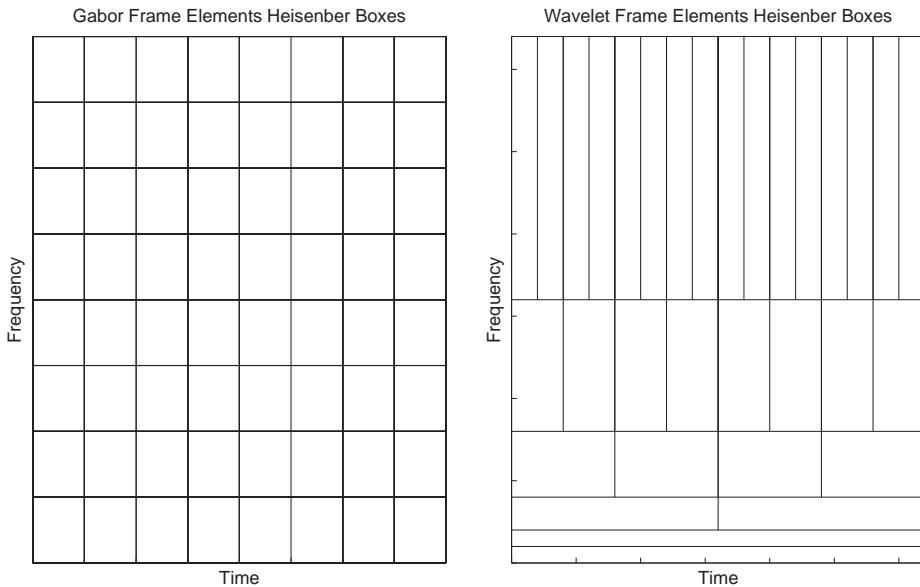
---

## 1.10.6 Some remarks and highlights on applications

### 1.10.6.1 Signal analysis

We have discussed the construction of different frames from a fixed prototype function. We have also presented examples that span the possibility of signal representation, analysis and synthesis using functions with prescribed characteristics. These are some of the main features of frames that are relevant for signal processing applications: detection of signal features, analyzing and synthesizing signals using a set of pre-defined signals. Frames bring freedom when compared to basis as the number of elements to be employed in the analysis-synthesis process can be larger than the signal space dimension.

In the literature, there is a large number of examples of different frame constructions using a fixed prototype signal; we have presented some in previous sections. Figure 10.3 illustrates different functions with different characteristics that can be used in this framework. These figures illustrate possible fixed functions to be employed to detect localized phenomena in a signal.

**FIGURE 10.4**

Heisenberg boxes in the time-frequency plane for Gabor (left) and wavelet (right) frames.

The Gabor and wavelet approaches commonly used for building frames from a fixed function were discussed. In some cases, the density in time, frequency or scale of the phenomena to be analyzed or detected in the signal [24,25,57] may be such that techniques for reducing the number of frame elements may be handy [58].

**Example 10.10.** As the reader may have already noticed, Wavelet frames can be used in a similar way to Gabor frames for time-frequency analysis. However, in this case a better phrasing would be *time-scale analysis*. Figure 10.4 depicts the different tilings of the time-frequency plane obtained by these approaches. These tilings depict the Heisenberg boxes [4] of the frame elements—these boxes roughly correspond to the time-frequency plane area where most of the elements' energy is concentrated. □

### 1.10.6.2 Gaborgram

In the beginning of Section 1.10.5.4 we have presented the idea of Gabor. That was representing a signal  $s(t)$  by means of

$$s(t) = \sum_m \sum_n c_{m,n} E_{mb} T_{na} g(t) = \sum_m \sum_n c_{m,n} e^{jmb} g(t - na). \quad (10.75)$$

As it can be noticed, this equation assumes that there is a representation for any  $s(t)$  using several  $g_{mb,na}(t) = E_{mb} T_{na} g(t)$ . We have seen that for these to be true it is enough for the set  $\{E_{mb} T_{na} g(t)\}_{m,n}$  to be a frame. What is commonly referred to as the Gaborgram is the plot of the coefficients  $c_{m,n}$  using the lattice presented in Figure 10.2. As discussed above a similar concept can be used for Wavelet Frames.

We have also seen that to compute the  $\{c_{m,n}\}_{m,n}$ , the inverse frame to  $\{E_{mb}T_{na}g(t)\}_{m,n}$  must be used. The next section illustrates how to obtain the inverse frame in the case of an exponential function. A Gabor frame of damped sinusoids is inspired on the connection among transients and damped sinusoids in physical systems.

### 1.10.6.3 Inverse gabor frame

The power of Gabor system for signal analysis and synthesis is not difficult to perceive. The problem is how to find the coefficients  $c_{m,n}$  that provide the reconstruction

$$s(t) = \sum_m \sum_n c_{m,n} E_{mb} T_{na} g(t) = \sum_m \sum_n c_{m,n} e^{jmb} g(t - na). \quad (10.76)$$

The inverse frame elements  $\{\tilde{g}(t)\}_{m,n}$  can be used. In this case one has that

$$s(t) = \sum_m \sum_n \langle c, \tilde{g}_{m,n}(t) \rangle E_{mb} T_{na} g(t). \quad (10.77)$$

Daubechies [13] has shown that, for Gabor frames, the inverse frame must be such that

$$\tilde{g}_{m,n}(t) = E_{mb} T_{na} \tilde{g}(t). \quad (10.78)$$

And one has that

$$s(t) = \sum_m \sum_n \langle c, E_{mb} T_{na} \tilde{g}(t) \rangle E_{mb} T_{na} g(t). \quad (10.79)$$

However, one still has to find  $\tilde{g}(t)$ . Equation (10.79) provides a biorthogonality condition (which is the “frame decomposition” formulas discussed in Section 1.10.3.1). In that sense, in [59] it is shown that Eq. (10.79) leads to

$$\frac{1}{ab} \int g(t) \tilde{g}\left(t - \frac{n}{b}\right) e^{-j2\pi m \frac{t}{a}} dt = \delta(n)\delta(m), \quad (10.80)$$

where  $\delta(x)$  denotes the impulse of  $x$ . In [59] it is also discussed that Eq. (10.80) accepts more than one solution in the oversampled case ( $ab < 1$ ) [27].

**Example 10.11.** In [24] the problem of building a Gabor system/frame based on the one sided exponential is presented. In this case, one has that

$$g(t) = \begin{cases} \sqrt{2\alpha} e^{-\alpha t}, & t \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (10.81)$$

Using the Zak Transform [47, 60] in [24] it is derived that for large oversampling ( $ab < 1$ )

$$\tilde{g}(t) \approx \begin{cases} -ke^{-\alpha(t+\frac{2}{b})}, & -1 \leq tb < 0 \\ ke^{-\alpha t}, & 0 \leq tb < 1 \\ 0, & \text{otherwise} \end{cases} \quad (10.82)$$

Where  $k$  is a constant that depends on  $a, b$  and  $\alpha$  [24].

Figure 10.5 illustrates the  $\tilde{g}(t)$  for different cases of  $a$  and  $b$ . In order to just focus on the waveform of  $\tilde{g}(t)$ , we ignore here the value of the constant  $k$  and make all functions scaled so that  $\max_t \tilde{g}(t) = 1$ .  $\square$

### 1.10.6.4 Gabor frames in discrete spaces

Obviously, due to the time-shift frequency-shift structure of Gabor frames, these have a broad range of applications for signal processing. Since most algorithms take place in a discrete space we should evaluate how to construct Gabor frames in this space.

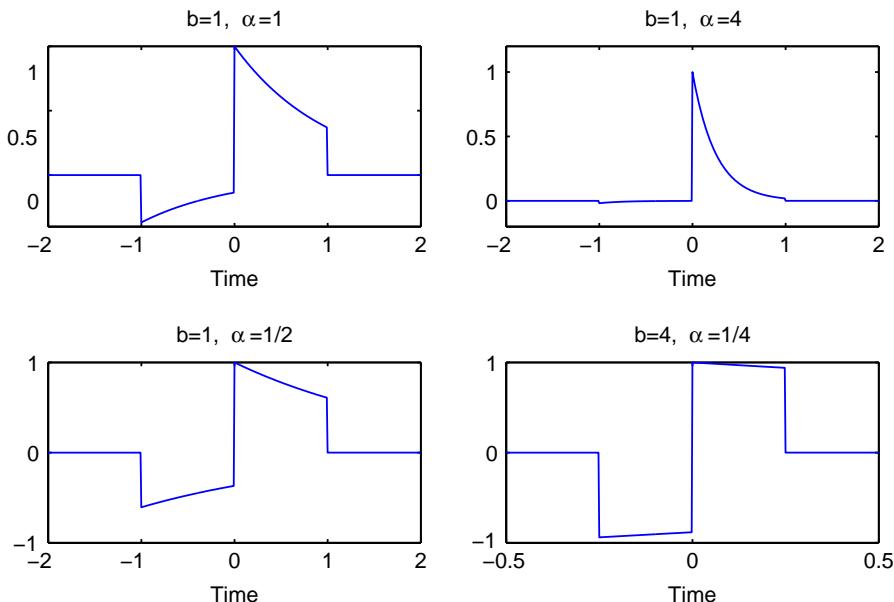
Frames in discrete spaces ( $l^2(\mathbb{Z})$ ) can be obtained by time-sampling the elements of frames in  $L^2(\mathbb{R})$  [6]. If  $g(t) \in L^2(\mathbb{R})$  is such that

$$\lim_{\epsilon \rightarrow 0} \sum_{k \in \mathbb{Z}} \frac{1}{\epsilon} \int_{-\frac{1}{2}\epsilon}^{\frac{1}{2}\epsilon} |g(k+t) - g(k)|^2 dt = 0, \quad (10.83)$$

and if  $\{E_{m/Q} T_{n/P} g(t)\}_{m,n \in \mathbb{Z}}$  with  $Q, P \in \mathbb{N}$  is a frame for  $L^2(\mathbb{R})$  with frame bounds  $A$  and  $B$ , then  $\{E_{m/Q} T_{n/P} \mathbf{g}_D\}_{n \in \mathbb{Z}, m=0,1,\dots,M-1}$  is a frame for  $l^2(\mathbb{Z})$  with frame bounds  $A$  and  $B$  [6]. The vector  $\mathbf{g}_D$  is the discretized version of  $g(t)$  with one sample per time unit, i.e.,  $\mathbf{g}_D = \{g(j)\}_{j \in \mathbb{Z}}$ .

### 1.10.6.5 Fast analysis and synthesis operators for gabor frames

Gabor or Weyl-Heisenberg frames in  $N$ -dimensional spaces are also interesting because one can find fast algorithms to compute their analysis and synthesis operators. Now, we consider that the number of “points” of the Weyl-Heisenberg frame in the frequency axis is such that  $Q = N/r$ ,  $Q, r \in \mathbb{N}$  and



**FIGURE 10.5**

Inverse Gabor frame prototype function of the one sided exponential—Biorthogonal function  $\tilde{g}(t)$  to  $g(t) = \sqrt{2\alpha} e^{-\alpha t} u(t)$ , for different values of  $\alpha$ .

also that the number of points in the time axis  $P \in \mathbb{N}$  (the same restrictions were employed in Section 1.10.6.4 above. These restrictions actually lead to  $a = 1/Q$  and  $b = N/P$  in Eq. (10.59). In this case, the  $c_{n,m}$  are defined for  $n \in \{0 \dots P - 1\}$  and  $m \in \{0 \dots Q - 1\}$  and are given by

$$c_{n,m} = \langle \mathbf{x}, E_{m \frac{1}{Q}} T_{n \frac{N}{P}} \mathbf{g} \rangle = \sum_{l=0}^{N-1} \mathbf{x}[l] e^{\frac{-j2\pi lm}{Q}} T_{n \frac{N}{P}} \mathbf{g}[l]. \quad (10.84)$$

Defining

$$\mathbf{f}_{n \frac{N}{P}}[k] = \mathbf{x}[k] T_{n \frac{N}{P}} \mathbf{g}[k] = \mathbf{x}[k] \mathbf{g}\left[\left(k - n \frac{N}{P}\right) \bmod N\right] \quad (10.85)$$

$$\mathbf{f}_{n \frac{N}{P}} = \left[ \mathbf{f}_{n \frac{N}{P}}[0], \dots, \mathbf{f}_{n \frac{N}{P}}[N-1] \right] \quad (10.86)$$

we obtain

$$c_{n,m} = \langle \mathbf{x}, E_{m \frac{1}{Q}} T_{n \frac{N}{P}} \mathbf{g} \rangle = \sum_{l=0}^{N-1} \mathbf{f}_{n \frac{N}{P}}[l] e^{\frac{-j2\pi lm}{Q}}. \quad (10.87)$$

For  $Q = N/r, r \in \mathbb{N}$  we obtain

$$c_{n,m} = \langle \mathbf{x}, E_{m \frac{1}{Q}} T_{n \frac{N}{P}} \mathbf{g} \rangle = \sum_{l=0}^{N-1} \mathbf{f}_{n \frac{N}{P}}[l] e^{\frac{-j2\pi l}{N} mr} = DFT\{\mathbf{f}_{n \frac{N}{P}}\}[mr], \quad (10.88)$$

where  $DFT\{\mathbf{x}\}[k]$  is the  $k^{th}$  sample of the Discrete Fourier Transform of  $\mathbf{x}$ , which can be computed using fast algorithms (FFT) [31].

Using the result in Eq. (10.88) the analysis operator (which was discussed in Section 1.10.4.2) of such Weyl-Heisenberg frames is given by

$$\mathbf{T}^*\{\cdot\} : \mathbb{H}^N \rightarrow \mathbb{C}^{PQ}, \quad (10.89)$$

$$\mathbf{T}^*\{\mathbf{x}\} : c_{n,m} = \{\langle \mathbf{x}, E_{m \frac{1}{Q}} T_{n \frac{N}{P}} \mathbf{g} \rangle\}, n \in [0, P-1], m \in [0, Q-1] \quad (10.90)$$

$$c_{n,m} = FFT\{\mathbf{f}_{n \frac{N}{P}}\}[mr], \mathbf{f}_{n \frac{N}{P}}[k] = \mathbf{x}[k] \mathbf{g}\left[\left(k - n \frac{N}{P}\right) \bmod N\right] \quad (10.91)$$

Similarly, for the synthesis operator of such frames we have that

$$\begin{aligned} \mathbf{T}\{\cdot\} : \mathbb{C}^{PQ} &\rightarrow \mathbb{H}^N, \\ \mathbf{T}\{c_{n,m}\} = [\mathbf{t}[0], \dots, \mathbf{t}[N-1]] &= \sum_{n=0}^{P-1} \sum_{m=0}^{Q-1} c_{n,m} E_{m \frac{1}{Q}} T_{n \frac{N}{P}} \mathbf{g}. \end{aligned} \quad (10.92)$$

The last equation is the sum of  $PQ$   $N$ -length vectors and can be computed using the Inverse Discrete Fourier Transform as below

$$\mathbf{t}[l] = \sum_{n=0}^{P-1} \sum_{m=0}^{Q-1} c_{n,m} e^{j \frac{2\pi ml}{Q}} \mathbf{f}_{n \frac{N}{P}}[l] = \sum_{n=0}^{P-1} \mathbf{f}_{n \frac{N}{P}}[l] \sum_{m=0}^{Q-1} c_{n,m} e^{j \frac{2\pi}{N} lmr}. \quad (10.93)$$

Denoting  $c_n(m) = c_{n,m}$  we can define

$$\mathbf{c}_n = [c_n[0], c_n[1], \dots, c_n[Q-1]], \quad (10.94)$$

and its up-sampled version

$$U(\mathbf{c}_n)[k] = \begin{cases} c_n[k/r], & k/r \in \{0, 1, \dots, Q-1\} \\ 0, & \text{otherwise} \end{cases} \quad (10.95)$$

Thus, the synthesis operator is such that

$$\begin{aligned} t[l] &= \sum_{n=0}^{P-1} f_{n \frac{N}{P}}[l] \sum_{m=0}^{Q-1} U(\mathbf{c}_n)[mr] e^{j \frac{2\pi}{N} lmr} \\ &= \sum_{n=0}^{P-1} f_{n \frac{N}{P}}[l] IFFT\{U(\mathbf{c}_n)\}[l]. \end{aligned} \quad (10.96)$$

Figure 10.6 presents a way to compute the expansion coefficients or the reconstructed signal when a frame is used either as an analysis or as a synthesis operator. Note that in the outputs of each FFT block, in Figure 10.6, there is a serial to parallel converter, while at the IFFT inputs a parallel to serial converter exists. Figure 10.6 shows that all the frame coefficients can be obtained using  $PN(1 + \log_2(N))$  operations, which can be reduced if one takes into account that just  $Q$  FFT/IFFT coefficients need to be computed at each FFT/IFFT branch.

### 1.10.6.6 Time-frequency content analysis using frame expansions

As we have seen in several cases, frames provide a powerful tool for signal analysis. This is specially true when one considers Gabor and Wavelet frames. These provide a natural tiling of the time-frequency plane. We now discuss how can a signal expansion be mapped into a time-frequency plot.

The Wigner-Ville distribution (WD) is probably one of the most well-known and widely used tools for time-frequency analysis of signals. The WD of a signal  $x(t)$  is defined as [4, 18, 20, 61]

$$WD_x(t, f) = \int_{-\infty}^{+\infty} x\left(t + \frac{\tau}{2}\right) x^*\left(t - \frac{\tau}{2}\right) e^{-2\pi j f \tau} d\tau, \quad (10.97)$$

Some applications have used the WD of signal decompositions in order to analyze the time-frequency content of signals [4, 20]. The Wigner-Ville distribution of a signal  $x(t)$ ,  $WD_x(t, f)$ , is a “measure” of the energy density in the signal in both time ( $t$ ) and frequency ( $f$ ) simultaneously. However, it is just meaningful when regions of the time-frequency plane are considered, that is as local averages, and it can not be considered at a given time-frequency point  $(t', f')$  due to uncertainty principle [4, 20]. In addition, the WD of a signal has another drawback since it is not restricted to be positive, a mandatory characteristic for an energy density.

In Section 1.10.3, we have seen that a signal  $x$  can be decomposed into a frame  $\{g_k\}_{k \in \mathcal{K}}$  or into its dual  $\{\tilde{g}_k\}_{k \in \mathcal{K}}$ , and reconstructed by means of

$$x = \sum_{k \in \mathcal{K}} \langle x, \tilde{g}_k \rangle g_k = \sum_{k \in \mathcal{K}} \langle x, g_k \rangle \tilde{g}_k. \quad (10.98)$$

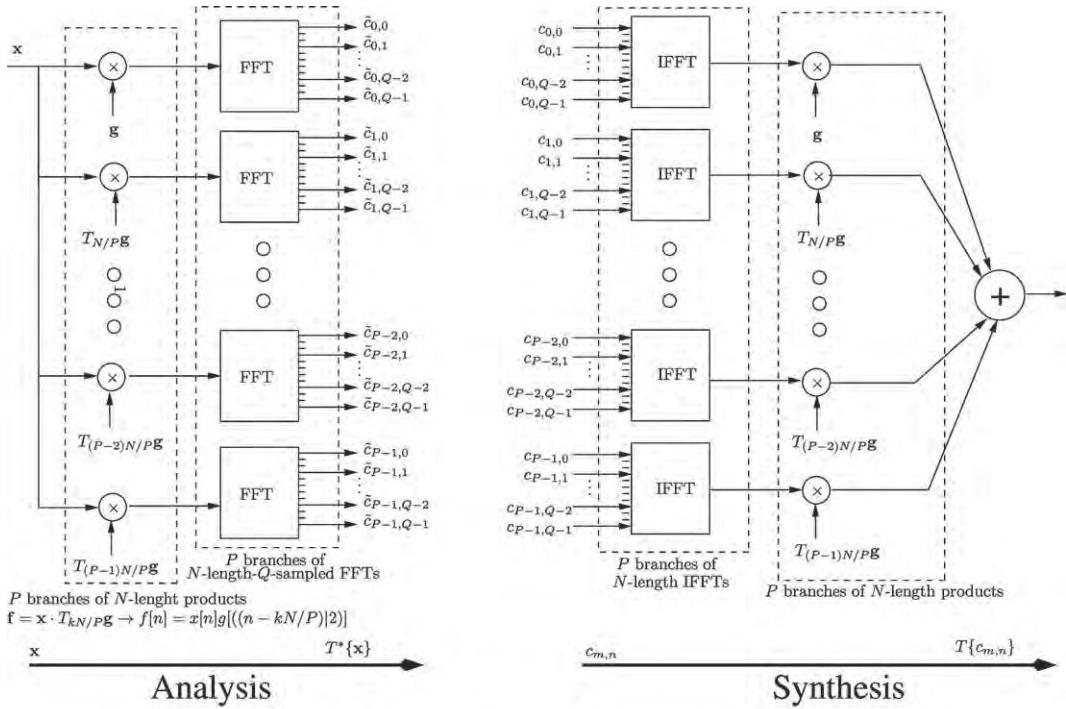


FIGURE 10.6

Weyl-Heisenberg fast analysis and synthesis using FFTs and IFFTs.

Once  $x$  is decomposed into the frame coefficients  $\langle x, \tilde{g}_k \rangle$ , its time-frequency content can be analyzed using

$$WD_{\tilde{x}}(t, f) = \sum_{k \in \mathcal{K}} \langle x(t), \tilde{g}_k(t) \rangle WD_{g_k}(t, f). \quad (10.99)$$

Using approaches like this, the inference of signal characteristics from signal decompositions is a common and powerful tool for signal analysis, detection and estimation. Specifically, in the case of time-frequency analysis, such approaches avoids the cross terms that appear when the original signal is directly analyzed [4, 18–22, 42, 61].

However, as we have seen the coefficients  $\langle x(t), \tilde{g}_k(t) \rangle$  used above may be obtained with a  $\tilde{g}_k(t)$  that does not, neither physically, nor visually and nor in its time-frequency content, resembles the associated  $g_k(t)$ . That is why in general this approach is left aside for others.

For instance, in [42] the time-frequency content of signals is estimated using the Matching Pursuits (MP) signal decomposition algorithm - a step based decomposition algorithm that iterates for successively obtaining better signal approximations [62]. The approximation is accomplished by

selecting the dictionary element that best matches the signal. This match is evaluated by projecting the signal on the dictionary elements. The best matching element is scaled by its correspondent projection and subtracted from the signal generating a residual error. The process is then applied to the residual error iteratively. This algorithm is considered to be greedy [63–65], in the sense that it minimizes the approximation error at each iteration. However, this may not be optimal as a sequence of iterations occur, and due to this fact several variants for the MP have been presented [65–68].

Although the dictionary employed in the MP is not required to be a frame, it has been a common procedure to employ a frame [58, 69, 70] as a dictionary. This is so because this choice guarantees the analysis and synthesis of any signal. In general, a mix of Gabor and Wavelet frames is employed to build the dictionary used together with the MP [42]. Doing this, the dictionary elements are placed in different locations in the time-frequency plane and the several scales provide different concentrations for the energy of the elements in time and frequency. It is common to employ a dictionary formed by Gaussian functions in different scales with different time and frequency shifts. One then obtains the MP expansion of signal  $x(t)$  into the Gabor dictionary

$$x(t) \approx \sum_{n=1}^M \gamma_n g_{i(n)}(t), \quad (10.100)$$

where the  $i(n)$  indexes the element selected for approximating the residual error in the decomposition step  $n$ . From this representation the time-frequency content of  $x(t)$  is analyzed by means of

$$\sum_{n=1}^M \gamma_n W D_{g_{i(n)}}(t, f). \quad (10.101)$$

In [71] the same approach is used, but using a modified version of the MP to obtain the signal expansion. Note the resemblance between the approach in Eq. (10.99) and the ones in Eqs. (10.100) and (10.101).

## 1.10.7 Conclusion

In this chapter we have made an overview of frames. We started by introducing the concept of overcomplete decompositions, highlighting its main characteristics. We then commented on the dual frames, and how frames can be employed to perform analysis and synthesis of signals. After this, we provided formal definitions of the frame operator, inverse frames and frame bounds. We then formalized the use of frames in discrete and finite dimensional spaces. Shifting into more practical matters, we showed how to generate frames from a prototype function, by using translations, modulations and dilations. From this, we analyzed the widely used frames of translates, Gabor frames and wavelet frames. We then described some applications in signal analysis, including the Gaborgram, and presented a way to compute it more efficiently using the Fast Fourier Transform. We finished the chapter by presenting ways to compute time-frequency analysis using decompositions and the matching pursuits algorithm.

### *Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 2](#) Continuous-Time Signals and Systems

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 5](#) Sampling and Quantization

See this Volume, [Chapter 7](#) Multirate Signal Processing for Software Radio Architectures

See this Volume, [Chapter 8](#) Modern Transform Design for Practical Audio/Image/Video Coding Applications

See this Volume, [Chapter 9](#) Discrete Multi-Scale Transforms in Signal Processing

## References

- [1] R.J. Duffin, A.C. Schaeffer, A class of nonharmonic fourier series, *Trans. Am. Math. Soc.* 72 (1952) 341–366.
- [2] R. Young, *An Introduction to Nonharmonic Fourier Series*, Academic Press, New York, 1980.
- [3] R. Paley, N. Wiener, The fourier transforms in the complex domain, *Am. Math. Soc. Colloquium Publ. Ser.* 19 (1934).
- [4] S. Mallat, *A Wavelet Tour of Signal Processing*, first ed., Academic Press, San Diego, California, USA, 1998.
- [5] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, USA, 1991.
- [6] O. Christensen, *An Introduction To Frames And Riesz Bases*, Applied and Numerical Harmonic Analysis, first ed., Birkhäuser, Boston, Basel, Berlin, 2002.
- [7] C.S. Burrus, R.A. Gopinath, H. Guo, *Introduction To Wavelets and Wavelets Transforms A Primer*, first ed., Prentice Hall, Upper Saddle River, New Jersey 07458, USA, 1998.
- [8] Z. Cvetkovic, M. Vetterli, On simple oversampled a/d conversion in  $l^2(\mathbb{R})$ , *IEEE Trans. Inform. Theory* 47 (1) (2001) 146–154.
- [9] F. Marvasti (Ed.), *Nonuniform Sampling: Theory and Practice*, first ed., Springer, 2001.
- [10] M. Vetterli, P. Marziliano, T. Blu, Sampling signals with finite rate of innovation, *IEEE Trans. Signal Process.* 50 (6) (2002) 1417–1428.
- [11] R.G. Baraniuk, Compressive sensing [lecture notes], *IEEE Signal Process. Mag.* 24 (4) (2007) 118–121.
- [12] I. Daubechies, A. Grossman, Y. Meyer, Painless nonorthogonal expansions, *J. Math. Phys.* 27 (1986) 1271–1283.
- [13] I. Daubechies, The wavelet transform, time-frequency localization and signal analysis, *IEEE Trans. Inform. Theory* 36 (5) (1990) 961–1005.
- [14] C. Heil, D. Walnut, Continuous and discrete wavelet transforms, *SIAM Review* 31 (1989) 628–666.
- [15] V.K. Goyal, J. Kovacevic, J.A. Kelner, Quantized frame expansions with erasures, *Appl. Comput. Harmon. Anal.* 10 (2001) 203–233.
- [16] P.G. Casazza, J. Kovacevic, Equal-norm tight frames with erasures, *Adv. Comput. Math.–Especial ISSUE on Frames* (2002) 387–430.
- [17] J. Kovacevic, P.L. Dragotti, V.K. Goyal, Filter bank frame expansions with erasures, *IEEE Trans. Inform. Theory* 48 (6) (2002) 1439 –1450.
- [18] G. Matz, F. Hlawatsch, Wigner distributions (nearly) everywhere: Time-frequency analysis of signals, systems, random processes, signal spaces, and frames, *Elsevier Signal Process.* 83 (2003) 1355–1378.
- [19] L. Cohen, Time-frequency distributions—a review, *Proc. of the IEEE* 77 (7) (1989) 941–981.
- [20] F. Hlawatsch, G.F. Boudreux-Bartels, Linear and quadratic time-frequency signal representations, *IEEE Signal Process. Mag.* 9 (1992) 21–67.

- [21] Leon Cohen, Time-Frequency Analysis, Prentice Hall Signal Processing Series. Prentice Hall, Prentice Hall PTR, Englewood Cliffs, New Jersey 07632, 1995.
- [22] Kalheiz Grchenig, Foundations of Time-Frequency Analysis, Birkhäuser, New York, USA, 2001.
- [23] D. Gabor, Theory of communications, IEEE J. 93 (1946) 429–457.
- [24] B. Friedlander, A. Zeira, Oversampled gabor representation for transient signals, IEEE Trans. Signal Process. 43 (9) (1995) 2088–2094.
- [25] B. Friedlander, B. Porat, Detection of transient signals by the gabor representation, IEEE Trans. Acoust. Speech Signal Process. 37 (2) (1989) 169–180.
- [26] M. Zibulski, Y.Y. Zeevi, Discrete multiwindow gabor-type transforms, IEEE Trans. Signal Process. 45 (1997) 1428–1442.
- [27] M. Zibulski, Y.Y. Zeevi, Oversampling in the gabor scheme, IEEE Trans. Signal Process. 43 (9) (1993) 2679–2687.
- [28] D.L. Donoho, M. Vetterli, R.A. DeVore, I. Daubechies, Data compression and harmonic analysis, IEEE Trans. Inform. Theory. 44 (6) (1998) 2435–2476.
- [29] E.J. Candes, D.L. Donoho, Ridgelets: a key to higher-dimensional intermittency?, Philos. Trans.: Math., Phys. Eng. Sci. 357 (1999) 2495–2509.
- [30] D.L. Donoho, A.G. Flesia, Can recent innovations in harmonic analysis ‘explain’ key findings in natural image statistics?, Network: Computation in Neural Systems, Taylor & Francis, vol. 12, pp. 371–393, 2001.
- [31] P.S.R. Diniz, E.A.B. da Silva, S. Lima Netto, Digital Signal Processing: System Analysis and Design, Cambridge University Press, 2001.
- [32] T. Strohmer, R.W. Heath Jr., Grassmannian frames with applications to coding and communications, App. Comput. Harmon. Anal. 14 (3) (2003) 257–275.
- [33] Z. Cvetkovic, Martin Vetterli, Oversampled filter banks, IEEE Trans. Signal Process. 46 (5) (1998) 1245–1255.
- [34] H. Bölksei, F. Hlawatsh, H.G. Feichtinger, Frame-theoretic analysis of oversampled filter banks, IEEE Trans. Signal Process. 46 (12) (1998) 3256–3268.
- [35] V.K. Goyal, M. Vetterli, N.T. Thao, Quantized overcomplete expansions in  $\mathbb{R}^N$ : Analysis, synthesis, and algorithms, IEEE Trans. Infor. Theory. 44 (1) (1998) 16–31.
- [36] N J Munch, Noise reduction in tight weyl-heisenberg frames, IEEE Trans. Inform. Theory. 38 (2) (1992) 608–616.
- [37] J.A. Tropp, I.S. Dhillon, R.W. Heath Jr., T. Strohmer, Designing structured tight frames via an alternating projection method, IEEE Trans. Inform. Theory. 51 (2005) 188–209.
- [38] Y.C. Eldar, G.D. Forney Jr., Optimal tight frames and quantum measurement, IEEE Trans. Inform. Theory. 48 (3) (2002) 599–610.
- [39] Y.C. Eldar, H. Bolcskei, Geometrically uniform frames,” IEEE Trans. Inform. Theory. 49 (4) (2003) 993–1006.
- [40] J.J. Benedetto, M. Fickus, Frame potentials, Adv. Comput. Math 18 (2003) 357–385.
- [41] R. Vale, S. Waldron, Tight frames and their symmetries, Constr. Approx. 21 (2005) 83–112.
- [42] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. 41 (12) (1993) 3397–3415.
- [43] A.J. Jerri, The shannon sampling theorem—its various extensions and applications: A tutorial review, Proc. IEEE 65 (11) (1977) 1565–1596.
- [44] R.J. Marks II, Introduction to Shannon Sampling and Interpolation Theory, Springer-Verlag, 1991.
- [45] J. von Neumann, Mathematische Grundlagen der Quantenmechanik, Springer Berlin, 1932, English version: Mathematical Foundations of Quantum Mechanichs, Princeton Univ. Press, 1955.
- [46] S. Qian, D. Chen, Joint time-frequency analysis, IEEE Signal Process. Mag. (1999) 52–67.
- [47] A.J.E.M. Janssen, Zak Transforms With Few Zeros and The Tie, in: H.G. Feichtinger, T.Strohmer (Eds.), Advances in Gabor Analysis, Birkhäuser, Boston, 2002.

- [48] P.G. Casazza, O. Christensen, Weyl-heisenberg frames for subspaces of  $L^2(\mathbb{R})$ , Proc. Am. Math. Soc. 129 (2001) 145–154.
- [49] W. Sun, X. Zhou, Irregular wavelet/gabor frames, Appl. Comput. Harmon. Anal. 13 (2002) 63–76.
- [50] Mladen Victor Wickerhauser, Adapted Wavelet Analysis from Theory to Software, IEEE Press, Piscataway, NJ, USA, 1998.
- [51] A.P. Calderon, Intermediate spaces and interpolation, the complex method, Studia Math. 24 (1964) 113–190.
- [52] A. Grossmann, J. Morlet, Decomposition of hardy functions into square integrable wavelets of constant shape, SIAM J. Math. Anal. 15 (1984) 723–736.
- [53] A. Haar, Zur theorie der orthogonalen funktionensysteme, Math. Ann. 69 (1910) 331–371. Translated Version: On the theory of orthogonal function systems, by G Zimmermann.
- [54] Christopher Heil, David F. Walnut, Fundamental Papers in Wavelet Theory, Princeton University Press, 2006.
- [55] C. Chui, X. Shi, Inequalities of Littlewood-Paley type for frames and wavelets, Siam J. Math. Anal. 24 (1993) 263–277.
- [56] T. Strohmer, Numerical analysis of the non-uniform sampling problem, Comput. Appl. Math. 122 (2000) 297–316.
- [57] R. Balan, I. Daubechies, V. Vaishampayan, The analysis and design of windowed fourier frame based multiple description source coding schemes, IEEE Trans. Inform. Theory. 46 (2000) 2491–2536.
- [58] K. Engan, S.O. Aase, J.H. Husoy, Designing frames for matching pursuits algorithms, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, May 1998, pp. 1817–1820.
- [59] J. Wexler, S. Raz, Discrete gabor expansions, Signal Process. 21 (3) (1990) 207–221.
- [60] J. Zak, Finite translations in solid-state physics, Phys. Rev. Lett. 19 (1967) 1385.
- [61] T. Claasen W. Mecklenbrauker, The aliasing problem in discrete-time wigner distributions, IEEE Trans. Acoust. Speech Signal Process. 31 (1983) 1067–1072.
- [62] Lisandro Lovisolo, Eduardo A.B. da Silva, Paulo S.R. Diniz, On the statistics of matching pursuit angles, Signal Process. 90 (2010).
- [63] R.A. DeVore, V.N. Temlyakov, Some remarks in greedy algorithms, Adv. Comput. Math. 5 (1996) 173–187.
- [64] G. Davis, S. Mallat, M. Avellaneda, Adaptive greedy approximations, J. Constr. Approx. 13 (1997) 57–98.
- [65] J.A. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Trans. Inform. Theory. 50 (2004) 2231–2241.
- [66] Y.C. Pati, R. Rezaifar, P.S. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, in: IEEE Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, 1993.
- [67] J.A. Tropp, A.C. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, IEEE Trans. Inform. Theory. 53 (2007) 4655–4666.
- [68] L. Rebollo-Neira, D. Lowe, Optimized orthogonal matching pursuit approach, IEEE Signal Process. Lett. 9 (2002) 137–140.
- [69] L. Lovisolo, M.A.M. Rodrigues, E.A.B. da Silva, P.S.R. Diniz, Efficient coherent decompositions of power systems signals using damped sinusoids, IEEE Trans. Signal Process. 53 (2005) 3831–3846.
- [70] P.J. Durka, D. Ircha, K.J. Blinowska, Stochastic time-frequency dictionaries for matching pursuit, IEEE Trans. Signal Process. 49 (2001) 507–510.
- [71] A. Papandreou-Suppappola, S.B. Suppappola, Analysis and classification of time-varying signals with multiple time-frequency structures, IEEE Signal Process. Lett. 9 (2002) 92–95.

# Parametric Estimation

# 11

Suleyman Serdar Kozat\* and Andrew C. Singer†

\*Department of Electrical and Electronics Engineering, Koc University Istanbul, Turkey

†110 CSL, 1308 West Main Street, Urbana, IL, USA

## 1.11.1 Introduction

This chapter considers the topic of parametric estimation, which is an important engineering concept that is often used for modeling signals and systems. When a complex signal or system is encountered, it is often desirable to construct a model for the signal or system in question such that it can be readily analyzed. For an input-output system, this may be to understand its behavior, its potential weaknesses, or its stability to bounded inputs. For signals of interest, this may be to understand the content of the signal, such as to determine the words that were spoken in an acoustic signal containing a recording of human speech. In many such cases, models for the signals or systems of interest are sought that can be practically analyzed without excess computation. As in Occam's razor, a parsimonious model is better than a needlessly complicated one. A good model would be one that is sufficiently rich to enable insights into the salient characteristics of the signals or systems under study, but not any more so than necessary. By using parametric models, whose behavior is uniquely determined by a finite set of parameters, the complexity of the model can often be controlled by limiting the number of parameters. In this chapter, we focus on the estimation of such model parameters under a variety of scenarios. Using both statistical and deterministic formulations, rational models (linear models with a finite number of poles and zeros) are considered in both the batch and online recursive formulations. Such autoregressive and moving average models form the basis for our initial investigation, after which we explore advanced topics, such as spectrum estimation, prediction, and filtering, as well as a number of nonlinear parametric modeling techniques.

The background assumed in this chapter is a working knowledge of discrete-time signals and systems, some basic probability theory including knowledge of stationary random processes, and the mathematical skills of a senior level undergraduate student in electrical engineering or a similar discipline. An exemplary treatment of these topics includes the texts by Oppenheim et al. [1], Rabiner and Gold [2], and Rabiner and Schafer [3]. These texts cover discrete-time signal and system theory, with the latter covering some random processes. Additional material on probability and stochastic processes as applied to the problems in this chapter can be found in [4,5].

## 1.11.2 Deterministic and stochastic signals

In this chapter, we will consider discrete-time signals through a number of different lenses with the goal of developing parametric models for signals and systems that are useful in a variety of engineering contexts.

In order to proceed, we will often make use of two distinct models for signals in our discussions. The first, and perhaps the simplest formulation, is to assume that the signals of interest are deterministic, that is, that the signal values  $x[n]$  can take on any finite real value and there are no probabilities or other constraints to these values. This might be a useful way for us to capture the inability of our modeling methods to provide any other meaningful measure to the data we observe than the simple statement that the data can take on any values.

In this light, we will often use measures of performance such as the accumulated square error of an estimate  $\hat{x}[n]$  for a signal  $x[n]$  over a block of samples,  $n = 0, \dots, N$ ,

$$E_{LS} = \sum_{n=0}^N (x[n] - \hat{x}[n])^2, \quad (11.1)$$

or we may formulate this as an integral square error measured in the frequency domain,

$$E_{IS} = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X_d(\omega) - \hat{X}(\omega)|^2 d\omega, \quad (11.2)$$

as a means of capturing the degree to which the approximating time domain signal  $\hat{x}[n]$  or frequency domain estimate  $\hat{X}(\omega)$  match or model the deterministic signal of interest  $x[n]$  or  $X_d(\omega)$ .

As a simple example, suppose that we have measured a signal  $x[n]$  that corresponds to the daily temperature of the water in a local pond. If we wanted to compute a good approximating signal,  $\hat{x}[n] = c$ , i.e., a signal whose value remains constant over the entire interval, and we use the accumulated square error over the collected data signal as a measure of performance, we could write

$$E_{LS} = \sum_{n=0}^N (x[n] - c)^2 \quad (11.3)$$

and seek to find the value of that minimizes the total accumulated square error over the observation interval. Differentiating  $E_{LS}$  with respect to  $c$ , and setting the result equal to zero, we obtain,

$$\frac{\partial E_{LS}}{\partial c} = -2 \sum_{n=0}^N (x[n] - c), \quad (11.4)$$

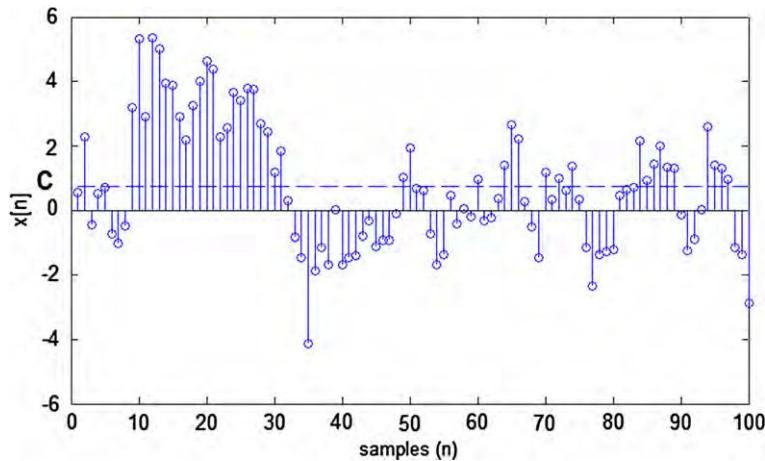
$$0 = -2 \sum_{n=0}^N (x[n] - c), \quad (11.5)$$

which can be solved for the minimizing value of  $c$ , yielding

$$c = \frac{1}{N+1} \sum_{n=0}^N x[n]. \quad (11.6)$$

This is the well-known result that the sample-average of a sequence is the best approximating value for the sequence, if we restrict the set of possible approximating signals to those that take on only a single constant value.

Figure 11.1 shows the result of approximating a signal  $x[n]$  with another signal that takes on only a single constant value. The signal in this example has an average value of  $c$  over the interval shown.

**FIGURE 11.1**

Approximating a signal with a constant value.

As a result, the best approximating constant valued signal, in terms of minimizing the squared approximation error over the interval shown, is the mean of the signal,  $c$ .

Another approach that we will often pursue is the use of stochastic (random) signal models in our development. Random signal models enable the use of probabilistic methods to model some of the uncertainty, or relative uncertainty, in the evolution of a time-series or a given signal or system model. As a result, we may choose measures of performance that are similar in spirit to those of our deterministic signal analyses, but for which we make explicit use of a probabilistic model for the signals of interest. In statistical signal processing, square-error methods are often used, again for their mathematical tractability as well as our ability to capture much of the behavior of the signals of interest with a relatively small number of statistical quantities. Specifically, we will often use a model for random signals that makes use of the so-called second-order statistics of the signal of interest [6]. Specifically, for a discrete-time random signal,  $x[n]$ , to have a complete statistical characterization of the signal would require knowledge of the probability density function for all possible probabilistic quantities such as  $f_x(x[n_1], x[n_2], x[n_3], \dots, x[n_k])$  for all possible  $n_1, n_2, n_3, \dots, n_k$ , and for all possible  $k$ . This complete statistical knowledge may be available for certain signal or system models, however, in general such a complete characterization is either not available or not appropriate.

As a result, we will often use second-order statistical models for random signals of interest by assuming that we either have knowledge of, or can make estimates of, the following two quantities. Specifically, the mean-value of the signal is defined as

$$m_x[n] = E\{x[n]\}, \quad (11.7)$$

where the  $E\{ \}$  denotes the statistical expectation operation which is taken with respect to the complete stochastic characterization of the random signal  $x[n]$ . The covariance function of the signal is defined as

$$\begin{aligned} C_x[m, n] &= E\{(x[m] - m_x[m])(x[n] - m_x[n])^*\}, \\ C_x[m, n] &= E\{(x[m]x[n]^*)\} - m_x[m]m_x[n]^*, \end{aligned} \quad (11.8)$$

where, again, the expectation is taken with respect to the complete stochastic characterization of the random signal, and the second term in the definition is conjugated, if the random signal is complex valued.

When the random signal  $x[n]$  is *wide-sense stationary* (WSS), then both the mean-value of the signal and also the covariance function take on a special form [1,6]. Specifically, for wide-sense stationary random signals, the mean-value of the signal is not a function of time, i.e.,  $m_x[n] = m_x$ , and the covariance function is only a function of the difference in times between the two samples, i.e.,

$$C_x[m, n] = C_x[0, n - m] \triangleq C_x[n - m], \quad (11.9)$$

where, with a slight abuse of notation, the same name is given to both the general and the wide-sense stationary covariance functions. In general terms, a random signal is stationary if the statistical properties of the signal do not depend on the specific time (of day, of the year, etc.) but rather samples of the signal have correlation (or other statistical dependence) that is merely a function of how close in time the samples are taken with respect to one another.

Returning to our earlier example of representing a signal, now a random signal, by a single constant estimate of that signal, we might seek to minimize the following mean square error, rather than the accumulated square error as before,

$$\begin{aligned} E_{MS} &= E\{(x[n] - \hat{x}[n])^2\}, \\ E_{MS} &= E\{(x[n] - c)^2\}, \end{aligned} \quad (11.10)$$

where, once again we seek a single value of the constant to represent the signal over the region of interest. If the signal  $x[n]$  is stationary, then we can find a simple solution to this estimation problem by differentiating with respect to the parameter as follows:

$$\begin{aligned} \frac{\partial E_{MS}}{\partial c} &= -E\{2(x[n] - c)\}, \\ 0 &= -2E\{x[n]\} + 2c, \\ c &= m_x. \end{aligned} \quad (11.11)$$

This provides the stochastic version of the deterministic least squares estimation problem, whereby the minimum mean square error (MMSE) constant estimate of a wide-sense stationary random signal is given by the mean of the signal.

### 1.11.3 Parametric models for signals and systems

There are a wide variety of ways in which a signal can be represented, or *modeled* using the tools of signal processing and random processes. In this section, we will consider models that are *parametric* in that they contain a number of free parameters that can be adjusted to optimize a given criterion of interest, such as the least square or mean square error metrics described previously. Perhaps the simplest such models are the so-called *autoregressive* (AR) and *moving average* (MA) models that use as free parameters, the coefficients of the numerator and denominator polynomials of a rational transfer function that is driven by a deterministic or random process.

In contrast to such parametric signal models, methods that attempt to model or approximate a given signal without the use of such a global parametric model are often called *non-parametric* signal models. A few examples of non-parametric methods for estimating a signal of interest include estimates of histograms for signal values, piecewise constant or piecewise linear signal models, truncated Fourier series models, and nearest neighbor models. We will not explore such non-parametric methods in this chapter.

### 1.11.3.1 Autoregressive (AR) models

A discrete-time signal  $x[n]$  is an autoregressive (AR) process of order  $p$  if it can be written in the form [6]

$$x[n] + a_1x[n - 1] + \cdots + a_px[n - p] = w[n], \quad (11.12)$$

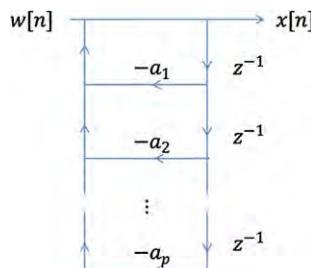
where  $a_1, \dots, a_p$  are parameters of the AR model and the signal  $w[n]$  is a zero mean white noise process of variance  $\sigma^2$ , i.e., we have that  $C_w[m] = \sigma^2\delta[m]$ . By rearranging the order of the terms in the definition, we can write the autoregressive process recursively, as

$$x[n] = -a_1x[n - 1] - a_2x[n - 2] - \cdots - a_px[n - p] + w[n], \quad (11.13)$$

which shows that a  $p$ th order autoregressive process can be written as a linear combination of the  $p$  most recent values of the signal plus a term that is statistically independent of the past values of the process; this is the so-called innovations sequence, and represents the new information, or the unpredictable components in the random signal (see Figure 11.2).

Another way to view autoregressive signals is by recognizing that the definition of the process also represents a convolution of the input sequence  $x[n]$  with the autoregressive parameters, such that the output signal is the innovations sequence, i.e.,

$$\sum_{k=0}^p a_k x[n - k] = w[n], \quad (11.14)$$



**FIGURE 11.2**

Autoregressive process flow diagram.

where we set  $a_0 = 1$ . The significance of this can be seen if we slightly rearrange terms, yielding

$$x[n] - \underbrace{\left( -\sum_{k=1}^p a_k x[n-k] \right)}_{\hat{x}[n]} = w[n] \quad (11.15)$$

in which we can view the summation as an estimate of the signal  $x[n]$ , given by its most recent  $p$  parameters, i.e., we have

$$x[n] - \hat{x}[n] = w[n]. \quad (11.16)$$

By writing the process in this form, we see that the sequence  $x[n]$  is rather unique in that it can be best estimated by its most recent past values and that after this *predictable component* is subtracted off, all that remains is a white-noise sequence that is independent of the past values. The term “autoregressive” comes from the ability to write the sequence  $x[n]$  using a “regression model” over its own past values, i.e., using a finite linear combination of its own past values.

By using the linear regression, or auto-regression model for the sequence  $x[n]$ , we see that there is an input-output relationship between the sequences  $w[n]$  and  $x[n]$  that is, we can view the creation of the sequence  $x[n]$  by processing the sequence  $w[n]$  with a linear, time-invariant filter. Ignoring that the sequences  $x[n]$  and  $w[n]$  may be random for now, and considering only the input-output relationship between them, we can take the  $z$ -transform of both sides (assuming for now that the sequences are deterministic, or given), we have

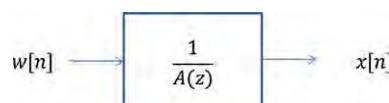
$$x(z) + \sum_{k=1}^p a_k z^{-k} X(z) = W(z), \quad (11.17)$$

$$x(z) \left( 1 + \sum_{k=1}^p a_k z^{-k} \right) = W(z), \quad (11.18)$$

$$x(z) = \left( \frac{1}{1 + \sum_{k=1}^p a_k z^{-k}} \right) W(z). \quad (11.19)$$

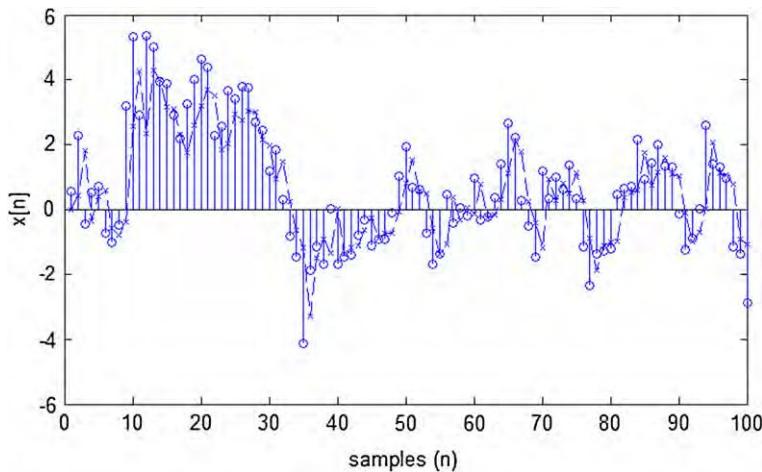
This input-output relationship implies that we can view the sequence  $x[n]$  as being generated by filtering the sequence  $w[n]$  with an “all-pole” filter, since the transfer function  $H(z) = X(z)/W(z)$ , relating  $X(z)$  and  $W(z)$  can be written as a rational transfer function where the numerator polynomial is given by  $B(z) = 1$ , and the denominator polynomial is given by  $A(z) = 1 + \sum_{k=1}^p a_k z^{-k}$  (see Figure 11.3).

This input-output interpretation of an AR process provides two interesting ways to view the relationship between the signals  $x[n]$  and  $w[n]$ . Specifically, we can use the relationship  $X(z) = H(z)W(z)$ ,



**FIGURE 11.3**

Input-output transfer function perspective of the autoregressive model.

**FIGURE 11.4**

Tracking colored noise with an AR model.

to look at the signal  $x(n)$  as being *generated* by filtering a white-noise signal  $w[n]$  with an all-pole filter. This provides us with a simple way to generate random processes (or deterministic signals) with a spectral shape that can be parametrically controlled through the coefficients of  $A[z]$ . An alternative viewpoint arises from writing the signal in the so-called “prediction error form” [3,6] whereby we have  $x[n] - \hat{x}[n] = w[n]$ . By writing it this way, we can view the signal  $x[n]$  as being filtered using the prediction error filter with transfer function  $A(z)$ , such that the output is the white-noise signal  $w[n]$ . In this sense, the predictable component of  $x[n]$  can be completely removed by the FIR prediction error filter with transfer function  $A(z)$ , leaving only the “innovation” sequence, or unpredictable component,  $x[n] - \hat{x}[n] = w[n]$ .

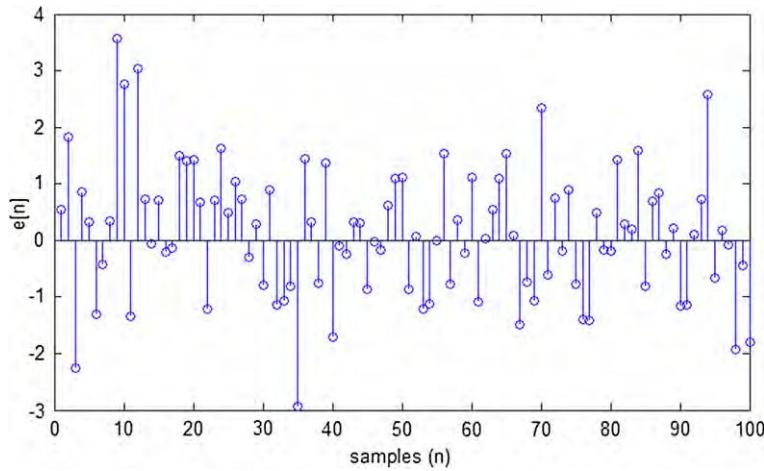
One application of the AR model is to use the parametric model for prediction of the sequence  $x[n]$ . For example, one might attempt to predict the next value of a colored noise signal with an AR model. Figure 11.4 shows the input  $w[n]$  and output  $x[n]$  of an FIR prediction filter that is designed to track the input using only past values of the input. The signal used in Figure 11.1 is reused in Figure 11.4.

Since colored noise can be generated by filtering a white-noise signal, we can view the signal of interest as if it were originally generated by filtering such a white-noise signal. Thus, we expect that the error between the predicted and measured signals will be a white-noise sequence. Figure 11.5 shows the prediction error sequence, which indeed appears uncorrelated from sample to sample, i.e., white.

### 1.11.3.2 Moving average (MA) models

A discrete-time signal  $x[n]$  is called a moving average (MA) process of order  $q$  if it can be written in the form [3,6]

$$x[n] = b_0 w[n] + b_1 w[n-1] + \cdots + b_q w[n-q], \quad (11.20)$$

**FIGURE 11.5**

Prediction error from estimating colored noise using an AR model.

where  $b_0, \dots, b_q$  are parameters of the MA model and the signal  $w[n]$  is a white noise process, i.e., we have that  $C_w[m] = \sigma^2 \delta[m]$ . Since the sequence  $x[n]$  can be viewed as a linear combination of the most recent values of the sequence then it is called a “moving average.” From this structure, whenever the sequence  $x[n]$  is given as the output of an FIR filter whose input is a white-noise sequence, then we can refer to as a moving average sequence. More generally, even when the input is not a white-noise sequence, the process of filtering with an FIR filter can be viewed as taking a “moving average” of the input. Financial models often use 50-day and 100-day moving averages as indicators of trends in various stock prices and indices. In such a context,  $q$  and  $b_k$  are usually taken such that  $b_k = \frac{1}{q+1}$ ,  $k = 0, \dots, q$ , such that  $x[n] = \frac{1}{q+1} \sum_{k=0}^q w[n-k]$ , i.e., the arithmetic mean of the most recent  $q+1$  values of  $w[n]$ .

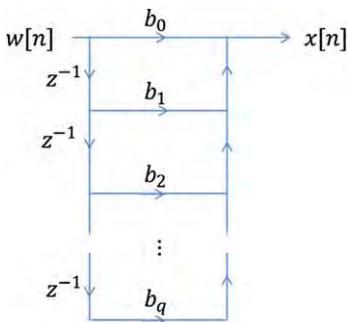
Viewing the relationship between the sequences  $x[n]$  and  $w[n]$  with a signal flowgraph model, in Figure 11.6a, we see that  $x[n]$  can be viewed as an FIR filtered version of the sequence  $w[n]$ . Similarly, we can view this in Figure 11.6b, via an input-output transfer function relationship between the two sequences.

### 1.11.3.3 Autoregressive moving average (ARMA) models

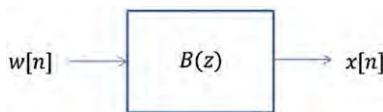
A discrete-time signal  $x[n]$  is an autoregressive moving average (ARMA) process of order  $q, p$  if it can be written in the form [3, 6]

$$x[n] + a_1 x[n-1] + \cdots + a_p x[n-p] = b_0 w[n] + \cdots + b_q w[n-q], \quad (11.21)$$

where  $a_1, \dots, a_p, b_0, \dots, b_q$  are parameters of the ARMA model and the signal  $w[n]$  is a white noise process, i.e., we have that  $C_w[m] = \sigma^2 \delta[m]$ . We can once again relate the sequences  $x[n]$  and  $w[n]$

**FIGURE 11.6a**

Moving average flowgraph model.

**FIGURE 11.6b**

Transfer function perspective of the MA model relating  $x[n]$  and  $w[n]$ .

through a transfer function relationship, such as

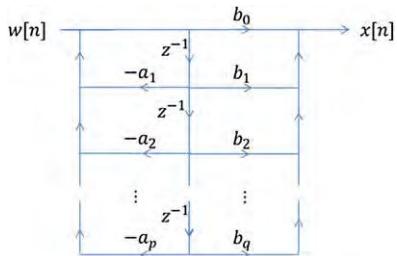
$$X(z) = \left( \frac{\sum_{k=0}^q b_k z^{-k}}{1 + \sum_{k=1}^p a_k z^{-k}} \right) W(z). \quad (11.22)$$

This input-output relationship implies that we can view the sequence  $x[n]$  as being generated by filtering the sequence  $w[n]$  with a rational function, i.e., a “pole-zero” filter with finite order numerator and denominator polynomials, where the numerator polynomial is given by  $B(z) = \sum_{k=0}^q b_k z^{-k}$ , and the denominator polynomial is given by  $A(z) = 1 + \sum_{k=1}^p a_k z^{-k}$ . It is usually assumed that the rational function is “strictly proper,” which means that the numerator order is strictly less than that of the denominator, i.e.,  $q < p$ . For this reason, we may refer to the sequence as an ARMA( $p$ ) process, rather than ARMA( $q, p$ ). Figures 11.7a and 11.7b illustrate the flowgraph and transfer function perspective of the ARMA model relating the sequences  $x[n]$  and  $w[n]$ .

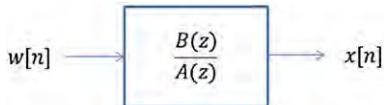
#### 1.11.3.4 Parametric modeling for system function approximation

One of the primary reasons that parametric models are used is to obtain finite-order (rational system) approximations to systems that we encounter in real-world engineering problems. Whether the data of interest are taken from a mechanical system, an electrical device, a digital communications link, or even a financial or economic model, we often seek input-output relationships that can explain the behaviors we observe such that we might be able to better understand them, or even hope to control them.

There are a number of ways in which input-output behavior can be modeled though two are perhaps most common. One is based on observations of input-output pairs in the time domain, from which

**FIGURE 11.7a**

ARMA flowgraph model.

**FIGURE 11.7b**

Transfer function perspective of the ARMA model relating  $x[n]$  and  $w[n]$ .

an explanation of such behavior is sought. Another other comes from attempts to fit a given system function or frequency response that either arose from a more detailed physical model, or which was measured using instrumentation, such as a spectrum analyzer in response to a variety of probe signals. These two approaches can certainly be linked, through the Fourier transform, and we will move back and forth between the two domains depending on the methods at hand. For example, given a frequency response  $H(e^{j\omega})$  that we sought to model, we could readily transform this into the input-output pair  $x[n] = \delta[n]$ ,  $y[n] = h[n]$ , defining the impulse response of the system as the response of the system to an impulse as input, using the definition of the discrete-time Fourier transform [1],

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}, \quad (11.23)$$

$$h[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n} dw. \quad (11.24)$$

In general, direct ARMA modeling is difficult, since the formulation of the optimization of interest is highly nonlinear in the parameters of the model. For example, given a frequency response  $H(e^{j\omega})$ , the deterministic least squares optimal ARMA( $q, p$ ) model satisfies the following minimization

$$\min_{\{a_k\}_{k=1}^p \{b_k\}_{k=0}^q} \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) - \frac{\sum_{k=0}^q b_k e^{-jk\omega}}{1 + \sum_{k=1}^p a_k e^{-jk\omega}} \right|^2 d\omega, \quad (11.25)$$

which is highly nonlinear in the parameters of the ARMA model. This can be rewritten in the time domain, by application of Parseval's relation [1], which equates energy calculated in the time domain

with that calculated in the frequency domain, yielding,

$$\min_{\{a_k\}_{k=1}^p \{b_k\}_{k=0}^q} \sum_{n=-\infty}^{\infty} \left| h[n] - \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\sum_{k=0}^q b_k e^{-jk\omega}}{1 + \sum_{k=1}^p a_k e^{-jk\omega}} e^{j\omega n} d\omega \right|^2, \quad (11.26)$$

which is equally cumbersome.

This problem formulation is difficult due to the manner in which the AR parameters of interest are nonlinearly related to the optimization criterion. If we were only concerned with a MA estimate and wanted to use this least-squares criterion, then the time-domain formulation can provide a simple solution, whereby

$$\min_{\{b_k\}_{k=0}^q} \sum_{n=-\infty}^{\infty} \left| h[n] - \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{k=0}^q b_k e^{-jk\omega} e^{j\omega n} d\omega \right|^2, \quad (11.27)$$

which can be rewritten

$$\min_{\{b_k\}_{k=0}^q} \sum_{n=-\infty}^{\infty} \left| h[n] - \sum_{k=0}^q b_k \delta[n-k] \right|^2, \quad (11.28)$$

yielding the simple, though somewhat unsatisfying, solution,

$$b_k = h[k], \quad k = 0, \dots, q. \quad (11.29)$$

This is nothing more than a simple truncation approximation to the desired impulse response for the system of interest. If a more elaborate response is desired, for example, one that is IIR, then poles are necessary, in addition to the zeros of the approximating transfer function, and an AR or ARMA model should be considered. This again would lead to the nonlinearities we just encountered.

An approach that had proven useful for AR modeling is one that takes the engineering approach to problem solving, that is, if a problem is difficult to solve directly, attempt to change it into one that is more readily solvable. In this light, we will see how the nonlinear optimization for the AR formulation can be transformed into a linear optimization through a clever insight. First, we recall the form of the AR model for the transfer function, namely,

$$H(z) = \frac{1}{1 + \sum_{k=1}^p a_k z^{-k}}. \quad (11.30)$$

Beginning with this expression and taking the inverse z-transform, we see that the impulse response of the modeled system must satisfy

$$h[n] + \sum_{k=1}^p a_k h[n-k] = \delta[n]. \quad (11.31)$$

Now given a desired (or measured) impulse response, which we will denote,  $h_d[n]$ , we could attempt to minimize the deviation of this expression from that which is expected from this recursion and replace the nonlinear optimization above with the simpler form

$$\min_{\{a_k\}_{k=1}^p} \sum_{n=-\infty}^{\infty} \left| h_d[n] + \sum_{k=1}^p a_k h_d[n-k] \right|^2, \quad (11.32)$$

which is a quadratic minimization in the parameters of interest and therefore admits a closed-form solution! Specifically, denoting the quadratic error term in the minimization above as  $E$  we can write

$$0 = \frac{\partial E}{\partial a_\ell} = 2 \sum_{n=-\infty}^{\infty} \left( h_d[n] + \sum_{k=1}^p a_k h_d[n-k] \right) h_d[n-\ell], \quad \ell = 1, \dots, p, \quad (11.33)$$

which comprise a set of linear equations in  $a_\ell$ ,  $\ell = 1, \dots, p$ . We can write these equations more compactly by exchanging the order of summation, as

$$\sum_{k=1}^p a_k \underbrace{\sum_{n=-\infty}^{\infty} h_d[n-k] h_d[n-\ell]}_{\hat{R}_{h_d h_d}[k-\ell]} = - \underbrace{\sum_{n=-\infty}^{\infty} h_d[n] h_d[n-\ell]}_{\hat{R}_{h_d h_d}[\ell]}, \quad \ell = 1, \dots, p. \quad (11.34)$$

We have made use of the definition which is often called the deterministic auto-correlation sequence for the sequence. We can then summarize the set of equations in matrix form, sometimes referred to as the Normal Equations [3,6],

$$R\vec{a} = -\vec{r}, \quad (11.35)$$

or

$$\begin{bmatrix} \hat{R}_{h_d h_d}[0] & \dots & \hat{R}_{h_d h_d}[p-1] \\ \vdots & \ddots & \vdots \\ \hat{R}_{h_d h_d}[p-1] & \dots & \hat{R}_{h_d h_d}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} \hat{R}_{h_d h_d}[1] \\ \vdots \\ \hat{R}_{h_d h_d}[p] \end{bmatrix}, \quad (11.36)$$

which has the solution,

$$\vec{a} = -R_{-1}\vec{r}. \quad (11.37)$$

The matrix  $R$  has special structure that makes solving such linear equations not only straightforward, but also fast. Namely,  $R$  is known as a symmetric Toeplitz matrix, for which fast linear equation solvers can be readily obtained [3,6]. This special structure, together with the right hand side of the linear equations led to a particularly fast algorithm for its solution, known as the Levinson-Durbin recursion [3]. If we were to replace the least-squares enforcement of the prediction error criterion with a mean square error criterion, i.e., we sought to minimize the mean square error

$$\min_{\{a_k\}_{k=1}^p} E \left\{ \left| h_d[n] + \sum_{k=1}^p a_k h_d[n-k] \right|^2 \right\}, \quad (11.38)$$

we would naturally arrive at a similar form for its solution, namely, we would have

$$R\vec{a} = -\vec{r} \quad (11.39)$$

or

$$\begin{bmatrix} R_{h_d h_d}[0] & \dots & R_{h_d h_d}[p-1] \\ \vdots & \ddots & \vdots \\ R_{h_d h_d}[p-1] & \dots & R_{h_d h_d}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} R_{h_d h_d}[1] \\ \vdots \\ R_{h_d h_d}[p] \end{bmatrix}, \quad (11.40)$$

which has the solution,

$$\vec{a} = -R^{-1}\vec{r}, \quad (11.41)$$

where, the only difference in our solution lies in replacing the deterministic auto-correlation with the statistical auto-correlation,  $R_{h_d h_d}[\ell] = E\{h_d[n]h_d[n - \ell]\}$ , where the expectation is taken over the distribution of the random sequence  $h_d[n]$ . This approach to autoregressive modeling is often called “linear prediction” or “linear predictive modeling” due to the use of the prediction error form that is used to make the optimization problem linear in the parameters of interest [3,6,7].

Returning to the ARMA modeling problem, there are a number of ways of incorporating zeros into the modeling problem while still using the much simpler prediction-error formulation developed so far. Perhaps the simplest is to just use the zeros to match the first values of the impulse response exactly, that is to first solve for the denominator coefficients as above (predictive AR modeling) and then select

$$b_\ell = h_d[\ell] + \sum_{k=1}^p a_k h_d[\ell - k], \quad k = 0, \dots, q, \quad (11.42)$$

which follows from the system function relation

$$H_d(z) = \left( \frac{\sum_{k=0}^q b_k z^{-k}}{1 + \sum_{k=1}^p a_k z^{-k}} \right). \quad (11.43)$$

This two-step process, namely (1) determine the AR parameters using linear predictive modeling, followed by (2) select the MA parameters by exactly matching the first  $q$  samples of the impulse response is often called Prony’s method [3,7]. In general, this method provides a reasonable set of AR parameters; however the MA parameters are not particularly well-modeled. That is, if a system were known to have a given ARMA form, and its parameters were to be estimated from noisy observations using Prony’s method, the AR parameters would likely well-match the true underlying system, however the MA parameters would likely be a poor match.

An improvement over this approach is to once again use a least squares, or MMSE criterion to obtain the MA parameters in a modified two-step approach. The first of the two steps remains the same, that is, use the predictive-least squares (or MMSE) approach to find the AR parameters according to the Normal equations as before. However, now that a set of AR parameters are given, the sequence  $v[n]$  is defined as follows:

$$v[n] = - \sum_{k=1}^p a_k v[n - k] + \delta[n], \quad (11.44)$$

i.e.,  $v[n]$  is defined to be the impulse response of the AR model derived in step (1). Now, the goal is to minimize the discrepancy between the desired impulse response  $h_d[n]$  and the output of the cascade of the AR model and the MA model, such that the sequence  $v[n]$  passed through the FIR filter with transfer function  $B(z)$  is close to the desired impulse response. Specifically, we select the MA coefficients to minimize the following criterion,

$$\min_{\{b_k\}_{k=0}^q} \sum_{n=-\infty}^{\infty} \left| h_d[n] + \sum_{k=0}^q b_k v[n - k] \right|^2, \quad (11.45)$$

which is once again a quadratic minimization admitting a closed-form solution. The linear equations reduce to

$$\sum_{k=0}^q b_k \underbrace{\sum_{n=-\infty}^{\infty} v[n-k]v[n-\ell]}_{\hat{R}_{vv}[k-\ell]} = - \underbrace{\sum_{n=-\infty}^{\infty} h_d[n]v[n-\ell]}_{\hat{R}_{h_dv}[\ell]}, \quad \ell = 0, \dots, q, \quad (11.46)$$

which admits the solution,

$$\vec{b} = - \begin{bmatrix} R_{vv}[0] & \dots & R_{vv}[q] \\ \vdots & \ddots & \vdots \\ R_{vv}[q] & \dots & R_{vv}[0] \end{bmatrix}^{-1} \begin{bmatrix} R_{h_dv}[0] \\ \vdots \\ R_{h_dv}[q] \end{bmatrix}, \quad (11.47)$$

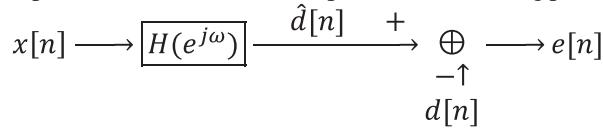
$$\vec{b} = -(R_{vv})^{-1} \vec{r}_{hv}, \quad (11.48)$$

where, we denote

$$R_{vv} = \begin{bmatrix} R_{vv}[0] & \dots & R_{vv}[q] \\ \vdots & \ddots & \vdots \\ R_{vv}[q] & \vdots & R_{vv}[0] \end{bmatrix}, \quad \vec{r}_{hv} = \begin{bmatrix} R_{h_dv}[0] \\ \vdots \\ R_{h_dv}[q] \end{bmatrix}. \quad (11.49)$$

### 1.11.3.5 Parametric modeling for joint process estimation

One of the classical signal estimation and modeling problems that makes use of parametric estimation methods like those developed here, include the adaptive filtering and joint process estimation problem [2]. Most generally, these problems can be viewed as a parametric modeling problem depicted as follows:



where the parameters of a model for  $H(e^{j\omega})$  are adjusted to minimize the discrepancy  $e[n] = (d[n] - \hat{d}[n])$  in a suitable manner [8]. When the parameters of the model are adjusted in an online (sequential) manner, such that the filter parameters  $\{h[k]\}_{k=0}^p$  are updated after each observation of the desired signal, this is commonly referred to as “adaptive filtering” [8]. When a batch of data  $\{x[k]\}_{k=0}^N$  is observed and the parameters  $\{h[k]\}_{k=0}^p$  are selected based on this entire set of observations, this problem is commonly referred to as batch or block processing. We will first consider the stochastic case, where the discrepancy to be minimized takes the form

$$\{h_{\text{MMSE}}[k]\}_{k=0}^p = \arg \min_{\{h[k]\}_{k=0}^p} \epsilon \{(d[n] - \hat{d}[n])^2\}. \quad (11.50)$$

Writing out this minimization using the parameters of the model, leads to

$$\{h_{\text{MMSE}}[k]\}_{k=0}^p = \arg \min_{\{h[k]\}_{k=0}^p} \epsilon_{\text{MMSE}} = E \left\{ \left( d[n] - \sum_{k=0}^p h[k]x[n-k] \right)^2 \right\}. \quad (11.51)$$

To proceed, we consider that the sequences  $x[n]$  and  $d[n]$  are wide-sense stationary stochastic processes, with zero mean and covariance functions [6]

$$C_x[n, n - m] = E\{x[n]x[n - m]\} = R_x[m] = R_x[-m], \quad (11.52)$$

$$C_d[n, n - m] = E\{d[n]d[n - m]\} = R_d[m] = R_d[-m], \quad (11.53)$$

$$C_{xd}[n, n - m] = E\{x[n]d[n - m]\} = R_{xd}[m] = R_{dx}[-m]. \quad (11.54)$$

We can now solve for the minimizing set of parameters by differentiating this quadratic expression with respect to each of the unknown parameters, which leads to

$$0 = \frac{\partial \epsilon_{\text{MMSE}}}{\partial h[\ell]} = -2E \left\{ \left( d[n] - \sum_{k=0}^p h[k]x[n - k] \right) x[n - \ell] \right\}, \quad \ell = 0, \dots, p. \quad (11.55)$$

Rearranging terms, we arrive at

$$0 = E\{d[n]x[n - \ell]\} - \sum_{k=0}^p h[k]E\{x[n - k]x[n - \ell]\}, \quad \ell = 0, \dots, p, \quad (11.56)$$

$$E\{d[n]x[n - \ell]\} = \sum_{k=0}^p h[k]E\{x[n - k]x[n - \ell]\}, \quad \ell = 0, \dots, p, \quad (11.57)$$

$$R_{dx}[\ell] = \sum_{k=0}^p h[k]R_{xx}[k - \ell], \quad \ell = 0, \dots, p, \quad (11.58)$$

which can be recognized as another form of the auto-correlation normal equations we have seen previously for AR modeling. We have

$$\begin{bmatrix} R_{xx}[0] & \dots & R_{xx}[p] \\ \vdots & \ddots & \vdots \\ R_{xx}[p] & \dots & R_{xx}[0] \end{bmatrix} \begin{bmatrix} h[0] \\ \vdots \\ h[p] \end{bmatrix} = \begin{bmatrix} R_{dx}[0] \\ \vdots \\ R_{dx}[p] \end{bmatrix}, \quad (11.59)$$

which has the solution,

$$\begin{bmatrix} h_{\text{MMSE}}[0] \\ \vdots \\ h_{\text{MMSE}}[p] \end{bmatrix} = \begin{bmatrix} R_{xx}[0] & \vdots & R_{xx}[p] \\ \vdots & \ddots & \vdots \\ R_{xx}[p] & \vdots & R_{xx}[0] \end{bmatrix}^{-1} \begin{bmatrix} R_{dx}[0] \\ \vdots \\ R_{dx}[p] \end{bmatrix}, \quad (11.60)$$

$$\vec{h}_{\text{MMSE}} = R^{-1} \vec{p}, \quad (11.61)$$

taking a similar form to the AR modeling case. Once again, since the matrix to be inverted is an auto-correlation matrix, and is symmetric, positive semidefinite, and Toeplitz, there exist fast, and numerically

stable algorithms for solving for the MMSE-optimal parameters [6,8]. This set of parameters minimizes the mean square error, and is hence called the MMSE-optimal set of model parameters. They are also often referred to as the Wiener solution, or the Wiener filter [8].

A deterministic formulation of the problem could be made, using the least-squares criterion, which would again lead to a set of equations in a similar form, specifically, defining the error criterion to be

$$\{h_{LS}[k]\}_{k=0}^p = \arg \min_{\{h[k]\}_{k=0}^p} \epsilon_{LS} = \sum_{n=-\infty}^{\infty} \left( d[n] - \sum_{k=0}^p h[k]x[n-k] \right)^2. \quad (11.62)$$

This leads to

$$\sum_{n=-\infty}^{\infty} d[n]x[n-\ell] = \sum_{k=0}^p h[k] \sum_{n=-\infty}^{\infty} x[n-k]x[n-\ell], \quad \ell = 0, \dots, p, \quad (11.63)$$

$$\hat{R}_{dx}[\ell] = \sum_{k=0}^p h[k] \hat{R}_{xx}[k-\ell], \quad \ell = 0, \dots, p, \quad (11.64)$$

where  $\hat{R}_{dx}[\ell]$  and  $\hat{R}_{xx}[\ell]$  are defined using summations over the observed data, rather than taking a statistical expectation. The resulting set of normal equations are identical, with the statistical auto- and cross-correlation functions replaced by their deterministic counterparts, arriving at the solution,

$$\begin{bmatrix} h_{LS}[0] \\ \vdots \\ h_{LS}[p] \end{bmatrix} = \begin{bmatrix} \hat{R}_{xx}[0] & \dots & \hat{R}_{xx}[p] \\ \vdots & \ddots & \vdots \\ \hat{R}_{xx}[p] & \vdots & \hat{R}_{xx}[0] \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}_{dx}[0] \\ \vdots \\ \hat{R}_{dx}[p] \end{bmatrix}, \quad (11.65)$$

$$\vec{h}_{LS} = \vec{R}^{-1} \hat{\vec{p}}. \quad (11.66)$$

### 1.11.3.6 Sequential parametric modeling

In certain signal processing problems, the observations may arrive sequentially and the application may require immediate output based on only the available data. In other applications, different model parameters may be suitable for different parts of the observations such that parameters are updated in time for better fit. A common approach to consolidate these is to produce the estimated parameters in a sequential manner. Unlike the previous approaches where we have the batch data  $\{x[k]\}_{k=0}^N$ , the parameters  $\{x[k]\}_{k=0}^p$  are selected based on only currently available data  $\{x[k]\}_{k=-\infty}^n$ . As in the previous sections, a deterministic formulation of the problem could be made, using the least-squares criterion, based only on the available data as

$$\{h_{LS}[k, n]\}_{k=0}^p = \operatorname{argmin}_{\{h[k]\}_{k=0}^p} \sum_{t=-\infty}^n \left( d[t] - \sum_{k=0}^p h[k]x[t-k] \right)^2. \quad (11.67)$$

The estimated parameters are now function of time and updated as new observations arrive. To emphasize the most recent data in the estimate, the least-squares criterion is often defined over a window of observations or a weighted least squares problem is formulated, such as

$$\{h_{LS}[k, n]\}_{k=0}^p = \underset{\{h[k]\}_{k=0}^p}{\operatorname{argmin}} \sum_{t=-\infty}^n \lambda_{n-t} \left( d[t] - \sum_{k=0}^p h[k]x[t-k] \right)^2, \quad (11.68)$$

where the emphasized window length is intrinsically set with weighting parameter  $0 < \lambda \leq 1$ , e.g., when  $\lambda = 0.9$  the most recent data is emphasized in that any samples that are more than  $(\frac{1}{1-\lambda}) = 10$  samples old are weighted by  $1/e$  or less, as compared with the most recent data sample. The sequential and weighted least-squares criterion leads to a similar set of normal equations

$$\begin{bmatrix} h_{LS}[0, n] \\ \vdots \\ h_{LS}[p-n] \end{bmatrix} = \begin{bmatrix} \hat{R}_{xx}[0, n] & \dots & \hat{R}_{xx}[p, n] \\ \vdots & \ddots & \vdots \\ \hat{R}_{xx}[p, n] & \dots & \hat{R}_{xx}[0, n] \end{bmatrix}^2 \begin{bmatrix} \hat{R}_{dx}[0, n] \\ \vdots \\ \hat{R}_{dx}[p, n] \end{bmatrix}, \quad (11.69)$$

$$\vec{h}_{LS}[n] = \hat{R}[n]^{-1} \hat{p}[n]. \quad (11.70)$$

However, here, the sample auto-correlation matrix and cross-correlation vector are calculated using only the available data, and hence, are time varying,

$$\hat{R}_{xx}[k-\ell, n] = \sum_{t=-\infty}^n \lambda^{n-t} x[t-k]x[t-\ell], \quad \ell = 0, \dots, p, \quad (11.71)$$

$$\hat{R}_{dx}[\ell, n] = \sum_{t=-\infty}^n \lambda^{n-t} d[t]x[t-\ell], \quad \ell = 0, \dots, p. \quad (11.72)$$

Although successful in most signal processing applications, this online processing requires calculating the sample correlation matrix, cross-correlation vector and performing inversion for each new sample, which may be computationally infeasible for some applications. However, by recognizing the time recursions in

$$\hat{R}_{xx}[k-\ell, n+1] = \lambda \hat{R}_{xx}[k-\ell, n] + x[n+1-k]x[n+1-\ell], \quad \ell = 0, \dots, p, \quad (11.73)$$

$$\hat{R}_{dx}[\ell, n+1] = \lambda \hat{R}_{dx}[\ell, n] + d[n+1]x[n+1-\ell], \quad \ell = 0, \dots, p, \quad (11.74)$$

and using the matrix inversion lemma [8], we can solve the corresponding estimation problem in a recursive form such that the estimated model parameters at time  $n+1$  can be readily obtained from the estimated parameters at time  $n$  as

$$e[n] = d[n] - \hat{d}[n], \quad (11.75)$$

$$\vec{g}[n] = p[n] \vec{x}[n] \{ \lambda + \vec{x}[n]^T p[n] \vec{x}[n] \}^{-1}, \quad (11.76)$$

$$p[n+1] = \lambda^{-1} p[n] - \vec{g}[n] \vec{x}[n]^T \lambda^{-1} p[n], \quad (11.77)$$

$$\vec{h}[n+1] = \vec{h}[n] + \vec{g}[n] e[n] \quad (11.78)$$

yielding the “recursive least squares” (RLS) algorithm [8], where  $P[n] = \hat{R}[n]^{-1}$ . While there exist more computationally efficient lattice implementations of this update without the matrix inversion lemma, the RLS algorithm is shown to be numerically more stable [8].

We observe from this recursive update that the corresponding estimated parameter vector  $\vec{h}[n]$  at time  $n$  is basically updated by an additive vector, called the gain vector, times the estimation error to yield the parameters at time  $n + 1$  as

$$\vec{h}[n + 1] = \vec{h}[n] + \vec{g}[n]e[n], \quad (11.79)$$

where the gain vector is the inverse correlation matrix multiplied by the data vector  $\vec{x}[n]$  with certain power scaling. A common approach in signal processing to reduce the computational complexity of this update is to replace the scaled inverse correlation matrix by the inverse of the trace of the correlation matrix

$$\vec{h}[n + 1] = \vec{h}[n] + \frac{1}{\text{Trace}(\hat{R}[n])}\vec{x}[n]e[n]. \quad (11.80)$$

Subsequently, the inverse of the trace, which corresponds to the power in the stochastic process, is further replaced by a positive constant  $\mu > 0$  as

$$\vec{h}[n + 1] = \vec{h}[n] + \mu\vec{x}[n]e[n], \quad (11.81)$$

yielding the celebrated least mean squares (LMS) adaptive algorithm [9].

When the underlying stochastic model is known, i.e., the cross and auto-correlation functions are known, the parameters that minimize the MSE are evaluated by solving the normal equations

$$\vec{h}_{\text{MMSE}} = R^{-1}\vec{p}. \quad (11.82)$$

The same normal equations can be iteratively solved by applying a gradient descent algorithm [8] to the mean square error cost function, yielding a recursive update

$$\vec{h}^{(k+1)} = \vec{h}^{(k)} - \mu \nabla_{\vec{h}^{(k)}} E[e^2[n]], \quad (11.83)$$

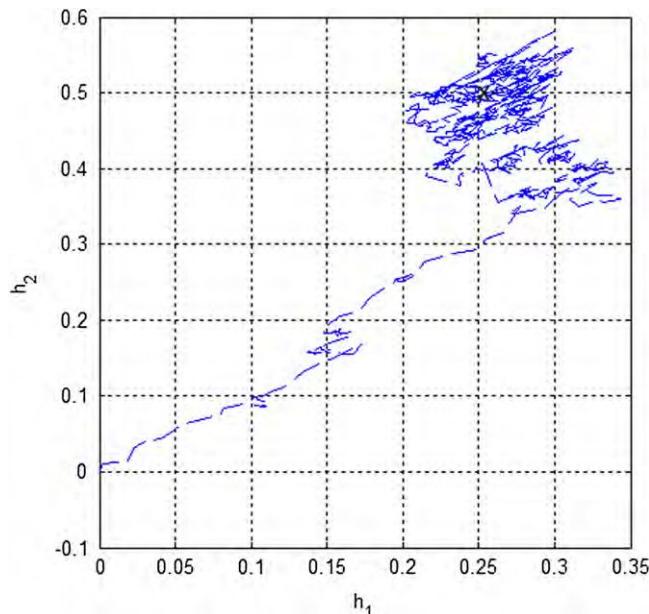
$$\vec{h}^{(k+1)} = \vec{h}^{(k)} - \mu(\vec{p} - R \vec{h}^{(k)}), \quad (11.84)$$

which converges to  $\vec{h}_{\text{MMSE}}$  as  $k$  increases provided that  $\mu$  is selected appropriately [8,9]. If the underlying  $R$  and  $p$  are not known, then the same iteration can be carried in time domain by replacing the expected variables by their temporal values, in other words by replacing  $E[e^2[n]]$  with  $e^2[n]$ , to yield

$$\vec{h}^{[n+1]} = \vec{h}[n] - \mu \nabla_{\vec{h}[n]} e^2[n], \quad (11.85)$$

$$\vec{h}^{[n+1]} = \vec{h}^{[n]} - \mu(\vec{x}[n]e[n]), \quad (11.86)$$

which yields the stochastic interpretation of the LMS algorithm. Hence, the LMS algorithm can be considered a gradient descent algorithm, where a “stochastic” gradient is used in the update instead of the true gradient as shown in Figure 11.8 [8].

**FIGURE 11.8**

Graphical depiction of the convergence of the stochastic gradient (LMS) algorithm, depicted for a two tap filter initialized at  $h = [0, 0]$ .

### 1.11.3.7 Universal model order estimation

Even when a parametric model is known to be appropriate for a particular application, e.g., an AR, ARMA or MA model, we still need to decide on the number of parameters in the corresponding parametric model. Determination of the proper number of parameters to use for a parametric model is known to be a difficult problem and has a rich history in a number of applications in signal processing, machine learning and communications [8, 10]. If the complete statistical properties of the underlying signals were known, then increasing the number of parameters would have only increased the modeling power, i.e., more parameters are better. However, since the model parameters should be learned from a limited amount of training data, this can cause over fitting, especially in short-data regimes, and lead to poor extrapolation outside of the training set, i.e., too many parameters can be bad. To avoid such over fitting problems, much of the early work in the model order determination literature focused on methods that are based on assumed probabilistic models, such as the minimum description length (MDL) or the Akaike information criterion (AIC) [6, 11]. These methods establish a balance between modeling power of the parameters using certain maximum likelihood, or information theoretic measures, while penalizing the number of parameters using certain regularization terms. For example, for AR modeling with white Gaussian noise, the leading term of the penalty term in the MDL for a model of order  $p$  and a signal length  $n$  is given by  $\frac{p}{2} \log(n)$ . Hence, for a signal  $d[t]$  according to the original definition of

MDL, the model order is selected by minimizing

$$\min_p \left\{ \min_{\{h[k]\}_{k=0}^p} \sum_{t=1}^n \left( d[t] - \sum_{k=1}^p h[k]d[t-k] \right)^2 + \left( \frac{p}{2} \right) \log(n) \right\}. \quad (11.87)$$

While these statistical approaches have been widely used in different signal processing applications, they require considerable statistical information on the signals of interest and can be fragile to operating in regimes in which the signals, for example, do not behave according to the assumed statistical model.

However, recent work in the machine learning and information theory communities have applied a new approach to tackling the model order selection problem [12, 13]. As an example, in the universal model order selection approach, instead of making hard decisions at each instant based on an assumed statistical model, one uses a performance based mixture of all models of different orders in an attempt to perform as well as or better than the best model in the mixture. The resulting algorithms then successfully navigate the short-data record regime by placing more emphasis on lower order models, while achieving the ultimate precision of higher order models as the data record grows to accommodate them. The elegance of the universal mixture methods is that this occurs naturally by monitoring the performance of each model on the data observed so far and placing more emphasis on the results of better performing models in an online manner.

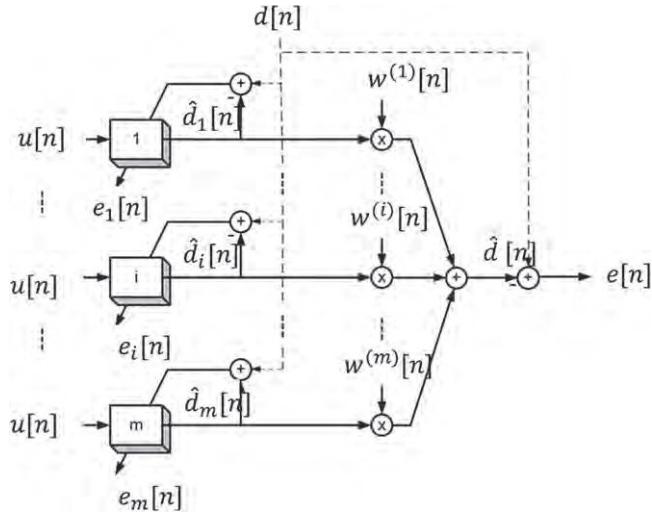
This model combination approach is successfully applied to model order selection in the context of RLS adaptive filtering when several different linear predictors, e.g., AR models, are used for prediction, regression or estimation. Clearly, the order of linear predictor heavily depends on the available amount of data and assumed statistical models. Here, instead of fixing a particular model order, one can use an explicit adaptive convex combination of different order linear predictors  $\hat{x}_p[n]$ , up to some order  $M$ , as

$$\hat{x}[n] = \sum_{p=1}^M \mu_p[n] \hat{x}_p[n], \quad (11.88)$$

where the corresponding adaptive combination weights  $\mu_p[n]$  are calculated intuitively as  $\mu_p[n] \sim \exp(-\sum_{t=1}^n (x(t) - \hat{x}_p[t])^2)$ , i.e., based on the performance of the predictor on the observed data so far. An example weighted mixture that combines  $m$  adaptive filters is shown in Figure 11.9. The output is simply calculated as a performance-weighted mixture. For these applications, an efficient lattice filter structure can be used to not only implement each of the elements of the competing class, but to also construct the universal predictor output all with complexity that is only linear in the largest model order in the competing class of models [9]. Such a model combination is shown to achieve, for any  $n$  [12]

$$\frac{1}{n} \sum_{n=1}^N (x[n] - \hat{x} - \hat{x}[n])^2 \leq \min_{p < M} \frac{1}{n} \sum_{n=1}^N (x[n] - \hat{x}_p[n])^2 + \frac{1}{n} \ln(M). \quad (11.89)$$

The ability to asymptotically achieve the performance of the best model order in the set of models gives rise to the term “universal filter” and hence such methods are often called “universal” methods.

**FIGURE 11.9**

Adaptively combining the outputs of multiple adaptive filters.

### 1.11.3.8 Universal linear predictor

In parameter estimation, the universal algorithms approach can be generalized to construct estimators that are, in a deterministic sense, min-max optimal [13]. As an example, once again let us focus on AR parameter estimation under the deterministic total square estimation error. However, instead of constructing sequential normal equations, as in the previous section, we hypothetically implement all of the continuum of AR models for each  $\vec{h} \in R^P$  and construct a sequential estimate based on a performance-weighted mixture of all these AR models as in the previous section. By taking an implicit performance-weighted mixture over the outputs of all adaptive filters  $\vec{h} \in R^P$  and using a priori Gaussian weighting  $p(\hat{h})$ , the integral

$$\int_{R^P} \vec{h}^T \vec{x}[n] p(\vec{h}) \sum_{t=1}^{n-1} e^2[t] d\vec{h}, \quad (11.90)$$

analogous to the weighted combination in the previous section, can be evaluated in closed form and leads to a diagonally loaded RLS adaptive filtering algorithm [12]

$$\vec{h}_{uni}[n] = \vec{R}[n-1]^{-1} \hat{\vec{p}}[n]. \quad (11.91)$$

However, in this new recursive update, we have  $\hat{R}[n-1] = \sum_{t=1}^{n-1} \vec{x}[t]\vec{x}^T + \delta I$ . Hence, the diagonal loading term is the missing ingredient that takes the standard RLS adaptive filtering algorithm from being simply universal to attaining the min-max optimal performance [13]. When applied to any bounded and arbitrary sequence, this universal recursive algorithm yields the min-max optimal performance bound

for any  $N$  as

$$\frac{1}{n} \sum_{n=1}^N (x[n] - \vec{h}_{uni}[n]^T \vec{x}[n])^2 \leq \min_{\vec{h}} \frac{1}{n} \sum_{n=1}^N (d[n] - \vec{h}^T \vec{x}[n])^2 + \frac{1}{n} O(p \ln N). \quad (11.92)$$

This is min-max optimal, in that no other approach can achieve a lower “regret” than the  $\frac{1}{n} O(p \ln N)$  term above [14].

### 1.11.3.9 Universality with respect to classes that vary in space: nonlinear classes/piecewise in space

Linear AR, MA, or ARMA models discussed in the previous sections are extensively used in signal processing due to their tractability and adequate accuracy in modeling [9]. Recently, nonlinear models and those based on piecewise linear and locally linear approximations have gained significant attention as they capture the salient characteristics of many physical phenomena. Nonlinear parametric estimation methods, for example, Volterra filters [8] have proven attractive for a number of applications. By removing structural constraints on linearity, nonlinear models are perhaps more appropriate for a variety of data exhibiting saturation effects, threshold phenomena or other nonlinear behavior.

Piecewise linear modeling can be viewed as a natural extension to linear modeling, in which the space spanned by past observations is partitioned into a union of disjoint regions over each of which an affine model is fitted. In each region, a linear model can be estimated, and as the number of regions grows, the piecewise linear model can better approximate any smoothly varying nonlinear estimator. As an example to a piecewise linear estimator, suppose the space of past observations  $\vec{x}[n] \in R^p$  is partitioned into  $k$  disjoint regions  $R^p = \bigcup_{k=1}^K V_k$ . Based on this partitioning, the observations can be divided into  $k$  disjoint sets, where  $k$  different normal equations are written as

$$\vec{R}_k = \sum_{n=1}^N \vec{x}[n] \vec{x}[n]^T I_k(\vec{x}[n]), \quad (11.93)$$

$$\vec{p}_k = \sum_{n=1}^N d[n] \vec{x}[n]^T I_k(\vec{x}[n]), \quad (11.94)$$

and  $I_k(\vec{x}[n])$  is the indicator function of the  $k$ th region, i.e.,  $I_k(\vec{x}[n]) = 1$  if  $\vec{x}[n] \in V_k$ , for each region. This partitioning of the observations yields a linear estimator,

$$\vec{h}_k = \hat{R}_k^{-1} \hat{p}_k, \quad (11.95)$$

$k = 1, \dots, k$ , at each region, yielding a piecewise linear model. A similar formulation is possible when the data arrives sequentially, however, in that case, the AR parameters are estimated sequentially using time dependent normal equations. As an example, for sequential processing, we can run  $k$  different LMS algorithms, one for each region, to get an adaptive piecewise linear model trained as

$$\vec{h}_k[n+1] = \vec{h}_k[n] + \mu e_k[n] \vec{x}[n] I_k(\vec{x}[n]), \quad (11.96)$$

where  $e_k[n] = d[n] - \vec{h}_k[n]^T \vec{x}[n]$ . These types of piecewise linear models have been referred to in the signal processing literature as “nonlinear autoregressive models” and in the signal processing and statistics literature as “self-exciting threshold auto regressive models,” and have been used in modeling a wide range of data in fields ranging from population biology to econometrics to glottal flow in voiced speech [15].

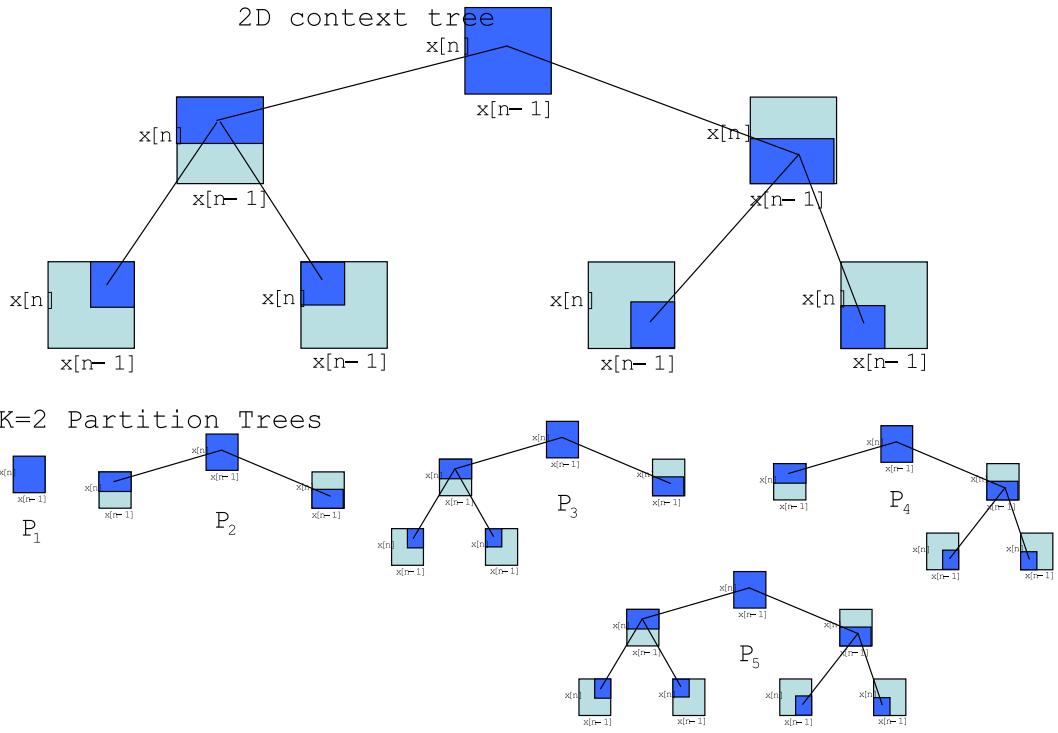
To fully exploit the potential modeling power of piecewise linear estimators, we need to be cautious while selecting the number of regions and boundaries of these regions. Piecewise linear models can approximate a wide class of nonlinear models, which satisfy certain regularity conditions. As the number of regions increases, we point out that the normal equations are calculated using a finite number of observations assigned to each region, which on a per-region basis, necessarily decreases with the number of regions. Hence, over fitting may occur if there are not enough observations assigned to a particular region. Clearly, how the boundaries are selected also affects the effectiveness of the piecewise linear models and how the observations are assigned to particular regions. These design considerations are especially severe in sequential processing, where the observations can only accumulate in time and available data is sparse in the initial phase of the adaptation.

One solution to avoid possible over fitting is to include the boundaries of regions as design parameters along with the corresponding AR parameters. Along these lines, recently, context tree based methods have been used to compactly represent a doubly exponential number of partitions of the space of past observations and provide an elegant way to design the partition of the past observations together with the AR parameters.

An example context tree is given in Figure 11.10 which partitions the space of past observations, where the depth of this context tree is 2, i.e.,  $K = 2$  [16]. For a context tree of depth  $K = 2$ , we have four leaves, where each leaf is assigned to a region of the space of past observations. Subsequently, each node on the context tree is assigned to a region that is the union of regions assigned to its children nodes. Using this context tree, we can define different partitions of the space of past observations as the union of regions assigned to nodes and the leaves. A context tree of depth can define a doubly exponential number of different partitions, e.g., in Figure 11.10, we present five different partitions of the space constructed by the context of depth  $K = 2$ . We represent a partition of the space of past observations as  $P_k = \{V_{k,1}, \dots, V_{k,K_k}\}$ , where  $\mathbf{U}_{j=1}^{K_k} V_{k,j}$  and each  $V_{k,j}$  assigned to a node on the context tree. Each of these partitions can be used to construct a piecewise linear model since they define complete partitions of the space of past observations. Some of the partitions are coarser and require fewer data samples to accurately calculate their linear estimators, while, as an example, the partition corresponding to the leaves has the finest partitioning and requires substantially more data for training. However, if there is sufficient data, then the finest partition naturally achieves the best modeling accuracy.

By using the context tree weighting method, we can construct a sequential piecewise AR model that hypothetically constructs all the piecewise linear estimators corresponding to all of these different partitions. This context tree based estimator achieves the performance of the best piecewise region with the corresponding sequential estimated parameters in each node as

$$\sum_{n=1}^N (x[n] - \vec{h}_{ctw}^T[n] \vec{x}[n])^2 \leq \min_k \left\{ \sum_{n=1}^N (x[n] - \vec{h}_{P_k}^T[t] \vec{x}[n])^2 + C(P_k) \right\}, \quad (11.97)$$

**FIGURE 11.10**

A context tree of depth  $K = 2$ , which has  $K = 4$  leaves and 3 inner nodes. This tree can represent five different partitions,  $P_1, \dots, P_5$ , where each partition defines a different piecewise linear estimator.

where  $C(P_k)$  is a certain constant that depends on the partition  $P_k$ ,  $\vec{h}_{P_k}[n]$  are the sequentially estimated AR parameters by the piecewise estimator assigned to the  $P_k$  partition. As an example, if the LMS algorithms are used in each node to train the corresponding linear models in different regions, then we have  $\vec{h}_{P_k}[n] = \vec{h}_{V_{k,j}}[n]$  if  $\vec{x}[n] \in V_{k,j}$ , and

$$\vec{h}_{V_{k,j}}[n+1] = \vec{h}_{V_{k,j}}[n] + \mu e_{V_{k,j}}[n] \vec{x}[n] I_{V_{k,j}}(\vec{x}[n]). \quad (11.98)$$

We can use different adaptation methods, e.g., the RLS algorithm instead of the LMS algorithm, in each node or region such that only the update equation should be changed accordingly. This implies that the context tree algorithm can uniformly achieve the performance of the best partition that can be represented on the context tree for any bounded deterministic signal, where the computational complexity of this algorithm is only in the order of the depth of the context tree.

### 1.11.3.10 Universality with respect to classes that vary in time

For non-stationary data models or time varying environments, the best choice of an adaptive estimator as well as the structure of this best estimator may change over time. As an example, different order AR models are needed to accurately model voiced and unvoiced time segments of a speech signal [3]. While RLS algorithm is known to converge faster than the LMS algorithm, since it minimizes a true least square error criterion, the LMS algorithm is known to better track certain non-stationary data [8]. In classical adaptive filtering, this kind of time variation is usually overcome by incorporating windowing or weighting by defining appropriate cost functions as done in the previous sections.

Recently, the universal model combination approaches gave rise to a host of model combination methods, where several different sequential estimation methods are combined in order to outperform the best in the combination [12]. In this sense, instead of using windowing or weighting, the best of choice of algorithms are intrinsically selected by the underlying data based on the performance so far. For example, by using a combination of adaptive algorithms that are fast tracking along with others that have superior steady state mean square error performance, one can strike a balance and achieve the tracking performance of the fast-converging adaptive filter, while maintaining the superior steady state performance of the slower, more precise adaptive filter counterpart.

Hence, rather than pushing the time variation into a known statistical model or adaptation, one can try to exploit the time varying nature of the best choice of algorithm for any given realization. As an example, given the batch data we can partition the observations in time into contiguous segments

$$\{\vec{x}[1], \dots, \vec{x}[n_1 - 1]\} \{\vec{x}[n_1], \dots, \vec{x}[n_2 - 1]\} \dots \{\vec{x}[n_k], \dots, \vec{x}[N]\} \quad (11.99)$$

and construct independent normal equations using

$$\hat{R}_k = \sum_{n=n_{k-1}+1}^{n_k} \vec{x}[n] \vec{x}[n]^T, \quad (11.100)$$

$$\hat{p}_k = \sum_{n=n_{k-1}+1}^{n_k} x[n] \vec{x}[n], \quad (11.101)$$

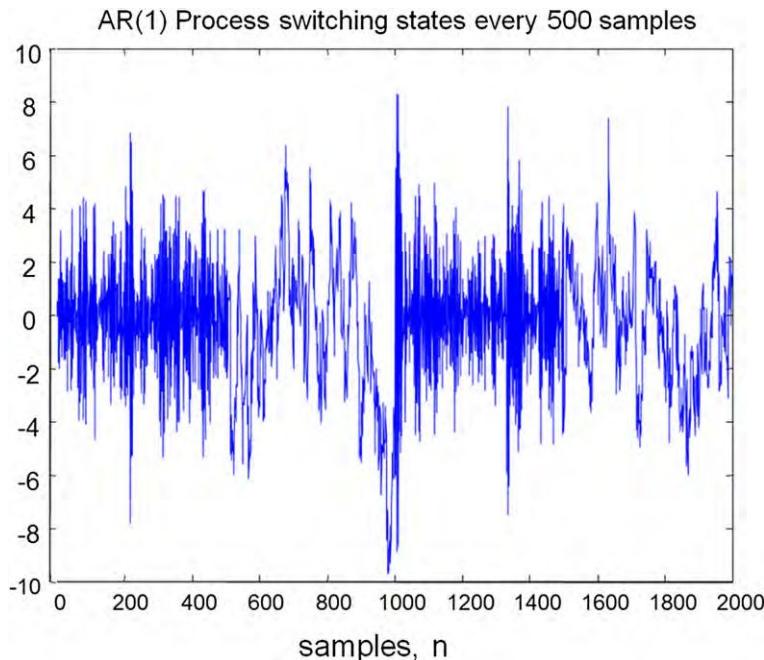
where  $n_0 = 0$ . This yields  $K$  independent AR models one for each contiguous segment as

$$\vec{h}_k = \hat{R}_k^{-1} \hat{p}_k, \quad (11.102)$$

$$\vec{h}[n] = \vec{h}_k, \quad n_{k-1} + 1 \leq n \leq n_k. \quad (11.103)$$

Since each normal equation only depends on the observations belonging to a specific segment, this adaptive processing can readily track variations in time. However, the best partition, i.e., how to optimally divide the observations in time can only be chosen after observing the whole data. Hence, we cannot construct a sequential version of such optimization and are forced to resort to windowing or weighting to emulate such piecewise partition in time.

However, inspired by the universal source coding methods, rather than trying to find the best partition (possible best switching points) or the best number of transitions, we can construct a sequential adaptive algorithm that simply achieves the performance of the best partition directly and simultaneously for all

**FIGURE 11.11**

A time-varying AR(1) model used as input to the predictor that permits switching among different linear predictors in time.

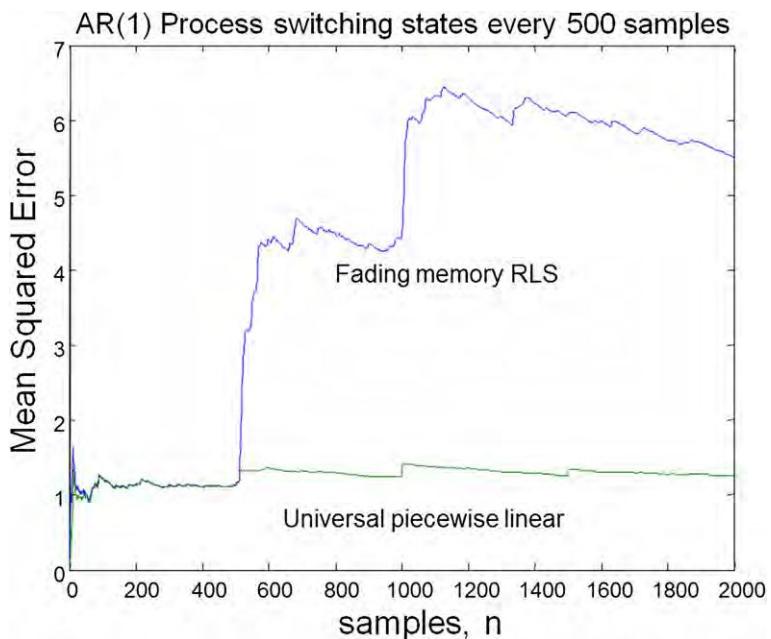
different partitions and for any possible observation sequence. This switching approach has been successfully applied by a number of researchers in a variety of problems, including tracking the best among a fixed class of algorithms as well as tracking the best from parametrically-continuous classes of algorithms [13]. By using transition diagrams, we can construct sequential algorithms that asymptotically achieve the performance of the best switching algorithm, i.e.,

$$\sum_{n=1}^N (x[n] - \vec{h}_{ctw}^T[n] \vec{x}[n])^2 \leq \min_{n_1, \dots, n_K, \vec{h}_1, \dots, \vec{h}_K} \left\{ \sum_{k=0}^K \sum_{n=n_k+1}^{n_k} (x[n] - \vec{h}_k^T \vec{x}[n])^2 + L(K) \right\} \quad (11.104)$$

tuned to the underlying sequence of observations without any stochastic assumptions, where  $L(K)$  is a constant that only depends on the number of contiguous segments.

The performance of a universal linear predictor that switches in time is demonstrated with the following example.

In this example, we are tracking a process similar to a speech sample such that the state of the process switches states abruptly every 500 samples, as shown in Figure 11.11. The state of the process at each segment is represented by an AR model driven by a Gaussian noise [8]. Figure 11.12 shows the mean

**FIGURE 11.12**

Mean square prediction error exhibited by (i) a fading memory RLS linear predictor and (ii) the universal piecewise constant (in time) linear predictor.

square error that results when we attempt to track the process using a fading memory RLS algorithm versus that obtained using a universal piecewise linear algorithm.

It can be seen that the universal piecewise linear algorithm recovers from the state transitions much more quickly, since it competes against the best switching pattern, while the fading memory RLS algorithm cannot recover in the short-data regime between transitions. Therefore, in such a situation, a universal piecewise linear model gives superior performance relative to the standard single filter model.

*Relevant Theory:* Signal Processing Theory, Machine Learning and Statistical Signal Processing

See this Volume, [Chapter 2](#) Continuous-Time Signals and Systems

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 4](#) Random Signals and Stochastic Processes

See this Volume, [Chapter 6](#) Digital Filter Structures and their Implementation

See this Volume, [Chapter 9](#) Discrete Multi-Scale Transforms in Signal Processing

See this Volume, Chapter 11 Parametric Estimation

See this Volume, [Chapter 12](#) Adaptive Filters

See this Volume, [Chapter 14](#) Learning Theory

See this Volume, [Chapter 17](#) Online Learning

See this Volume, [Chapter 25](#) A Tutorial on Model Selection

See [Vol. 3, Chapter 2](#) Model Order Selection

See [Vol. 3, Chapter 3](#) Non-Stationary Signal Analysis Time-Frequency Approach

See [Vol. 3, Chapter 8](#) Performance Analysis and Bounds

---

## References

- [1] A. Oppenheim, R.W. Schafer, J.R. Buck, Discrete-Time Signal Processing, second ed., Prentice Hall, 1999.
- [2] L. Rabiner, B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, 1975.
- [3] L.R. Rabiner, R.W. Schafer, Introduction to Digital Speech Processing, Prentice Hall, Inc., 1978.
- [4] W. Feller, An Introduction to Probability Theory and its Applications, Wiley, 1968.
- [5] B. Hajek, An Exploration of Random Processes for Engineers, 2011. Retrieved from An Exploration of Random Processes for Engineers: <<http://www.ifp.illinois.edu/~hajek/Papers/randomprocesses.html>>.
- [6] S. Kay, Fundamentals of Statistical Signal Processing, Estimation Theory, vol. 1, Prentice Hall, 1993.
- [7] J. Mahkoul, Linear prediction: a tutorial review, Proc. IEEE (1975) 561–580.
- [8] A.H. Sayed, Fundamentals of Adaptive Filtering, Wiley-IEEE Press, 2003.
- [9] M.L. Honig, D.G. Messerschmitt, Adaptive Filters: Structures, Algorithms and Applications, Springer, 1984.
- [10] J.G. Proakis, Digital Communications, McGraw-Hill, 1983.
- [11] H.V. Poor, An Introduction to Signal Detection and Estimation, Springer, New York, 1988.
- [12] A.C. Singer, M. Feder, Universal linear prediction by model order weighting, IEEE Trans. Signal Process. 47 (10) (1999) 2685–2699.
- [13] N. Cesa-Bianchi, G. Lugosi, Prediction Learning and Games, Cambridge University Press, 2006.
- [14] A.C. Singer, S.S. Kozat, M. Feder, Universal linear least squares prediction upper and lower bounds, IEEE Trans. Inform. Theory 48 (8) (2002) 2354–2362.
- [15] O.J. Michel, A.O. Hero, A.E. Badel, Tree-structured nonlinear signal modeling and prediction, IEEE Trans. Signal Process. 47 (11) (1999) 3027–3040.
- [16] S.S. Kozat, A.C. Singer, G.C. Zeitler, Universal piecewise linear prediction via context trees, IEEE Trans. Signal Process. 55 (7) (2007) 3730–3745.

# Adaptive Filters

# 12

Vítor H. Nascimento and Magno T. M. Silva

*Department of Electronic Systems Engineering, University of São Paulo, São Paulo, SP, Brazil*

---

### 1.12.1 Introduction

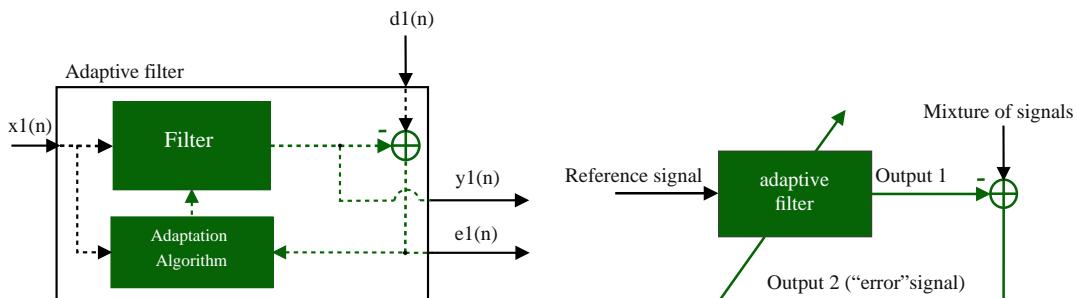
Adaptive filters are employed in situations in which the environment is constantly changing, so that a fixed system would not have adequate performance. As they are usually applied in real-time applications, they must be based on algorithms that require a small number of computations per input sample.

These algorithms can be understood in two complementary ways. The most straightforward way follows directly from their name: an adaptive filter uses information from the environment and from the very signal it is processing to change itself, so as to optimally perform its task. The information from the environment may be sensed in real time (in the form of a so-called desired signal), or may be provided *a priori*, in the form of previous knowledge of the statistical properties of the input signal (as in blind equalization).

On the other hand, we can think of an adaptive filter also as an algorithm to separate a mixture of two signals. The filter must have some information about the signals to be able to separate them; usually this is given in the form of a reference signal, related to only one of the two terms in the mixture. The filter has then two outputs, corresponding to each signal in the mixture (see Figure 12.1). As a byproduct of separating the signals, the reference signal is processed in useful ways. This way of viewing an adaptive filter is very useful, particularly when one is learning the subject.

In the following sections we give an introduction to adaptive filters, covering from basic principles to the most important recent developments. Since the adaptive literature is vast, and the topic is still an active area of research, we were not able to treat all interesting topics. Along the text, we reference important textbooks, classic papers, and the most promising recent literature. Among the many very good textbooks on the subject, four of the most popular are [1–4].

We start with an application example, in the next section. In it, we show how an adaptive filter is applied to cancel acoustic echo in hands-free telephony. Along the text we return frequently to this example, to illustrate new concepts and methods. The example is completed by an overview of how adaptive filters work, in which we use only deterministic arguments and concepts from basic linear systems theory, in Section 1.12.1.2. This overview shows many of the compromises that arise in adaptive filter design, without the more complicated math. Section 1.12.1.3 gives a small glimpse of the wide variety of applications of adaptive filters, describing a few other examples. Of course, in order to fully understand the behavior of adaptive filters, one must use estimation theory and stochastic processes,

**FIGURE 12.1**

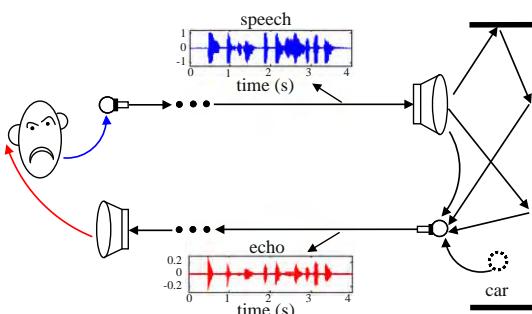
Inputs and outputs of an adaptive filter. Left: detailed diagram showing inputs, outputs and internal variable. Right: simplified diagram.

which is the subject of Sections 1.12.2 and 1.12.3. The remaining sections are devoted to extensions to the basic algorithms and recent results.

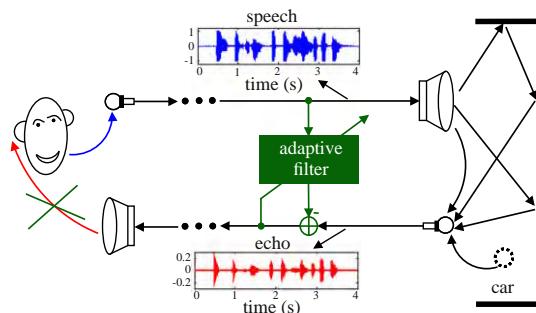
Adaptive filter theory brings together results from several fields: signal processing, matrix analysis, control and systems theory, stochastic processes, and optimization. We tried to provide short reviews for most of the necessary material in separate links (“boxes”), that the reader may follow if needed.

### 1.12.1.1 Motivation—acoustic echo cancellation

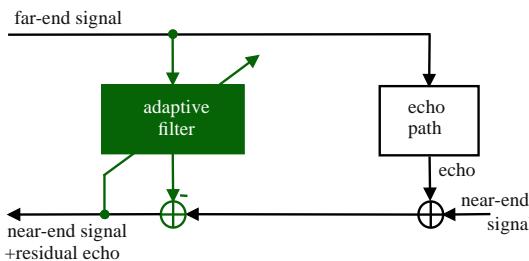
Suppose that you are talking with a person who uses a hands-free telephone inside a car. Let us assume that the telephone does not have any acoustic echo canceler. In this case, the sound of your voice will reach the loudspeaker, propagate inside the car, and suffer reflections every time it encounters the walls of the car. Part of the sound can be absorbed by the walls, but part of it is reflected, resulting in an echo signal. This signal is fed back to the microphone and you will hear your own voice with delay and attenuation, as shown in Figure 12.2.

**FIGURE 12.2**

Echo in hands-free telephony. Click [speech.wav](#) to listen to a speech signal and [echo.wav](#) to listen to the corresponding echo signal.

**FIGURE 12.3**

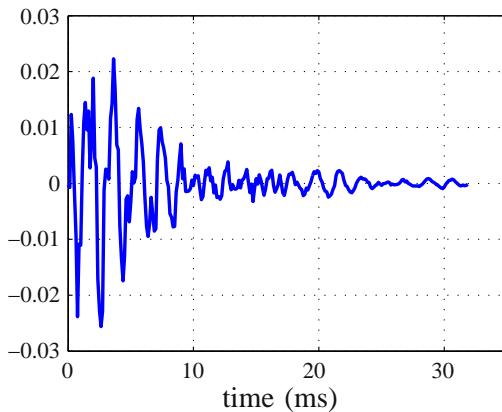
Acoustic echo cancellation using an adaptive filter.

**FIGURE 12.4**

General echo canceler configuration.

The echo signal depends on the acoustic characteristics of each car and also on the location of the loudspeaker and microphone. Furthermore, the acoustic characteristics can change during a phone call, since the driver may open or close a window, for example. In order to follow the variations of the environment, the acoustic echo must be canceled by an adaptive filter, which performs on-line updating of its parameters through an algorithm. Figure 12.3 shows the insertion of an adaptive filter in the system of Figure 12.2 to cancel the echo signal. In this configuration, the adaptive filter must identify the echo path and provide at its output an estimate of the echo signal. Thus, a synthesized version of the echo is subtracted from the signal picked up by the car microphone. In an ideal scenario, the resulting signal will be free of echo and will contain only the signal of interest. The general set-up of an echo canceler is shown in Figure 12.4. The *near-end* signal (near with respect to the echo canceler) may be simply noise when the person inside the car is in silence, as considered in Figures 12.2 and 12.3, or it can be a signal of interest, that is, the voice of the user in the car.

You most likely have already heard the output of an adaptive echo canceler. However, you probably did not pay attention to the initial period, when you can hear the adaptive filter learning the echo impulse response, if you listen carefully. We illustrate this through a simulation, using a measured impulse response of 256 samples (sampled at 8 kHz), and an echo signal produced from a recorded

**FIGURE 12.5**

Measured impulse response (256 samples).

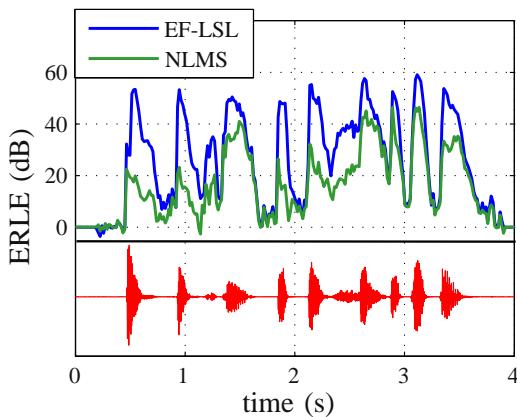
voice signal. The measured impulse response is shown in Figure 12.5. In the first example, the near-end signal is low-power noise (more specifically, white Gaussian noise with zero mean and variance  $\sigma^2 = 10^{-4}$ ). Listen to [e\\_nlms.wav](#), and pay attention to the beginning. You will notice that the voice (the echo) is fading away, while the adaptive filter is learning the echo impulse response. This file is the output of an echo canceler adapted using the *normalized least-mean squares algorithm (NLMS)*, which is described in Section 1.12.3.2.

Listen now to [e\\_lsl.wav](#) in this case the filter was adapted using the modified *error-feedback least-squares lattice algorithm (EF-LSL)* [5], which is a low-cost and stable version of the *recursive least-squares algorithm (RLS)* described in Section 1.12.3.3. Now you will notice that the echo fades away much faster. This fast convergence is characteristic of the EF-LSL algorithm, and is of course a very desirable property. On the other hand, the NLMS algorithm is simpler to implement and more robust against imperfections in the implementation. An adaptive filtering algorithm must satisfy several requirements, such as convergence rate, tracking capability, noise rejection, computational complexity, and robustness. These requirements are conflicting, so that there is no best algorithm that will outperform all the others in every situation. The many algorithms available in the literature are different compromises between these requirements [1–3].

Figure 12.6 shows 4 s of the echo signal, prior to cancellation (bottom) and a measure of the quality of echo cancellation, the so-called *echo return loss enhancement (ERLE)* (top). The ERLE is defined as the ratio

$$\text{ERLE} = \frac{\text{Power of original echo}}{\text{Power of residual echo}}.$$

The higher the ERLE, the better the performance of the canceler. The ERLE drops when the original echo is small (there is nothing to cancel!), and increases again when the echo is strong. The figure shows that EF-LSL performs better than NLMS (but at a higher computational cost).

**FIGURE 12.6**

Echo return loss enhancement for NLMS ( $\tilde{\mu} = 0.5$ ,  $\delta = 10^{-5}$ ,  $M = 256$ ) and EF-LSL ( $\lambda = 0.999$ ,  $\delta = 10^{-5}$ ) algorithms. Bottom trace: echo signal prior to cancellation. The meaning of the several parameters is explained in Sections 1.12.3.1, 1.12.3.2, and 1.12.3.3.

Assuming now that the near-end signal is another recorded voice signal (Click [DT\\_echo.wav](#) to listen to this signal), we applied again the EF-LSL algorithm to cancel the echo. This case simulates a person talking inside the car. Since the speech of this person is not correlated to the far-end signal, the output of the adaptive filter converges again to an estimate of the echo signal. Thus, a synthesized version of the echo is subtracted from the signal picked up by the car microphone and the resulting signal converges to the speech of the person inside the car. This result can be verified by listening to the file [e\\_1s1DT.wav](#).

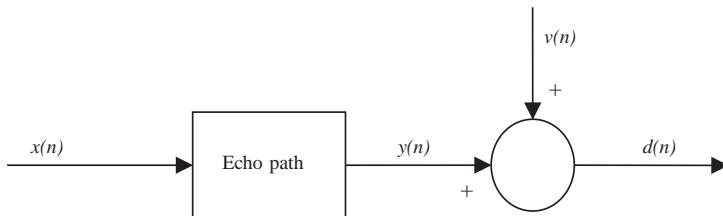
This is a typical example of use of an adaptive filter. The environment (the echo impulse response) changes constantly, requiring a constant re-tuning of the canceling filter. Although the echo itself is not known, we have access to a reference signal (the clean speech of the far-end user). This is the signal that is processed to obtain an estimate of the echo. Finally, the signal captured by the near-end microphone (the so-called *desired* signal) is a mixture of two signals: the echo, plus the near-end speech (or ambient noise, when the near-end user is silent). The task of the adaptive filter can be seen as to separate these two signals.

We should mention that the name desired signal is somewhat misleading, although widespread in the literature. Usually we are interested in separating the “desired” signal into two parts, one of which is of no interest at all!

We proceed now to show how an algorithm can perform this task, modeling all signals as periodic to simplify the arguments.

### 1.12.1.2 A quick tour of adaptive filtering

Before embarking in a detailed explanation of adaptive filtering and its theory, it is a good idea to present the main ideas in a simple setting, to help the reader create intuition. In this section, we introduce the least-mean squares (LMS) algorithm and some of its properties using only deterministic arguments and basic tools from linear systems theory.

**FIGURE 12.7**

A model for the echo.

#### 1.12.1.2.1 Posing the problem

Consider again the echo-cancellation problem, seen in the previous section (see Figure 12.7). Let  $y(n)$  represent the echo,  $v(n)$  represent the voice of the near-end user, and  $x(n)$  the voice of the far-end user (the reference).  $d(n)$  represents the signal that would be received by the far-end user without an echo canceler (the mixture, or desired signal).

The echo path represents the changes that the far-end signal suffers when it goes through the digital-to-analog (D/A) converter, the loudspeaker, the air path between the loudspeaker and the microphone (including all reflections), the microphone itself and its amplifier, and the analog-to-digital (A/D) converter. The microphone signal  $d(n)$  will always be a mixture of the echo  $y(n)$  and an unrelated signal  $v(n)$ , which is composed of the near-end speech plus noise. Our goal is to remove  $y(n)$  from  $d(n)$ . The challenge is that we have no means to measure  $y(n)$  directly, and the echo path is constantly changing.

#### 1.12.1.2.2 Measuring how far we are from the solution

How is the problem solved, then? Since we cannot measure  $y(n)$ , the solution is to estimate it from the measurable signals  $x(n)$  and  $d(n)$ . This is done as follows: imagine for now that all signals are periodic, so they can be decomposed as Fourier series

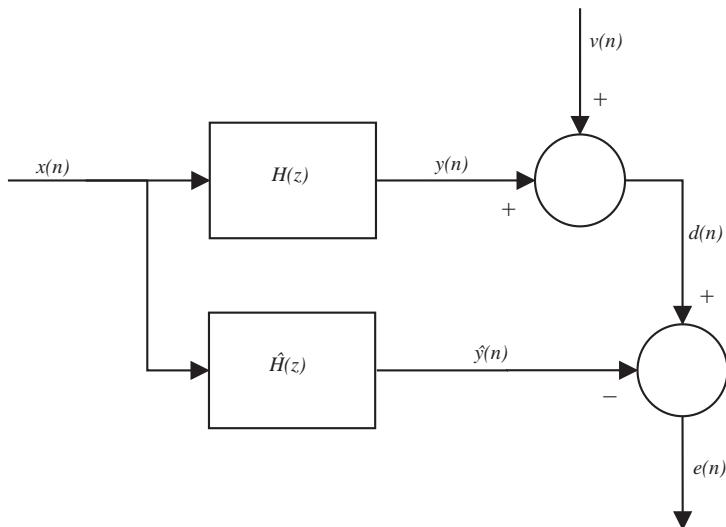
$$x(n) = \sum_{k=1}^{K_0} A_k \cos(k\omega_0 n + \varphi_k), \quad v(n) = \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell), \quad (12.1)$$

for certain amplitudes  $A_k$  and  $B_\ell$ , phases  $\varphi_k$  and  $\theta_\ell$ , and frequencies  $\omega_0$  and  $\omega_1$ . The highest frequencies appearing in  $x(n)$  and  $v(n)$  must satisfy  $K_0\omega_0 < \pi$ ,  $K_1\omega_1 < \pi$ , since we are dealing with sampled signals (Nyquist criterion). We assume for now that the echo path is fixed (not changing), and can be modeled by an unknown linear transfer function  $H(z)$ . In this case,  $y(n)$  would also be a periodic sequence with fundamental frequency  $\omega_0$ ,

$$y(n) = \sum_{k=1}^{K_0} C_k \cos(k\omega_0 n + \psi_k),$$

with  $C_k e^{j\psi_k} = H(e^{jk\omega_0}) A_k e^{j\varphi_k}$ . The signal picked by the microphone is then

$$d(n) = y(n) + v(n) = \sum_{k=1}^{K_0} C_k \cos(k\omega_0 n + \psi_k) + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell).$$

**FIGURE 12.8**

Echo canceler.

Since we know  $x(n)$ , if we had a good approximation  $\hat{H}(z)$  to  $H(z)$ , we could easily obtain an approximation  $\hat{y}(n)$  to  $y(n)$  and subtract it from  $d(n)$ , as in Figure 12.8.

How could we find  $\hat{H}(z)$  in real time? Recall that only  $d(n)$ ,  $x(n)$ ,  $\hat{y}(n)$ , and  $e(n)$  can be observed. For example, how could we know that we have the exact filter, that is, that  $\hat{H}(z) = H(z)$ , by looking only at these signals? To answer this question, take a closer look at  $e(n)$ . Let the output of  $\hat{H}(z)$  be

$$\begin{aligned}
 \hat{y}(n) &= \sum_{k=1}^{k_0} \hat{C}_k \cos(k\omega_0 n + \hat{\psi}_k), \quad \text{and thus} \\
 e(n) &= d(n) - \hat{y}(n) = y(n) - \hat{y}(n) + v(n) \\
 &= \sum_{k=1}^{K_0} [C_k \cos(k\omega_0 n + \psi_k) - \hat{C}_k \cos(k\omega_0 n + \hat{\psi}_k)] \\
 &\quad + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell) \\
 &= \sum_{k=1}^{K_0} \tilde{C}_k \cos(k\omega_0 n + \tilde{\psi}_k) + \sum_{\ell=1}^{K_1} B_\ell \cos(\ell\omega_1 n + \theta_\ell),
 \end{aligned} \tag{12.2}$$

where  $\tilde{C}_k e^{j\tilde{\psi}_k} = C_k e^{j\psi_k} - \hat{C}_k e^{j\hat{\psi}_k}$ ,  $k = 1, \dots, K_0$ .

Assuming that the cosines in  $x(n)$  and  $v(n)$  have no frequencies in common, the simplest approach is to measure the average power of  $e(n)$ . Indeed, if all frequencies  $k\omega_0$  and  $\ell\omega_1$  appearing in  $e(n)$  are different from one another, then the average power of  $e(n)$  is

$$P = \sum_{k=1}^{K_0} \frac{\tilde{C}_k^2}{2} + \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2}. \quad (12.3)$$

If  $\hat{y}(n) = y(n)$ , then  $P$  is at its minimum,

$$P_o = \min_{\hat{H}(z)} P = \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2},$$

which is the average power of  $v(n)$ . In this case,  $e(n)$  is at its optimum:  $e(n) = e_o(n) = v(n)$ , and the echo is completely canceled.

It is important to see that this works because the two signals we want to separate,  $y(n)$  and  $v(n)$ , are *uncorrelated*, that is, they are such that

$$\lim_{N \rightarrow \infty} \frac{1}{N+1} \sum_{-N/2}^{N/2} y(n)v(n) = \lim_{N \rightarrow \infty} \frac{1}{N+1} \sum_{-N/2}^{N/2} y(n)e_o(n) = 0. \quad (12.4)$$

This property is known as the *orthogonality condition*. A form of orthogonality condition will appear in general whenever one tries to minimize a quadratic function, as is the case here. This is discussed in more detail in Section 1.12.2.

The average power  $P$  could then be used as a measure of how good is our approximation  $\hat{y}(n)$ . The important point is that it is very easy to find a good approximation to  $P$ . It suffices to low-pass filter  $e^2(n)$ , for example,

$$\hat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^2(n-k) \quad (12.5)$$

is a good approximation if the window length  $N$  is large enough.

The adaptive filtering problem then becomes an optimization problem, with the particularity that the cost function that is being minimized is not known exactly, but only through an approximation, as in (12.5). In the following sections, we discuss in detail the consequences of this fact.

It is also important to remark that the average error power is not the only possible choice for cost function. Although it is the most popular choice for a number of reasons, in some applications other choices are more adequate. Even posing the adaptive filtering problem as an optimization problem is not the only alternative, as shown by recent methods based on projections on convex sets [6].

### 1.12.1.2.3 Choosing a structure for the filter

Our approximation  $\hat{y}(n)$  will be built by minimizing the estimated average error power (12.5) as a function of the echo path model  $\hat{H}(z)$ , that is, our estimated echo path will be the solution of

$$\hat{H}_o(z) = \arg \min_{\hat{H}(z)} \hat{P}(n). \quad (12.6)$$

Keep in mind that we are looking for a real-time way of solving (12.6), since  $\widehat{P}(n)$  can only be obtained by measuring  $e(n)$  for a certain amount of time. We must then be able to implement the current approximation  $\widehat{H}(z)$ , so that  $\hat{y}(n)$  and  $e(n)$  may be computed.

This will impose practical restrictions on the kind of function that  $\widehat{H}(z)$  may be: since memory and processing power are limited, we must choose beforehand a structure for  $\widehat{H}(z)$ . Memory and processing power will impose a constraint on the maximum order the filter may have. In addition, in order to write the code for the filter, we must decide if it will depend on past outputs (IIR filters) or only on past inputs (FIR filters). These choices must be made based on some knowledge of the kind of echo path that our system is likely to encounter.

To explain how these choices can be made, let us simplify the problem a little more and restrict the far-end signal  $x(n)$  to a simple sinusoid (i.e., assume that  $K_0 = 1$ ). In this case,

$$y(n) = C_1 \cos(\omega_0 n + \varphi_1),$$

with  $C_1 e^{j\psi_1} = H(e^{j\omega_0}) A_1 e^{j\varphi_1}$ . Therefore,  $\widehat{H}(z)$  must satisfy

$$\left[ \widehat{H}(e^{j\omega}) = H(e^{j\omega}) \right]_{\omega=\omega_0} \quad (12.7)$$

only for  $\omega = \omega_0$ —the value of  $\widehat{H}(e^{j\omega})$  for other frequencies is irrelevant, since the input  $x(n)$  has only one frequency. Expanding (12.7), we obtain

$$\operatorname{Re}\{\widehat{H}(e^{j\omega_0})\} = \operatorname{Re}\{H(e^{j\omega_0})\}, \quad \operatorname{Im}\{\widehat{H}(e^{j\omega_0})\} = \operatorname{Im}\{H(e^{j\omega_0})\}. \quad (12.8)$$

The optimum  $\widehat{H}(z)$  is defined through two conditions. Therefore an FIR filter with just two coefficients would be able to satisfy both conditions for any value of  $H(e^{j\omega_0})$ , so we could define

$$\widehat{H}(z) = w_0 + w_1 z^{-1},$$

and the values of  $w_0$  and  $w_1$  would be chosen to solve (12.6). Note that the argument generalizes for  $K_0 > 1$ : in general, if  $x(n)$  has  $K_0$  harmonics, we could choose  $\widehat{H}(z)$  as an FIR filter with length  $2K_0$ , two coefficients per input harmonic.

This choice is usually not so simple: in practice, we would not know  $K_0$ , and in the presence of noise (as we shall see later), a filter with fewer coefficients might give better performance, even though it would not completely cancel the echo. The structure of the filter also affects the dynamic behavior of the adaptive filter, so simply looking at equations such as (12.7) does not tell the whole story. Choosing the structure for  $\widehat{H}(z)$  is one of the most difficult steps in adaptive filter design. In general, the designer must test different options, initially perhaps using recorded signals, but ultimately building a prototype and performing some tests.

#### 1.12.1.2.4 Searching for the solution

Returning to our problem, assume then that we have decided that an FIR filter with length  $M$  is adequate to model the echo path, that is,

$$\widehat{H}(z) = \sum_{k=0}^{M-1} w_k z^{-k}.$$

Our minimization problem (12.6) now reduces to finding the coefficients  $w_0, \dots, w_{M-1}$  that solve

$$\min_{w_0, \dots, w_{M-1}} \widehat{P}(n). \quad (12.9)$$

Since  $\widehat{P}(n)$  must be computed from measurements, we must use an iterative method to solve (12.9). Many algorithms could be used, as long as they depend only on measurable quantities ( $x(n)$ ,  $d(n)$ ,  $\hat{y}(n)$ , and  $e(n)$ ). We will use the steepest descent algorithm (also known as gradient algorithm) as an example now. Later we show other methods that also could be used. If you are not familiar with the gradient algorithm, see Box 1 for an introduction.

The cost function in (12.9) is

$$\widehat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^2(n-k) = \frac{1}{N} \sum_{k=0}^{N-1} [d(n-k) - \hat{y}(n-k)]^2.$$

We need to rewrite this equation so that the steepest descent algorithm can be applied. Recall also that now the filter coefficients will change, so define the vectors

$$\mathbf{w}(n) = [w_0(n) \quad w_1(n) \quad \cdots \quad w_{M-1}(n)]^T,$$

and

$$\mathbf{x}(n) = [x(n) \quad x(n-1) \quad \cdots \quad x(n-M+1)]^T,$$

where  $(\cdot)^T$  denotes transposition. At each instant, we have  $\hat{y}(n) = \mathbf{w}^T(n)\mathbf{x}(n)$ , and our cost function becomes

$$\widehat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} [d(n-k) - \mathbf{w}^T(n-k)\mathbf{x}(n-k)]^2, \quad (12.10)$$

which depends on  $\mathbf{w}(n), \dots, \mathbf{w}(n-N+1)$ . In order to apply the steepest descent algorithm, let us keep the coefficient vector  $\mathbf{w}(n)$  constant during the evaluation of  $\widehat{P}$ , as follows. Starting from an initial condition  $\mathbf{w}(0)$ , compute for  $n = 0, N, 2N, \dots$  (The notation is explained in Boxes 2 and 6.)

$$\mathbf{w}(n+N) = \mathbf{w}(n) - \alpha \left. \frac{\partial \widehat{P}(n+N-1)}{\partial \mathbf{w}^T} \right|_{\mathbf{w}=\mathbf{w}(n)}, \quad (12.11)$$

where  $\alpha$  is a positive constant, and  $\mathbf{w}(n+N-1) = \mathbf{w}(n+N-2) = \cdots = \mathbf{w}(n)$ . Our cost function now depends on only one  $\mathbf{w}(n)$  (compare with (12.10)):

$$\begin{aligned} \widehat{P}(n+N-1) &= \frac{1}{N} \sum_{k=0}^{N-1} [d(n+N-1-k) - \mathbf{w}^T(n)\mathbf{x}(n+N-1-k)]^2 \\ &= \frac{1}{N} \sum_{k=0}^{N-1} [d(n+k) - \mathbf{w}^T(n)\mathbf{x}(n+k)]^2, \quad n = 0, N, 2N, \dots \end{aligned} \quad (12.12)$$

We now need to evaluate the gradient of  $\widehat{P}$ . Expanding the expression above, we obtain

$$\widehat{P}(n+N-1) = \frac{1}{N} \sum_{k=0}^{N-1} \left[ d(n+k) - \sum_{\ell=0}^{M-1} w_\ell(n)x(n+k-\ell) \right]^2,$$

so

$$\begin{aligned}\frac{\partial \widehat{P}(n - N + 1)}{\partial w_m(n)} &= -\frac{2}{N} \sum_{k=0}^{N-1} [d(n+k) - \mathbf{w}^T(n)\mathbf{x}(n+k)] x(n+k-m) \\ &= -\frac{2}{N} \sum_{k=0}^{N-1} e(n+k)x(n+k-m),\end{aligned}$$

and

$$\frac{\partial \widehat{P}(n - N + 1)}{\partial \mathbf{w}^T} = -\frac{2}{N} \sum_{k=0}^{N-1} e(n+k)\mathbf{x}(n+k), n = 0, N, 2N, \dots \quad (12.13)$$

As we needed, this gradient depends only on measurable quantities,  $e(n+k)$  and  $\mathbf{x}(n+k)$ , for  $k = 0, \dots, N-1$ . Our algorithm for updating the filter coefficients then becomes

$$\mathbf{w}(n+N) = \mathbf{w}(n) + \mu \frac{1}{N} \sum_{k=0}^{N-1} e(n+k)\mathbf{x}(n+k), n = 0, N, 2N, \dots, \quad (12.14)$$

where we introduced the overall step-size  $\mu = 2\alpha$ .

We still must choose  $\mu$  and  $N$ . The choice of  $\mu$  is more complicated and will be treated in Section 1.12.1.2.5. The value usually employed for  $N$  may be a surprise: in almost all cases, one uses  $N = 1$ , resulting in the so-called *least-mean squares* (LMS) algorithm

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n), \quad (12.15)$$

proposed initially by Widrow and Hoff in 1960 [7] (Widrow [8] describes the history of the creation of the LMS algorithm). The question is, how can this work, if no average is being used for estimating the error power? An intuitive answer is not complicated: assume that  $\mu$  is a very small number so that  $\mu = \mu_0/N$  for a large  $N$ . In this case, we can approximate  $e(n+k)$  from (12.15) as follows.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu_0}{N} e(n)\mathbf{x}(n) \approx \mathbf{w}(n), \quad \text{if } N \text{ is large.}$$

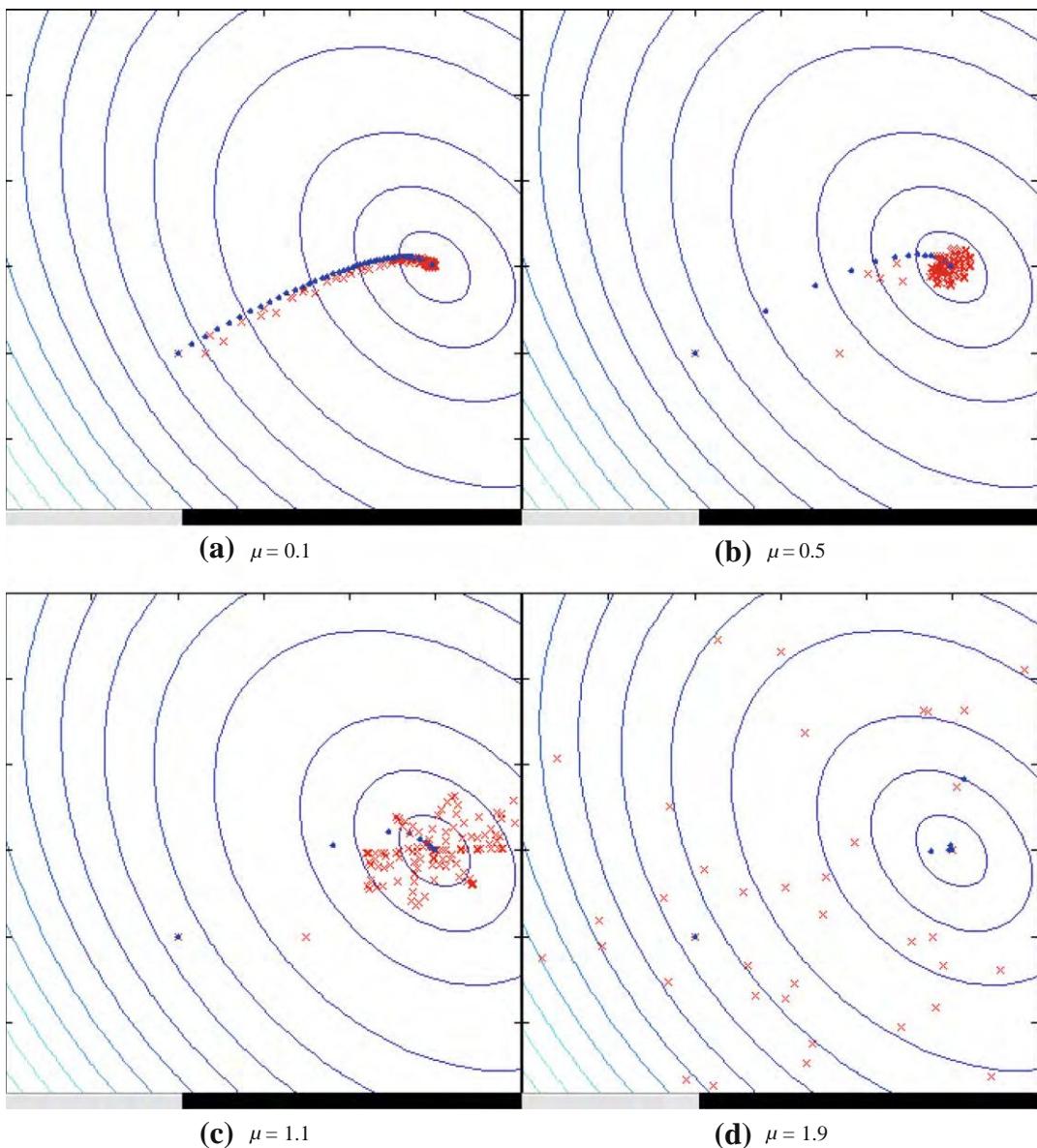
Since  $\mathbf{w}(n+1) \approx \mathbf{w}(n)$ , we have  $e(n+1) = d(n+1) - \mathbf{w}^T(n+1)\mathbf{x}(n+1) \approx d(n+1) - \mathbf{w}^T(n)\mathbf{x}(n+1)$ . Therefore, we could approximate

$$e(n+k) \approx d(n+k) - \mathbf{w}^T(n)\mathbf{x}(n+k), \quad k = 0, \dots, N-1,$$

so  $N$  steps of the LMS recursion (12.15) would result

$$\mathbf{w}(n+N) \approx \mathbf{w}(n) + \frac{\mu_0}{N} \sum_{k=0}^{N-1} [d(n+k) - \mathbf{w}^T(n)\mathbf{x}(n+k)] \mathbf{x}(n+k),$$

just what would be obtained from (12.14). The conclusion is that, although there is no explicit average being taken in the LMS algorithm (12.15), the algorithm in fact computes an implicit, approximate average if the step-size is small. This is exactly what happens, as can be seen in the animations in

**FIGURE 12.9**

LMS algorithm for echo cancelation with sinusoidal input. [Click](#) LMS video files to see animations on your browser.

Figure 12.9. These simulations were prepared with

$$x(n) = \cos(0.4\pi n + \varphi_1), \quad v(n) = 0.2 \cos(0.2\pi n + \theta_1),$$

where  $\varphi_1$  and  $\theta_1$  were chosen randomly in the interval  $[0, 2\pi]$ . Several step-sizes were used. The estimates computed with the LMS algorithm are marked by the crosses (red in the web version). The initial condition is at the left, and the theoretical optimum is at the right end of each figure. In the simulations, the true echo path was modeled by the fixed filter

$$H(z) = 1.5 + 0.5z^{-1}.$$

For comparison, we also plotted the estimates that would be obtained by the gradient algorithm using the exact value of the error power at each instant. These estimates are marked with small dots (blue in the web version).

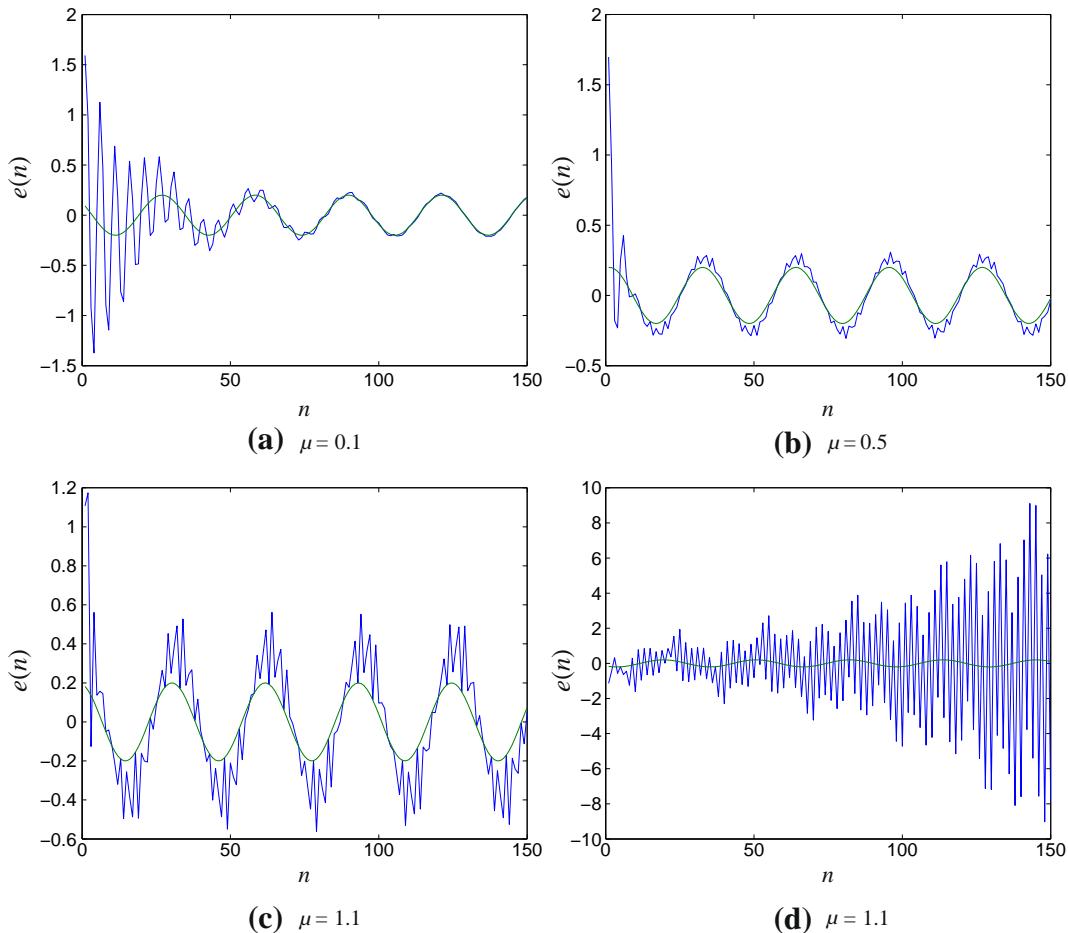
Note that when a small step-size is used, such as  $\mu = 0.1$  in Figure 12.9a, the LMS filter stays close to the dots obtained assuming exact knowledge of the error power. However, as the step-size increases, the LMS estimates move farther away from the dots. Although convergence is faster, the LMS estimates do not reach the optimum and stay there: instead, they hover around the optimum. For larger step-sizes, the LMS estimates can go quite far from the optimum (Figure 12.9b and c). Finally, if the step-size is too large, the algorithm will diverge, that is, the filter coefficients grow without bounds (Figure 12.9d).

#### 1.12.1.2.5 Tradeoff between speed and precision

The animations in Figure 12.9 illustrate an important problem in adaptive filters: even if you could magically choose as initial conditions the exact optimum solutions, the adaptive filter coefficients would not stay there! This happens because the filter does not know the exact value of the cost function it is trying to minimize (in this case, the error power): since  $\hat{P}(n)$  is an approximation, noise (and the near-end signal in our echo cancellation example) would make the estimated gradient non-zero, and the filter would wander away from the optimum solution. This effect keeps the minimum error power obtainable using the adaptive filter always a little higher than the optimum value. The difference between the (theoretical, unattainable without perfect information) optimum and the actual error power is known as *excess mean-square error* (EMSE), and the ratio between the EMSE and the optimum error is known as *misadjustment* (see Eqs. (12.171) and (12.174)).

For small step-sizes the misadjustment is small, since the LMS estimates stay close to the estimates that would be obtained with exact knowledge of the error power. However, a small step-size also means slow convergence. This trade-off exists in all adaptive filter algorithms, and has been an intense topic of research: many algorithms have been proposed to allow faster convergence without increasing the misadjustment. We will see some of them in the next sections.

The misadjustment is central to evaluate the performance of an adaptive filter. In fact, if we want to eliminate the echo from the near-end signal, we would like that after convergence,  $e(n) \approx v(n)$ . This is indeed what happens when the step-size is small (see Figure 12.10a). However, when the step-size is increased, although the algorithm converges more quickly, the performance after convergence is not very good, because of the wandering of the filter weights around the optimum (Figure 12.10b–d—in the last case, for  $\mu = 1.9$ , the algorithm is diverging.).

**FIGURE 12.10**

Compromise between convergence rate and misadjustment in LMS. Plots of  $e(n)x_n$  for several values of  $\mu$ .

We will stop this example here for the moment, and move forward to a description of adaptive filters using probability theory and stochastic processes. We will return to it during discussions of stability and model order selection.

The important message from this section is that an adaptive filter is able to separate two signals in a mixture by minimizing a well-chosen cost function. The minimization must be made iteratively, since the true value of the cost function is not known: only an approximation to it may be computed at each step. When the cost function is quadratic, as in the example shown in this section, the components of the mixture are separated such that they are in some sense orthogonal to each other.

---

**Box 1: Steepest descent algorithm**


---

Given a cost function

$$J(\mathbf{w}) = J(w_0, \dots, w_{M-1}),$$

with  $\mathbf{w} = [w_0, \dots, w_{M-1}]^T$  ( $(\cdot)^T$  denotes transposition), we want to find the minimum of  $J$ , that is, we want to find  $\mathbf{w}_o$  such that

$$\mathbf{w}_o = \arg \min_{\mathbf{w}} J(\mathbf{w}).$$

The solution can be found using the gradient of  $J(\mathbf{w})$ , which is defined by

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}^T} = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_{M-1}} \end{bmatrix}.$$

See Box 2 for a brief explanation about gradients and Hessians, that is, derivatives of functions of several variables.

The notation  $\partial J / \partial \mathbf{w}^T$  is most convenient when we deal with complex variables, as in Box 6. We use it for real variables for consistency.

Since the gradient always points to the direction in which  $J(\mathbf{w})$  increases most quickly, the steepest descent algorithm searches iteratively for the minimum of  $J(\mathbf{w})$  taking at each iteration a small step in the opposite direction, i.e., towards  $-\nabla_{\mathbf{w}} J(\mathbf{w})$ :

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{2} \nabla_{\mathbf{w}} J(\mathbf{w}(n)), \quad (12.16)$$

where  $\mu$  is a step-size, i.e., a constant controlling the speed of the algorithm. As we shall see, this constant should not be too small (or the algorithm will converge too slowly) neither too large (or the recursion will diverge).

As an example, consider the quadratic cost function with  $M = 2$

$$J(\mathbf{w}) = 1 - 2\mathbf{w}^T \mathbf{r} + \mathbf{w}^T \mathbf{R} \mathbf{w}, \quad (12.17)$$

with

$$\mathbf{r} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \frac{4}{3} & -\frac{2}{3} \\ -\frac{2}{3} & 4 \end{bmatrix}. \quad (12.18)$$

The gradient of  $J(\mathbf{w})$  is

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}^T} = -2\mathbf{r} + 2\mathbf{R}\mathbf{w}. \quad (12.19)$$

Of course, for this example we can find the optimum  $\mathbf{w}_o$  by equating the gradient to zero:

$$\nabla_{\mathbf{w}} J(\mathbf{w}_o) = \mathbf{0} \implies \mathbf{R}\mathbf{w}_o = \mathbf{r} \implies \mathbf{w}_o = \mathbf{R}^{-1}\mathbf{r} = \begin{bmatrix} 0.9877 \\ 0.4902 \end{bmatrix}. \quad (12.20)$$

Note that in this case the *Hessian* of  $J(\mathbf{w})$  (the matrix of second derivatives, see Box 2) is simply  $\nabla_{\mathbf{w}}^2 J = 2\mathbf{R}$ . As  $\mathbf{R}$  is symmetric with positive eigenvalues ( $\lambda_1 = 2$  and  $\lambda_2 = \frac{2}{3}$ ), it is positive-definite, and we can conclude that  $\mathbf{w}_o$  is indeed a minimum of  $J(\mathbf{w})$  (See Boxes 2 and 3. Box 3 lists several useful properties of matrices.)

Even though in this case we could compute the optimum solution through (12.20), we will apply the gradient algorithm with the intention of understanding how it works and which are its main limitations. From (12.16) and (12.17), we obtain the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{r} - \mathbf{R}\mathbf{w}(n)). \quad (12.21)$$

The user must choose an initial condition  $\mathbf{w}(0)$ .

In Figure 12.11 we plot the evolution of the approximations computed through (12.21) against the level sets (i.e., curves of constant value) of the cost function (12.17). Different choices of step-size  $\mu$  are shown. In Figure 12.11a, a small step-size is used. The algorithm converges to the correct solution, but slowly. In Figure 12.11b, we used a larger step-size—convergence is now faster, but the algorithm still needs several iterations to reach the solution. If we try to increase the step-size even further, convergence at first becomes slower again, with the algorithm oscillating towards the solution (Figure 12.11c). For even larger step-sizes, such as that in Figure 12.11d, the algorithm diverges: the filter coefficients get farther and farther away from the solution.

It is important to find the maximum step-size for which the algorithm (12.21) remains stable. For this, we need concepts from linear systems and linear algebra, in particular eigenvalues, their relation to stability of linear systems, and the fact that symmetric matrices always have real eigenvalues (see Box 3).

The range of allowed step-sizes is found as follows. Rewrite (12.21) as

$$\mathbf{w}(n+1) = [\mathbf{I} - \mu\mathbf{R}] \mathbf{w}(n) + \mu\mathbf{r},$$

which is a linear recursion in state-space form ( $\mathbf{I}$  is the identity matrix).

Linear systems theory [9] tells us that this recursion converges as long as the largest eigenvalue of  $\mathbf{A} = \mathbf{I} - \mu\mathbf{R}$  has absolute value less than one. The eigenvalues of  $\mathbf{A}$  are the roots of

$$\det(\beta\mathbf{I} - (\mathbf{I} - \mu\mathbf{R})) = \det((\beta - 1)\mathbf{I} + \mu\mathbf{R}) = -\det((1 - \beta)\mathbf{I} - \mu\mathbf{R}) = 0.$$

Let  $1 - \beta = v$ . Then  $\beta$  is an eigenvalue of  $\mathbf{A}$  if and only if  $v = 1 - \beta$  is an eigenvalue of  $\mu\mathbf{R}$ . Therefore, if we denote by  $\lambda_i$  the eigenvalues of  $\mathbf{R}$ , the stability condition is

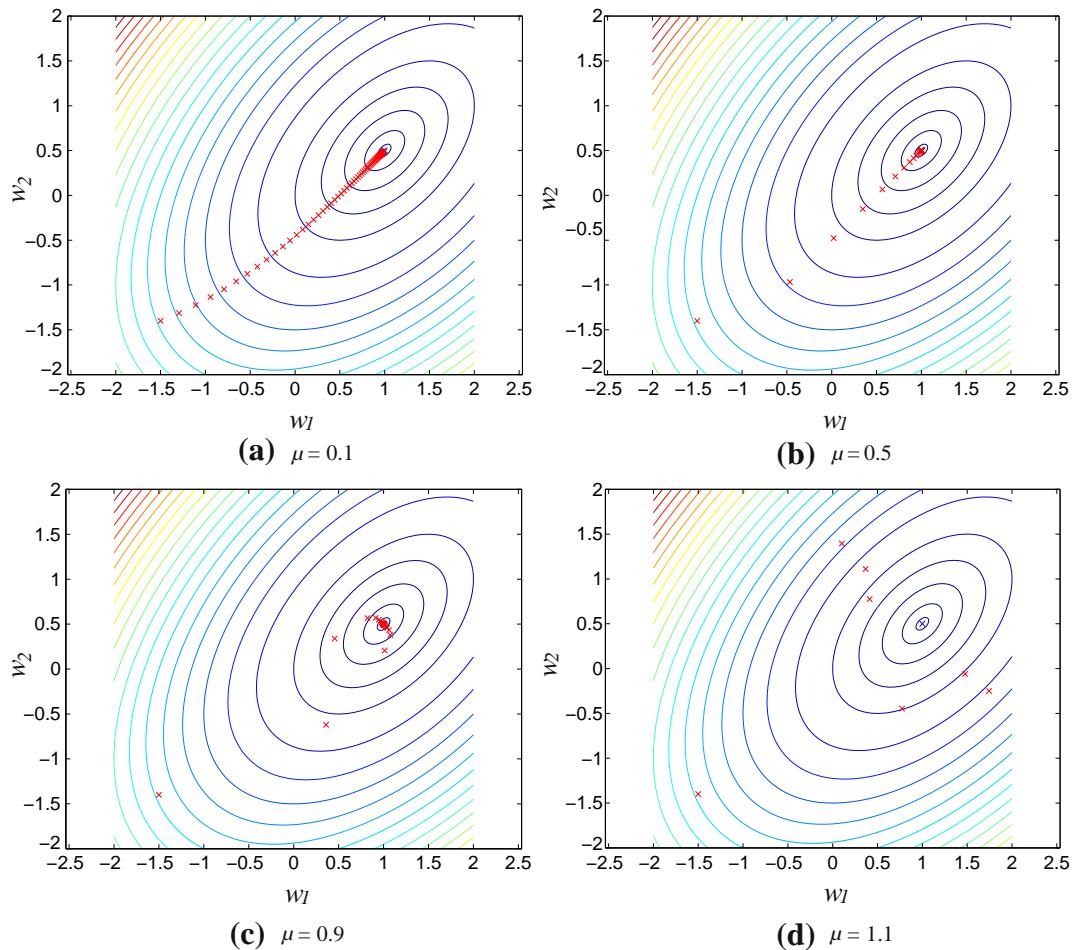
$$-1 < 1 - \mu\lambda_i < 1 \quad \forall i \Leftrightarrow 0 < \mu < \frac{2}{\lambda_i} \quad \forall i \Leftrightarrow 0 < \mu < \frac{2}{\max\{\lambda_i\}}.$$

The stability condition for our example is thus  $0 < \mu < 1$ , in agreement with what we saw in Figure 12.11.

The gradient algorithm leads to relatively simple adaptive filtering algorithms; however, it has an important drawback. As you can see in Figure 12.11a, the gradient does not point directly to the direction of the optimum. This effect is heightened when the level sets of the cost function are very elongated, as in Figure 12.12. This case corresponds to a quadratic function in which  $\mathbf{R}$  has one eigenvalue much smaller than the other. In this example, we replaced the matrix  $\mathbf{R}$  in (12.18) by

$$\mathbf{R}' = \frac{1}{9} \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix}.$$

This matrix has eigenvalues  $\lambda'_1 = 5$  and  $\lambda'_2 = \frac{5}{9}$ .

**FIGURE 12.11**

Performance of the steepest descent algorithm for different step-sizes. The crosses ( $x$ ) represent the successive approximations  $\mathbf{w}(n)$ , plotted against the level curves of (12.17). The initial condition is the point in the lower left.

Let us see why this is so. Recall from Box 3 that for every symmetric matrix there exists an orthogonal matrix  $\mathbf{U}$  (that is,  $\mathbf{U}^{-1} = \mathbf{U}^T$ ) such that

$$\mathbf{U}^T \mathbf{R} \mathbf{U} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \stackrel{\Delta}{=} \mathbf{\Lambda}, \quad (12.22)$$

where we defined the diagonal eigenvalue matrix  $\mathbf{\Lambda}$ .

Let us apply a change of coordinates to (12.19). The optimum solution to (12.17) is

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{r}.$$

Our first change of coordinates is to replace  $\mathbf{w}(n)$  by  $\tilde{\mathbf{w}}(n) = \mathbf{w}_o - \mathbf{w}(n)$  in (12.19). Subtract (12.19) from  $\mathbf{w}_o$  and replace  $\mathbf{r}$  in (12.19) by  $\mathbf{r} = \mathbf{R}\mathbf{w}_o$  to obtain

$$\tilde{\mathbf{w}}(n+1) = \tilde{\mathbf{w}}(n) - \mu \mathbf{R} \tilde{\mathbf{w}}(n) = [\mathbf{I} - \mu \mathbf{R}] \tilde{\mathbf{w}}(n). \quad (12.23)$$

Next, multiply this recursion from both sides by  $\mathbf{U}^T = \mathbf{U}^{-1}$

$$\mathbf{U}^T \tilde{\mathbf{w}}(n+1) = \mathbf{U}^T [\mathbf{I} - \mu \mathbf{R}] \underbrace{\mathbf{U} \mathbf{U}^T}_{=\mathbf{I}} \tilde{\mathbf{w}}(n) = [\mathbf{I} - \mu \Lambda] \mathbf{U}^T \tilde{\mathbf{w}}(n).$$

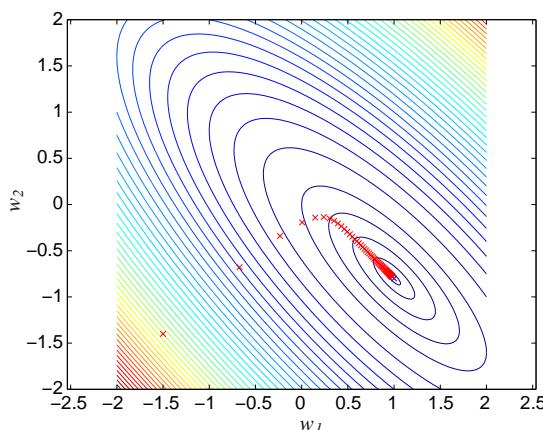
Defining  $\bar{\mathbf{w}}(n) = \mathbf{U}^T \tilde{\mathbf{w}}(n)$ , we have rewritten the gradient equation in a new set of coordinates, such that now the equations are uncoupled. Given that  $\Lambda$  is diagonal, the recursion is simply

$$\begin{bmatrix} \bar{w}_1(n+1) \\ \bar{w}_2(n+1) \end{bmatrix} = \begin{bmatrix} (1 - \mu \lambda_1) \bar{w}_1(n) \\ (1 - \mu \lambda_2) \bar{w}_2(n) \end{bmatrix}. \quad (12.24)$$

Note that, as long as  $|1 - \mu \lambda_i| < 1$  for  $i = 1, 2$ , both entries of  $\bar{\mathbf{w}}(n)$  will converge to zero, and consequently  $\mathbf{w}(n)$  will converge to  $\mathbf{w}_o$ .

The stability condition for this recursion is

$$|1 - \mu \lambda_i| < 1, i = 1, 2 \Rightarrow \mu < \frac{2}{\max\{\lambda_1, \lambda_2\}}.$$



**FIGURE 12.12**

Performance of the steepest descent algorithm for a problem with large ratio of eigenvalues. The crosses ( $\times$ ) represent the successive approximations  $\mathbf{w}(n)$ , plotted against the level curves of (12.17). The initial condition is the point in the lower left. Step-size  $\mu = 0.1$ .

When one of the eigenvalues is much larger than the other, say, when  $\lambda_1 \gg \lambda_2$ , the rate of convergence for the direction relative to the smaller eigenvalue becomes

$$1 > 1 - \mu\lambda_2 > 1 - 2\frac{\lambda_{\min}}{\lambda_{\max}} \approx 1,$$

and even if one of the coordinates in  $\tilde{\mathbf{w}}(n)$  converges quickly to zero, the other will converge very slowly. This is what we saw in Figure 12.12. The ratio  $\lambda_{\max}/\lambda_{\min}$  is known as the *eigenvalue spread* of a matrix.

In general, the gradient algorithm converges slowly when the eigenvalue spread of the Hessian is large. One way of solving this problem is to use a different optimization algorithm, such as the *Newton* or *quasi-Newton* algorithms, which use the inverse of the Hessian matrix, or an approximation to it, to improve the search direction used by the gradient algorithm.

Although these algorithms converge very quickly, they require more computational power, compared to the gradient algorithm. We will see more about them when we describe the RLS (*recursive least-squares*) algorithm in Section 1.12.3.3.

For example, for the quadratic cost-function (12.17), the Hessian is equal to  $2\mathbf{R}$ . If we had a good approximation  $\hat{\mathbf{R}}$  to  $\mathbf{R}$ , we could use the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\hat{\mathbf{R}}^{-1}[\mathbf{r} - \mathbf{R}\mathbf{w}(n)]. \quad (12.25)$$

This class of algorithms is known as *quasi-Newton* (if  $\hat{\mathbf{R}}^{-1}$  is an approximation) or *Newton* (if  $\hat{\mathbf{R}}^{-1}$  is exact). In the Newton case, we would have

$$\mathbf{w}(n+1) = (1 - \mu)\mathbf{w}(n) + \mathbf{w}_o,$$

that is, the algorithm moves at each step precisely in the direction of the optimum solution. Of course, this is not a very interesting algorithm for a quadratic cost function as in this example (the optimum solution is used in the recursion!) However, this method is very useful when the cost function is not quadratic and in adaptive filtering, when the cost function is not known exactly.

## Box 2: Gradients

Consider a function of several variables  $J(\mathbf{w})$ , where

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}.$$

Its gradient is defined by

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}^T} = \begin{bmatrix} \frac{\partial J}{\partial w_0} \\ \frac{\partial J}{\partial w_1} \\ \vdots \\ \frac{\partial J}{\partial w_{M-1}} \end{bmatrix}.$$

The real value of the notation  $\partial J / \partial \mathbf{w}^T$  will only become apparent when working with functions of complex variables, as shown in Box 6. We also define, for consistency,

$$\frac{\partial J}{\partial \mathbf{w}} = \left[ \begin{array}{c} \frac{\partial J}{\partial w_0} \quad \frac{\partial J}{\partial w_1} \quad \cdots \quad \frac{\partial J}{\partial w_{M-1}} \end{array} \right].$$

As an example, consider the quadratic cost function

$$J(\mathbf{w}) = b - 2\mathbf{w}^T \mathbf{r} + \mathbf{w}^T \mathbf{R} \mathbf{w}. \quad (12.26)$$

The gradient of  $J(\mathbf{w})$  is (verify!)

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}^T} = -2\mathbf{r} + 2\mathbf{R}\mathbf{w}. \quad (12.27)$$

If we use the gradient to find the minimum of  $J(\mathbf{w})$ , it would be necessary also to check the second-order derivative, to make sure the solution is not a maximum or a saddle point. The second order derivative of a function of several variables is a matrix, the *Hessian*. It is defined by

$$\nabla_{\mathbf{w}}^2 J = \frac{\partial^2 J}{\partial \mathbf{w} \partial \mathbf{w}^T} = \left[ \begin{array}{c} \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{\partial J}{\partial w_0} \right\} \\ \vdots \\ \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{\partial J}{\partial w_{M-1}} \right\} \end{array} \right] = \left[ \begin{array}{ccc} \frac{\partial^2 J}{\partial w_0^2} & \cdots & \frac{\partial^2 J}{\partial w_0 \partial w_{M-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial w_{M-1} \partial w_0} & \cdots & \frac{\partial^2 J}{\partial w_{M-1}^2} \end{array} \right]. \quad (12.28)$$

Note that for the quadratic cost-function (12.26), the Hessian is equal to  $\mathbf{R} + \mathbf{R}^T$ . It is equal to  $2\mathbf{R}$  if  $\mathbf{R}$  is symmetric, that is, if  $\mathbf{R}^T = \mathbf{R}$ . This will usually be the case here. Note that  $\mathbf{R} + \mathbf{R}^T$  is always symmetric: in fact, since for well-behaved functions  $J$  it holds that

$$\frac{\partial^2 J}{\partial w_i \partial w_j} = \frac{\partial^2 J}{\partial w_j \partial w_i},$$

the Hessian is usually symmetric.

Assume that we want to find the minimum of  $J(\mathbf{w})$ . The first-order conditions are then

$$\left. \frac{\partial J}{\partial \mathbf{w}} \right|_{\mathbf{w}_0} = \mathbf{0}.$$

The solution  $\mathbf{w}_0$  is a minimum of  $J(\mathbf{w})$  if the Hessian at  $\mathbf{w}_0$  is a positive semi-definite matrix, that is, if

$$\mathbf{x}^T \nabla_{\mathbf{w}}^2 J(\mathbf{w}_0) \mathbf{x} \geq 0$$

for all directions  $\mathbf{x}$ .

### Box 3: Useful results from Matrix Analysis

Here we list without proof a few useful results from matrix analysis. Detailed explanations can be found, for example, in [10–13].

**Fact 1 (Traces).** The *trace* of a matrix  $\mathbf{A} \in \mathcal{C}^{M \times M}$  is the sum of its diagonal elements:

$$\text{Tr}(\mathbf{A}) = \sum_{i=1}^M a_{ii}, \quad (12.29)$$

in which the  $a_{ii}$  are the entries of  $\mathbf{A}$ . For any two matrices  $\mathbf{B} \in \mathcal{C}^{M \times N}$  and  $\mathbf{C} \in \mathcal{C}^{N \times M}$ , it holds that

$$\text{Tr}(\mathbf{BC}) = \text{Tr}(\mathbf{CB}). \quad (12.30)$$

In addition, if  $\lambda_i$ ,  $i = 1, \dots, M$  are the eigenvalues of a square matrix  $\mathbf{A} \in \mathcal{C}^{M \times M}$ , then

$$\text{Tr}(\mathbf{A}) = \sum_{i=1}^M \lambda_i. \quad (12.31)$$

**Fact 2 (Singular matrices).** A matrix  $\mathbf{A} \in \mathcal{C}^{M \times M}$  is singular if and only if there exists a nonzero vector  $\mathbf{u}$  such that  $\mathbf{Au} = \mathbf{0}$ .

The *null space*, or *kernel*  $N(\mathbf{A})$  of a matrix  $\mathbf{A} \in \mathcal{C}^{M \times N}$  is the set of vectors  $\mathbf{u} \in \mathcal{C}^N$  such that  $\mathbf{Au} = \mathbf{0}$ . A square matrix  $\mathbf{A} \in \mathcal{C}^{M \times M}$  is nonsingular if, and only if,  $N(\mathbf{A}) = \{\mathbf{0}\}$ , that is, if its null space contains only the null vector.

Note that, if  $\mathbf{A} = \sum_{k=1}^K \mathbf{u}_k \mathbf{v}_k^H$ , with  $\mathbf{u}_k, \mathbf{v}_k \in \mathcal{C}^M$ , then  $\mathbf{A}$  cannot be invertible if  $K < M$ . This is because when  $K < M$ , there always exists a nonzero vector  $\mathbf{x} \in \mathcal{C}^M$  such that  $\mathbf{v}_k^H \mathbf{x} = 0$  for  $1 \leq k \leq K$ , and thus  $\mathbf{Ax} = \mathbf{0}$  with  $\mathbf{x} \neq \mathbf{0}$ .

**Fact 3 (Inverses of block matrices).** Assume the square matrix  $\mathbf{A}$  is partitioned such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix},$$

with  $\mathbf{A}_{11}$  and  $\mathbf{A}_{22}$  square. If  $\mathbf{A}$  and  $\mathbf{A}_{11}$  are invertible (nonsingular), then

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{I} & -\mathbf{A}_{11}^{-1} \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \Delta^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{21} \mathbf{A}_{11}^{-1} & \mathbf{I} \end{bmatrix},$$

where  $\Delta = \mathbf{A}_{22} - \mathbf{A}_{21} \mathbf{A}_{11}^{-1} \mathbf{A}_{12}$  is the *Schur complement* of  $\mathbf{A}_{11}$  in  $\mathbf{A}$ . If  $\mathbf{A}_{22}$  and  $\mathbf{A}$  are invertible, then

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{22}^{-1} \mathbf{A}_{21} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Delta'^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}_{12} \mathbf{A}_{22}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where  $\Delta' = \mathbf{A}_{11} - \mathbf{A}_{12} \mathbf{A}_{22}^{-1} \mathbf{A}_{21}$ .

**Fact 4 (Matrix inversion Lemma).** Let  $A$  and  $B$  be two nonsingular  $M \times M$  matrices,  $D \in \mathcal{C}^{K \times K}$  be also nonsingular, and  $C, E \in \mathcal{C}^{M \times K}$  be such that

$$A = B + CDE^H. \quad (12.32)$$

The inverse of  $A$  is then given by

$$A^{-1} = B^{-1} - B^{-1}C(D^{-1} + E^H B^{-1}C)^{-1}E^H B^{-1}. \quad (12.33)$$

The lemma is most useful when  $K \ll M$ . In particular when  $K = 1$ ,  $D + E^H B^{-1}C$  is a scalar, and we have

$$A^{-1} = B^{-1} - \frac{B^{-1}CE^H B^{-1}}{D^{-1} + E^H B^{-1}C}.$$

**Fact 5 (Symmetric matrices).** Let  $A \in \mathcal{C}^{M \times M}$  be *Hermitian Symmetric* (or simply *Hermitian*), that is,  $A^H = (A^T)^* = A$ . Then, it holds that

1. All eigenvalues  $\lambda_i, i = 1, \dots, M$  of  $A$  are real.
2.  $A$  has a complete set of orthonormal eigenvectors  $u_i, i = 1, \dots, M$ . Since the eigenvectors are orthonormal,  $u_i^H u_j = \delta_{ij}$ , where  $\delta_{ij} = 1$  if  $i = j$  and 0 otherwise.
3. Arranging the eigenvectors in a matrix  $U = [u_1 \dots u_M]$ , we find that  $U$  is *unitary*, that is,  $U^H U = UU^H = I$ . In addition,

$$U^H A U = \Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{bmatrix}.$$

4. If  $A = A^T \in \mathcal{R}^{M \times M}$ , it is called simply *symmetric*. In this case, not only the eigenvalues, but also the eigenvectors are real.
5. If  $A$  is symmetric and invertible, then  $A^{-1}$  is also symmetric.

**Fact 6 (Positive-definite matrices).** If  $A \in \mathcal{C}^{M \times M}$  is Hermitian symmetric and moreover for all  $u \in \mathcal{C}^M$  it holds that

$$u^H A u \geq 0, \quad (12.34)$$

then  $A$  is *positive semi-definite*. Positive semi-definite matrices have non-negative eigenvalues:  $\lambda_i \geq 0$ ,  $i = 1, \dots, M$ . They may be singular, if one or more eigenvalue is zero.

If the inequality (12.34) is strict, that is, if

$$u^H A u > 0 \quad (12.35)$$

for all  $u \neq 0$ , then  $A$  is *positive-definite*. All eigenvalues  $\lambda_i$  of positive-definite matrices satisfy  $\lambda_i > 0$ . Positive-definite matrices are always nonsingular.

Finally, all positive-definite matrices admit a Cholesky factorization, i.e., if  $A$  is positive-definite, there is a lower-triangular matrix  $L$  such that  $A = LL^H$  [14].

**Fact 7 (Norms and spectral radius).** The *spectral radius*  $\rho(\mathbf{A})$  of a matrix  $\mathbf{A} \in \mathcal{C}^{M \times M}$  is

$$\rho(\mathbf{A}) = \max_{1 \leq i \leq M} |\lambda_i|, \quad (12.36)$$

in which  $\lambda_i$  are the eigenvalues of  $\mathbf{A}$ . The spectral radius is important, because a linear system

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n)$$

is stable if and only if  $\rho(\mathbf{A}) \leq 1$ , as is well known. A useful inequality is that, for any matrix norm  $\|\cdot\|$  such that for any  $\mathbf{A}, \mathbf{B} \in \mathcal{C}^{M \times M}$

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|,$$

it holds that

$$\rho(\mathbf{A}) \leq \|\mathbf{A}\|. \quad (12.37)$$

This property is most useful when used with norms that are easy to evaluate, such as the 1-norm,

$$\|\mathbf{A}\|_1 = \max_{1 \leq i \leq M} \sum_{j=1}^M |a_{ij}|,$$

where  $a_{ij}$  are the entries of  $\mathbf{A}$ . We could also use the infinity norm,

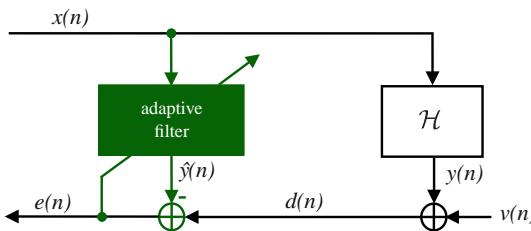
$$\|\mathbf{A}\|_\infty = \max_{1 \leq j \leq M} \sum_{i=1}^M |a_{ij}|.$$

### 1.12.1.3 Applications

Due to the ability to adjust themselves to different environments, adaptive filters can be used in different signal processing and control applications. Thus, they have been used as a powerful device in several fields, such as communications, radar, sonar, biomedical engineering, active noise control, modeling, etc. It is common to divide these applications into four groups:

1. interference cancellation;
2. system identification;
3. prediction; and
4. inverse system identification.

In the first three cases, the goal of the adaptive filter is to find an approximation  $\hat{y}(n)$  for the signal  $y(n)$ , which is contained in the signal  $d(n) = y(n) + v(n)$ . Thus, as  $\hat{y}(n)$  approaches  $y(n)$ , the signal  $e(n) = d(n) - \hat{y}(n)$  approaches  $v(n)$ . The difference between these applications is in what we are interested. In interference cancellation, we are interested in the signal  $v(n)$ , as is the case of acoustic echo cancellation, where  $v(n)$  is the speech of the person using the hands-free telephone (go back to Figures 12.4 and 12.7). In system identification, we are interested in the filter parameters, and in

**FIGURE 12.13**

Scheme for interference cancellation and system identification.  $x(n)$  and  $y(n)$  are correlated to each other.

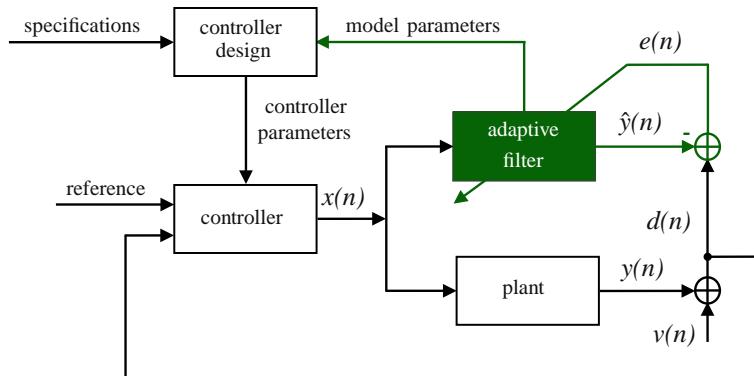
prediction, we may be interested in the signal  $v(n)$  and/or in the filter parameters. Note that in all these applications (interference cancellation, system identification and prediction), the signal  $e(n)$  is an approximation for  $v(n)$  and should not converge to zero, except when  $v(n) \equiv 0$ . In inverse system identification, differently from the other applications, the signal at the output of the adaptive filter must be as close as possible to the signal  $d(n)$  and thus, ideally, the signal  $e(n)$  should be zeroed. In the sequel, we give an overview of these four groups in order to arrive at a common formulation that will simplify our analysis of adaptive filtering algorithms in further sections.

#### 1.12.1.3.1 Interference cancellation

In a general interference cancellation problem, we have access to a signal  $d(n)$ , which is a mixture of two other signals,  $v(n)$  and  $y(n)$ . We are interested in one of these signals, and want to separate it from the other (the interference) (see Figure 12.13). Even though we do not know  $v(n)$  or  $y(n)$ , we have some information about  $y(n)$ , usually in the form of a reference signal  $x(n)$  that is related to  $y(n)$  through a filtering operation  $\mathcal{H}$ . We may know the general form of this operation, for example, we may know that the relation is linear and well approximated by an FIR filter with 200 coefficients. However, we do not know the parameters (the filter coefficients) necessary to reproduce it. The goal of the adaptive filter is to find an approximation  $\hat{y}(n)$  to  $y(n)$ , given only  $x(n)$  and  $d(n)$ . In the process of finding this  $\hat{y}$ , the adaptive filter will construct an approximation for the relation  $\mathcal{H}$ . A typical example of an interference cancellation problem is the echo cancellation example we gave in Section 1.12.1.1. In that example,  $x(n)$  is the far-end voice signal,  $v(n)$  is the near-end voice signal, and  $y(n)$  is the echo. The relation  $\mathcal{H}$  is usually well approximated by a linear FIR filter with a few hundred taps.

The approximation for  $\mathcal{H}$  is a by-product, that does not need to be very accurate as long as it leads to a  $\hat{y}(n)$  close to  $y(n)$ . This configuration is shown in Figure 12.13 and is called *interference cancellation*, since the interference  $y(n)$  should be canceled. Note that we do not want to make  $e(n) \equiv 0$ , otherwise we would not only be killing the interference  $y(n)$ , but also the signal of interest  $v(n)$ .

There are many applications of interference cancellation. In addition to acoustic and line echo cancellation, we can mention, for example, adaptive notch filters for cancellation of sinusoidal interference (a common application is removal of 50 or 60 Hz interference from the mains line), cancellation of the maternal electrocardiography in fetal electrocardiography, cancellation of echoes in long distance telephone circuits, active noise control (as in noise-canceling headphones), and active vibration control.



**FIGURE 12.14**

Scheme for plant identification in a control system.

### 1.12.1.3.2 System identification

In interference cancellation, the coefficients of the adaptive filter converge to an approximation for the true relation  $\mathcal{H}$ , but as mentioned before, this approximation is a by-product that does not need to be very accurate. However, there are some applications in which the goal is to construct an approximation as accurate as possible for the unknown relation  $\mathcal{H}$  between  $x(n)$  and  $y(n)$ , thus obtaining a model for the unknown system. This is called *system identification* or *modeling* (the diagram in Figure 12.13 applies also for this case). The signal  $d(n)$  is now composed of the output  $y(n)$  of an unknown system  $\mathcal{H}$ , plus noise  $v(n)$ . The reference signal  $x(n)$  is the input to the system which, when possible, is chosen to be white noise. In general, this problem is harder than interference cancellation, as we shall see in Section 1.12.2.1.5, due to some conditions that the reference signal  $x(n)$  must satisfy. However, one point does not change: in the ideal case, the signal  $e(n)$  will be equal to the noise  $v(n)$ . Again, we do not want to make  $e(n) \equiv 0$ , otherwise we would be trying to model the noise (however, the smaller the noise the easier the task, as one would expect).

In many control systems, an unknown dynamic system (also called as *plant* in control system terminology) is identified online and the result is used in a self-tuning controller, as depicted in Figure 12.14. Both the plant and the adaptive filter have the same input  $x(n)$ . In practical situations, the plant to be modeled is noisy, which is represented by the signal  $v(n)$  added to the plant output. The noise  $v(n)$  is generally uncorrelated with the plant input. The task of the adaptive filter is to minimize the error model  $e(n)$  and track the time variations in the dynamics of the plant. The model parameters are continually fed back to the controller to obtain the controller parameters used in the self-tuning regulator loop [15].

There are many applications that require the identification of an unknown system. In addition to control systems, we can mention, for example, the identification of the room impulse response, used to study the sound quality in concert rooms, and the estimation of the communication channel impulse response, required by maximum likelihood detectors and blind equalization techniques based on second-order statistics.

### 1.12.1.3.3 Prediction

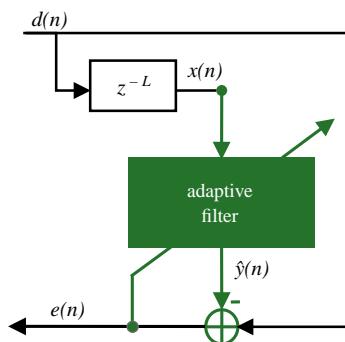
In *prediction*, the goal is to find a relation between the current sample  $d(n)$  and previous samples  $d(n-L), \dots, d(n-L-M+1)$ , as shown in Figure 12.15. Therefore, we want to model  $d(n)$  as a part  $y(n)$  that depends only on  $d(n-L), \dots, d(n-L-M+1)$ , and a part  $v(n)$  that represents new information. For example, for a linear model, we would have

$$d(n) = \underbrace{\sum_{k=0}^{M-1} h_k d(n-L-k)}_{y(n)} + v(n),$$

Thus, we in fact have again the same problem as in Figure 12.13. The difference is that now the reference signal is a delayed version of  $d(n)$ :  $x(n) = d(n-L)$ , and the adaptive filter will try to find an approximation  $\hat{y}(n)$  for  $y(n)$ , thereby separating the “predictable” part  $y(n)$  of  $d(n)$  from the “new information”  $v(n)$ , to which the signal  $e(n) = d(n) - \hat{y}(n)$  should converge. The system  $\mathcal{H}$  in Figure 12.13 now represents the relation between  $d(n-L)$  and the predictable part of  $d(n)$ .

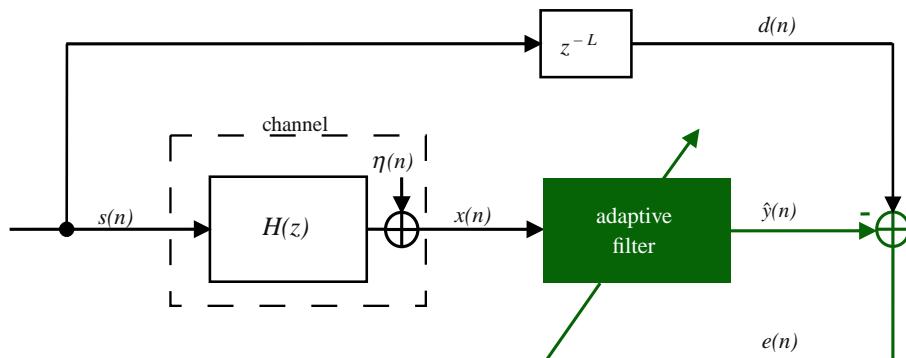
Prediction finds application in many fields. We can mention, for example, linear predictive coding (LPC) and adaptive differential pulse code modulation (ADPCM) used in speech coding, adaptive line enhancement, autoregressive spectral analysis, etc.

For example, adaptive line enhancement (ALE) seeks the solution for a classical detection problem, whose objective is to separate a narrowband signal  $y(n)$  from a wideband signal  $v(n)$ . This is the case, for example, of finding a low-level sine wave (predictable narrowband signal) in noise (non-predictable wideband signal). The signal  $d(n)$  is constituted by the sum of these two signals. Using  $d(n)$  as the reference signal and a delayed replica of it, i.e.,  $d(n-L)$ , as input of the adaptive filter as in Figure 12.15, the output  $\hat{y}(n)$  provides an estimate of the (predictable) narrowband signal  $y(n)$ , while the error signal  $e(n)$  provides an estimate of the wideband signal  $v(n)$ . The delay  $L$  is also known as *decorrelation parameter* of the ALE, since its main function is to remove the correlation between the wideband signal  $v(n)$  present in the reference  $d(n)$  and in the delayed predictor input  $d(n-L)$ .

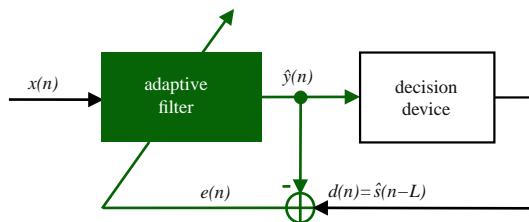


**FIGURE 12.15**

Scheme for prediction.  $x(n) = d(n-L)$  is a delayed version of  $d(n)$ .

**FIGURE 12.16**

Simplified communications system with an adaptive equalizer in the training mode.

**FIGURE 12.17**

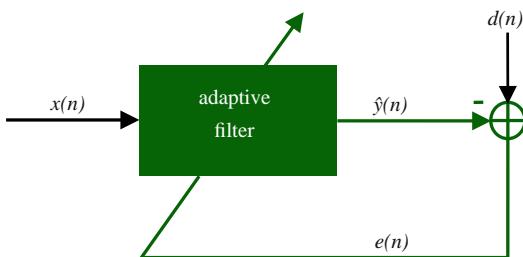
Adaptive equalizer in the decision-directed mode.

#### 1.12.1.3.4 Inverse system identification

*Inverse system identification*, also known as *deconvolution*, has been widely used in different fields as communications, acoustics, optics, image processing, control, among others. In communications, it is also known as *channel equalization* and the adaptive filter is commonly called as *equalizer*. Adaptive equalizers play an important role in digital communications systems, since they are used to mitigate the inter-symbol interference (ISI) introduced by dispersive channels. Due to its importance, we focus on the equalization application, which is explained in the sequel.

A simplified baseband communications system is depicted in Figure 12.16. The signal  $s(n)$  is transmitted through an unknown channel, whose model is constituted by an FIR filter with transfer function  $H(z) = h_0 + h_1z^{-1} + \dots + h_{K-1}z^{K-1}$  and additive noise  $\eta(n)$ . Due to the channel memory, the signal at the receiver contains contributions not only from  $s(n)$ , but also from the previous symbols  $s(n-1), s(n-2), \dots, s(n-K+1)$ , i.e.,

$$x(n) = \underbrace{\sum_{k=0}^{L-1} h_k s(n-k)}_{\text{pre-ISI}} + h_L s(n-L) + \underbrace{\sum_{k=L+1}^{K-1} h_k s(n-k)}_{\text{post-ISI}} + \eta(n). \quad (12.38)$$

**FIGURE 12.18**

Inputs and output of an adaptive filter.

Assuming that the overall channel-equalizer system imposes a delay of  $L$  samples, the adaptive filter will try to find an approximation  $\hat{y}(n)$  for  $d(n) = s(n - L)$  and for this purpose, the two summations in (12.38), which constitute the inter-symbol interference, must be mitigated. Of course, when you are transmitting information the receiver will not have access to  $d(n - L)$ . The filter will adapt during a *training* phase, in which the transmitter sends a pre-agreed signal. You can see that in this case, the role of  $x(n)$  and  $d(n)$  is the reverse of that in system identification. In this case, you would indeed like to have  $e(n) \equiv 0$ . The problem is that this is not possible, given the presence of noise in  $x(n)$ . The role of the adaptive filter is to approximately invert the effect of the channel, at the same time trying to suppress the noise (or at least, not to amplify it too much). In one sense, we are back to the problem of separating two signals, but now the mixture is in the reference signal  $x(n)$ , so what can and what cannot be done is considerably different from the other cases.

The scheme of Figure 12.16 is also called training mode, since the delayed version of the transmitted sequence  $d(n) = s(n - L)$  (training sequence) is known at the receiver. After the convergence of the filter, the signal  $d(n)$  is changed to the estimate  $\hat{s}(n - L)$  obtained at the output of a decision device, as shown in Figure 12.17. In this case, the equalizer works in the so-called decision-directed mode. The decision device depends on the signal constellation—for example, if  $s(n) = \pm 1$ , the decision device returns +1 for  $\hat{y}(n) \geq 0$ , and -1 for  $\hat{y}(n) < 0$ .

#### 1.12.1.3.5 A common formulation

In all the four groups of applications, the inputs of the adaptive filter are given by the signals  $x(n)$  and  $d(n)$  and the output by  $\hat{y}(n)$ . Note that the input  $d(n)$  appears effectively in the signal  $e(n) = d(n) - \hat{y}(n)$ , which is computed and fed back at each time instant  $n$ , as shown in Figure 12.18. In the literature, the signal  $d(n)$  is referred to as desired signal,  $x(n)$  as reference signal and  $e(n)$  as error signal. These names unfortunately are somewhat misleading, since they give the impression that our goal is to recover exactly  $d(n)$  by filtering (possibly in a nonlinear way) the reference  $x(n)$ . In almost all applications, this is far from the truth, as previously discussed. In fact, except in the case of channel equalization, exactly zeroing the error would result in very poor performance.

The only application in which  $d(n)$  is indeed a “desired signal” is channel equalization, in which  $d(n) = s(n - L)$ . Despite this particularity in channel equalization, the common feature in all adaptive filtering applications is that the filter must learn a relation between the reference  $x(n)$  and the desired

signal  $d(n)$  (we will use this name to be consistent with the literature.) In the process of building this relation, the adaptive filter is able to perform useful tasks, such as separating two mixed signals; or recovering a distorted signal. Section 1.12.2 explains this idea in more detail.

## 1.12.2 Optimum filtering

We now need to extend the ideas of Section 1.12.1.2, that applied only to periodic (deterministic) signals, to more general classes of signals. For this, we will use tools from the theory of stochastic processes. The main ideas are very similar: as before, we need to choose a structure for our filter and a means of measuring how far or how near we are to the solution. This is done by choosing a convenient cost function, whose minimum we must search iteratively, based only on measurable signals.

In the case of periodic signals, we saw that minimizing the error power (12.3), repeated below,

$$P = \sum_{k=1}^{K_0} \frac{\tilde{C}_k^2}{2} + \sum_{\ell=1}^{K_1} \frac{B_\ell^2}{2},$$

would be equivalent to zeroing the echo. However, we were not able to compute the error power exactly—we used an approximation through a time-average, as in (12.5), repeated here:

$$\hat{P}(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^2(n-k),$$

where  $N$  is a convenient window length (in Section 1.12.1.2, we saw that choosing a large window length is approximately equivalent to choosing  $N = 1$  and a small step-size). Since we used an approximated estimate of the cost, our solution was also an approximation: the estimated coefficients did not converge exactly to their optimum values, but instead hovered around the optimum (Figures 12.9 and 12.10).

The minimization of the time-average  $\hat{P}(n)$  also works in the more general case in which the echo is modeled as a random signal—but what is the corresponding exact cost function? The answer is given by the property of *ergodicity* that some random signals possess. If a random signal  $s(n)$  is such that its mean  $E\{s(n)\}$  does not depend on  $n$  and its autocorrelation  $E\{s(n)s(k)\}$  depends only on  $n - k$ , it is called wide-sense stationary (WSS) [16]. If  $s(n)$  is also (mean-square) ergodic, then

$$\text{Average power of } s(n) = \lim_{N \rightarrow \infty} \frac{1}{N+1} \sum_{n=0}^N s^2(n) = E\{s^2(n)\}. \quad (12.39)$$

Note that  $E\{s^2(n)\}$  is an ensemble average, that is, the expected value is computed over all possible realizations of the random signal. Relation (12.39) means that for an ergodic signal, the time average of a single realization of the process is equal to the ensemble-average of all possible realizations of the process. This ensemble average is the exact cost function that we would like to minimize.

In the case of adaptive filters, (12.39) holds approximately for  $e(n)$  for a finite value of  $N$  if the environment does not change too fast, and if the filter adapts slowly. Therefore, for random variables we will still use the average error power as a measure of how well the adaptive filter is doing. The difference

is that in the case of periodic signals, we could understand the effect of minimizing the average error power in terms of the amplitudes of each harmonic in the signal, but now the interpretation will be in terms of ensemble averages (variances).

Although the error power is not the only possible choice for cost function, it is useful to study this choice in detail. Quadratic cost functions such as the error power have a number of properties that make them popular. For example, they are differentiable, and so it is relatively easy to find a closed-form solution for the optimum, and in the important case where all signals are Gaussian, the optimum filters are linear. Of course, quadratic cost functions are not the best option in all situations: the use of other cost functions, or even of adaptive filters not based on the minimization of a cost function, is becoming more common. We will talk about some of the most important alternatives in Sections 1.12.5.2 and 1.12.5.4.

Given the importance of quadratic cost functions, in this section we study them in detail. Our focus is on what could and what could not be done if the filter were able to measure perfectly  $E\{e^2(n)\}$  at each instant. This discussion has two main goals: the first is to learn what is feasible, so we do not expect more from a filter than it can deliver. The second is to enable us to make more knowledgeable choices when designing a filter—for example, which filter order should be used, and for system identification, which input signal would result in better performance. In Section 1.12.4.4.3 we will study the performance of adaptive filters taking into consideration their imperfect knowledge of the environment.

### 1.12.2.1 Linear least-mean squares estimation

In the examples we gave in previous sections, a generic adaptive filtering problem resumed to this: we are given two sequences,  $\{x(n)\}$  and  $\{d(n)\}$ , as depicted in Figure 12.18. It is known, from physical arguments, that  $d(n)$  has two parts, one that is in some sense related to  $x(n)$ ,  $x(n-1)$ , ..., and another that is not, and that both parts are combined additively, that is,

$$d(n) = \underbrace{\mathcal{H}(x(n), x(n-1), \dots)}_{y(n)} + v(n), \quad (12.40)$$

where  $\mathcal{H}(\cdot)$  is an unknown relation, and  $v(n)$  is the part “unrelated” to  $\{x(n)\}$ . In our examples so far,  $\mathcal{H}(\cdot)$  was linear, but we do not need to restrict ourselves to this case. Our objective is to extract  $y(n)$  and  $v(n)$  from  $d(n)$ , based only on observations of  $\{x(n)\}$  and  $\{d(n)\}$ . We saw in Section 1.12.1.2 that the average power of the difference  $e(n)$  between  $d(n)$  and our current approximation  $\hat{y}(n)$  could be used as a measure of how close we are to the solution. In general,  $\mathcal{H}(\cdot)$  is time-variant, i.e., depends directly on  $n$ . We will not write this dependence explicitly to simplify the notation.

In this section we ask in some sense the inverse problem, that is: given two sequences  $\{x(n)\}$  and  $\{d(n)\}$ , what sort of relation will be found between them if we use  $E\{e^2(n)\}$  as a standard? The difference is that we now do not assume a model such as (12.40); instead, we want to know what kind of model results from the exact minimization of  $E\{e^2(n)\}$ , where  $e(n)$  is defined as the difference between  $d(n)$  and a function of  $x(n)$ ,  $x(n-1)$ , ... .

We can answer this question in an entirely general way, without specifying beforehand the form of  $\mathcal{H}$ . This is done in Box 4. However, if we have information about the physics of a problem and know that

the relation  $\mathcal{H}(x(n), x(n-1), \dots)$  is of a certain kind, we can restrict the search for the solution  $\hat{\mathcal{H}}$  to this class. This usually reduces the complexity of the filter and increases its convergence rate (because less data is necessary to estimate a model with less unknowns, as we will see in Section 1.12.3). In this section we focus on this second option.

The first task is to describe the class  $\mathcal{F}$  of allowed functions  $\hat{\mathcal{H}}$ . This may be done by choosing a relation that depends on a few parameters, such as (recall that the adaptive filter output is  $\hat{y}(n) = \hat{\mathcal{H}}(x(n), x(n-1), \dots)$ )

$$\mathcal{F}_{\text{FIR}} \text{ (FIR filter): } \hat{y}(n) = w_0 x(n) + \dots + w_{M-1} x(n-M+1), \quad (12.41)$$

$$\mathcal{F}_{\text{IIR}} \text{ (IIR filter): } \hat{y}(n) = -a_1 \hat{y}(n-1) + b_0 x(n), \quad (12.42)$$

$$\begin{aligned} \mathcal{F}_{\text{V}} \text{ (Volterra filter): } \hat{y}(n) = & w_0 x(n) + w_1 x(n-1) + w_{0,0} x^2(n) \\ & + w_{0,1} x(n)x(n-1) + w_{1,1} x(n-1)^2, \end{aligned} \quad (12.43)$$

$$\mathcal{F}_{\text{S}} \text{ (Saturation): } \hat{y}(n) = \arctan(ax(n)). \quad (12.44)$$

In each of these cases, the relation between the input sequence  $\{x(n)\}$  and  $d(n)$  is constrained to a certain class, for example, linear length- $M$  FIR filters in (12.41), first-order IIR filters in (12.42), and second-order Volterra filters in (12.43). Each class is described by a certain number of parameters: the filter coefficients  $w_0, \dots, w_{M-1}$  in the case of FIR filters,  $a_1$  and  $b_0$  in the case of IIR filters, and so on. The task of the adaptive filter will then be to choose the values of the parameters that best fit the data.

For several practical reasons, it is convenient if we make the filter output depend linearly on the parameters, as happens in (12.41) and in (12.43). It is important to distinguish linear in the parameters from input-output linear: (12.43) is linear in the parameters, but the relation between the input sequence  $\{x(n)\}$  and the output  $\hat{y}(n)$  is nonlinear. What may come as a surprise is that the IIR filter of Eq. (12.42) is *not* linear in the parameters: in fact,  $\hat{y}(n) = -a_1 \hat{y}(n-1) + b_0 x(n) = a_1^2 \hat{y}(n-2) - a_1 b_0 x(n-1) + b_0 x(n) = \dots$  — you can see that  $\hat{y}(n)$  depends nonlinearly on  $a_1$  and  $b_0$ .

Linearly parametrized classes  $\mathcal{F}$ , such as  $\mathcal{F}_{\text{FIR}}$  (12.41) and  $\mathcal{F}_{\text{V}}$  (12.43) are popular because in general it is easier to find the optimum parameters, both theoretically and in real time. In fact, when the filter output depends linearly on the parameters and the cost function is a *convex* function of the error, it can be shown that the optimal solution is unique (see Box 5). This is a very desirable property, since it simplifies the search for the optimal solution.

In the remainder of this section we will concentrate on classes of relations that are linear in the parameters. As  $\mathcal{F}_{\text{V}}$  shows, this does not imply that we are restricting ourselves to linear models. There are, however, adaptive filtering algorithms that use classes of relations that are not linear in the parameters, such as IIR adaptive filters. On the other hand, blind equalization algorithms are based on non-convex cost functions.

Assume then that we have chosen a convenient class of relations  $\mathcal{F}$  that depends linearly on its parameters. That is, we want to solve the problem

$$\min_{\hat{\mathcal{H}} \in \mathcal{F}} E \left\{ \left[ d(n) - \hat{\mathcal{H}}(x(n), x(n-1), \dots) \right]^2 \right\}. \quad (12.45)$$

We want to know which properties the solution to this problem will have when  $\mathcal{F}$  depends linearly on a finite number of parameters. In this case,  $\mathcal{F}$  is a linear combinations of certain functions  $\phi_i$  of  $x(n), x(n - 1), \dots$

$$\hat{y}(n) = w_0\phi_0 + w_1\phi_1 + \dots + w_{M-1}\phi_{M-1} \stackrel{\Delta}{=} \mathbf{w}^T \boldsymbol{\phi}, \quad (12.46)$$

where in general  $\phi_i = \phi_i(x(n), x(n - 1), \dots), 0 \leq i \leq M - 1$ . The vector  $\boldsymbol{\phi}$  is known as *regressor*. In the case of length- $M$  FIR filters, we would have

$$\phi_0 = x(n), \phi_1 = x(n - 1), \dots, \phi_{M-1} = x(n - M + 1),$$

whereas in the case of second-order Volterra filters with memory 1, we would have

$$\phi_0 = x(n), \phi_1 = x(n - 1), \phi_2 = x^2(n), \phi_3 = x(n)x(n - 1), \phi_4 = x^2(n - 1).$$

Our problem can then be written in general terms as: find  $\mathbf{w}_o$  such that

$$\mathbf{w}_o = \arg \min_{\mathbf{w}} E\{(d - \mathbf{w}^T \boldsymbol{\phi})^2\}. \quad (12.47)$$

We omitted the dependence of the variables on  $n$  to lighten the notation. Note that, in general,  $\mathbf{w}_o$  will also depend on  $n$ .

To solve this problem, we use the facts that  $\mathbf{w}^T \boldsymbol{\phi} = \boldsymbol{\phi}^T \mathbf{w}$  to expand the expected value

$$J(\mathbf{w}) \stackrel{\Delta}{=} E\left\{\left(d - \mathbf{w}^T \boldsymbol{\phi}\right)^2\right\} = E\{d^2\} - 2\mathbf{w}^T E\{d\boldsymbol{\phi}\} + \mathbf{w}^T E\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}\mathbf{w}.$$

Recall that the weight vector  $\mathbf{w}$  is *not* random, so we can take it out of the expectations.

Define the autocorrelation of  $d$ , and also the cross-correlation vector and autocorrelation matrix

$$r_d = E\{d^2\}, \quad \mathbf{r}_{d\boldsymbol{\phi}} = E\{d\boldsymbol{\phi}\}, \quad \mathbf{R}_{\boldsymbol{\phi}} = E\{\boldsymbol{\phi}\boldsymbol{\phi}^T\}. \quad (12.48)$$

The cost function then becomes

$$J(\mathbf{w}) = r_d - 2\mathbf{w}^T \mathbf{r}_{d\boldsymbol{\phi}} + \mathbf{w}^T \mathbf{R}_{\boldsymbol{\phi}} \mathbf{w}. \quad (12.49)$$

Differentiating  $J(\mathbf{w})$  with respect to  $\mathbf{w}$ , we obtain

$$\frac{\partial J}{\partial \mathbf{w}^T} = -2\mathbf{r}_{d\boldsymbol{\phi}} + 2\mathbf{R}_{\boldsymbol{\phi}} \mathbf{w},$$

and equating the result to zero, we see that the optimal solution must satisfy

$$\mathbf{R}_{\boldsymbol{\phi}} \mathbf{w}_o = \mathbf{r}_{d\boldsymbol{\phi}}. \quad (12.50)$$

These are known as the *normal*, or *Wiener-Hopf*, equations. The solution,  $\mathbf{w}_o$ , is known as the *Wiener solution*. A note of caution: the Wiener solution is not the same thing as the Wiener filter. The Wiener filter is the linear filter that minimizes the mean-square error, without restriction of filter order [17]. The difference is that the Wiener solution has the filter order pre-specified (and is not restricted to linear filters, as we saw).

When the autocorrelation matrix is non-singular (which is usually the case), the Wiener solution is

$$\mathbf{w}_o = \mathbf{R}_{\phi}^{-1} \mathbf{r}_{d\phi}. \quad (12.51)$$

Given  $\mathbf{w}_o$ , the optimum error will be

$$v_o = d - \mathbf{w}_o^T \phi. \quad (12.52)$$

Note that the expected value of  $v_o$  is not necessarily zero:

$$E\{v_o\} = E\{d\} - \mathbf{w}_o^T E\{\phi\}. \quad (12.53)$$

If, for example,  $E\{\phi\} = \mathbf{0}$  and  $E\{d\} \neq 0$ , then  $E\{v_o\} = E\{d\} \neq 0$ . In practice, it is good to keep  $E\{v_o\} = 0$ , because we usually know that  $v_o$  should approximate a zero-mean signal, such as speech or noise. We will show shortly, in Section 1.12.2.1.4, how to guarantee that  $v_o$  has zero mean.

### 1.12.2.1.1 Orthogonality condition

A key property of the Wiener solution is that the optimum error is orthogonal to the regressor  $\phi$ , that is,

$$E\{v_o \phi\} = E\{\phi(d - \underbrace{\mathbf{w}_o^T \phi}_{= \phi^T \mathbf{w}_o})\} = E\{d\phi\} - E\{\phi \phi^T\} \mathbf{w}_o = \mathbf{r}_{d\phi} - \mathbf{R}_{\phi} \mathbf{w}_o = \mathbf{0}. \quad (12.54)$$

We saw a similar condition in Section 1.12.1.2. It is very useful to remember this result, known as the *orthogonality condition*: from it, we will find when to apply the cost function (12.47) to design an adaptive filter. Note that when  $v_o$  has zero mean, (12.54) also implies that  $v_o$  and  $\phi$  are uncorrelated.

**Remark 1.** You should not confuse orthogonal with uncorrelated. Two random variables  $x$  and  $y$  are orthogonal if  $E\{xy\} = 0$ , whereas they are uncorrelated if  $E\{xy\} = E\{x\} E\{y\}$ . The two concepts coincide only if either  $x$  or  $y$  have zero mean.

**Remark 2.** The orthogonality condition is an intuitive result, if we remember that we can think of the space of random variables with finite variance as a vector space. In fact, define the cross-correlation  $r_{xy} = E\{xy\}$  between two random variables  $x$  and  $y$  as the inner product. Then the autocorrelation of a random variable  $x$ ,  $E\{x^2\}$ , would be interpreted as the square of the “length” of  $x$ . In this case, our problem (12.47) is equivalent to finding the vector in the subspace spanned by  $\phi_0, \dots, \phi_{M-1}$  that is closest to  $d$ . As we know, the solution is such that the error is orthogonal to the subspace. So, the orthogonality condition results from the vector space structure of our problem and the quadratic nature of our cost function. See Figure 12.19.

**Remark 3.** The difference between (12.54) and the corresponding result obtained in Box 4, Eq. (12.105), is that here the optimum error is orthogonal only to the functions  $\phi_i$  of  $x(n), x(n-1), \dots$  that were included in the regressor (and their linear combinations), not to any function, as in (12.105). This is not surprising: in this section the optimization was made only over the functions in class  $\mathcal{F}$ , and in Box 4 we allow any function. Both solutions, (12.106) and (12.51), will be equal if the general solution according to Box 4 is indeed in  $\mathcal{F}$ .

**Remark 4.** The optimal mean-square error is (recall that  $E\{v_o \phi\} = \mathbf{0}$ )

$$\begin{aligned} J_{\min} = r_{v,o} &= E\{v_o^2\} = E\{v_o(d - \mathbf{w}_o^T \phi)\} = E\{v_o d\} \\ &= E\{(d - \mathbf{w}_o^T \phi)d\} = E\{d^2\} - \mathbf{w}_o^T \mathbf{r}_{d\phi} = r_d - \mathbf{r}_{d\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}. \end{aligned} \quad (12.55)$$

Now that we have the general solution to (12.47), we turn to the question: for which class of problems is the quadratic cost function adequate?

#### 1.12.2.1.2 Implicit vs. physical models

From (12.54), we see that the fact that we are minimizing the mean-square error between  $d$  and a linear combination of the regressor  $\phi$  induces a model

$$d = \mathbf{w}_o^T \phi + v_o, \quad (12.56)$$

in which  $v_o$  is orthogonal to  $\phi$ . This is not an assumption, as we sometimes see in the literature, but a consequence of the quadratic cost function we chose. In other words, there is always a relation such as (12.56) between any pair  $(d, \phi)$ , as long as both have finite second-order moments. This relation may make sense from a physical analysis of the problem (as we saw in the echo cancellation example), or be simply a consequence of solving (12.47) (see Section 1.12.2.1.3 for examples).

On the other hand, the orthogonality condition allows us to find out when the solution of (12.47) will be able to successfully solve a problem: assume now that we know beforehand, by physical arguments

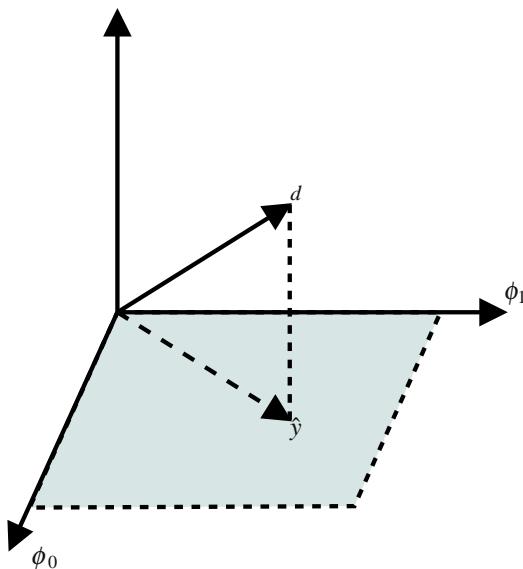


FIGURE 12.19

Orthogonality in vector spaces with inner products.

about the problem, that  $d$  and  $\phi$  must be related through

$$d = \mathbf{w}_*^T \phi + v, \quad (12.57)$$

where  $\mathbf{w}_*$  is a certain coefficient vector and  $v$  is orthogonal to  $\phi$ . Will the solution to (12.47) be such that  $\mathbf{w}_o = \mathbf{w}_*$  and  $v_o = v$ ?

To check if this is the case, we only need to evaluate the cross-correlation vector  $\mathbf{r}_{d\phi}$  assuming the model (12.57):

$$\begin{aligned} \mathbf{r}_{d\phi} &= E\{d\phi\} = E\{\phi(\underbrace{\mathbf{w}_*^T \phi}_{= \phi^T \mathbf{w}_*} + v)\} = E\{\phi\phi^T\}\mathbf{w}_* + \underbrace{E\{\phi v\}}_{= 0} = \mathbf{R}_\phi \mathbf{w}_*. \end{aligned} \quad (12.58)$$

Therefore,  $\mathbf{w}_*$  also obeys the normal equations, and we can conclude that  $\mathbf{w}_o = \mathbf{w}_*$ ,  $v_o = v$ , as long as  $\mathbf{R}_\phi$  is nonsingular.

Even if  $\mathbf{R}_\phi$  is singular, the optimal error  $v_o$  will always be equal to  $v$  (in a certain sense). In fact, if  $\mathbf{R}_\phi$  is singular, then there exists a vector  $a$  such that  $\mathbf{R}_\phi a = \mathbf{0}$  (see Fact 2 in Box 3), and thus any solution  $\mathbf{w}_o$  of the normal equations (we know from (12.58) that there is at least one,  $\mathbf{w}_*$ ) will have the form

$$\mathbf{w}_o = \mathbf{w}_* + a, \quad a \in N(\mathbf{R}_\phi), \quad (12.59)$$

where  $N(\mathbf{R}_\phi)$  is the null-space of  $\mathbf{R}_\phi$ . In addition,

$$0 = a^T \mathbf{R}_\phi a = a^T E\{\phi\phi^T\}a = E\{a^T \phi\phi^T a\} = E\{(a^T \phi)^2\}.$$

Therefore,  $a^T \phi = 0$  with probability one. Then,

$$v_o = d - \mathbf{w}_o^T \phi = d - (\mathbf{w}_* + a)^T \phi = d - \mathbf{w}_*^T \phi,$$

with probability one. Therefore, although we cannot say that  $v = v_o$  always, we can say that they are equal with probability one. In other words, when  $\mathbf{R}_\phi$  is singular we may not be able to identify  $\mathbf{w}_*$ , but (with probability one) we are still able to separate  $d$  into its two components. More about this in Section 1.12.2.1.5.

### 1.12.2.1.3 Undermodeling

We just saw that the solution to the quadratic cost function (12.47) is indeed able to separate the two components of  $d$  when the regressor that appears in physical model (12.57) is the same  $\phi$  that we used to describe our class  $\mathcal{F}$ , that is, when our choice of  $\mathcal{F}$  includes the general solution from Box 4. Let us check now what happens when this is not the case.

Assume there is a relation

$$d = \mathbf{w}_*^T \phi_e + v, \quad (12.60)$$

in which  $v$  is orthogonal to  $\phi_e$ , but our class  $\mathcal{F}$  is defined through a regressor  $\phi$  that is a subset of the “correct” one,  $\phi_e$ . This situation is called *undermodeling*: our class is not rich enough to describe the true relation between  $\{x(n)\}$  and  $\{d(n)\}$ .

Assume then that

$$\phi_e = \begin{bmatrix} \phi \\ \theta \end{bmatrix}, \quad (12.61)$$

where  $\phi \in \mathcal{R}^M$ ,  $\theta \in \mathcal{R}^{K-M}$ , with  $K > M$ . The autocorrelation of  $\phi_e$  is

$$\mathbf{R}_{\phi_e} = E\{\phi_e \phi_e^T\} = \begin{bmatrix} \mathbf{R}_\phi & \mathbf{R}_{\phi\theta} \\ \mathbf{R}_{\theta\phi}^T & \mathbf{R}_\theta \end{bmatrix},$$

where  $\mathbf{R}_{\phi\theta} = E\{\phi\theta^T\}$ . From our hypothesis that  $\phi_e$  is orthogonal to  $v$ , that is,

$$\mathbf{0} = E\{v\phi_e\} = \begin{bmatrix} E\{v\phi\} \\ E\{v\theta\} \end{bmatrix},$$

we conclude that  $\phi$  is also orthogonal to  $v$ , so

$$\mathbf{r}_{d\phi} = E\{\phi(\phi_e^T w_* + v)\} = E\{\phi\phi_e^T\}w_* + \underbrace{E\{\phi v\}}_{=0} = [\mathbf{R}_\phi \ \mathbf{R}_{\phi\theta}]w_*.$$

Assuming that  $\mathbf{R}_\phi$  is nonsingular, solving the normal Eq. (12.50) we obtain

$$\mathbf{w}_o = \mathbf{R}_\phi^{-1} [\mathbf{R}_\phi \ \mathbf{R}_{\phi\theta}] w_* = [I \ \mathbf{R}_\phi^{-1} \mathbf{R}_{\phi\theta}] w_*. \quad (12.62)$$

Now we have two interesting special cases: first, if  $\phi$  and  $\theta$  are orthogonal, i.e., if  $\mathbf{R}_{\phi\theta} = \mathbf{0}$ , then  $\mathbf{w}_o$  contains the first  $M$  elements of  $w_*$ . Let us consider a specific example: assume that

$$\phi = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-M+1) \end{bmatrix}, \quad \text{but} \quad \phi_e = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-K+1) \end{bmatrix}, \quad (12.63)$$

with  $M < K$ . Then  $\theta = [x(n-M) \dots x(n-K+1)]^T$ . This situation is very common in practice.

In this case,  $\mathbf{R}_{\phi\theta} = \mathbf{0}$  when  $\{x(n)\}$  is zero-mean white noise. This is one reason why white noise is the preferred input to be used in system identification: even in the (very likely in practice) case in which  $M < K$ , at least the first elements of  $w_*$  are identified without bias.

On the other hand, if  $\mathbf{R}_{\phi\theta} \neq \mathbf{0}$ , then  $\mathbf{w}_o$  is a mixture of the elements of  $w_*$ . This happens in (12.63) when the input sequence  $\{x(n)\}$  is not white. In this case, the optimum filter  $\mathbf{w}_o$  takes advantage of the correlation between the entries of  $\phi$  and  $\theta$  to estimate  $\theta$  given  $\phi$ , and uses these estimated values to obtain a better approximation for  $d$ . Sure enough, if you write down the problem of approximating  $\theta$  linearly from  $\phi$ , namely,

$$\mathbf{W}_o = \arg \min_{\mathbf{W}} E \left\{ \left\| \theta - \mathbf{W}^T \phi \right\|^2 \right\},$$

you will see that the solution is exactly  $\mathbf{W}_o = \mathbf{R}_\phi^{-1} \mathbf{R}_{\phi\theta}$ .

What is important to notice is that in both cases the optimum error  $v_o$  is not equal to  $v$ :

$$v_o = d - \mathbf{w}_o^T \boldsymbol{\phi} = \mathbf{w}_*^T \boldsymbol{\phi}_e + v - \mathbf{w}_o^T \boldsymbol{\phi}.$$

Partitioning  $\mathbf{w}_* = [\mathbf{w}_{*,1}^T \mathbf{w}_{*,2}^T]^T$ , with  $\mathbf{w}_{*,1} \in \mathcal{R}^M$ ,  $\mathbf{w}_{*,2} \in \mathcal{R}^{K-M}$ , we have

$$\begin{aligned} v_o &= (\mathbf{w}_{*,1} - \mathbf{w}_o)^T \boldsymbol{\phi} + \mathbf{w}_{*,2}^T \boldsymbol{\theta} + v = \mathbf{w}_{*,2}^T \mathbf{R}_{\boldsymbol{\phi}\boldsymbol{\theta}}^T \mathbf{R}_{\boldsymbol{\phi}}^{-1} \boldsymbol{\phi} + \mathbf{w}_{*,2}^T \boldsymbol{\theta} + v \\ &= \mathbf{w}_{*,2}^T (\boldsymbol{\theta} - \mathbf{R}_{\boldsymbol{\phi}\boldsymbol{\theta}}^T \mathbf{R}_{\boldsymbol{\phi}}^{-1} \boldsymbol{\phi}) + v. \end{aligned}$$

Although  $v_o \neq v$ , you may check that indeed  $E\{v_o \boldsymbol{\phi}\} = \mathbf{0}$ .

As another example, assume that we have two variables such that  $d = ax + bx^3 + v$ , in which  $v$  has zero mean and is independent of  $x$ , and  $a$  and  $b$  are constants. Assume that we choose as regressor simply  $\boldsymbol{\phi} = x$ , so we try to solve

$$w_o = \arg \min_w E \left\{ (d - wx)^2 \right\}.$$

In this case we have

$$R_{\boldsymbol{\phi}} = E\{x^2\}, \quad r_{d\boldsymbol{\phi}} = E\{xd\} = E\{ax^2 + bx^4\},$$

since  $E\{x^k v\} = E\{x^k\}E\{v\} = 0$ . The optimum solution is thus

$$w_o = \frac{aE\{x^2\} + bE\{x^4\}}{E\{x^2\}},$$

and the optimum error is

$$v_o = d - w_o x = ax + bx^3 + v - \frac{aE\{x^2\} + bE\{x^4\}}{E\{x^2\}} x = v + b \left( x^2 - \frac{E\{x^4\}}{E\{x^2\}} \right) x \neq v.$$

However,  $v_o$  is again orthogonal to  $x$  (but not to  $x^3$ , which we did not include in the regressor).

What happens in the opposite situation, when we include more entries in  $\boldsymbol{\phi}$  than necessary? As one would expect, in this case the parameters related to these unnecessary entries will be zeroed in the optimum solution  $\mathbf{w}_o$ . This does not mean, however, that we should hasten to increase  $\boldsymbol{\phi}$  to make  $\mathcal{F}$  as large as possible, and leave to the filter the task of zeroing whatever was not necessary. This is because increasing the number of parameters to be estimated has a cost. The first and more obvious cost is in terms of memory and number of computations. However, we saw in Section 1.12.1.2 that an actual adaptive filter usually does not converge to the exact optimum solution, but rather hovers around it. Because of this, increasing the number of parameters may in fact decrease the quality of the solution. We explain this in more detail in Section 1.12.4.

#### 1.12.2.1.4 Zero and non-zero mean variables

Up to now we did not assume anything about the means of  $d$  and  $\boldsymbol{\phi}$ . If both  $d$  and  $\boldsymbol{\phi}$  have zero mean, we can interpret all autocorrelations as variances or covariance matrices, according to the case. To see this, assume that

$$E\{d\} = 0, \quad E\{\boldsymbol{\phi}\} = \mathbf{0}. \tag{12.64}$$

Then  $E\{\hat{y}\} = E\{\mathbf{w}_o^T \boldsymbol{\phi}\} = 0$ , and thus

$$E\{v_o\} = E\{d\} - \mathbf{w}_o^T E\{\boldsymbol{\phi}\} = 0 + \mathbf{w}_o^T \mathbf{0} = 0,$$

so the optimum error has zero mean. In this case,  $r_d = E\{d^2\} = \sigma_d^2$  is the variance of  $d$ , and  $E\{v_o^2\} = \sigma_{v,o}^2$  is the variance of  $v_o$ , so (12.55) becomes

$$J_{\min} = \sigma_{v,o}^2 = E\{v_o^2\} = \sigma_d^2 - \mathbf{r}_{d\boldsymbol{\phi}}^T \mathbf{R}_{\boldsymbol{\phi}}^{-1} \mathbf{r}_{d\boldsymbol{\phi}}. \quad (12.65)$$

Since  $\mathbf{R}_{\boldsymbol{\phi}}$  is positive-definite, we conclude that the uncertainty in  $v_o$  (its variance) is never larger than the uncertainty in  $d$  (Box 3).

However,  $v_o$  may have a nonzero mean if either  $d$  or  $\boldsymbol{\phi}$  has nonzero mean. This is verified as follows. Define

$$\bar{d} = E\{d\}, \quad \bar{\boldsymbol{\phi}} = E\{\boldsymbol{\phi}\}, \quad (12.66)$$

then

$$E\{v_o\} = E\{d - \mathbf{r}_{d\boldsymbol{\phi}}^T \mathbf{R}_{\boldsymbol{\phi}}^{-1} \boldsymbol{\phi}\} = \bar{d} - \mathbf{r}_{d\boldsymbol{\phi}}^T \mathbf{R}_{\boldsymbol{\phi}}^{-1} \bar{\boldsymbol{\phi}} \neq 0$$

in general (for example, if  $\bar{\boldsymbol{\phi}} = \mathbf{0}$ , but  $\bar{d} \neq 0$ ).

In many applications,  $v_o$  should approximate some signal that is constrained to have zero mean, such as speech or noise, or because  $v_o$  will be passed through a system with a high DC gain (such as an integrator), so it may be useful to guarantee that  $v_o$  has zero mean. How can this be done if the means of  $d$  and  $\boldsymbol{\phi}$  are not zero? This problem is quite common, particularly when  $\boldsymbol{\phi}$  includes nonlinear functions of  $x(n)$ . Fortunately, there are two easy solutions: First, if one knew the means of  $d$  and  $\boldsymbol{\phi}$ , one could define zero-mean variables

$$d_c = d - E\{d\}, \quad \boldsymbol{\phi}_c = \boldsymbol{\phi} - E\{\boldsymbol{\phi}\}, \quad (12.67)$$

and use them instead of the original variables in (12.47). This is the simplest solution, but we often do not know the means  $\bar{d}$  and  $\bar{\boldsymbol{\phi}}$ .

The second solution is used when the means are not known: simply add an extra term to  $\boldsymbol{\phi}$ , defining an extended regressor (not to be confused with the extended regressor from Section 1.12.2.1.3)

$$\boldsymbol{\phi}_e = \begin{bmatrix} 1 \\ \boldsymbol{\phi} \end{bmatrix}, \quad (12.68)$$

and an extended weight vector  $\mathbf{w}_e \in \mathcal{R}^{M+1}$ . The first coefficient of  $\mathbf{w}_e$  will take care of the means, as we show next. The extended cross-correlation vector and autocorrelation matrix are

$$\mathbf{r}_{d\boldsymbol{\phi}_e} = \begin{bmatrix} \bar{d} \\ \mathbf{r}_{d\boldsymbol{\phi}} \end{bmatrix}, \quad \mathbf{R}_{\boldsymbol{\phi}_e} = \begin{bmatrix} 1 & \bar{\boldsymbol{\phi}}^T \\ \bar{\boldsymbol{\phi}} & \mathbf{R}_{\boldsymbol{\phi}} \end{bmatrix}.$$

Assuming that  $\mathbf{R}_{\phi_e}$  and  $\mathbf{R}_\phi$  are nonsingular, we can compute  $\mathbf{w}_{o,e}$  if we evaluate the inverse of  $\mathbf{R}_{\phi_e}$  using Schur complements (See Fact 3 in Box 3), as follows

$$\begin{aligned}\mathbf{w}_{o,e} &= \mathbf{R}_{\phi_e}^{-1} \mathbf{r}_{d\phi_e} \\ &= \begin{bmatrix} 1 & \mathbf{0}^T \\ -\mathbf{R}_\phi^{-1} \bar{\phi} & \mathbf{I} \end{bmatrix} \begin{bmatrix} (1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi})^{-1} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{R}_\phi^{-1} \end{bmatrix} \begin{bmatrix} 1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \bar{d} \\ \mathbf{r}_{d\phi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mathbf{0}^T \\ -\mathbf{R}_\phi^{-1} \bar{\phi} & \mathbf{I} \end{bmatrix} \begin{bmatrix} (1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi})^{-1} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{R}_\phi^{-1} \end{bmatrix} \begin{bmatrix} \bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi} \\ \mathbf{r}_{d\phi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mathbf{0}^T \\ -\mathbf{R}_\phi^{-1} \bar{\phi} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} \\ \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi} \end{bmatrix} = \begin{bmatrix} \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} \\ \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi} - \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} \mathbf{R}_\phi^{-1} \bar{\phi} \end{bmatrix}.\end{aligned}$$

Using this result, we can evaluate now  $E\{v_{o,e}\} = E\{d\} - \mathbf{w}_{o,e}^T E\{\phi_e\}$  as follows (we used the fact that the inverse of a symmetric matrix is also symmetric from Box 3):

$$\begin{aligned}E\{v_{o,e}\} &= \bar{d} - \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} - \mathbf{r}_{d\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi} + \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi} \\ &= \bar{d} - \mathbf{r}_{d\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi} - \frac{\bar{d} - \bar{\phi}^T \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}}{1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}} (1 - \bar{\phi}^T \mathbf{R}_\phi^{-1} \bar{\phi}) = 0.\end{aligned}$$

### 1.12.2.1.5 Sufficiently rich signals

We saw in Section 1.12.2.1.2 that even when  $\mathbf{R}_\phi$  is singular, we can recover the optimum  $v_o$  (with probability one), but not the optimum  $\mathbf{w}_*$ . Let us study this problem in more detail, since it is important for system identification. So, what does it mean in practice to have a singular autocorrelation function  $\mathbf{R}_\phi$ ? The answer to this question can be quite complicated when the input signals are nonstationary (i.e., when  $\mathbf{R}_\phi$  is not constant), so in this case we refer the interested reader to texts in adaptive control (such as [18]), which treat this problem in detail (although usually from a deterministic point of view). We consider here only the case in which the signals  $\{d(n), x(n)\}$  are jointly wide-sense stationary.

In this case,  $\mathbf{R}_\phi$  may be singular because some entries of  $\phi$  are linear combinations of the others. For example, if we choose  $\phi_0 = x(n)$  and  $\phi_1 = 2x(n)$ , then

$$\mathbf{R}_\phi = E\{x^2(n)\} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix},$$

which is singular.

However, there are more subtle ways in which  $\mathbf{R}_\phi$  may become singular. Assume for example that our input sequence is given by

$$x(n) = \cos(\omega_0 n + \varphi), \quad (12.69)$$

where  $0 < \omega_0 \leq \pi$  is the frequency, and  $\varphi$  is a random phase, uniformly distributed in the interval  $[0, 2\pi)$ . This kind of model (sinusoidal signals with random phases), by the way, is the bridge between the periodic signals from Section 1.12.1.2 and the stochastic models discussed in this section. In this case, consider a vector  $\phi$  formed from a delay line with  $x(n)$  as input,

$$\phi = [x(n) \ x(n-1) \ \dots \ x(n-M+1)]^T.$$

If  $M = 2$ , we have

$$\mathbf{R}_\phi = \begin{bmatrix} E\{x^2(n)\} & E\{x(n)x(n-1)\} \\ E\{x(n)x(n-1)\} & E\{x^2(n-1)\} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & \cos(\omega_0) \\ \cos(\omega_0) & 1 \end{bmatrix}.$$

The determinant of  $\mathbf{R}_\phi$  is  $1 - \cos^2(\omega_0) = \sin^2(\omega_0) \neq 0$  if  $\omega_0 \neq 0$  and  $\omega_0 \neq \pi$ .

However, for any  $M \geq 3$ , the resulting  $\mathbf{R}_\phi$  will be singular: we can write

$$\phi = \frac{1}{2} e^{j(\omega_0 n + \varphi)} \begin{bmatrix} 1 \\ e^{-j\omega_0} \\ \vdots \\ e^{-j(M-1)\omega_0} \end{bmatrix} + \frac{1}{2} e^{-j(\omega_0 n + \varphi)} \begin{bmatrix} 1 \\ e^{j\omega_0} \\ \vdots \\ e^{j(M-1)\omega_0} \end{bmatrix},$$

and since  $E\{(e^{j\omega_0 n + j\varphi})^2\} = E\{e^{j2(\omega_0 n + \varphi)}\} = 0$ ,  $E\{e^{j(\omega_0 n + \varphi)}e^{-j(\omega_0 n + \varphi)}\} = 1$ , we have

$$\begin{aligned} \mathbf{R}_\phi &= \frac{1}{4} \begin{bmatrix} 1 \\ e^{-j\omega_0} \\ \vdots \\ e^{-j(M-1)\omega_0} \end{bmatrix} \begin{bmatrix} 1 & e^{j\omega_0} & \dots & e^{j(M-1)\omega_0} \end{bmatrix} \\ &\quad + \frac{1}{4} \begin{bmatrix} 1 \\ e^{j\omega_0} \\ \vdots \\ e^{j(M-1)\omega_0} \end{bmatrix} \begin{bmatrix} 1 & e^{-j\omega_0} & \dots & e^{-j(M-1)\omega_0} \end{bmatrix}. \end{aligned}$$

Since  $\mathbf{R}_\phi$  is the sum of two rank-one matrices, its rank is at most two. It will be two when  $0 < \omega_0 < \pi$ , since in this case the vectors are linearly independent. We conclude that when the input signal is a sinusoid as in (12.69), the optimum solution to (12.47) will never be able to identify more than two coefficients. This argument can be generalized: if  $x(n)$  is composed of  $K$  sinusoids of different frequencies, then  $\mathbf{R}_\phi$  will be nonsingular if and only if  $M \leq 2K$ .

The general conclusion is that  $\mathbf{R}_\phi$  will be singular or not, depending on which functions are chosen as inputs, and on the input signal. We say that the input is *sufficiently rich* for a given problem if it is such that  $\mathbf{R}_\phi$  is nonsingular. We show a few examples next.

### 1.12.2.1.6 Examples

The concepts we saw in Sections 1.12.2.1.2–1.12.2.1.5 can be understood intuitively if we return to our example in Section 1.12.1.2. In that example, the true echo was obtained by filtering the input  $x(n)$  by

an unknown  $H(z)$ , so we had

$$d(n) = \sum_{k=0}^K h_k x(n-k) + v(n),$$

where  $K$  is the true order of the echo path,  $v(n) = B \cos(\omega_1 n + \theta)$  and  $x(n) = A \cos(\omega_0 n + \varphi)$ . We considered only one harmonic for simplicity.

We are trying to cancel the echo, by constructing an approximated transfer function  $\hat{H}(z)$ , which is modeled as an FIR filter with  $M$  coefficients. As we saw in Section 1.12.1.2, if  $x(n)$  is a simple sinusoid, the echo will be completely canceled if we satisfy (12.8), reproduced below

$$\operatorname{Re}\{\hat{H}(e^{j\omega_0})\} = \operatorname{Re}\{H(e^{j\omega_0})\}, \quad \operatorname{Im}\{\hat{H}(e^{j\omega_0})\} = \operatorname{Im}\{H(e^{j\omega_0})\}.$$

If  $K = M = 1$ , so  $H(z) = h_0 + h_1 z^{-1}$  and  $\hat{H}(z) = w_0 + w_1 z^{-1}$ , this condition becomes

$$\begin{aligned} w_0 + w_1 \cos(-\omega_0) &= h_0 + h_1 \cos(-\omega_0), \\ w_1 \sin(-\omega_0) &= h_1 \sin(-\omega_0), \end{aligned}$$

with a single solution  $w_0 = h_0$ ,  $w_1 = h_1$ .

On the other hand, if the input were as before (a single sinusoid), but  $K = 2$ , so  $H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2}$ , the equations would be

$$\begin{aligned} w_0 + w_1 \cos(-\omega_0) &= h_0 + h_1 \cos(-\omega_0) + h_2 \cos(-2\omega_0), \\ w_1 \sin(-\omega_0) &= h_1 \sin(-\omega_0) + h_2 \sin(-2\omega_0), \end{aligned}$$

and the solution would be unique (since  $\mathbf{R}_\phi$  is nonsingular), but would not approximate  $h_0$  and  $h_1$  in general.

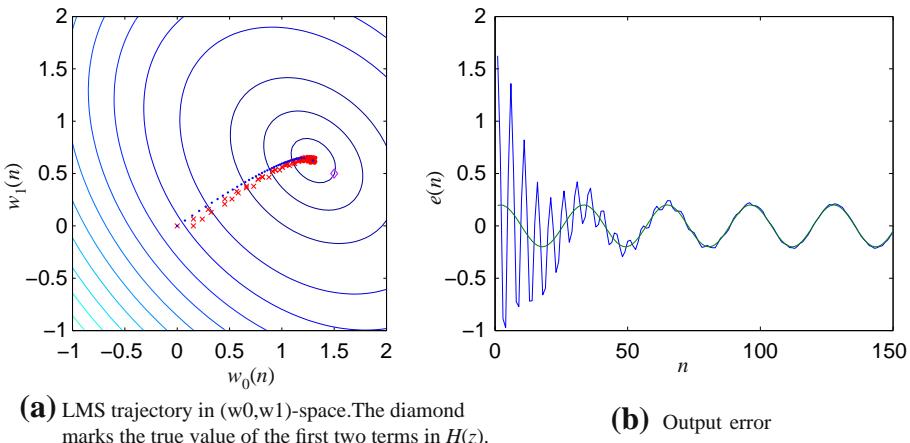
This is an example of an undermodeled system, as we saw in Section 1.12.2.1.3. Figure 12.20 shows the LMS algorithm derived in Section 1.12.1.2 applied to this problem. Note that, even though  $w_0(n)$ ,  $w_1(n)$  do not converge to  $h_0$ ,  $h_1$ , in this example the error  $e(n)$  does approximate well  $v(n)$ . This happens because in this case  $x(n)$  is periodic, and the term  $x(n-2)$  can be obtained exactly as a linear combination of  $x(n)$  and  $x(n-1)$  (Check that using the expressions for  $v_0$  from Section 1.12.2.1.3, we obtain  $v_0 = v$  in this case.)

If we tried to identify  $H(z)$  by adding an extra coefficient to  $\hat{H}(z)$ , the equations would be

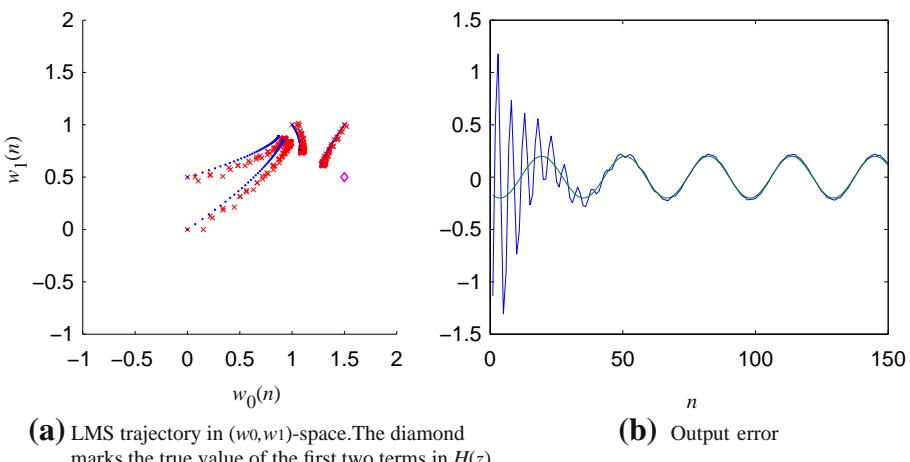
$$\begin{aligned} w_0 + w_1 \cos(-\omega_0) + w_2 \cos(-2\omega_0) &= h_0 + h_1 \cos(-\omega_0) + h_2 \cos(-2\omega_0), \\ w_1 \sin(-\omega_0) + w_2 \sin(-2\omega_0) &= h_1 \sin(-\omega_0) + h_2 \sin(-2\omega_0). \end{aligned} \tag{12.70}$$

Now  $w_0 = h_0$ ,  $w_1 = h_1$ , and  $w_2 = h_2$  is a solution, but not the only one. Depending on the initial condition, the filter would converge to a different solution, as can be seen in Figure 12.21 (the level sets are not shown, because they are not closed curves in this case). The input is not rich enough (in harmonics) to excite the unknown system  $H(z)$  so that the adaptive filter might identify it, as we saw in Section 1.12.2.1.5. However, for all solutions the error converges to  $v(n)$ .

If the input had a second harmonic ( $K_0 = 2$ ), we would add two more equations to (12.70), corresponding to  $\hat{H}(e^{2j\omega_0}) = H(e^{2j\omega_0})$ , and the filter would be able to identify systems with up to four coefficients.

**FIGURE 12.20**

LMS in undermodeled case.

**FIGURE 12.21**

LMS under non-sufficiently rich input.

The important message is that the adaptive filter only “sees” the error  $d(n) - \hat{y}(n)$ . If a mismatch between the true echo path  $H(z)$  and the estimated one  $\hat{H}(z)$  is not observable through looking only at the error signal, the filter will not converge to  $H(z)$ . Whether a mismatch is observable or not through the error depends on both  $H(z)$  and the input signal being used: the input must excite a large enough

number of frequencies so that  $H(z)$  can be estimated correctly. For this reason, in system identification a good input would be white noise.

### 1.12.2.2 Complex variables and multi-channel filtering

Adaptive filters used in important applications such as communications and radar have complex variables as input signals. In this section, we extend the results of Section 1.12.2.1 to consider complex signals. We start with the general solution, then restrict the result to the more usual case of circular signals.

We first show that the general solution for complex signals is equivalent to an adaptive filter with two inputs and two outputs: Indeed, if all input variables are complex, then the error  $e(n) = d(n) - \hat{y}(n)$  will also be complex, that is,  $e(n) = e_r(n) + j e_i(n)$ , where  $e_r(n), e_i(n) \in \mathcal{R}$  are the real and imaginary parts of  $e(n)$ . We must then minimize  $E\{|e(n)|^2\} = E\{e_r^2(n)\} + E\{e_i^2(n)\}$ , the total power of the complex signal  $e(n)$ . The quadratic cost function (12.45) must be changed to

$$J(\mathbf{w}) = E\{|e(n)|^2\} = E\{e(n)e^*(n)\} = E\{e_r^2(n)\} + E\{e_i^2(n)\},$$

where  $e^*(n)$  is the complex conjugate of  $e(n)$ . Now, we need to be careful about what exactly is the coefficient vector  $\mathbf{w}$  in this case: If the regressor is  $\phi = \phi_r + j\phi_i$  and  $d = d_r + j d_i$ , in principle  $\hat{y} = \hat{y}_r + j\hat{y}_i$  should be such that both its real and imaginary parts depend on both the real and imaginary parts of  $\phi$ , that is,

$$\begin{aligned}\hat{y}_r &= \mathbf{w}_{rr}^T \phi_r + \mathbf{w}_{ri}^T \phi_i, \\ \hat{y}_i &= \mathbf{w}_{ir}^T \phi_r + \mathbf{w}_{ii}^T \phi_i.\end{aligned}\quad (12.71)$$

Note that there is no *a priori* reason for us to imagine that there should be a special relation between the pairs  $(\mathbf{w}_{rr}, \mathbf{w}_{ri})$  and  $(\mathbf{w}_{ir}, \mathbf{w}_{ii})$ . That is, we are dealing with two different coefficient vectors and one extended regressor:

$$\mathbf{w}_r = \begin{bmatrix} \mathbf{w}_{rr} \\ \mathbf{w}_{ri} \end{bmatrix}, \quad \mathbf{w}_i = \begin{bmatrix} \mathbf{w}_{ir} \\ \mathbf{w}_{ii} \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \phi_r \\ \phi_i \end{bmatrix}. \quad (12.72)$$

Let us proceed for the time being separating the real and imaginary channels of our filter. Later, in Section 1.12.2.2.1, we return to complex algebra. With these definitions, we can write

$$\hat{y}_r = \mathbf{w}_r^T \boldsymbol{\theta}, \quad \hat{y}_i = \mathbf{w}_i^T \boldsymbol{\theta}. \quad (12.73)$$

The cost function then becomes

$$\begin{aligned}J(\mathbf{w}_r, \mathbf{w}_i) &= E\{(d_r - \mathbf{w}_r^T \boldsymbol{\theta})^2\} + E\{(d_i - \mathbf{w}_i^T \boldsymbol{\theta})^2\} \\ &= E\{d_r^2 - 2\mathbf{w}_r^T \boldsymbol{\theta} d_r + \mathbf{w}_r^T \boldsymbol{\theta} \boldsymbol{\theta}^T \mathbf{w}_r\} + E\{d_i^2 - 2\mathbf{w}_i^T \boldsymbol{\theta} d_i + \mathbf{w}_i^T \boldsymbol{\theta} \boldsymbol{\theta}^T \mathbf{w}_i\} \\ &= r_{d_r} - 2\mathbf{w}_r^T \mathbf{r}_{d_r \boldsymbol{\theta}} + \mathbf{w}_r^T \mathbf{R}_{\boldsymbol{\theta}} \mathbf{w}_r + r_{d_i} - 2\mathbf{w}_i^T \mathbf{r}_{d_i \boldsymbol{\theta}} + \mathbf{w}_i^T \mathbf{R}_{\boldsymbol{\theta}} \mathbf{w}_i,\end{aligned}\quad (12.74)$$

where the autocorrelation matrix and cross-correlation vectors are given by

$$\mathbf{R}_{\boldsymbol{\theta}} = \begin{bmatrix} \mathbf{R}_r & \mathbf{R}_{ri} \\ \mathbf{R}_{ir} & \mathbf{R}_i \end{bmatrix}, \quad \mathbf{r}_{d_r \boldsymbol{\theta}} = \begin{bmatrix} \mathbf{r}_{rr} \\ \mathbf{r}_{ri} \end{bmatrix}, \quad \mathbf{r}_{d_i \boldsymbol{\theta}} = \begin{bmatrix} \mathbf{r}_{ir} \\ \mathbf{r}_{ii} \end{bmatrix}, \quad (12.75)$$

where we simplified the notation, writing  $\mathbf{R}_r$  instead of  $\mathbf{R}_{\phi_r}$ ,  $\mathbf{R}_{ri}$  instead of  $\mathbf{R}_{\phi_r \phi_i}$ ,  $\mathbf{r}_{rr}$  instead of  $\mathbf{r}_{d_r \phi_r}$ , and so on.

Differentiating (12.74) to find the minimum, we obtain

$$\frac{\partial J}{\partial \mathbf{w}_r^T} = -2\mathbf{r}_{d_r \theta} + 2\mathbf{R}_\theta \mathbf{w}_r, \quad \frac{\partial J}{\partial \mathbf{w}_i^T} = -2\mathbf{r}_{d_i \theta} + 2\mathbf{R}_\theta \mathbf{w}_i. \quad (12.76)$$

Therefore, the optimum filters satisfy

$$\begin{bmatrix} \mathbf{R}_r & \mathbf{R}_{ri} \\ \mathbf{R}_{ir} & \mathbf{R}_i \end{bmatrix} \begin{bmatrix} \mathbf{w}_{rr,o} \\ \mathbf{w}_{ri,o} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{rr} \\ \mathbf{r}_{ri} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{R}_r & \mathbf{R}_{ri} \\ \mathbf{R}_{ir} & \mathbf{R}_i \end{bmatrix} \begin{bmatrix} \mathbf{w}_{ir,o} \\ \mathbf{w}_{ii,o} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{ir} \\ \mathbf{r}_{ii} \end{bmatrix}. \quad (12.77)$$

The optimum error is then

$$v_{r,o} = d_r - [\mathbf{w}_{rr,o}^T \mathbf{w}_{ri,o}^T] \boldsymbol{\theta}, \quad v_{i,o} = d_i - [\mathbf{w}_{ir,o}^T \mathbf{w}_{ii,o}^T] \boldsymbol{\theta}. \quad (12.78)$$

There is also an orthogonality condition for this case: using (12.77), we obtain

$$\mathbb{E}\{v_{r,o} \boldsymbol{\theta}\} = \mathbb{E}\{d_r \boldsymbol{\theta}\} - \mathbb{E}\{\boldsymbol{\theta} \boldsymbol{\theta}^T\} \begin{bmatrix} \mathbf{w}_{rr,o} \\ \mathbf{w}_{ri,o} \end{bmatrix} = \mathbf{0}, \quad (12.79a)$$

$$\mathbb{E}\{v_{i,o} \boldsymbol{\theta}\} = \mathbb{E}\{d_i \boldsymbol{\theta}\} - \mathbb{E}\{\boldsymbol{\theta} \boldsymbol{\theta}^T\} \begin{bmatrix} \mathbf{w}_{ir,o} \\ \mathbf{w}_{ii,o} \end{bmatrix} = \mathbf{0}. \quad (12.79b)$$

The value of the cost function at the minimum can be obtained using this result. As in the case of real variables,

$$\begin{aligned} \mathbb{E}\{v_{r,o}^2\} &= \mathbb{E}\{v_{r,o} (d_r - [\mathbf{w}_{rr,o}^T \mathbf{w}_{ri,o}^T] \boldsymbol{\theta})\} \\ &= \mathbb{E}\{v_{r,o} d_r\} = \mathbb{E}\{d_r^2\} - [\mathbf{w}_{rr,o}^T \mathbf{w}_{ri,o}^T] \mathbb{E}\{d_r \boldsymbol{\theta}\}, \end{aligned}$$

where we used (12.79a) in the second equality. Doing similarly for  $v_{i,o}$ , we obtain

$$r_{v_{r,o}} = \mathbb{E}\{v_{r,o}^2\} = r_{d_r} - [\mathbf{w}_{rr,o}^T \mathbf{w}_{ri,o}^T] \mathbf{r}_{d_r \theta}, \quad (12.80a)$$

$$r_{v_{i,o}} = \mathbb{E}\{v_{i,o}^2\} = r_{d_i} - [\mathbf{w}_{ir,o}^T \mathbf{w}_{ii,o}^T] \mathbf{r}_{d_i \theta}, \quad (12.80b)$$

$$J_{\min} = r_{v_{r,o}} + r_{v_{i,o}}. \quad (12.80c)$$

What we have just described is a two-input/two-output optimal filter. In the next section, we re-derive it using complex algebra. Note that multi-channel filters have applications that are unrelated to complex variables, particularly in control [18], stereo echo cancellation [19, 20], and active noise control [21–24].

### 1.12.2.2.1 Widely-linear complex least-mean squares

This general solution can be obtained in a more straightforward way if we use the complex gradients described in Box 6. Keeping the regressor  $\boldsymbol{\theta}$  as in (12.72) and defining the extended complex weight vector as  $\boldsymbol{\omega} = \mathbf{w}_r + j\mathbf{w}_i$ , we can write the cost function as

$$J(\boldsymbol{\omega}) = \mathbb{E}\{(d - \boldsymbol{\omega}^H \boldsymbol{\theta})(d - \boldsymbol{\omega}^H \boldsymbol{\theta})^*\} = r_d - \boldsymbol{\omega}^H \mathbf{r}_{d\theta} - \mathbf{r}_{d\theta}^H \boldsymbol{\omega} + \boldsymbol{\omega}^H \mathbf{R}_\theta \boldsymbol{\omega},$$

where we defined  $r_d = E\{|d|^2\}$  and  $\mathbf{r}_{d\theta} = E\{d^*\boldsymbol{\theta}\}$ . Note that  $(\cdot)^H$  represents the Hermitian transpose, that is, transpose followed by conjugation. Differentiating  $J(\boldsymbol{\omega})$  with respect to  $\boldsymbol{\omega}^H$ , as described in Box 6, we obtain

$$\frac{\partial J}{\partial \boldsymbol{\omega}^H} = -\mathbf{r}_{d\theta} + \mathbf{R}_\theta \boldsymbol{\omega}. \quad (12.81)$$

The normal equations then become

$$\mathbf{R}_\theta \boldsymbol{\omega}_0 = \mathbf{r}_{d\theta}. \quad (12.82)$$

Expanding the real and imaginary parts of (12.82), we recover (12.77). Similarly, the optimum error and the minimum value of the cost are

$$v_0 = d - \boldsymbol{\omega}_0^H \boldsymbol{\theta}, \quad J_{\min} = E\{|v_0|^2\} = E\{|d|^2\} - \boldsymbol{\omega}_0^H \mathbf{r}_{d\theta}. \quad (12.83)$$

Comparing with (12.80), we see that  $v_0 = v_{r,o} + j v_{i,o}$ , and the value of  $J_{\min}$  is the same as before. However, as you noticed, using a complex weight vector  $\boldsymbol{\omega}$  and the complex derivatives of Box 6, all expressions are much simpler to derive.

The filter using a complex weight vector  $\boldsymbol{\omega}$  and the real regressor  $\boldsymbol{\theta}$  constitutes a version of what is known as *widely-linear* complex filter. This version was recently proposed in [25] as a reduced-complexity alternative to the widely-linear complex filter originally proposed in [26, 27], which uses as regressor the extended vector

$$\boldsymbol{\phi}_e = \begin{bmatrix} \boldsymbol{\phi} \\ \boldsymbol{\phi}^* \end{bmatrix}, \quad (12.84)$$

composed of the original regressor  $\boldsymbol{\phi}$  and its complex conjugate  $\boldsymbol{\phi}^*$  (computational complexity is reduced because the number of operations required to compute  $\boldsymbol{\omega}^H \boldsymbol{\theta}$  is about half of what is necessary to evaluate  $\boldsymbol{\omega}^H \boldsymbol{\phi}_e$ ). Widely-linear estimation has been receiving much attention lately, due to new applications that have appeared [28–31].

But why the name *widely linear*? The reason for this name is that the complex regressor  $\boldsymbol{\phi}$  does not appear linearly in the expressions, but only by separating its real and imaginary parts (as in  $\boldsymbol{\theta}$ ) or by including its complex conjugate (as in  $\boldsymbol{\phi}_e$ ). Both these expressions, from the point of view of a complex variable, are nonlinear.

What would be a *linear* complex filter, then? This is the subject of the next section.

### 1.12.2.2 Linear complex least-mean squares

There are many applications in which the data and regressor are such that their real and imaginary parts are similarly distributed, so that

$$\mathbf{R}_r = \mathbf{R}_i, \quad r_{dr} = r_{di}, \quad \mathbf{R}_{ri} = -\mathbf{R}_{ir}, \quad \mathbf{r}_{rr} = \mathbf{r}_{ii}, \quad \mathbf{r}_{ri} = -\mathbf{r}_{ir}. \quad (12.85)$$

Under these conditions, we say that the pair  $(d, \boldsymbol{\phi})$  is *complex circular*, or simply *circular*.

In this case, we can rewrite (12.77) as

$$\begin{bmatrix} \mathbf{R}_r & \mathbf{R}_{ri} \\ -\mathbf{R}_{ri} & \mathbf{R}_r \end{bmatrix} \begin{bmatrix} \mathbf{w}_{rr} \\ \mathbf{w}_{ri} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{rr} \\ \mathbf{r}_{ri} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{R}_r & \mathbf{R}_{ri} \\ -\mathbf{R}_{ri} & \mathbf{R}_r \end{bmatrix} \begin{bmatrix} \mathbf{w}_{ir} \\ \mathbf{w}_{ii} \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_{ri} \\ \mathbf{r}_{rr} \end{bmatrix}.$$

Expand these equations as follows (notice the change in order in (12.86b))

$$\mathbf{R}_r \mathbf{w}_{rr} + \mathbf{R}_{ri} \mathbf{w}_{ri} = \mathbf{r}_{rr}, \quad -\mathbf{R}_{ri} \mathbf{w}_{rr} + \mathbf{R}_r \mathbf{w}_{ri} = \mathbf{r}_{ri}, \quad (12.86a)$$

$$\mathbf{R}_r \mathbf{w}_{ii} - \mathbf{R}_{ri} \mathbf{w}_{ir} = \mathbf{r}_{rr}, \quad \mathbf{R}_{ri} \mathbf{w}_{ii} + \mathbf{R}_r \mathbf{w}_{ir} = -\mathbf{r}_{ri}. \quad (12.86b)$$

Now we can see that, if  $(\mathbf{w}_{rr}, \mathbf{w}_{ri})$  is a solution to (12.86a), then

$$(\mathbf{w}_{ii} = \mathbf{w}_{rr}, \mathbf{w}_{ir} = -\mathbf{w}_{ri}) \quad (12.87)$$

is a solution to (12.86b). Therefore, when the input signals are circular, the number of degrees of freedom in the problem is actually smaller. In this case, there is a very nice way of recovering the solutions, working only with complex algebra. Indeed, defining  $\mathbf{w} = \mathbf{w}_{rr} + j\mathbf{w}_{ri}$ , we have

$$\begin{aligned} \mathbf{w}^H \boldsymbol{\phi} &= (\mathbf{w}_{rr} - j\mathbf{w}_{ri})^T (\boldsymbol{\phi}_r + j\boldsymbol{\phi}_i) \\ &= \left( \mathbf{w}_{rr}^T \boldsymbol{\phi}_r + \mathbf{w}_{ri}^T \boldsymbol{\phi}_i \right) + j \left( \mathbf{w}_{rr}^T \boldsymbol{\phi}_i - \mathbf{w}_{ri}^T \boldsymbol{\phi}_r \right), \end{aligned} \quad (12.88)$$

which is equivalent to (12.71) with the identities (12.87). Using this definition, it is possible to obtain the optimum solution (12.86a), (12.86b), and (12.87) using only complex algebra, following exactly the same steps as our derivation for the real case in the previous section.

This path is very useful, we can derive almost all results for real or complex circular adaptive filters using the same arguments. Let us see how it goes. First, define  $\hat{y} = \mathbf{w}^H \boldsymbol{\phi}$ , so our problem becomes

$$\min_{\mathbf{w} \in \mathcal{C}^M} E \left\{ |d - \mathbf{w}^H \boldsymbol{\phi}|^2 \right\}. \quad (12.89)$$

Expanding the cost, recalling that now  $|d - \mathbf{w}^H \boldsymbol{\phi}|^2 = (d - \mathbf{w}^H \boldsymbol{\phi})(d - \mathbf{w}^H \boldsymbol{\phi})^* = (d - \mathbf{w}^H \boldsymbol{\phi})(d^* - \boldsymbol{\phi}^H \mathbf{w})$ , we obtain

$$\begin{aligned} J(\mathbf{w}) &= E \left\{ |d|^2 - \mathbf{w}^H \boldsymbol{\phi} d^* - d \boldsymbol{\phi}^H \mathbf{w} + \mathbf{w}^H \boldsymbol{\phi} \boldsymbol{\phi}^H \mathbf{w} \right\} \\ &= r_d - \mathbf{w}^H \mathbf{r}_{d\boldsymbol{\phi}} - \mathbf{r}_{d\boldsymbol{\phi}}^H \mathbf{w} + \mathbf{w}^H \mathbf{R}_{\boldsymbol{\phi}} \mathbf{w}, \end{aligned} \quad (12.90)$$

where we defined  $r_d = E\{|d|^2\}$ ,  $\mathbf{r}_{d\boldsymbol{\phi}} = E\{d^* \boldsymbol{\phi}\}$ ,  $\mathbf{R}_{\boldsymbol{\phi}} = E\{\boldsymbol{\phi} \boldsymbol{\phi}^H\}$ .

We need to find the minimum of this cost. One easy solution is to use the rules for differentiation of real functions of complex variables, as described in Box 6. These rules are very useful, and surprisingly easy to use: basically, we can treat  $\mathbf{w}$  and its conjugate  $\mathbf{w}^H$  as if they were independent variables, that is

$$\frac{\partial J}{\partial \mathbf{w}^H} = -\mathbf{r}_{d\boldsymbol{\phi}} + \mathbf{R}_{\boldsymbol{\phi}} \mathbf{w}. \quad (12.91)$$

Equating the gradient to zero, the solution is (assuming that  $\mathbf{R}_{\boldsymbol{\phi}}$  is nonsingular)

$$\mathbf{w}_o = \mathbf{R}_{\boldsymbol{\phi}}^{-1} \mathbf{r}_{d\boldsymbol{\phi}}, \quad (12.92)$$

which is exactly equal to (12.51). This is the advantage of this approach: the expressions for complex and real problems are *almost* equal. The important differences are the conjugates that appear in the

definitions of  $\mathbf{R}_\phi$  and  $\mathbf{r}_{d\phi}$ , and the absence of a factor of 2 in the gradient (12.91). This follows from the definition of complex derivative we used (see Box 6). Here this difference is canceled out in the solution (12.92), but further on we will find situations in which there will be a different factor for the case of real and for the case of complex variables.

Let us check that (12.92) is really equivalent to (12.86a). Expanding  $\mathbf{R}_\phi$  and  $\mathbf{r}_{d\phi}$  and using the circularity conditions (12.85), we obtain

$$\mathbf{R}_\phi = \mathbb{E}\{\boldsymbol{\phi}\boldsymbol{\phi}^H\} = \mathbb{E}\{(\boldsymbol{\phi}_r + j\boldsymbol{\phi}_i)(\boldsymbol{\phi}_r - j\boldsymbol{\phi}_i)^T\} = \mathbf{R}_r + \mathbf{R}_i + j(\mathbf{R}_{ir} - \mathbf{R}_{ri}), \quad (12.93a)$$

$$= 2\mathbf{R}_r - 2j\mathbf{R}_{ri}, \quad (12.93b)$$

and

$$\begin{aligned} \mathbf{r}_{d\phi} &= \mathbb{E}\{d^*\boldsymbol{\phi}\} = \mathbb{E}\{(d_r - jd_i)(\boldsymbol{\phi}_r + j\boldsymbol{\phi}_i)\} = \mathbf{r}_{rr} + \mathbf{r}_{ii} + j(\mathbf{r}_{ri} - \mathbf{r}_{ir}) \\ &= 2\mathbf{r}_{rr} + 2jr_{ri}, \end{aligned} \quad (12.94)$$

so, equating the gradient (12.91) to zero,  $\mathbf{w}_o = \mathbf{w}_{r,o} + j\mathbf{w}_{i,o}$  must satisfy

$$\begin{aligned} 2(\mathbf{R}_r - j\mathbf{R}_{ri})(\mathbf{w}_{r,o} + j\mathbf{w}_{i,o}) &= 2(\mathbf{r}_{rr} + jr_{ri}), \\ \mathbf{R}_r\mathbf{w}_{r,o} + \mathbf{R}_{ri}\mathbf{w}_{i,o} + j(\mathbf{R}_r\mathbf{w}_{i,o} - \mathbf{R}_{ri}\mathbf{w}_{r,o}) &= \mathbf{r}_{rr} + jr_{ri}. \end{aligned} \quad (12.95)$$

Comparing the real and imaginary parts of (12.95) with (12.86a), we see that they indeed correspond to the same set of equations.

A few important remarks:

**Remark 5.** The orthogonality condition still holds in the complex circular case, but with a small difference: defining the optimum error as before,

$$v_o = d - \mathbf{w}_o^H \boldsymbol{\phi},$$

we obtain, noting that  $(\mathbf{w}_o^H \boldsymbol{\phi})^* = \boldsymbol{\phi}^H \mathbf{w}_o$ ,

$$\mathbb{E}\{v_o^* \boldsymbol{\phi}\} = \mathbb{E}\{d^* \boldsymbol{\phi} - \boldsymbol{\phi} \boldsymbol{\phi}^H \mathbf{w}_o\} = \mathbf{r}_{d\phi} - \mathbf{R}_\phi \mathbf{w}_o = \mathbf{0}. \quad (12.96)$$

**Remark 6.** The complex gradient (12.91) seems strange because it does not include the factors of 2 that would appear in the real case. This difference will appear in many expressions in the following sections, whenever we need to differentiate between the real and complex cases. However, the difference is illusory: if you go back to (12.93b) and (12.94), you will see that the factor of two is actually there, just hidden by the definition of autocorrelation and cross-correlation under the circularity conditions.

**Remark 7.** The optimum error power can be obtained as in the real case, and is equal to

$$\mathbb{E}\{|v_o|^2\} = r_d - \mathbf{w}_o^H \mathbf{r}_{d\phi} = r_d - \mathbf{r}_{d\phi}^H \mathbf{R}_\phi^{-1} \mathbf{r}_{d\phi}. \quad (12.97)$$

---

**Box 4: Least mean-squares estimation**


---

Here we consider the general problem of finding the optimum relation  $\hat{\mathcal{H}}_o$  between  $d(n)$  and  $x(n)$ , such that

$$\hat{\mathcal{H}}_o = \arg \min_{\hat{\mathcal{H}}} E \left\{ d(n) - \hat{\mathcal{H}}(x(n), x(n-1), \dots) \right\}, \quad (12.98)$$

without restricting  $\mathcal{H}$  to a particular class.

In order to understand well this problem, let us consider first the simpler case where the relation between  $x(n)$  and  $d(n)$  is memoryless, that is, our function depends only on the current value of  $x(n)$ . Dropping the time index to simplify the notation, our problem then becomes how to find a function  $\hat{\mathcal{H}}$  that, applied to a certain random variable  $x$ , approximates a certain random variable  $d$  so that the mean-square error is minimized, that is, we want to solve

$$\min_{\hat{\mathcal{H}}} E \{ [d - \hat{\mathcal{H}}(x)]^2 \}. \quad (12.99)$$

Assume that the joint probability density function  $p_{dx}(d, x)$  of  $d$  and  $x$  is known (we are assuming that  $d$  and  $x$  are continuous random variables, but the final result in (12.100) also holds if they are discrete). Then,

$$\begin{aligned} E \{ [d - \hat{\mathcal{H}}(x)]^2 \} &= \int \int [d - \hat{\mathcal{H}}(x)]^2 p_{dx}(d, x) d d dx \\ &= \int \underbrace{\left\{ \int [d - \hat{\mathcal{H}}(x)]^2 p_{d|x}(d|x) d d \right\}}_{E_d \{ [d - \hat{\mathcal{H}}(x)]^2 | x \}} p_x(x) dx \\ &= E_x \left\{ E_d \left\{ [d - \hat{\mathcal{H}}(x)]^2 | x \right\} \right\}, \end{aligned} \quad (12.100)$$

where we use the subscripts  $E_d$  and  $E_x$  to highlight that the expectations are taken with respect to  $d$  or  $x$  only.

Note that the inner integrand  $E_d \{ [d - \hat{\mathcal{H}}(x)]^2 | x \}$  in (12.100) is nonnegative. Therefore, if we minimize it for each value of  $x$ , we will obtain the minimum of the full expression. That is, the optimum  $\hat{\mathcal{H}}_o$  is the function defined by

$$\hat{\mathcal{H}}_o(x) = \arg \min_{\hat{\mathcal{H}}(x)} E_d \left\{ [d - \hat{\mathcal{H}}(x)]^2 | x \right\}. \quad (12.101)$$

It is important to remember that, fixed  $x$ ,  $\hat{\mathcal{H}}(x)$  is simply a number. This can be seen more easily expanding the expectations in (12.101)

$$\begin{aligned} E_d \left\{ [d - \hat{\mathcal{H}}(x)]^2 | x \right\} &= \int (d^2 - 2d\hat{\mathcal{H}}(x) + \hat{\mathcal{H}}^2(x)) p_{d|x}(d|x) dd \\ &= \int d^2 p_{d|x}(d|x) dd - 2\hat{\mathcal{H}}(x) \int dp_{d|x}(d|x) dd + \hat{\mathcal{H}}^2(x) \int p_{d|x}(d|x) dd \\ &= E_d \{ d^2 | x \} - 2E \{ d | x \} \hat{\mathcal{H}}(x) + \hat{\mathcal{H}}^2(x). \end{aligned}$$

Differentiating with respect to  $\hat{\mathcal{H}}(x)$  and equating to zero, we obtain the solution

$$\hat{\mathcal{H}}_o(x) = E_d\{d|x\} \stackrel{\Delta}{=} \hat{y}(n), \quad (12.102)$$

which is indeed a function of  $x$ , as desired.

Let us study the properties of this solution. First, define

$$v_o = d - \hat{\mathcal{H}}_o(x).$$

Note that  $v_o$  and  $\hat{\mathcal{H}}_o$  are not necessarily equal to  $v$  and  $\mathcal{H}$  from (12.40). Evaluating the average of  $\hat{\mathcal{H}}_o(x)$ , we obtain

$$E_x\{\hat{\mathcal{H}}_o(x)\} = E_x\{E_d\{d|x\}\} = E\{d\},$$

by an argument similar to that used to obtain (12.100). We conclude that  $v_o$  has zero mean. In addition, for any function  $f(x)$  of  $x$  for which the expected values exist, we have

$$E\{v_o f(x)\} = E_x\{E_d\{df(x)\}\} - E\{E_d\{d|x\}f(x)\} = 0 = E\{v_o\}E\{f(x)\}, \quad (12.103)$$

since  $E_d\{df(x)|x\} = f(x)E_d\{d|x\}$ . This result means that the error  $v_o$  is uncorrelated to any function of the reference data  $x$ , which is a very strong condition. It basically means that we have extracted all second-order information available in  $x$  about  $d$ .

We remark that (12.103) does not imply that  $x$  and  $v_o$  are independent. A simple counter-example is the following: consider two discrete-valued random variables  $r$  and  $s$  with joint probabilities given by Table 12.1. Then  $E\{sf(r)\} = 1/4 \cdot (-1) \cdot f(+1) + 1/4 \cdot (+1) \cdot f(+1) + 1/2 \cdot 0 \cdot f(-1) = 0, \forall f(\cdot)$ , but  $r$  and  $s$  are not independent.

In the more general case in which  $\hat{\mathcal{H}}$  is allowed to depend on previous samples of  $x(n)$ , the arguments and the final conclusion are similar:

$$\arg \min_{\hat{\mathcal{H}}} E \left\{ \left[ d - \hat{\mathcal{H}}(x(n), x(n-1), \dots) \right]^2 \right\} = E\{d | x(n), x(n-1), \dots\}, \quad (12.104)$$

and defining

$$v_o(n) = d(n) - E\{d | x(n), x(n-1), \dots\},$$

**Table 12.1** Counter-Example: (12.103) does not Imply Independence

$r \setminus s$	-1	0	1
-1	0	1/2	0
1	1/4	0	1/4

it holds that  $v_o(n)$  has zero mean and is uncorrelated with any function of  $x(n), x(n-1), \dots$ :

$$\mathbb{E}\{v_o(n)\} = 0, \quad \mathbb{E}\{v_o(n)f(x(n), x(n-1), \dots)\} = 0. \quad (12.105)$$

Note that, in general, the optimal solution of (12.104) depends on  $n$ , so we should write

$$\hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \dots) = \mathbb{E}\{d|x(n), x(n-1), \dots\}, \quad (12.106)$$

to make the time dependence explicit.

We can now return to the question at the start of this section: From our results thus far, we see that minimizing  $\mathbb{E}\{e^2(n)\}$  leads to a model which relates  $d(n)$  and the sequence  $\{x(n)\}$  such that

$$d(n) = \hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \dots) + v_o(n),$$

in which  $v_o(n)$  is uncorrelated with any function  $f(\cdot)$  of  $x(n), x(n-1), \dots$ . In other words, the solution of (12.104) imposes such a model on our data, even if this model makes no physical sense at all. This is an important point: such a model will always result from the minimization of the mean-square error, at least as long as the statistics of  $d(n)$  and  $\{x(n)\}$  are such that the expected values in (12.104) are finite.

Now, what if we knew beforehand, by physical arguments about the data, that a model such as (12.40) holds between  $d(n)$  and  $\{x(n)\}$ , such that  $v(n)$  has zero mean and is uncorrelated to any function of  $x(n), x(n-1), \dots$ ? Will it result that the solution of (12.104) will be such that

$$\hat{\mathcal{H}}_o(x(n), x(n-1), \dots) = \mathcal{H}(x(n), x(n-1), \dots), \quad v_o(n) = v(n)?$$

To answer this question, let us go back to (12.99), and use (12.40) to find the solution. We thus have (we omit the arguments of  $d(n), v(n)$  and  $\mathcal{H}(\cdot)$  for simplicity)

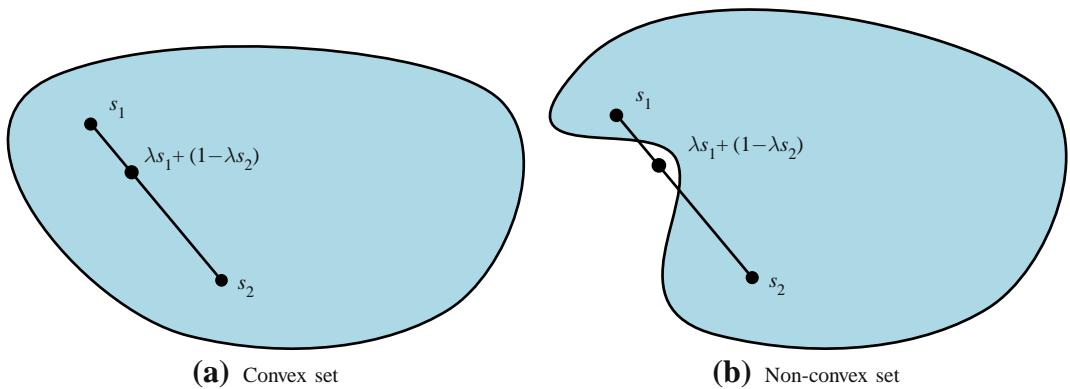
$$\min_{\hat{\mathcal{H}}} \mathbb{E}\{[d - \hat{\mathcal{H}}(x)]^2\} = \min_{\hat{\mathcal{H}}} \mathbb{E}\left\{[\mathcal{H}_n + v - \hat{\mathcal{H}}]^2\right\}.$$

Since  $v$  is orthogonal to any function of  $x(n), x(n-1), \dots$  by assumption, it will be orthogonal to  $\mathcal{H}_n$  and  $\hat{\mathcal{H}}$  in particular. Therefore, we can simplify the cost as

$$\min_{\hat{\mathcal{H}}} \mathbb{E}\{[d - \hat{\mathcal{H}}(x)]^2\} = \min_{\hat{\mathcal{H}}} \left[ \mathbb{E}\left\{(\mathcal{H} - \hat{\mathcal{H}})^2\right\} + \mathbb{E}\{v^2\} \right]. \quad (12.107)$$

Both terms in (12.107) are positive, and only the first depends on  $\hat{\mathcal{H}}$ . Therefore, the solution is indeed  $\hat{\mathcal{H}} = \mathcal{H}$  and  $v_o = v$ , as one would wish (to be precise,  $\mathcal{H}$  and  $\hat{\mathcal{H}}$  could differ on a set of probability zero).

We conclude that we can use (12.102) as a criterion for separating the two parts  $y(n)$  and  $v(n)$  of our model for  $d(n)$  if and only if  $v(n)$  satisfies (12.105). This holds, for example, if  $v(n)$  has zero mean and is independent of  $x(n-k)$  for all  $k \geq 0$ .

**FIGURE 12.22**

Convexity.

Note that since  $\hat{y}(n) = \hat{\mathcal{H}}_{o,n}(x(n), x(n-1), \dots)$  is itself a function of  $x(n), x(n-1), \dots$ , (12.105) implies that

$$\mathbb{E}\{v_o(n)\hat{y}(n)\} = 0,$$

and we have

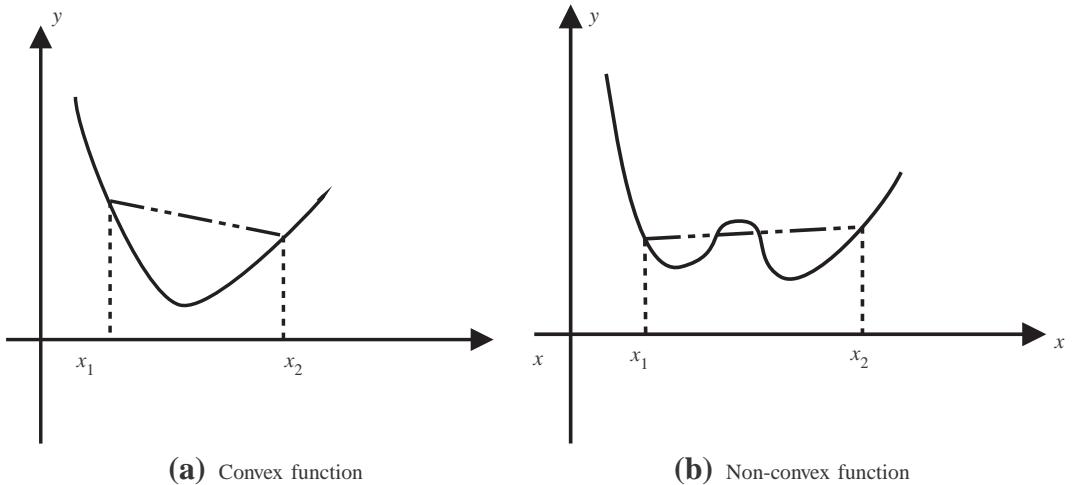
$$\begin{aligned} \mathbb{E}\{v_o^2(n)\} &= \mathbb{E}\{v_o[d(n) - \hat{y}(n)]\} = \mathbb{E}\{v_o d(n)\} \\ &= \mathbb{E}\{[d(n) - \hat{y}(n)]d(n)\} \\ &= \mathbb{E}\{d^2(n)\} - \mathbb{E}\{d(n)\hat{y}(n)\} \\ &= \mathbb{E}\{d^2(n)\} - \mathbb{E}\{[\hat{y}(n) + v_o(n)]\hat{y}(n)\} \\ &= \mathbb{E}\{d^2(n)\} - \mathbb{E}\{\hat{y}^2(n)\}. \end{aligned}$$

Recalling that  $\mathbb{E}\{\hat{y}(n)\} = \mathbb{E}\{d(n)\}$ , if we add and subtract  $\mathbb{E}^2\{d(n)\}$  to the last result, we conclude that

$$\mathbb{E}\{v_o^2(n)\} = \sigma_d^2 - \sigma_{\hat{y}}^2 \leq \sigma_d^2, \quad (12.108)$$

where  $\sigma_d^2$  and  $\sigma_{\hat{y}}^2$  are the variances of  $d(n)$  and of  $\hat{y}(n)$ .

The solution we just found is very general (it assumes little about  $\mathcal{H}$ ), and there are ways of computing it adaptively from the data, using for example algorithms known as particle filters [32]. The solution requires, however, that one builds approximations for the joint probability distribution of  $d(n)$  and  $x(n), x(n-1), \dots$ , which in general requires a large amount of data, resulting in relatively high complexity and relatively slow convergence. If you know more about the relation between  $\{x(n)\}$  and  $d(n)$ , this information might be used to constrain the search to a smaller class of problems, potentially leading to simpler algorithms with faster convergence. That is the goal of adaptive filters, and the reason why we restricted ourselves to this case in the main text.



**FIGURE 12.23**

## Convex functions.

## Box 5: Convex functions

Convex sets are such that lines between any two points in the set always stay entirely in the set (see Figure 12.22). More formally, a set  $\mathcal{S} \in \mathbb{R}^n$  is convex if

$$s_1, s_2 \in \mathcal{S} \Rightarrow \lambda s_1 + (1 - \lambda) s_2 \in \mathcal{S}, \quad \text{for all } \lambda \in [0, 1].$$

Convex functions, on the other hand, are functions  $f(\mathbf{x}) : \mathcal{R}^n \rightarrow \mathcal{R}$  such that for all  $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^n$  and  $0 < \lambda < 1$ ,

$$f(\lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1 - \lambda) f(\mathbf{x}_2).$$

The function is called *strictly convex* if the inequality is strict ( $<$  instead of  $\leq$ ). For example,  $f(x) = x^2$  is strictly convex. Figure 12.23 shows examples of convex and non-convex functions. An important property of strictly convex functions is that they always have a unique minimum, that is, they never have sub-optimal local minima.

### **Box 6: Wirtinger derivatives and minima of functions of a complex variable**

Suppose you want to minimize a function

$$f : \mathcal{C} \rightarrow \mathcal{R}, \\ w \in \mathcal{C} \mapsto f(w) \in \mathcal{R}, \quad (12.109)$$

that is, a function that takes a complex argument to a real value. The minimization problem is well defined, since the image of  $f$  is real. However, we cannot use standard arguments about derivatives and stationary points to find the minimum, because a function like this is never analytic, that is, the standard derivative used in complex analysis does not exist.

To see why, we need to think about  $f$  as a function of two variables,  $x$  and  $y$ , the real and imaginary parts of  $w$ . Take  $f(w) = |w|^2 = ww^*$  for example. We can write it as a function of  $x$  and  $y$  as

$$\hat{f}(x, y) = f(x + jy) = x^2 + y^2. \quad (12.110)$$

Since it is a function of two variables, the derivatives in principle depend on the direction we are moving along. Consider the derivative at a point  $w$  along the  $x$  axis, that is, keeping  $y$  constant:

$$\lim_{\epsilon \rightarrow 0} \frac{(x + \epsilon)^2 + y^2 - (x^2 + y^2)}{\epsilon} = 2x.$$

On the other hand, along the  $y$  axis, we have

$$\lim_{\epsilon \rightarrow 0} \frac{x^2 + (y + \epsilon)^2 - (x^2 + y^2)}{\epsilon} = 2y,$$

that is, the derivative at point  $w$  is different according to the direction we are moving. This is essentially what is meant by saying that the function is not analytic at  $w$ .

For contrast, consider now the analytic function  $g = w^2$ . Its derivative is the same, no matter which direction we take:

$$\begin{aligned} & \lim_{\epsilon \rightarrow 0} \frac{(x + \epsilon + jy)^2 - (x + jy)^2}{\epsilon} \\ &= \lim_{\epsilon \rightarrow 0} \frac{(x + \epsilon)^2 - y^2 + 2j(xy + \epsilon y) - (x^2 - y^2 + 2jxy)}{\epsilon} = 2w. \end{aligned}$$

Similarly, differentiating along the  $y$  axis we obtain again the same result

$$\lim_{\epsilon \rightarrow 0} \frac{(x + j(y + \epsilon))^2 - (x + jy)^2}{\epsilon} = 2w.$$

When the derivative at a certain point  $w_0$  is the same along every direction, we can define derivatives of complex functions writing simply

$$\frac{dg(w_0)}{dw} = \lim_{\Delta w \rightarrow 0} \frac{g(w_0 + \Delta w) - g(w_0)}{\Delta w}, \quad (12.111)$$

since we know the limit does not depend on the direction along which  $\Delta w$  goes to zero.

It can be shown that a function is analytic at a given point only if the *Cauchy-Riemann conditions* hold at that point. Separating a generic complex function  $g$  into its real and imaginary parts  $g(w) = u(x, y) + jv(x, y)$ , the Cauchy-Riemann conditions are [33]

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

In the case of a *real* function of a complex variable, such as  $f$  defined as in (12.109),  $v \equiv 0$ , and of course the conditions could hold only for points in which the real part has zero partial derivatives.

This means that we cannot search for the stationary points of a function such as (12.109) using the derivative defined as in (12.111). We could of course always go back and interpret the function as a function of two variables, as we did in (12.110). In order to find the minimum of  $f(w)$ , we would need to find the stationary points of  $\hat{f}(x, y)$

$$\frac{\partial \hat{f}}{\partial x} = 0, \quad \frac{\partial \hat{f}}{\partial y} = 0.$$

This works just fine, of course, but there is a nice trick, which allows us to keep working with complex variables in a simple way. The idea is to define derivatives differently, such that optimization problems become easy to solve, using what is known as *Wirtinger calculus*.

Real functions of complex variables are usually defined in terms of variables and its conjugates, since  $w + w^*$  and  $ww^*$  are both real. The idea is to define derivatives such that  $w$  and  $w^*$  can be treated as if they were independent variables, as follows. Despite the motivation, the new derivatives are defined for general functions  $g : \mathcal{C} \rightarrow \mathcal{C}$ .

Define partial derivatives of a function  $g : \mathcal{C} \rightarrow \mathcal{C}$  with respect to  $w$  and  $w^*$  as

$$\frac{\partial g}{\partial w} = \frac{1}{2} \left\{ \frac{\partial g}{\partial x} - j \frac{\partial g}{\partial y} \right\}, \quad \frac{\partial g}{\partial w^*} = \frac{1}{2} \left\{ \frac{\partial g}{\partial x} + j \frac{\partial g}{\partial y} \right\}. \quad (12.112)$$

You can check that if  $g$  is analytic at a given point (so  $g$  satisfies the Cauchy-Riemann conditions at that point), then  $\partial g / \partial w^* = 0$  at that point.

Consider, for example,  $g(w) = w^m (w^*)^n = (x + jy)^m (x - jy)^n$ . Then,

$$\begin{aligned} \frac{\partial g}{\partial w} &= \frac{1}{2} \left\{ m(x + jy)^{m-1} (x - jy)^n + n(x + jy)^m (x - jy)^{n-1} \right. \\ &\quad \left. - j(m(x + jy)^{m-1} (x - jy)^n - n(x + jy)^m (x - jy)^{n-1}) \right\} \\ &= \frac{1}{2} \left\{ mw^{m-1} (w^*)^n + nw^m (w^*)^{n-1} + mw^{m-1} (w^*)^n - nw^m (w^*)^{n-1} \right\} \\ &= mw^{m-1} (w^*)^n. \end{aligned}$$

As advertised, the final result is what we would obtain if we had treated  $w$  and  $w^*$  as two independent variables. Similarly, you can check that

$$\frac{\partial g}{\partial w^*} = nw^m (w^*)^{n-1}.$$

An important special case is the quadratic function  $f(w) = d - w^*r - r^*w + w^*wR$ , which assumes only real values if  $d$  and  $R$  are real. Using the previous result, we see that

$$\frac{\partial f}{\partial w} = -r^* + w^*R, \quad \frac{\partial f}{\partial w^*} = -r + wR.$$

Note that there is no factor of 2 in this case. This difference between Wirtinger derivatives and standard real derivatives will propagate to many expressions in this text.

Now, the definitions (12.112) are useful for minimizing real functions of complex variables as in (12.109), since from (12.112)

$$\frac{\partial f}{\partial w^*} = 0 \Leftrightarrow \frac{\partial \hat{f}}{\partial x} = 0 \quad \text{and} \quad \frac{\partial \hat{f}}{\partial y} = 0.$$

Applying this idea to our quadratic function, we see that

$$\frac{\partial f}{\partial w^*} = 0 \Rightarrow R w_0 = r \Rightarrow w_0 = r/R,$$

if  $R \neq 0$ . In addition,

$$\frac{\partial^2 f}{\partial w \partial w^*} = R,$$

and we can conclude that  $w_0$  is a minimum as long as  $R > 0$ .

The same conclusions could easily be obtained working directly with  $\hat{f}$ . The Wirtinger derivatives are merely a shortcut, that makes working with complex variables very similar to working with real variables. A good detailed description of Wirtinger derivatives can be found in [34].

In the vector case, the definitions and results are very similar. Given a function

$$g : \mathcal{C}^M \rightarrow \mathcal{R},$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} \in \mathcal{C}^M \mapsto g(\mathbf{w}),$$

with  $w_i = x_i + j y_i$ ,  $i = 0, \dots, M-1$ , we define

$$\frac{\partial g}{\partial \mathbf{w}} = \left[ \frac{\partial g}{\partial w_0} \cdots \frac{\partial g}{\partial w_{M-1}} \right] = \frac{1}{2} \left[ \frac{\partial g}{\partial x_0} - j \frac{\partial g}{\partial y_0} \cdots \frac{\partial g}{\partial x_{M-1}} - j \frac{\partial g}{\partial y_{M-1}} \right],$$

$$\frac{\partial g}{\partial \mathbf{w}^H} = \begin{bmatrix} \frac{\partial g}{\partial w_0^*} \\ \vdots \\ \frac{\partial g}{\partial w_{M-1}^*} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{\partial g}{\partial x_0} - j \frac{\partial g}{\partial y_0} \\ \vdots \\ \frac{\partial g}{\partial x_{M-1}} - j \frac{\partial g}{\partial y_{M-1}} \end{bmatrix}.$$

Using our definition of gradients, the second derivative of  $f$  (its *Hessian*) is

$$\frac{\partial^2 f}{\partial \mathbf{w} \partial \mathbf{w}^H} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{\partial f}{\partial w_0^*} \right\} \\ \vdots \\ \frac{\partial}{\partial \mathbf{w}} \left\{ \frac{\partial f}{\partial w_{M-1}^*} \right\} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial w_0 \partial w_0^*} & \cdots & \frac{\partial^2 f}{\partial w_{M-1} \partial w_0^*} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_0 \partial w_{M-1}^*} & \cdots & \frac{\partial^2 f}{\partial w_{M-1} \partial w_{M-1}^*} \end{bmatrix}.$$

The most important example is the quadratic function

$$f(\mathbf{w}) = r_d - \mathbf{w}^H \mathbf{r}_{d\phi} - \mathbf{r}_{d\phi}^H \mathbf{w} + \mathbf{w}^H \mathbf{R}_\phi \mathbf{w},$$

where  $r_d$  is a real variable, and  $\mathbf{R}_\phi$  is Hermitian symmetric. Then, we have

$$\frac{\partial f}{\partial \mathbf{w}^H} = -\mathbf{r}_{d\phi} + \mathbf{R}_\phi \mathbf{w}.$$

We see that a stationary point of  $f$  is such that  $\mathbf{R}_\phi \mathbf{w}_o = \mathbf{r}_{d\phi}$ . In addition,

$$\frac{\partial^2 f}{\partial \mathbf{w} \partial \mathbf{w}^H} = \mathbf{R}_\phi,$$

and we conclude that  $\mathbf{w}_o$  is the single point of minimum of  $f$  if and only if  $\mathbf{R}_\phi$  is positive-definite.

Note that in the complex case, the gradient and the Hessian do not have the factors of 2 that we saw in the real case (see Box 1).

---

### 1.12.3 Stochastic algorithms

In this section, we focus on three of the most important adaptive algorithms: the least-mean squares (LMS) algorithm, the normalized least-mean squares (NLMS) algorithm, and the recursive least-squares (RLS) algorithm, which are the basis of many other algorithms. These three stochastic algorithms are derived in detail and analyzed using different methods. We present different approaches for analyzing an adaptive filter, because the variety and complexity of their behavior makes it difficult for a single technique to be able to handle all situations. The analyses provided here allow us to predict the behavior of the algorithms in transient and steady-state, and choose values of parameters (such as the step-size) for which the filters operate adequately. In particular, we study conditions under which the filters will remain stable.

We assume linearly parametrized classes, such as FIR ( $\mathcal{F}_{\text{FIR}}$ ) and Volterra ( $\mathcal{F}_V$ ) filters (see Section 1.12.2.1). In these classes, each element of the input regressor vector is given by a certain function  $\phi_i$   $i = 0, 1, \dots, M-1$  of  $x(n), x(n-1), \dots, x(n-N)$ . In the case of length- $M$  FIR filters, we have

$$\begin{aligned} \boldsymbol{\phi}(n) &= [\phi_0(n) \phi_1(n) \cdots \phi_{M-1}(n)]^T \\ &= [x(n) x(n-1) \cdots x(n-M+1)]^T, \end{aligned} \quad (12.113)$$

whereas in the case of second-order Volterra filters with memory  $N = 1$  and real-valued signals, we have

$$\begin{aligned} \boldsymbol{\phi}(n) &= [\phi_0(n) \phi_1(n) \phi_2(n) \phi_3(n) \phi_4(n)]^T \\ &= [x(n) x(n-1) x^2(n) x(n)x(n-1) x^2(n-1)]^T. \end{aligned} \quad (12.114)$$

As much as possible, our presentation in this section is independent of the choices of  $\mathcal{F}$  and  $\phi(\cdot)$ . However, some algorithms are designed for a specific class  $\mathcal{F}$  (the most common case are fast algorithms designed for FIR filters (12.113)). In these cases, we will alert the reader of the restriction.

Using the results of Box 6, the derivation of algorithms for real or circular complex inputs is very similar, the only differences being the need of conjugating a few variables in some expressions, and the appearance of a factor  $\beta = 2$  in the expressions of gradients of functions of real variables, and  $\beta = 1$  in the complex case.

In general, the output of a length- $M$  adaptive filter is given by

$$\hat{y}(n) = \mathbf{w}^H(n)\boldsymbol{\phi}(n) = w_0^*(n)\phi_0(n) + w_1^*(n)\phi_1(n) + \cdots + w_{M-1}^*(n)\phi_{M-1}(n). \quad (12.115)$$

In the case of Volterra filters, it is common to use the notation  $w_{k,\ell}$  for the weight related to the input  $x(n-k)x(n-\ell)$  (see Eq. (12.43)). However, to obtain a common formulation independent of the class  $\mathcal{F}$ , we denote the weight vector as

$$\mathbf{w}(n) = [w_0(n) \ w_1(n) \ w_2(n) \ \cdots \ w_{M-1}(n)]^T. \quad (12.116)$$

The task of the adaptive filter will be to choose in a recursive form the values of the parameters  $w_0, w_1, \dots, w_{M-1}$  that best fit the data at each time instant  $n$ .

Figure 12.24 shows the scheme of a length- $M$  adaptive filter for linearly parametrized classes, where  $\Phi$  denotes the set of functions  $\phi_i(\cdot), i = 0, 1, \dots, M-1$  of  $x(n), x(n-1), \dots, x(n-N)$  that generate the elements  $\phi_0(n), \phi_1(n), \dots, \phi_{M-1}(n)$  of the input regressor vector  $\boldsymbol{\phi}(n)$ . The number of samples of the signal  $x(n)$  depends on the class  $\mathcal{F}$  and is given by  $N+1$ . An FIR filter with length  $M$  requires  $N+1 = M$  samples. On the other hand, a Volterra filter with memory  $N$ , requires  $N+1$  samples, but the length of the corresponding adaptive filter depends on the order of the Volterra series expansion. At each time instant, the output of the adaptive filter  $\hat{y}(n)$  is compared with the desired signal  $d(n)$  to compute the error  $e(n) = d(n) - \hat{y}(n)$ . In order to minimize a cost function of this error, a stochastic algorithm uses  $e(n)$  to adjust the filter weights.

### 1.12.3.1 LMS algorithm

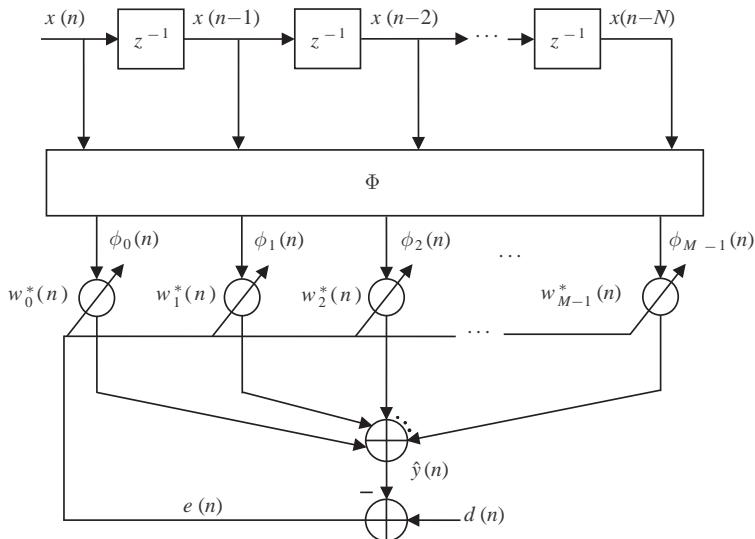
The update equation of the steepest-descent algorithm is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{\beta} \nabla_{\mathbf{w}} J(n), \quad (12.117)$$

where  $\mu$  is a step-size and  $\beta = 1$  (resp.,  $\beta = 2$ ) for complex (resp., real) data (see Box 6). This algorithm requires exact measurements of the gradient vector. For the case of minimization of the mean-square error, that is, if  $J(n) = E\{|e(n)|^2\}$ ,

$$\nabla_{\mathbf{w}} J(n) = -\beta \mathbf{r}_{d\phi} + \beta \mathbf{R}_{\phi} \mathbf{w}(n) \quad (12.118)$$

at each iteration  $n$  (see Box 1 for an introduction to gradient algorithms). Note that the exact gradient requires a prior knowledge of the correlation matrix  $\mathbf{R}_{\phi}$  and the cross-correlation vector  $\mathbf{r}_{d\phi}$ , which is not feasible in real-time applications. For example, in the hands-free telephone application described in Section 1.12.1.1, the input signal of the adaptive filter is the far-end speech (see Figures 12.3 and 12.4). A speech signal is nonstationary and therefore, it is not possible to obtain exact estimates for  $\mathbf{R}_{\phi}$  and  $\mathbf{r}_{d\phi}$  at each time instant. Furthermore, good estimates would demand much processing time, which

**FIGURE 12.24**

Scheme for a length- $M$  adaptive filter assuming linearly parametrized classes.

could insert an inadmissible delay in the system, making this approach not feasible, except in off-line applications. Similar restrictions appear in virtually all adaptive filtering applications.

To circumvent this problem, a stochastic version of the steepest descent algorithm was proposed, the least-mean squares (LMS) algorithm. Instead of minimizing the exact mean-square error, the LMS algorithm minimizes a straightforward approximation to it:  $|e(n)|^2$ . The idea, as we saw in Section 1.12.1.2.4, is that the algorithm will approximately average the cost-function if adaptation is slow (small step-size). The estimation error is

$$e(n) = d(n) - \mathbf{w}^H(n)\boldsymbol{\phi}(n), \quad (12.119)$$

so the gradient of the instantaneous cost function is

$$\hat{\nabla}_{\mathbf{w}} J(n) = -\beta\boldsymbol{\phi}(n)d^*(n) + \beta\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\mathbf{w}(n) = -\beta\boldsymbol{\phi}(n)e^*(n). \quad (12.120)$$

Since  $e(n)$  depends on the old estimate of the weight vector, it is called an *a priori* estimation error, by opposition to the *a posteriori* error that will be introduced in (12.129). Instead of minimizing the instantaneous cost function, the stochastic gradient defined in (12.120) can be obtained by replacing the instantaneous estimates

$$\hat{\mathbf{R}}_{\boldsymbol{\phi}}(n) = \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n) \quad (12.121)$$

and

$$\hat{\mathbf{r}}_{d\boldsymbol{\phi}}(n) = \boldsymbol{\phi}(n)d^*(n) \quad (12.122)$$

in (12.118).

**Table 12.2** Summary of the LMS Algorithm

Initialization:  
Set  $\mathbf{w}(0) = \mathbf{0}$   
For  $n = 0, 1, 2, \dots$ , compute  
 $\hat{y}(n) = \mathbf{w}^H(n)\phi(n)$   
 $e(n) = d(n) - \hat{y}(n)$   
 $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e^*(n)\phi(n)$

**Table 12.3** Computational Cost of the LMS Algorithm for Complex and Real-Valued Signals in Terms of Real Multiplications and Real Additions Per Iteration

Term	Real Signals		Complex Signals	
	x	+	x	+
$\hat{y}^*(n) = \phi^H(n)[\mathbf{w}(n)]$	M	$M - 1$	$4M$	$4M - 2$
$e^*(n) = d^*(n) - [\hat{y}^*(n)]$		1		2
$\mu[e^*(n)]$	1		2	
$[\mu e^*(n)]\phi(n)$	M		$4M$	$2M$
$\mathbf{w}(n+1) = [\mathbf{w}(n)] + [\mu e^*(n)\phi(n)]$		M		$2M$
TOTAL per iteration	$2M + 1$	$2M$	$8M + 2$	$8M$

Replacing (12.120) in (12.117), we arrive at the update equation for LMS, i.e.,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \phi(n) e^*(n). \quad (12.123)$$

As explained in Section 1.12.1.2.4 and illustrated in the animations of Figure 12.9, although there is no explicit average being taken in the LMS algorithm, it in fact computes an implicit approximate average if the step-size is small.

The LMS algorithm is summarized in Table 12.2. Its computational cost increases linearly with the length of the filter, i.e., the computational cost is  $\mathcal{O}(M)$ . Table 12.3 shows the number of real multiplications and real additions per iteration for real and complex-valued signals. The quantities inside brackets were computed in previous steps. We assume that a complex multiplication requires 4 real multiplications and 2 real additions (it is possible to use less multiplications and more additions), and a complex addition requires 2 real additions. Different ways of carrying out the calculations may result in a different computational cost. For example, if we first computed  $\mu \times \phi(n)$  followed by  $[\mu \phi(n)] \times e^*(n)$ , LMS would require  $M - 1$  more real multiplications per iteration than the ordering of Table 12.3, considering real-valued signals. For complex-valued signals, the cost would be even higher, since LMS would require  $4M - 2$  and  $2M$  more real multiplications and real additions, respectively.

### 1.12.3.1.1 A deterministic approach for the stability of LMS

Finding the range of step-sizes such that the recursion (12.123) converges is a complicated task. There are two main approaches to finding the maximum step-size: an average approach and a deterministic (worst-case) approach. The average approach will be treated in Section 1.12.4.3. Since the deterministic approach is easy to explain intuitively, we address it in the sequel (although a precise proof is not so simple, see [35,36] for a full analysis).

For simplicity, assume that our filter is such that  $d(n) \equiv 0$ . Although this is not a very likely situation to encounter in practice, it allows us to avoid technical details that would complicate considerably the discussion, and still arrive at the important conclusions.

Let us rewrite the LMS recursion, expanding the error with  $d(n) = 0$  as  $e^*(n) = -(\mathbf{w}^H(n)\boldsymbol{\phi}(n))^* = -\boldsymbol{\phi}^H(n)\mathbf{w}(n)$  to obtain

$$\mathbf{w}(n+1) = [\mathbf{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)]\mathbf{w}(n). \quad (12.124)$$

This is a linear system, but not a time-invariant one. It will be stable if the eigenvalues  $\lambda_i(n)$  of the matrix  $\mathbf{A}(n) = \mathbf{I} - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$  satisfy

$$\max_i |\lambda_i(n)| \leq 1, \quad \text{for all } n \geq 0. \quad (12.125)$$

Be careful, if  $d(n)$  were not identically zero, condition (12.125) would have to be slightly modified to guarantee stability (see, e.g., [36,37]).

The eigenvalues of  $\mathbf{A}(n)$  are easy to find, if we note that  $\boldsymbol{\phi}(n)$  is an eigenvector of  $\mathbf{A}(n)$  (the case  $\boldsymbol{\phi}(n) = \mathbf{0}$  is trivial):

$$\mathbf{A}(n)\boldsymbol{\phi}(n) = \boldsymbol{\phi}(n) - \mu\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n) = (1 - \mu\|\boldsymbol{\phi}(n)\|^2)\boldsymbol{\phi}(n),$$

where  $\|\boldsymbol{\phi}(n)\|^2 = \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)$  is the square of the Euclidean norm. The corresponding eigenvalue is  $\lambda_1 = 1 - \mu\|\boldsymbol{\phi}(n)\|^2$ . All other eigenvalues are equal to one (the eigenvectors are all vectors orthogonal to  $\boldsymbol{\phi}(n)$ .) Therefore, the LMS algorithm (12.124) will be stable whenever

$$-1 \leq 1 - \mu\|\boldsymbol{\phi}(n)\|^2 \leq 1 \iff 0 \leq \mu \leq \frac{2}{\sup_{n \geq 0} \|\boldsymbol{\phi}(n)\|^2}. \quad (12.126)$$

The *supremum* (sup) of a sequence is an extension to the maximum, defined to avoid a technicality. Consider the sequence  $f(n) = \{1 - 0.5^n\}$ ,  $n \geq 0$ . Strictly speaking, it has no maximum, since there is no  $n_0$  for which  $f(n_0) \geq f(n)$  for all  $n \geq 0$ . The supremum is defined so we can avoid this problem:  $\sup_{n \geq 0} f(n) = 1$ . Therefore, the supremum is equal to the maximum whenever the later exists, but is also defined for some sequences that do not have a maximum. The *infimum* (inf) has a similar relation to the minimum.

This condition is sufficient for stability, but is too conservative, since  $\|\boldsymbol{\phi}(n)\|^2$  is not at its highest value at all instants. A popular solution is to use *normalization*. The idea is simple: adopt a time-variant step-size  $\mu(n)$  such that

$$\mu(n) = \frac{\tilde{\mu}}{\varepsilon + \|\boldsymbol{\phi}(n)\|^2}, \quad (12.127)$$

with  $0 < \tilde{\mu} < 2$  and where  $\varepsilon > 0$  is used to avoid division by zero. The resulting algorithm is known as *normalized LMS* (NLMS). We will talk more about it in the next section.

### 1.12.3.2 Normalized LMS algorithm

One problem with the LMS algorithm is how to choose the step-size. How large should it be to enable a high convergence rate, provide an acceptable misadjustment, and even ensure the stability of LMS? More importantly, how can this choice be made so that the filter will work correctly even for very different input signals? One answer to this problem is the normalized LMS algorithm which we describe now. It uses a time varying step-size, which is particularly useful when the statistics of the input signals change quickly.

Thus, replacing the fixed step-size  $\mu$  by a time varying step-size  $\mu(n)$  in (12.123), we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)\phi(n)e^*(n). \quad (12.128)$$

In order to increase the convergence speed, we could try to find  $\mu(n)$  such that the *a posteriori* estimation error,

$$\xi(n) = d(n) - \mathbf{w}^H(n+1)\phi(n), \quad (12.129)$$

is zeroed. Note that, in contrast to the *a priori* estimation error defined in (12.119),  $\xi(n)$  depends on the updated estimate of the weight vector, hence the name *a posteriori* estimation error. Replacing (12.128) in (12.129), we obtain

$$\begin{aligned} \xi(n) &= \underbrace{d(n) - \mathbf{w}^H(n)\phi(n)}_{e(n)} - \mu(n)\phi^H(n)\phi(n)e(n) \\ &= [1 - \mu(n)\phi^H(n)\phi(n)]e(n). \end{aligned} \quad (12.130)$$

In order to enforce  $\xi(n) = 0$  at each time instant  $n$ , we must select

$$\mu(n) = \frac{1}{\phi^H(n)\phi(n)} = \frac{1}{\|\phi(n)\|^2}. \quad (12.131)$$

Replacing (12.131) in (12.128), we obtain the update equation of the normalized least-squares algorithm, i.e.,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{\|\phi(n)\|^2}\phi(n)e^*(n). \quad (12.132)$$

Note that the time varying step-size  $\mu(n)$  depends inversely on the instantaneous power of the input vector  $\phi(n)$ . Indeed, we will show in Section 1.12.4.3 that the maximum value of the fixed step-size  $\mu$  to ensure the convergence and mean-square stability of LMS depends inversely on the power of the input signal.

In order to make (12.132) more reliable for practical implementations, it is common to introduce two positive constants: a fixed step-size  $\tilde{\mu}$  to control the rate of convergence (and the misadjustment) and the regularization factor  $\varepsilon$  to prevent division by a small value when  $\|\phi(n)\|^2$  is small. With these constants, the NLMS update equation reduces to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{\varepsilon + \|\phi(n)\|^2}\phi(n)e^*(n). \quad (12.133)$$

Some authors refer to (12.132) as NLMS and to (12.133) as  $\varepsilon$ -NLMS. However, for simplicity we will refer to both as NLMS. The name “normalized” is due to a most useful property of NLMS: the range of values of  $\tilde{\mu}$  for which the algorithm remains stable is independent of the statistics of  $x(n)$ , i.e.,  $0 < \tilde{\mu} < 2$ , as can be observed in Section 1.12.3.1.1.

As pointed out in [3], the NLMS algorithm can also be interpreted as a quasi-Newton algorithm. As briefly explained in Box 1, a quasi-Newton algorithm updates the weights using approximations for the Hessian matrix and gradient vector, i.e.,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{\mu}{\beta} \left[ \widehat{\nabla}_{\mathbf{w}}^2 J(n) \right]^{-1} \widehat{\nabla}_{\mathbf{w}} J(n). \quad (12.134)$$

Assuming the following instantaneous approximations

$$\widehat{\nabla}_{\mathbf{w}} J(n) = -\beta \boldsymbol{\phi}(n) d^*(n) + \beta \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{w}(n) = -\beta \boldsymbol{\phi}(n) e^*(n) \quad (12.135)$$

and

$$\widehat{\nabla}_{\mathbf{w}}^2 J(n) = \varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n), \quad (12.136)$$

and replacing them in (12.134), we arrive at the stochastic recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \left[ \varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right]^{-1} \boldsymbol{\phi}(n) e^*(n). \quad (12.137)$$

The term  $\varepsilon \mathbf{I}$  guarantees that (12.136) is invertible (a rank-one matrix such as  $\boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n)$  is never invertible).

To compute  $[\varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n)]^{-1}$ , we can use the *matrix inversion lemma* (Box 3), which gives us an expression for the inverse of an  $M \times M$  matrix of the form

$$\mathbf{A} = \mathbf{B} + \mathbf{C} \mathbf{D} \mathbf{C}^H, \quad (12.138)$$

where  $\mathbf{B}$  and  $\mathbf{D}$  are two square matrices with dimensions  $M$  and  $N$ , respectively, and  $\mathbf{C}$  is an  $M \times N$  matrix. According to the matrix inversion lemma, the inverse of  $\mathbf{A}$  is given by

$$\mathbf{A}^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1} \mathbf{C} (\mathbf{D}^{-1} + \mathbf{C}^H \mathbf{B}^{-1} \mathbf{C})^{-1} \mathbf{C}^H \mathbf{B}^{-1}. \quad (12.139)$$

Thus, identifying

$$\mathbf{A} = \varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n), \quad \mathbf{B} = \varepsilon \mathbf{I}, \quad \mathbf{C} = \boldsymbol{\phi}(n), \quad \text{and} \quad \mathbf{D} = 1,$$

we obtain

$$[\varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n)]^{-1} = \varepsilon^{-1} \mathbf{I} - \frac{\varepsilon^{-2}}{1 + \varepsilon^{-1} \boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)} \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n). \quad (12.140)$$

Multiplying (12.140) from the right by  $\boldsymbol{\phi}(n)$ , after some algebraic manipulations, we get

$$[\varepsilon \mathbf{I} + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n)]^{-1} \boldsymbol{\phi}(n) = \frac{\boldsymbol{\phi}(n)}{\varepsilon + \boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)} = \frac{\boldsymbol{\phi}(n)}{\varepsilon + \|\boldsymbol{\phi}(n)\|^2}. \quad (12.141)$$

**Table 12.4** Summary of the NLMS Algorithm

Initialization:

Set  $\mathbf{w}(0) = \mathbf{0}$  and  $0 < \tilde{\mu} < 2$

For  $n = 0, 1, 2, \dots$ , compute

$$\hat{y}(n) = \mathbf{w}^H(n)\phi(n)$$

$$e(n) = d(n) - \hat{y}(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\tilde{\mu}}{\varepsilon + \|\phi(n)\|^2} e^*(n)\phi(n)$$

Replacing (12.141) in (12.137), we arrive at (12.133). Since Newton-based algorithms converge faster than gradient-based algorithms, it is expected that NLMS exhibits a higher convergence rate than that of LMS, which is indeed the case in general.

The NLMS algorithm is summarized in Table 12.4. To obtain its computational cost, we can use the same procedure considered in the computation of the number of operations of LMS (see Table 12.3). For complex-valued signals, NLMS requires  $10M + 2$  real multiplications,  $10M$  real additions and one real division per iteration. For real-valued signals, it requires  $3M + 1$  real multiplications and  $3M$  real additions per iteration. Note that for input vectors with a shift structure such as the FIR filters of Eq. (12.113), its computational cost can be reduced if  $\|\phi(n)\|^2$  is computed by the following recursion

$$\|\phi(n)\|^2 = \|\phi(n-1)\|^2 - |x(n-M)|^2 + |x(n)|^2 \quad (12.142)$$

with initialization  $\|\phi(-1)\| = 0$ .

### 1.12.3.3 RLS algorithm

The convergence rate of LMS-type algorithms varies considerably with the input signal  $x(n)$ , since they are based on steepest-descent optimization algorithms. If  $x(n)$  is a white noise, the convergence rate is high. On the other hand, if the correlation between successive samples of  $x(n)$  is high, LMS-type algorithms converge slowly. The RLS algorithm does not have this drawback. However, this advantage has a price: a considerable increase in the computational cost. Furthermore, numerical errors play a more important role in RLS and may even lead to divergence of the weight estimates. Different versions of RLS were proposed to circumvent this problem. In the sequel, we obtain the equations of the conventional RLS algorithm.

The RLS can be derived in different ways (much as NLMS). Each form of arriving at the algorithm highlights a different set of its properties: one that leads to many new algorithms and insights is the connection between RLS and Kalman filters described in [38].

We present here shorter routes, starting with a derivation based on deterministic arguments. In fact, RLS solves a deterministic least-squares problem, which is based on the weighted least-squares cost function

$$J_{\text{LS}}(n) = \sum_{\ell=0}^n \lambda^{n-\ell} |\xi(\ell)|^2, \quad (12.143)$$

where  $0 \ll \lambda < 1$  is a constant known as *forgetting factor*,

$$\xi(\ell) = d(\ell) - \mathbf{w}^H \boldsymbol{\phi}(\ell), \quad (12.144)$$

are *a posteriori* errors and

$$\mathbf{w}(n+1) = \arg \min_{\mathbf{w}} J_{\text{LS}}(n). \quad (12.145)$$

The factor  $\lambda^{n-\ell}$  emphasizes the most recent errors, forgetting the errors from the remote past. This is important, otherwise the algorithm would give too much weight to the remote past and would not be able to track variations in the input signals. Note that  $\mathbf{w}(n+1)$  minimizes  $J_{\text{LS}}(n)$  at each time instant. Thus, RLS provides an exact solution for the least-squares problem at each time instant. As the NLMS algorithm, RLS seeks to minimize the *a posteriori* error  $|\xi(n)|^2$ , searching for an exact solution to an optimization problem. The difference is that RLS searches for a weight vector  $\mathbf{w}(n+1)$  that takes into consideration the whole past history of inputs, minimizing  $\lambda^n |\xi(0)|^2 + \lambda^{n-1} |\xi(1)|^2 + \dots + \lambda |\xi(n-1)|^2 + |\xi(n)|^2$  at each time instant  $n$ , not only the current value  $|\xi(n)|^2$  as in NLMS.

The error sequence  $\xi(\ell)$  weighted by  $\lambda^{(n-\ell)/2}$  in the interval  $0 \leq \ell \leq n$  can be written in a vectorial form, i.e.,

$$\xi(n) = \mathbf{d}(n) - \Phi(n)\mathbf{w}^*, \quad (12.146)$$

where

$$\xi(n) = [\xi(n) \ \lambda^{1/2}\xi(n-1) \ \dots \ \lambda^{n/2}\xi(0)]^T, \quad (12.147)$$

$$\mathbf{d}(n) = [d(n) \ \lambda^{1/2}d(n-1) \ \dots \ \lambda^{n/2}d(0)]^T \quad (12.148)$$

and

$$\Phi(n) = \begin{bmatrix} \phi_0(n) & \phi_1(n) & \dots & \phi_{M-1}(n) \\ \lambda^{1/2}\phi_0(n-1) & \lambda^{1/2}\phi_1(n-1) & \dots & \lambda^{1/2}\phi_{M-1}(n-1) \\ \lambda\phi_0(n-2) & \lambda\phi_1(n-2) & \dots & \lambda\phi_{M-1}(n-2) \\ \vdots & \ddots & \ddots & \vdots \\ \lambda^{n/2}\phi_0(0) & \lambda^{n/2}\phi_1(0) & \dots & \lambda^{n/2}\phi_{M-1}(0) \end{bmatrix}. \quad (12.149)$$

Note that the weighted least-squares cost function can be rewritten in terms of the vector  $\xi(n)$ , i.e.,

$$J_{\text{LS}}(n) = \xi^H(n)\xi(n) = \|\xi(n)\|^2 = \sum_{\ell=0}^n \lambda^{n-\ell} |\xi(\ell)|^2. \quad (12.150)$$

Now, we have to find the weight vector  $\mathbf{w}(n+1)$  that minimizes (12.150). The gradient of  $J_{\text{LS}}(n)$  with relation to  $\mathbf{w}^H$  is given by

$$\nabla_{\mathbf{w}} J_{\text{LS}}(n) = -\beta \Phi^T(n) \mathbf{d}^*(n) + \beta \Phi^T(n) \Phi^*(n) \mathbf{w}, \quad (12.151)$$

where as usual,  $\beta = 1$  (resp.,  $\beta = 2$ ) for complex (resp., real) data. Defining

$$\widehat{\mathbf{R}}_{\boldsymbol{\phi}}(n) = \Phi^T(n) \Phi^*(n) = \sum_{\ell=0}^n \lambda^{n-\ell} \boldsymbol{\phi}(\ell) \boldsymbol{\phi}^H(\ell) \quad (12.152)$$

and

$$\hat{\mathbf{r}}_{d\phi}(n) = \Phi^T(n)\mathbf{d}^*(n) = \sum_{\ell=0}^n \lambda^{n-\ell} d^*(\ell) \boldsymbol{\phi}(\ell), \quad (12.153)$$

in which  $\boldsymbol{\phi}(n)$  is defined as before, (12.151) can be rewritten as

$$\nabla_{\mathbf{w}} J_{LS}(n) = -\beta \hat{\mathbf{r}}_{d\phi}(n) + \beta \hat{\mathbf{R}}_{\phi}(n) \mathbf{w}. \quad (12.154)$$

Equating  $\nabla_{\mathbf{w}} J_{LS}(n)$  to the null vector, we get the normal equations

$$\hat{\mathbf{R}}_{\phi}(n) \mathbf{w}(n+1) = \hat{\mathbf{r}}_{d\phi}(n). \quad (12.155)$$

Note that these are a deterministic version of the normal equations we saw in Section 1.12.2.1. Therefore, to minimize  $J_{LS}(n)$ , the weight vector  $\mathbf{w}(n+1)$  must satisfy

$$\mathbf{w}(n+1) = \hat{\mathbf{R}}_{\phi}^{-1}(n) \hat{\mathbf{r}}_{d\phi}(n), \quad (12.156)$$

which requires the computation of the inverse correlation matrix  $\hat{\mathbf{R}}_{\phi}^{-1}(n)$  at each time instant. Note that this is only valid if  $\hat{\mathbf{R}}_{\phi}^{-1}(n)$  is nonsingular. In particular, the inverse does not exist if  $n < M$  (Box 3). Assuming  $\lambda = 1$  and that the signals  $x(n)$  and  $d(n)$  are jointly stationary and ergodic, we obtain the following steady-state relations

$$\lim_{n \rightarrow \infty} \frac{1}{n+1} \hat{\mathbf{R}}_{\phi}(n)|_{\lambda=1} = \mathbf{R}_{\phi} \quad \text{and} \quad \lim_{n \rightarrow \infty} \frac{1}{n+1} \hat{\mathbf{r}}_{d\phi}(n)|_{\lambda=1} = \mathbf{r}_{d\phi}.$$

Therefore, the deterministic normal equations converge to the stochastic normal equations when  $\lambda = 1$ , and the weight vector obtained from (12.156) converges to the Wiener solution, i.e.,

$$\lim_{n \rightarrow \infty} \mathbf{w}(n+1)|_{\lambda=1} = \mathbf{R}_{\phi}^{-1} \mathbf{r}_{d\phi} = \mathbf{w}_o. \quad (12.157)$$

We should mention that with  $\lambda = 1$  the convergence rate of RLS goes to zero, i.e., it loses the ability to follow the statistical variations of the observed data. An equivalent result is obtained for LMS or NLMS when the fixed step-size  $\mu$  or  $\tilde{\mu}$  is replaced by  $1/n$  [39].

The computational cost of (12.156) is high due to the computation of the inverse matrix. In order to solve the normal equations in real-time and with less cost, we should rewrite (12.152) and (12.153) as recursions, i.e.,

$$\hat{\mathbf{R}}_{\phi}(n) = \lambda \hat{\mathbf{R}}_{\phi}(n-1) + \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n), \quad (12.158)$$

and

$$\hat{\mathbf{r}}_{d\phi}(n) = \lambda \hat{\mathbf{r}}_{d\phi}(n-1) + d^*(n) \boldsymbol{\phi}(n), \quad (12.159)$$

with initializations  $\hat{\mathbf{R}}_{\phi}(-1) = \delta \mathbf{I}$  and  $\hat{\mathbf{r}}_{d\phi}(-1) = \mathbf{0}$ , where  $\delta$  is a small positive constant. Note that (12.159) is identical to (12.153), but (12.158) leads to the estimate

$$\hat{\mathbf{R}}_{\phi}(n) = \lambda^{n+1} \delta \mathbf{I} + \sum_{\ell=0}^n \lambda^{n-\ell} \boldsymbol{\phi}(\ell) \boldsymbol{\phi}^H(\ell), \quad (12.160)$$

and therefore the initialization guarantees the existence of the inverse at all instants. For  $0 \ll \lambda < 1$ , the initialization  $\widehat{\mathbf{R}}_{\phi}(-1) = \delta \mathbf{I}$  is forgotten and (12.158) becomes close to (12.152) as  $n$  increases. Furthermore, if you want the algorithm to forget quickly the initialization, besides using  $0 \ll \lambda < 1$ , you should choose  $\delta$  small ( $\delta^{-1}$  large). Note however that a certain amount of regularization may be useful in practice, to avoid problems when the input signal has long stretches with low power.

In order to obtain a recursion for the inverse matrix  $\widehat{\mathbf{R}}_{\phi}^{-1}(n)$ , we can use the matrix inversion lemma (Box 3) by comparing (12.138) with (12.158) and identifying

$$\mathbf{A} = \widehat{\mathbf{R}}_{\phi}(n), \quad \mathbf{B} = \lambda \widehat{\mathbf{R}}_{\phi}(n-1), \quad \mathbf{C} = \boldsymbol{\phi}(n), \quad \text{and} \quad \mathbf{D} = 1.$$

Thus, using (12.139), we obtain

$$\widehat{\mathbf{R}}_{\phi}^{-1}(n) = \lambda^{-1} \widehat{\mathbf{R}}_{\phi}^{-1}(n-1) - \frac{\lambda^{-2} \widehat{\mathbf{R}}_{\phi}^{-1}(n-1) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \widehat{\mathbf{R}}_{\phi}^{-1}(n-1)}{1 + \lambda^{-1} \boldsymbol{\phi}^H(n) \widehat{\mathbf{R}}_{\phi}^{-1}(n-1) \boldsymbol{\phi}(n)}. \quad (12.161)$$

Defining  $\mathbf{P}(n) \triangleq \widehat{\mathbf{R}}_{\phi}^{-1}(n)$ , with some algebraic manipulations in (12.161), we arrive at

$$\mathbf{P}(n) = \frac{1}{\lambda} \left[ \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{P}(n-1)}{\lambda + \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)} \right] \quad (12.162)$$

with initialization  $\mathbf{P}(-1) = \delta^{-1} \mathbf{I}$ . This equation allow us to compute the inverse correlation matrix in a recurrent form, avoiding the explicit computation of a matrix inverse. However, the implementation of (12.162) in finite precision arithmetic requires some care to avoid numerical problems, as explained in Section 1.12.3.3.1.

To simplify the following equations, it is convenient to define the column vector

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1) \boldsymbol{\phi}(n)}{\lambda + \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)}, \quad (12.163)$$

which is the so-called Kalman gain. Thus, (12.162) can be rewritten as

$$\mathbf{P}(n) = \lambda^{-1} \left[ \mathbf{I} - \mathbf{k}(n) \boldsymbol{\phi}^H(n) \right] \mathbf{P}(n-1). \quad (12.164)$$

Using (12.164) and some algebraic manipulations in (12.163), we arrive at

$$\begin{aligned} \mathbf{k}(n) &= \lambda^{-1} \mathbf{P}(n-1) \boldsymbol{\phi}(n) - \lambda^{-1} \mathbf{k}(n) \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n) \\ &= \lambda^{-1} \left[ \mathbf{I} - \mathbf{k}(n) \boldsymbol{\phi}^H(n) \right] \mathbf{P}(n-1) \boldsymbol{\phi}(n) \\ &= \mathbf{P}(n) \boldsymbol{\phi}(n). \end{aligned} \quad (12.165)$$

Thus, using (12.159) and (12.165) in (12.156), the weight vector  $\mathbf{w}(n+1)$  can also be computed recursively, i.e.,

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{P}(n) \hat{\mathbf{r}}_{d\phi}(n) \\ &= \lambda \mathbf{P}(n) \hat{\mathbf{r}}_{d\phi}(n-1) + \mathbf{P}(n) \boldsymbol{\phi}(n) d^*(n) \\ &= \lambda \mathbf{P}(n) \hat{\mathbf{r}}_{d\phi}(n-1) + \mathbf{k}(n) d^*(n). \end{aligned} \quad (12.166)$$

**Table 12.5** Summary of the Conventional RLS Algorithm

Initialization:

Set  $\mathbf{P}(-1) = \delta^{-1} \mathbf{I}$ ;  $\delta = \text{small positive constant}$

$\mathbf{w}(0) = \mathbf{0}$  and  $0 \ll \lambda < 1$

For  $n = 0, 1, 2, \dots$ , compute

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1)\phi(n)}{\lambda + \phi^H(n)\mathbf{P}(n-1)\phi(n)}$$

$$\hat{y}(n) = \mathbf{w}^H(n)\phi(n)$$

$$e(n) = d(n) - \hat{y}(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mathbf{k}(n)e^*(n)$$

$$\mathbf{P}(n) = \lambda^{-1} [\mathbf{I} - \mathbf{k}(n)\phi^H(n)]\mathbf{P}(n-1)$$

Replacing (12.164) in (12.166), we arrive at

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{P}(n-1)\hat{r}_{d\phi}(n-1) - \mathbf{k}(n)\phi^H(n)\mathbf{P}(n-1)\hat{r}_{d\phi}(n-1) + \mathbf{k}(n)d^*(n) \\ &= \mathbf{w}(n) - \mathbf{k}(n)\phi^H(n)\mathbf{w}(n) + \mathbf{k}(n)d^*(n) \\ &= \mathbf{w}(n) + \mathbf{k}(n)\underbrace{[d(n) - \mathbf{w}^H(n)\phi(n)]^*}_{e(n)}. \end{aligned} \quad (12.167)$$

The RLS algorithm was derived here as an exact solution for the least-squares problem. However, it also can be interpreted as a *quasi-Newton* algorithm assuming the instantaneous approximation for the gradient vector as in (12.135) and considering  $\mathbf{P}(n)$  as an approximation for the inverse Hessian matrix. Replacing these approximations in (12.134), we arrive at the same stochastic recursion (12.167).

The RLS algorithm is summarized in Table 12.5. Its computational cost increases with  $M^2$ , much faster than those of LMS and NLMS. Different ways of carrying out the calculations may result in a slightly different computational cost, but all with the same order of magnitude, i.e.,  $\mathcal{O}(M^2)$ . Performing the calculations in the following order [3]:

$$\begin{aligned} &\phi^H(n) \times [\mathbf{w}(n)] \\ &d^*(n) - [\phi^H(n)\mathbf{w}(n)] \\ &\lambda^{-1}\phi(n) \\ &\mathbf{P}(n-1) \times [\lambda^{-1}\phi(n)] \\ &\phi^H(n) \times [\lambda^{-1}\mathbf{P}(n-1)\phi(n)] \\ &1 + [\lambda^{-1}\phi^H(n)\mathbf{P}(n-1)\phi(n)] \\ &1/[1 + \lambda^{-1}\phi^H(n)\mathbf{P}(n-1)\phi(n)] \\ &[\lambda^{-1}\phi^H(n)\mathbf{P}(n-1)\phi(n)] \times \left[ \frac{1}{1 + \lambda^{-1}\phi^H(n)\mathbf{P}(n-1)\phi(n)} \right] \end{aligned}$$

$$\begin{aligned}
& [\lambda^{-1} \mathbf{P}(n-1) \boldsymbol{\phi}(n)] \times \left[ \frac{\lambda^{-1} \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)}{1 + \lambda^{-1} \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)} \right] \\
& \mathbf{P}(n) \boldsymbol{\phi}(n) = [\lambda^{-1} \mathbf{P}(n-1) \boldsymbol{\phi}(n)] - \left[ \frac{\lambda^{-2} \mathbf{P}(n-1) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)}{1 + \lambda^{-1} \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)} \right] \\
& [\mathbf{P}(n) \boldsymbol{\phi}(n)] \times [d^*(n) - \boldsymbol{\phi}^H(n) \mathbf{w}(n)] \\
& \mathbf{w}(n+1) = [\mathbf{w}(n)] + [\mathbf{P}(n) \boldsymbol{\phi}(n) (d^*(n) - \boldsymbol{\phi}^H(n) \mathbf{w}(n))],
\end{aligned}$$

the conventional RLS requires  $4M^2 + 16M + 1$  real multiplications,  $4M^2 + 12M - 1$  real additions, and one real division per iteration for complex-valued signals. For real-valued data, it requires  $M^2 + 5M + 1$  real multiplications,  $M^2 + 3M$  real additions, and one real division per iteration. Note that the quantities inside brackets were computed in previous steps.

#### 1.12.3.3.1 Practical implementation of RLS

The RLS algorithm must be implemented carefully, because it is sensitive to numerical errors, which may accumulate and make the algorithm unstable. The problem can be understood intuitively as follows. In the conventional RLS algorithm, the inverse of the autocorrelation matrix is computed recursively by (expanding (12.161))

$$\mathbf{P}(n) = \lambda^{-1} \mathbf{P}(n-1) - \frac{\lambda^{-2} \mathbf{P}(n-1) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{P}(n-1)}{1 + \lambda^{-1} \boldsymbol{\phi}^H(n) \mathbf{P}(n-1) \boldsymbol{\phi}(n)},$$

i.e., as the difference between two positive (semi-) definite matrices. Note that, since  $\lambda \leq 1$ , the factor multiplying the current  $\mathbf{P}(n-1)$  is  $\lambda^{-1} \geq 1$ . In infinite precision arithmetic, any growth due to this factor will be compensated by the second term. However, in finite precision arithmetic this compensation may not take place, and the factor  $\lambda^{-1}$  may make the recursion numerically unstable—usually, what happens is that  $\mathbf{P}(n)$  loses its positive character, so the matrix will have a negative eigenvalue, which in its turn leads to the divergence of the coefficients. Thus, to avoid problems one would need to guarantee that  $\mathbf{P}(n)$  stays symmetric and positive-definite for all time instants  $n$  (these properties are known as *backward consistency* of RLS [40, 41]). Symmetry can be guaranteed by computing only the upper or lower triangular part of  $\mathbf{P}(n)$ , and copying the result to obtain the rest of elements. However, it is also necessary to guarantee positive-definiteness. Reference [42] describes how this can be achieved.

Due to the fast convergence rate of RLS, a large literature is devoted to finding numerically stable and low-cost (i.e.,  $\mathcal{O}(M)$ ) versions of RLS. This is not an easy problem, that required the work of numerous researchers to be solved. Nowadays the user has a very large number of different versions of RLS to choose from. There are versions with formal proofs of stability, versions that work well in practice, but without formal proofs, versions with  $\mathcal{O}(M^2)$  complexity, and versions with  $\mathcal{O}(M)$  complexity. The interested reader can find more references in Section 1.12.5.1.2.

One approach to ensure the consistency of  $\mathbf{P}(n)$  uses a property of symmetric and positive-definite matrices (see Box 3): Cholesky factorizations. A symmetric and positive-definite matrix  $\mathbf{P}(n)$  may

always be factored as

$$\mathbf{P}(n) = \mathcal{P}(n)\mathcal{P}^H(n), \quad (12.168)$$

where  $\mathcal{P}(n)$  is a lower triangular  $M \times M$  matrix, called Cholesky factor of  $\mathbf{P}(n)$ . Thus, an algorithm that computes  $\mathcal{P}(n)$  instead of  $\mathbf{P}(n)$  can avoid the numerical instability of the conventional RLS algorithm. There are many algorithms based on this main idea; for some of them a precise stability proof is available [40]. A recent and very complete survey of QR-RLS algorithms is available in [43].

Another approach is available when the regressor sequence  $\{\phi(n)\}$  has shift structure. In this case, it is possible to derive *lattice*-based RLS filters that are in practice stable, although no formal proof is available at this time, as described, for example, in [5].

A more practical solution for the implementation of the RLS algorithm was proposed recently in [44]. This approach avoids propagation of the inverse covariance matrix, using an iterative method to solve (12.155) at each step. The computational complexity is kept low by using the previous solution as initialization, and by restricting the majority of the multiplications to multiplications by powers of two (which can be implemented as simple shifts). See Section 1.12.5.1.3.

### 1.12.3.4 Comparing convergence rates

To compare the convergence of LMS, NLMS, and RLS, we show next two examples. In Example 1, we consider a system identification application and in Example 2, we use these algorithms to update the weights of an equalizer.

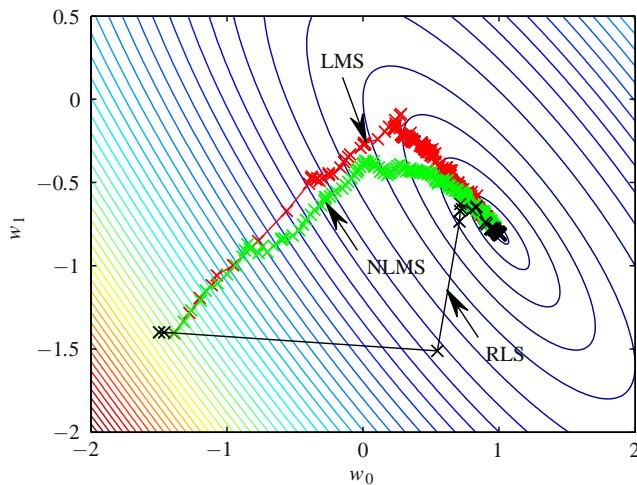
As we will see in more detail in Section 1.12.4, adaptive filters are compared based on how well they handle different classes of signals. For example, we could be interested in seeing how two filters handle white noise, or how they handle a certain level of correlation at the input. Given the stochastic nature of the inputs, the performance measures are defined in terms of averaged cost functions, most commonly the mean-square error (MSE), that is, the average of  $|e(n)|^2$  over all possible inputs in a given class. These averages can be computed theoretically, as we show in Section 1.12.4, or via simulations.

For example, the MSE as a function of time, that is, the curve  $E\{|e(n)|^2\} \times n$ , is called a filter's *learning curve*. A comparison of learning curves obtained theoretically and from simulations can be seen in Figure 12.30 further ahead. When simulations are used, one runs a filter  $L$  times for an interval of  $N$  samples, starting always from the same conditions. For each run  $\ell$ , the error  $e^{(\ell)}(n)$  is computed for  $1 \leq n \leq N$ , and one obtains the so-called *ensemble-average learning curve*

$$\widehat{E}(n) \triangleq \frac{1}{L} \sum_{\ell=1}^L |e^{(\ell)}(n)|^2.$$

Usually the ensemble-average learning curve will be a reasonable approximation of the true learning curve for a reasonably small value of  $L$  (say, from 50 to 1000), but in some situations this may not be true—see [45, 46] for examples and a detailed explanation.

**Example 1.** The aim of this example is to identify the system  $\mathbf{w}_o = [1 - 0.8]^T$ , assuming that the regressor  $\phi(n)$  is obtained from a process  $x(n)$  generated with a first-order autoregressive model, whose transfer function is  $1/(1 - 0.8z^{-1})$ . This model is fed with an i.i.d. Gaussian random process with unitary variance. Moreover, additive i.i.d. noise  $v(n)$  with variance  $\sigma_v^2 = 0.01$  is added to form the desired signal.

**FIGURE 12.25**

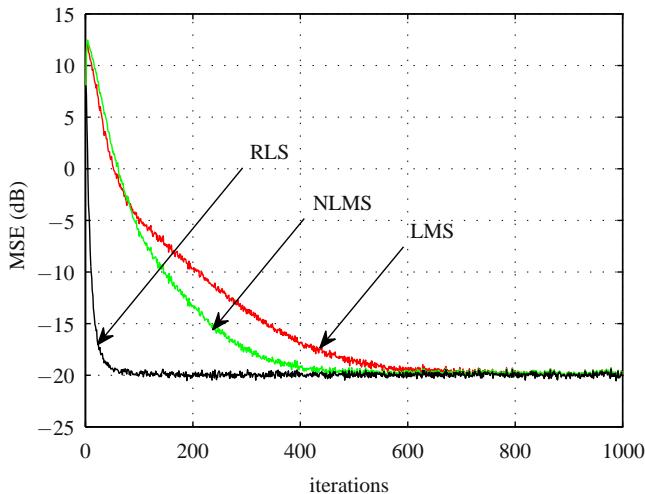
LMS ( $\mu = 0.01$ ), NLMS ( $\tilde{\mu} = 0.05$ ,  $\varepsilon = 0.5$ ) and RLS ( $\lambda = 0.99$ ,  $\delta = 1$ ) initialized at  $\mathbf{w}(0) = [-1.5 \ -1.4]^T$ ,  $\sigma_v^2 = 0.01$ . [Click](#) the animation video file.

Considering an adaptive filter with  $M = 2$  coefficients, we can obtain a contour plot of the mean-square error cost as a function of  $w_0$  and  $w_1$  as shown in Figure 12.25. The animations in Figure 12.25 illustrate the behavior of LMS ( $\mu = 0.01$ ), NLMS ( $\tilde{\mu} = 0.05$ ,  $\varepsilon = 0.5$ ), and RLS ( $\lambda = 0.99$ ,  $\delta = 1$ ) initialized at  $\mathbf{w}(0) = [-1.5 \ -1.4]^T$ . The correspondent curves of MSE along the iterations, estimated from the ensemble-average of 1000 independent runs, are shown in Figure 12.26. As expected for a colored input signal, RLS converges much faster than NLMS and LMS. NLMS converges faster than LMS and achieves the solution through a different path. To obtain a good behavior of NLMS in this example, we had to choose a relatively large regularization factor  $\varepsilon$ . This is always necessary when NLMS is used with few coefficients since in this case,  $\|\phi(n)\|^2$  has a high probability of becoming too close to zero and de-stabilizing the algorithm. This is shown in the analysis of Section 1.12.4.

**Example 2.** Now, we assume the transmission of a QPSK (quadrature phase shift-keying) signal, whose symbols belong to the alphabet  $\{\pm 1 \pm j1\}$ . This signal suffers the effect of the channel (taken from [47])

$$\begin{aligned} H(z) = & (-0.2 + j0.3) + (-0.5 + j0.4)z^{-1} + (0.7 - j0.6)z^{-2} \\ & +(0.4 + j0.3)z^{-3} + (0.2 + j0.1)z^{-4} \end{aligned} \quad (12.169)$$

and of additive white Gaussian noise (AWGN) under a signal-to-noise ratio (SNR) of 30 dB. The equalizer is an adaptive filter with  $M = 15$  coefficients initialized with zero and the desired signal is the transmitted sequence delayed by  $L = 9$  samples (go back to Figure 12.16 to see the equalization scheme). The adaptation parameters ( $\mu$ ,  $\tilde{\mu}$ ,  $\varepsilon$ ,  $\lambda$ ) were chosen in order to obtain the same steady-state performance of LMS, NLMS, and RLS (note that with a longer filter, the regularization parameter  $\varepsilon$  can be much

**FIGURE 12.26**

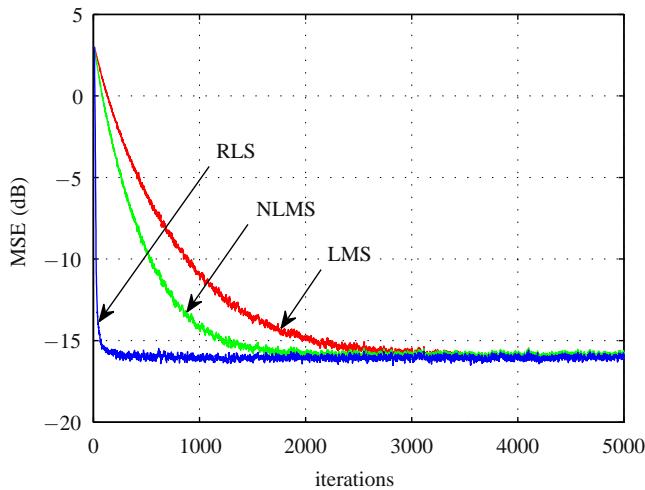
MSE along the iterations for LMS ( $\mu = 0.01$ ), NLMS ( $\hat{\mu} = 0.05$ ,  $\varepsilon = 0.5$ ) and RLS ( $\lambda = 0.99$ ,  $\delta = 1$ ) initialized at  $\mathbf{w}(0) = [-1.5 \ -1.4]^T$ ,  $\sigma_v^2 = 0.01$ , and ensemble-average of 1000 independent runs.

smaller than in the previous example). Figure 12.27 shows the MSE along the iterations, estimated from the ensemble-average of 1000 independent runs. Again, RLS presents the fastest convergence rate, followed by NLMS and LMS. These algorithms achieve the same steady-state MSE and therefore, present the same performance after convergence, which can be observed in Figure 12.28 where the equalizer output signal constellations are shown. The Matlab file used in this simulation is available at <http://www.lps.usp.br>.

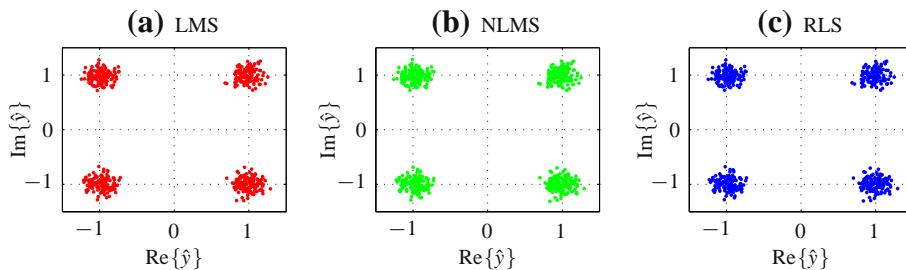
## 1.12.4 Statistical analysis

Closed-form expressions for the mean-square performance of adaptive filters facilitate comparisons between different algorithms and provide information about stability, convergence rate, mean-square error, and tracking capability. Since adaptive algorithms are obtained from stochastic approximations of the cost-functions, gradients and Hessians, their performance will degrade in comparison with the performance of the exact algorithms. Thus, stochastic analyses are also important to measure the performance degradation in relation to the exact algorithms.

The analysis of an adaptive algorithm constitutes a very hard problem, since adaptive filters are time-variant, stochastic, and nonlinear systems. Therefore, it is common to introduce simplifying approximations. They consist in most cases of disregarding the dependence between variables, for example, approximating the expected value of the product of two random variables, say,  $E\{xy\}$ , by the product of the expected values,  $E\{x\}E\{y\}$ . For historical reasons, these approximations are usually referred to as *assumptions* in the literature.

**FIGURE 12.27**

MSE along the iterations for LMS ( $\mu = 10^{-3}$ ), NLMS ( $\tilde{\mu} = 0.08$ ,  $\varepsilon = 10^{-5}$ ), and RLS ( $\lambda = 0.997$ ,  $\delta = 1$ ) initialized at  $\mathbf{w}(0) = \mathbf{0}$ ; QPSK,  $L = 9$ ,  $M = 15$ , and ensemble-average of 1000 independent runs; equalization of the channel (12.169).

**FIGURE 12.28**

Equalizer output signal constellations after convergence for (a) LMS ( $\mu = 10^{-3}$ ), (b) NLMS ( $\tilde{\mu} = 0.08$ ,  $\varepsilon = 10^{-5}$ ), and (c) RLS ( $\lambda = 0.997$ ,  $\delta = 1$ ); initialized at  $\mathbf{w}(0) = \mathbf{0}$ ; QPSK,  $L = 9$ ,  $M = 15$ ; equalization of the channel (12.169).

In some situations, in particular for LMS and some of its variants, the approximations are proved to converge to the exact solution in the limiting case of slow adaptation ( $\mu \rightarrow 0$  in the case of LMS), and their predictions tend to be reasonable for the range of step-sizes used in practice. We will not reproduce these proofs here, since they are in general very technical, but they can be found in [48–53] for the case of LMS, and [39] for more general filters.

The literature contains a few exact analytical results, but only for LMS, under very particular conditions [54–56]. In addition, the complexity of these exact models grows very quickly with the number of coefficients, so they are only useful for short filters (up to about five coefficients.).

The performance of an adaptive filter is usually described by evaluating the mean and the mean square (variance) of the weight error vector and of the error signal  $e(n)$ . This usually works very well for slow adaptation (i.e., small step-size). However, for fast adaptation, the behavior of an adaptive filter is considerably more complicated than can be seen by looking only at means and variances. Some of the phenomena that appear in LMS with large step-sizes are described in [46, 57]. References [45, 58–60] study algorithms with strong nonlinearities, for which the usual approximations are not always valid.

The analysis of NLMS was considered in [61–68]. Many works consider only tapped-delay line regressors, the case of Volterra filters is considered for example in [69–73]. For RLS filters, important models for infinite-precision arithmetic can be found in [74, 75]. The case of finite-precision arithmetic is discussed in Section 1.12.5.1.2.

The real importance of the approximations used in the adaptive filtering literature is that they lead to reasonably simple models that capture well the overall behavior of an adaptive filter. Simple models provide intuition about how a filter will behave if a given parameter is changed, which is important for a designer to make good decisions. Some approximations are made because without them the mathematical analysis would not be feasible, but others are used mainly because they lead to simpler models. Of course, there are situations in which it is important to study a filter in more detail, reducing the number of assumptions used in the analysis, as in the references just cited.

In this section we analyze three of the most common adaptive filters—LMS, NLMS, and RLS, starting with LMS in Section 1.12.4.3, and then showing how to extend these results to the other algorithms in Section 1.12.4.4. To start, we need to define some useful performance measures. The *mean-square error* (MSE) is simply the expected value of  $|e(n)|^2$ :

$$J(n) \triangleq E\{|e(n)|^2\}. \quad (12.170)$$

The most widely measure of performance used in the literature of adaptive filtering is the *excess mean-square error* (EMSE), which is defined as

$$\xi(n) \triangleq E\{|e_a(n)|^2\}, \quad (12.171)$$

where the *excess a priori error* is given by

$$e_a(n) = \tilde{\mathbf{w}}^H(n)\phi(n), \quad (12.172)$$

and the weight error vector by

$$\tilde{\mathbf{w}}(n) = \mathbf{w}_o(n) - \mathbf{w}(n). \quad (12.173)$$

Note that the optimal solution  $\mathbf{w}_o(n)$  is now allowed to be time-variant. Note also the difference between  $e_a(n)$  and  $e(n)$ , defined in (12.119). Both are *a priori* errors, but  $e_a(n)$  is computed with  $\tilde{\mathbf{w}}(n)$ , whereas  $e(n)$  is computed with  $\mathbf{w}(n)$ . The EMSE measures how much  $J(n)$  exceeds its minimum value  $J_{\min}(n)$  due to adaptation: if you were able to find the optimum filter coefficients  $\mathbf{w}_o(n)$  at each time, the EMSE would be zero. Since the actual filter coefficients are never exactly equal to the optimum values, the

EMSE measures the effect of this difference on the error variance. Closely related to the EMSE, the *misadjustment* is defined as

$$\mathcal{M}(n) \triangleq \zeta(n)/\sigma_v^2. \quad (12.174)$$

The *excess a posteriori error*  $e_p(n)$  is mostly used in the analysis and derivations of algorithms. It is the error obtained *after* using the current data  $\phi(n)$  to update the coefficient vector (thus the names *a priori* and *a posteriori*), i.e.,

$$e_p(n) = \tilde{\mathbf{w}}^H(n+1)\phi(n). \quad (12.175)$$

Again, note the difference with respect to  $\xi(n)$ , defined in (12.129).

Another common performance measure is the *mean-square deviation* (MSD). It measures how far  $\mathbf{w}(n)$  is from  $\mathbf{w}_o(n)$  in average:

$$\chi(n) \triangleq E\{\|\tilde{\mathbf{w}}(n)\|^2\}. \quad (12.176)$$

The EMSE is the most frequently performance measure used for comparisons between adaptive filters; however in some applications, particularly for system identification, the MSD is more adequate.

We now will discuss a general model for the input signals, considering a set of approximations that leads to reasonable, yet simple models.

#### 1.12.4.1 Data model

From the orthogonality principle (recall Section 1.12.2), the optimal solution  $\mathbf{w}_o(n)$  satisfies the property

$$E\{\phi(n)[d(n) - \mathbf{w}_o^H(n)\phi(n)]\} = \mathbf{0}, \quad (12.177)$$

i.e., the input regressor vector  $\phi(n)$  is uncorrelated with the optimal estimation error

$$v_o(n) \triangleq d(n) - \mathbf{w}_o^H(n)\phi(n), \quad (12.178)$$

whose mean is zero and whose variance  $\sigma_v^2(n) = E\{|v_o(n)|^2\}$  is equal to the minimum cost  $J_{\min}(n)$ . Thus, a linear regression model for  $d(n)$  holds, i.e.,

$$d(n) = \mathbf{w}_o^H(n)\phi(n) + v_o(n). \quad (12.179)$$

It then follows that

$$e(n) = d(n) - \mathbf{w}^H(n)\phi(n) = \tilde{\mathbf{w}}^H(n)\phi(n) + v_o(n) = e_a(n) + v_o(n), \quad (12.180)$$

and the MSE is given by

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} = E\{|e_a(n)|^2\} + 2E\{e_a(n)v_o(n)\} + E\{|v_o(n)|^2\} = \\ &= \zeta(n) + 2E\{e_a(n)v_o(n)\} + \sigma_v^2(n). \end{aligned}$$

The first approximation (or assumption) used in adaptive filtering analysis is to disregard the cross-term  $E\{e_a(n)v_o(n)\}$ . Since  $v_o(n)$  and  $\phi(n)$  are uncorrelated, but not necessarily independent, the cross-term does not vanish in general. From observations, however, one notes that the cross-term is usually negligible, compared to the other terms. In order to obtain a simple model for the adaptive filter, we will assume that  $\phi(n)$  and  $v_o(n)$  are independent of each other, and of previous samples  $\phi(n-k)$ ,  $v(n-k)$  for  $k > 0$ :

**Assumption 1.** The sequences  $\{\phi(n)\}$  and  $\{v_o(n)\}$  are independent and identically distributed (iid), and independent of each other (not only orthogonal). In addition, the sequence  $\{\phi(n)\}$  is assumed stationary and  $\{v_o(n)\}$  is zero-mean.

The last two assumptions are not in fact necessary, but they simplify the analysis. In general the sequence  $\{\phi(n)\}$  is not iid: for example if  $\phi(n)$  is formed from a tap-delay line or from a Volterra series expansion (see Eqs. (12.113) and (12.114))  $\phi(n)$  and  $\phi(n - 1)$  will have terms in common. However, assuming that the sequences are iid leads to simple models that in general are good approximations for the range of step-sizes usually found in practice, in particular for small step-sizes. The analysis based on Assumption 1 is known in the literature as *independence theory*.

Going back to the adaptive filter recursions (see Tables 12.2–12.5, and also Eq. (12.221) further ahead) we see that the current weight error  $\tilde{w}(n)$  is a function of past inputs  $\phi(n - 1), \phi(n - 2), \dots$  and noise samples  $v(n - 1), v(n - 2), \dots$ , but *not* of the current samples  $\phi(n)$  and  $v_o(n)$ . Therefore, under Assumption 1,  $\tilde{w}(n)$  is independent of  $v_o(n)$  and  $\phi(n)$ , so that

$$\text{E}\{e_a(n)v_o(n)\} = \text{E}\{\tilde{w}^H(n)\}\text{E}\{\phi(n)\}\text{E}\{v_o(n)\} = 0.$$

We conclude that under Assumption 1, the MSE  $J(n) = \text{E}\{|e(n)|^2\}$  is related to the EMSE via

$$J(n) = \zeta(n) + J_{\min}(n) = \zeta(n) + \sigma_v^2, \quad (12.181)$$

since Assumption 1 implies that  $J_{\min}(n)$  is a constant.

We also need a model for the variation of the optimal solution. Again, our choice is a model that captures the main consequences of a non-constant  $w_o(n)$ , without unnecessarily complicating the analysis.

**Assumption 2 (Random-walk model).** In a nonstationary environment,  $w_o$  varies as

$$w_o(n + 1) = w_o(n) + q(n), \quad (12.182)$$

where  $q(n)$  is zero-mean i.i.d vector with positive-definite autocorrelation matrix  $Q = \text{E}\{q(n)q^H(n)\}$ , independent of the initial condition  $w(0)$ , and of  $\{\phi(\ell), v(\ell)\}$  for all  $\ell$ .

The covariance matrix of  $w_o$ , according to (12.182), grows slowly to infinity. A more general model for the variation of  $w_o$  uses the recursion

$$\begin{cases} w_o(n + 1) = w_o(n) + \theta(n), \\ \theta(n) = \alpha\theta(n - 1) + q(n), \quad 0 \leq |\alpha| < 1, \end{cases}$$

whose covariance matrix remains bounded, but the analysis becomes more complicated [3]. Therefore, it is common in the literature to adopt the model (12.182). Note that both models assume that  $\{x(n)\}$  is a stationary random sequence, although  $w_o(n)$  is allowed to vary with  $n$ .

#### 1.12.4.2 Relating the covariance matrix of the weight-error vector to the EMSE and the MSD

The covariance matrix of the weight-error vector is defined as

$$S(n) \triangleq \text{E}\left\{\tilde{w}(n)\tilde{w}^H(n)\right\}. \quad (12.183)$$

This matrix is important in the statistical analysis of adaptive filters since it is related to the MSD and the EMSE as we show next.

Recall that the MSD is given by

$$\chi(n) = \mathbb{E} \left\{ \|\tilde{\mathbf{w}}(n)\|^2 \right\} = \sum_{i=0}^{M-1} \mathbb{E} \left\{ \tilde{w}_i^2(n) \right\}.$$

Now, the terms  $\mathbb{E}\{\tilde{w}_i^2(n)\}$  are exactly the terms that appear in the diagonal of  $\mathbf{S}(n)$ , therefore, recalling that the *trace* of a square matrix is the sum of its main-diagonal elements (see Box 3), we obtain

$$\chi(n) = \sum_{i=0}^{M-1} \mathbb{E}\{\tilde{w}_i^2(n)\} = \sum_{m=0}^{M-1} s_{mm}(n) = \text{Tr}(\mathbf{S}(n)). \quad (12.184)$$

Under Assumption 1,  $\phi(n)$  and  $\tilde{\mathbf{w}}(n)$  are independent, as we just saw. Therefore,

$$\begin{aligned} \xi(n) &= \mathbb{E}\{|\tilde{\mathbf{w}}^H(n)\phi(n)|^2\} = \mathbb{E}\{\tilde{\mathbf{w}}^H(n)\phi(n)(\tilde{\mathbf{w}}^H(n)\phi(n))^*\} \\ &= \mathbb{E}\{\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)\} = \mathbb{E} \left\{ \mathbb{E}\{\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)|\tilde{\mathbf{w}}(n)\} \right\} \\ &= \mathbb{E} \left\{ \tilde{\mathbf{w}}^H(n) \mathbb{E}\{\phi(n)\phi^H(n)|\tilde{\mathbf{w}}(n)\} \tilde{\mathbf{w}}(n) \right\} = \mathbb{E}\{\tilde{\mathbf{w}}^H(n)\mathbf{R}_\phi\tilde{\mathbf{w}}(n)\}, \end{aligned}$$

where we used Assumption 1 to write  $\mathbb{E}\{\phi(n)\phi^H(n)\} = \mathbf{R}_\phi$  (constant), and two properties of expected values. The first is that, for any two random variables  $a, b$ ,

$$\mathbb{E}\{ab\} = \mathbb{E}\{\mathbb{E}\{ab|b\}\} = \mathbb{E}\{\mathbb{E}\{a|b\}b\},$$

where the inner expected value is taken with respect to  $a$ , and the second with respect to  $b$ . The second property is that, if  $a$  and  $b$  are independent, then  $\mathbb{E}\{a|b\}$  does not depend on  $b$ , i.e.,  $\mathbb{E}\{a|b\} = \mathbb{E}\{a\}$ .

To proceed, we apply a useful trick from matrix theory. You can check by direct computation that, if  $\mathbf{A} \in \mathcal{C}^{K \times L}$  and  $\mathbf{B} \in \mathcal{C}^{L \times K}$ , then  $\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA})$ . Applying this property with  $\mathbf{A} \leftarrow \tilde{\mathbf{w}}^H(n)$  and  $\mathbf{B} \leftarrow \mathbf{R}_\phi\tilde{\mathbf{w}}(n)$ , we obtain

$$\xi(n) = \text{Tr} \left( \mathbf{R}_\phi \mathbb{E}\{\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\} \right) = \text{Tr} \left( \mathbf{R}_\phi \mathbf{S}(n) \right). \quad (12.185)$$

Since  $\mathbf{R}_\phi$  is symmetric, there always exists an orthogonal matrix  $\mathbf{U}$  (i.e.,  $\mathbf{U}^{-1} = \mathbf{U}^H$ ) that diagonalizes  $\mathbf{R}_\phi$ , that is,

$$\mathbf{U}^H \mathbf{R}_\phi \mathbf{U} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_M \end{bmatrix} \triangleq \boldsymbol{\Lambda}, \quad (12.186)$$

where  $\lambda_1, \lambda_2, \dots, \lambda_M$  are the eigenvalues of  $\mathbf{R}_\phi$  (see Box 3). Defining the rotated matrix

$$\mathbf{S}'(n) \triangleq \mathbf{U}^H \mathbf{S}(n) \mathbf{U}$$

and recalling that  $\mathbf{U}\mathbf{U}^H = \mathbf{I}$ , (12.184) and (12.185) can be rewritten respectively as

$$\chi(n) = \text{Tr}(\mathbf{U} \underbrace{\mathbf{U}^H \mathbf{S}(n) \mathbf{U}}_{\mathbf{S}'(n)} \mathbf{U}^H) = \text{Tr}(\mathbf{U} \mathbf{S}'(n) \mathbf{U}^H) \quad (12.187)$$

and

$$\zeta(n) = \text{Tr}(\mathbf{U} \underbrace{\mathbf{U}^H \mathbf{R}_\phi \mathbf{U}}_{\Lambda} \underbrace{\mathbf{U}^H \mathbf{S}(n) \mathbf{U}}_{\mathbf{S}'(n)} \mathbf{U}^H) = \text{Tr}(\mathbf{U} \Lambda \mathbf{S}'(n) \mathbf{U}^H). \quad (12.188)$$

Using again the fact that  $\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA})$ , with  $\mathbf{A} \leftarrow \mathbf{U}$ , we obtain

$$\chi(n) = \text{Tr}(\mathbf{S}'(n)) \quad (12.189)$$

and

$$\zeta(n) = \text{Tr}(\Lambda \mathbf{S}'(n)). \quad (12.190)$$

Note that both the MSD (12.189) and the EMSE (12.190) depend only on the diagonal entries of  $\mathbf{S}'(n)$ . We can work therefore only with these diagonal entries, defining the vectors

$$\mathbf{s}'(n) = \text{diag}(\mathbf{S}'(n)) = [s'_0(n) s'_1(n) \cdots s'_{M-1}(n)]^T, \quad (12.191)$$

and

$$\mathbf{l} = [\lambda_0, \dots, \lambda_{M-1}]^T, \quad (12.192)$$

where  $\text{diag}(\mathbf{A})$  represents a column vector with the diagonal elements of  $\mathbf{A}$ , and evaluating the MSD and EMSE respectively as

$$\chi(n) = \sum_{k=0}^{M-1} s'_k(n) \quad (12.193)$$

and

$$\zeta(n) = \mathbf{l}^T \mathbf{s}'(n) = \sum_{k=0}^{M-1} \lambda_k s'_k(n). \quad (12.194)$$

Again, the MSD and EMSE depend only on the elements of the diagonal of the rotated matrix  $\mathbf{S}'(n)$ . In the case of EMSE, these elements are weighed by the eigenvalues of the autocorrelation matrix  $\mathbf{R}_\phi$ . As we shall see in Section 1.12.4.3, the stability of LMS can be studied through a recursion for the vector  $\mathbf{s}'(n)$ .

Thus, according to the previous results, in order to obtain closed-form expressions for the MSD and the EMSE of an adaptive algorithm, one path is to find a recursion for the covariance matrix  $\mathbf{S}(n)$  or for the vector  $\mathbf{s}'(n)$ . However, this is not the only method to analyze adaptive algorithms, as we shall see in Section 1.12.4.4.

### 1.12.4.3 Statistical analysis of the LMS algorithm

In this section, we present a statistical analysis for the LMS algorithm, assuming that the optimal solution  $\mathbf{w}_o$  remains constant ( $\mathbf{q}(n) \equiv \mathbf{0}$ , for all  $n$ ). This analysis predicts the behavior of LMS in steady-state and in transient, and provides a range of the step-size  $\mu$  for which LMS operates adequately. If you are familiar with the LMS analysis, go to Section 1.12.4.4, where we present a unified analysis for LMS, NLMS, and RLS algorithms in a nonstationary environment.

We first rewrite (12.123) in terms of the weight-error vector  $\tilde{\mathbf{w}}$ . Thus, subtracting both sides of (12.123) from  $\mathbf{w}_o$ , we obtain

$$\begin{aligned}\mathbf{w}_o - \mathbf{w}(n+1) &= \mathbf{w}_o - \mathbf{w}(n) - \mu\phi(n)e^*(n), \\ \tilde{\mathbf{w}}(n+1) &= \tilde{\mathbf{w}}(n) - \mu\phi(n)e^*(n).\end{aligned}\quad (12.195)$$

Replacing  $e^*(n) = e_a^*(n) + v_o^*(n) = \phi^H(n)\tilde{\mathbf{w}}(n) + v_o^*(n)$  in (12.195), we arrive at

$$\tilde{\mathbf{w}}(n+1) = \left[ \mathbf{I} - \mu\phi(n)\phi^H(n) \right] \tilde{\mathbf{w}}(n) - \mu\phi(n)v_o^*(n). \quad (12.196)$$

**First-order analysis.** Taking expectations on both sides of (12.196), we obtain

$$\begin{aligned}\mathbb{E}\{\tilde{\mathbf{w}}(n+1)\} &= \mathbb{E}\left\{\left[ \mathbf{I} - \mu\phi(n)\phi^H(n) \right] \tilde{\mathbf{w}}(n)\right\} - \mu\mathbb{E}\{\phi(n)v_o^*(n)\} \\ &= \mathbb{E}\left\{\left[ \mathbf{I} - \mu\phi(n)\phi^H(n) \right] \tilde{\mathbf{w}}(n)\right\},\end{aligned}\quad (12.197)$$

where the last equality follows from the fact that  $\phi(n)$  is orthogonal to  $v_o^*(n)$ , according to the orthogonality condition (see Eq. (12.96)). Since  $\tilde{\mathbf{w}}(n)$  is assumed to be independent of  $\phi(n)$  (see Assumption 1), we arrive at

$$\mathbb{E}\{\tilde{\mathbf{w}}(n+1)\} = [\mathbf{I} - \mu\mathbf{R}_\phi]\mathbb{E}\{\tilde{\mathbf{w}}(n)\}. \quad (12.198)$$

This is the same recursion we saw in Box 1. According to linear systems theory, this recursion converges as long as the largest eigenvalue of  $\mathbf{A} = \mathbf{I} - \mu\mathbf{R}_\phi$  has absolute value less than one, which implies (see Box 1)

$$\begin{aligned}|1 - \mu\lambda| < 1 &\Leftrightarrow 0 < \mu < \frac{2}{\lambda_k}, \quad \forall k \Leftrightarrow \\ 0 < \mu &< \frac{2}{\lambda_{\max}},\end{aligned}\quad (12.199)$$

where  $\lambda_k$ ,  $k = 1, 2, \dots, M$  are the eigenvalues of  $\mathbf{R}_\phi$ , and  $\lambda_{\max}$  is its maximum eigenvalue. We should notice that this range for  $\mu$  does not ensure the stability of LMS, since it ensures the convergence of the mean of  $\tilde{\mathbf{w}}(n) = \mathbf{w}_o - \mathbf{w}(n)$  towards 0, but the autocovariance of  $\tilde{\mathbf{w}}(n)$  may still be diverging. We show next that the range for  $\mu$  based on a second-order analysis is indeed more restrict.

**Second-order analysis.** Now, we use (12.196) to find a recursion for  $S(n)$  and (12.185) to evaluate the EMSE of LMS. Multiplying (12.196) by its Hermitian, taking the expectations of both sides, we arrive at

$$\begin{aligned}
 E\{\tilde{\mathbf{w}}(n+1)\tilde{\mathbf{w}}^H(n+1)\} &= E\{\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\} \\
 &\quad - \overbrace{\mu E\{\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\}}^{\mathcal{A}} \\
 &\quad - \overbrace{\mu E\{\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\}}^{\mathcal{B}} \\
 &\quad + \overbrace{\mu^2 E\{\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\}}^{\mathcal{C}} \\
 &\quad + \overbrace{\mu^2 E\{|v_o(n)|^2 \phi(n)\phi^H(n)\}}^{\mathcal{D}} \\
 &\quad - \overbrace{\mu E\{v_o(n)\tilde{\mathbf{w}}(n)\phi^H(n)\}}^{\mathcal{E}} \\
 &\quad - \overbrace{\mu E\{v_o^*(n)\phi(n)\tilde{\mathbf{w}}^H(n)\}}^{\mathcal{F}} \\
 &\quad + \overbrace{\mu^2 E\{v_o(n)\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)\phi^H(n)\}}^{\mathcal{G}} \\
 &\quad + \overbrace{\mu^2 E\{v_o^*(n)\phi(n)\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\}}^{\mathcal{H}}. \tag{12.200}
 \end{aligned}$$

Now, using Assumption 1, we can evaluate the terms  $\mathcal{A} - \mathcal{H}$  of (12.200):

**A-** Recalling that Assumption 1 implies that  $\tilde{\mathbf{w}}(n)$  is independent from  $\phi(n)$ , the term  $\mathcal{A}$  can be approximated by

$$\begin{aligned}
 \mathcal{A} &= \mu E\left\{E\left\{\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)|\phi(n)\right\}\right\} \\
 &\approx \mu E\left\{E\left\{\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\right\}\phi(n)\phi^H(n)\right\} \\
 &= \mu S(n)E\left\{\phi(n)\phi^H(n)\right\} = \mu S(n)\mathbf{R}_\phi. \tag{12.201}
 \end{aligned}$$

**B-** Analogously, we obtain for  $\mathcal{B}$

$$\mathcal{B} \approx \mu \mathbf{R}_\phi S(n). \tag{12.202}$$

**C-** Using Assumption 1, the term  $\mathcal{C}$  can be rewritten as

$$\begin{aligned}
 \mathcal{C} &= \mu^2 E\left\{\phi(n)\phi^H(n)\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\phi(n)\phi^H(n)\right\} \\
 &\approx \mu^2 E\left\{\phi(n)\phi^H(n)S(n)\phi(n)\phi^H(n)\right\}. \tag{12.203}
 \end{aligned}$$

We return to this term further on.

$\mathcal{D}$ - Using the fact that  $v_o(n)$  is independent of  $\phi(n)$  and  $\tilde{\mathbf{w}}(n)$ ,  $\mathcal{D}$  reduces to

$$\mathcal{D} = \mu^2 E\{|v_o(n)|^2 \phi(n) \phi^H(n)\} \approx \mu^2 \sigma_v^2 \mathbf{R}_\phi. \quad (12.204)$$

$\mathcal{E}$ - Using the fact that  $v_o(n)$  is a zero-mean random variable, which is independent of  $\phi(n)$  and  $\tilde{\mathbf{w}}(n)$ ,  $\mathcal{E}$  is a  $M \times M$  null matrix. Using the same arguments, the terms  $\mathcal{F}$ ,  $\mathcal{G}$ , and  $\mathcal{H}$  are also null matrices. We should point out that the zero-mean assumption for  $v_o(n)$  is not necessary, but it really facilitates the analysis. Furthermore, as we saw in Section 1.12.2.1.4, we can enforce  $E\{v_o(n)\} = 0$  using simple methods.

From the previous results, (12.200) reduces to

$$\begin{aligned} \mathbf{S}(n+1) &\approx \mathbf{S}(n) - \mu \mathbf{S}(n) \mathbf{R}_\phi - \mu \mathbf{R}_\phi \mathbf{S}(n) \\ &\quad + \underbrace{\mu^2 E\{\phi(n) \phi^H(n) S(n) \phi(n) \phi^H(n)\}}_{\mathcal{C}} + \mu^2 \sigma_v^2 \mathbf{R}_\phi. \end{aligned} \quad (12.205)$$

This recursion is convenient to obtain closed-form expressions for the EMSE and the MSD of LMS. Observe that the four first terms on the right-hand side of (12.205) are linear in  $\mathbf{S}(n)$ . Assuming that  $\mu$  is sufficiently small, the term  $\mathcal{C}$  can be neglected with respect to the three first terms. Thus, for small step-sizes, (12.205) reduces to

$$\mathbf{S}(n+1) \approx \mathbf{S}(n) - \mu [S(n) \mathbf{R}_\phi + \mathbf{R}_\phi S(n)] + \mu^2 \sigma_v^2 \mathbf{R}_\phi, \quad (12.206)$$

with initialization  $\mathbf{S}(0) = [\mathbf{w}_o - \mathbf{w}(0)] [\mathbf{w}_o - \mathbf{w}(0)]^H$ .

Since we can obtain the EMSE and the MSD from  $\mathbf{S}(n)$  and  $\mathbf{R}_\phi$  through (12.185) and (12.184), we should use recursion (12.206) to compute  $\mathbf{S}(n)$  and then, evaluate  $\zeta(n) = \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi)$  and  $\chi(n) = \text{Tr}(\mathbf{S}(n))$ . Particularly for  $n \rightarrow \infty$ , since  $\mathbf{S}(n+1) \approx \mathbf{S}(n)$ , we obtain

$$\mathbf{S}(\infty) \mathbf{R}_\phi + \mathbf{R}_\phi \mathbf{S}(\infty) = \mu \sigma_v^2 \mathbf{R}_\phi. \quad (12.207)$$

Taking the trace of both sides of (12.207), we arrive at an analytical expression for the steady-state EMSE of LMS, i.e.,

$$\zeta^{\text{LMS}}(\infty) \approx \frac{\mu \sigma_v^2 \text{Tr}(\mathbf{R}_\phi)}{2} \quad (\text{for sufficiently small } \mu). \quad (12.208)$$

This expression indicates that the EMSE of LMS increases with  $\mu$  and with the length of the filter. The dependence with the filter length is through the trace of  $\mathbf{R}_\phi$ . It is more evident when the class of filters we are considering is FIR, that is, when  $\phi(n) = [x(n) \dots x(n-M+1)]^T$  (a tapped-delay line). In this case, if  $\{x(n)\}$  is stationary, then  $\text{Tr}(\mathbf{R}_\phi) = M E\{x^2(n)\}$ ,  $M$  times the average power of  $x(n)$ .

Turning now to the filter coefficients, if we multiply both sides of (12.207) from the right by  $\mathbf{R}_\phi^{-1}$ , taking the trace of both sides, and recalling that  $\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA})$ , we obtain an expression for the steady-state MSD of LMS, i.e.,

$$\chi^{\text{LMS}}(\infty) \approx \frac{\mu \sigma_v^2 M}{2} \quad (\text{for sufficiently small } \mu). \quad (12.209)$$

Again, the smaller the step-size  $\mu$  and the filter length  $M$ , the smaller the MSD of LMS.

These results explain what we observed in the simulations in Section 1.12.1.2 (Figures 12.9 and 12.10): for larger values of  $\mu$  the filter converges faster, but hovers around the optimum solution in a larger region. The size of this region is what the MSD attempts to measure—the EMSE measures how much this hovering affects the error.

These quantities are important in practical situations. For example, in an actual implementation of an acoustic echo canceler, when the near-end user is speaking, his voice is part of  $v(n)$ . Since the near-end speech is normally stronger than the echo,  $\sigma_v^2$  tends to be quite large when the near-end speaker is talking. If the adaptive filter were allowed to keep adapting at these moments, (12.208) shows that the filter estimates would wander too far from the optimum. To avoid this problem, the step-size is reduced to zero whenever the near-end user is speaking. A *double-talk detector* is used to decide when to stop adaptation.

It can be shown that (12.206) is stable for sufficiently small  $\mu$ . However, (12.206) cannot be used to find a range of values of  $\mu$  that guarantee stability, due to the approximations made, particularly the discarding of term  $\mathcal{C}$ . Thus, in order to study the stability of LMS, we consider next an approximation for this term, based the assumption that the input vector is Gaussian. This assumption allows us to approximate  $\mathcal{C}$  as

$$\mathcal{C} \approx \mathbf{R}_\phi \text{Tr}(S(n)\mathbf{R}_\phi) + \beta \mathbf{R}_\phi S(n)\mathbf{R}_\phi, \quad (12.210)$$

where as usual,  $\beta = 1$  (resp.,  $\beta = 2$ ) for complex (resp., real) data. See Box 7 for the derivation of (12.210).

Replacing (12.210) in (12.205), we arrive at

$$\begin{aligned} S(n+1) &\approx S(n) - \mu [S(n)\mathbf{R}_\phi + \mathbf{R}_\phi S(n)] \\ &\quad + \mu^2 [\mathbf{R}_\phi \text{Tr}(S(n)\mathbf{R}_\phi) + \beta \mathbf{R}_\phi S(n)\mathbf{R}_\phi + \sigma_v^2 \mathbf{R}_\phi]. \end{aligned} \quad (12.211)$$

The stability of (12.211) is determined through a recursion for the vector  $s'(n)$ , which contains the elements of the diagonal of the rotated matrix  $S'(n) = \mathbf{U}^H S(n) \mathbf{U}$  (see Section 1.12.4.2). A recursion for  $S'(n)$  can be obtained by multiplying (12.211) from the left by  $\mathbf{U}^H$  and from the right by  $\mathbf{U}$  and recalling that  $\mathbf{U}\mathbf{U}^H = \mathbf{I}$ , which leads to

$$\begin{aligned} S'(n+1) &\approx S'(n) - \mu [S'(n)\Lambda + \Lambda S'(n)] \\ &\quad + \mu^2 [\Lambda \text{Tr}(\Lambda S'(n)) + \beta \Lambda S'(n)\Lambda] + \mu^2 \sigma_v^2 \Lambda, \end{aligned} \quad (12.212)$$

where  $\Lambda$  is defined as in (12.186). Thus, a recursion for  $s'(n+1)$  is given by

$$s'(n+1) = [\mathbf{I} - 2\mu\Lambda + \mu^2 (\beta\Lambda^2 + \mathbf{U}^T)] s'(n) + \mu^2 \sigma_v^2 \mathbf{I}, \quad (12.213)$$

where  $\mathbf{I}$  is defined in (12.192).

The system matrix  $A$  of (12.213) is given by

$$A = \mathbf{I} - 2\mu\Lambda + \mu^2 (\beta\Lambda^2 + \mathbf{U}^T) = (\mathbf{I} - \mu\Lambda)^2 + \mu^2(\beta - 1)\Lambda^2 + \mu^2 \mathbf{U}^T.$$

Although it is possible to obtain the exact values of all eigenvalues of  $\mathbf{A}$ , as done in [36], we follow here a much simpler, approximate route. We can use Fact 7 in Box 3 to find an upper bound for the largest eigenvalue  $\nu_i$  of  $\mathbf{A}$ , using the 1-norm of  $\mathbf{A}$ :

$$\max_{0 \leq i \leq M-1} |\nu_i| \leq \|\mathbf{A}\|_1 = \max_{0 \leq i \leq M-1} \sum_{j=0}^{M-1} |a_{ij}|,$$

where  $a_{ij}$  are the entries of  $\mathbf{A}$ . If we find a range of  $\mu$  for which  $\|\mathbf{A}\|_1 \leq 1$ , then the recursion will be stable. This will be of course a conservative condition (sufficient, but not necessary).

Now note that the  $i$ th column of  $\mathbf{A}$  has entries  $\lambda_i \lambda_j$  if  $i \neq j$ , and  $(1 - \mu \lambda_i)^2 + \mu^2(\beta - 1)\lambda_i^2 + \mu^2\lambda_i^2$  in the term corresponding to the diagonal. Since all terms are positive, the 1-norm is

$$\begin{aligned} & \max_{0 \leq i \leq M-1} \left[ (1 - \mu \lambda_i)^2 + \mu^2(\beta - 1)\lambda_i^2 + \mu^2\lambda_i \sum_{j=0}^{M-1} \lambda_j \right] \\ &= \max_{0 \leq i \leq M-1} \left[ (1 - \mu \lambda_i)^2 + \mu^2(\beta - 1)\lambda_i^2 + \mu^2\lambda_i \text{Tr}(\mathbf{\Lambda}) \right]. \end{aligned}$$

Recall that  $\text{Tr}(\mathbf{\Lambda}) = \text{Tr}(\mathbf{R}_\phi)$  (Fact 1 from Box 3). Therefore, the recursion will be stable if

$$1 - 2\mu\lambda_i + \mu^2\lambda_i^2 + \mu^2(\beta - 1)\lambda_i^2 + \mu^2\lambda_i \text{Tr}(\mathbf{R}_\phi) \leq 1, \quad 0 \leq i \leq M - 1.$$

Simplifying, we obtain the condition

$$\mu \leq \frac{2}{\beta\lambda_i + \text{Tr}(\mathbf{R}_\phi)}, \quad 0 \leq i \leq M - 1.$$

The smallest bound is for  $\lambda_i = \lambda_{\max}$ :

$$\mu \leq \frac{2}{\beta\lambda_{\max} + \text{Tr}(\mathbf{R}_\phi)}. \quad (12.214)$$

If we replace  $\lambda_{\max}$  by  $\text{Tr}(\mathbf{R}_\phi) \geq \lambda_{\max}$  in the denominator, we obtain a simpler, but more conservative condition:

$$\mu \leq \frac{2}{(\beta + 1)\text{Tr}(\mathbf{R}_\phi)}. \quad (12.215)$$

For real data, this range reduces to

$$0 < \mu < \frac{2}{3\text{Tr}(\mathbf{R}_\phi)} \quad (\text{real data}) \quad (12.216)$$

and for complex data to

$$0 < \mu < \frac{1}{\text{Tr}(\mathbf{R}_\phi)} \quad (\text{complex data}). \quad (12.217)$$

**Table 12.6** Parameters of LMS, NLMS, and RLS Algorithms

Algorithm	$\rho(n)$	$\mathbf{M}^{-1}(n)$
LMS	$\mu$	$\mathbf{I}$
NLMS	$\frac{\tilde{\mu}}{\varepsilon + \ \boldsymbol{\phi}(n)\ ^2}$	$\mathbf{I}$
RLS	$1 - \lambda$	$(1 - \lambda) \sum_{\ell=1}^n \lambda^{n-\ell} \boldsymbol{\phi}(\ell) \boldsymbol{\phi}^H(\ell)$

If the input is a tapped-delay line as in (12.113),  $\text{Tr}(\mathbf{R}_\phi) = \sum_{k=0}^{M-1} \mathbb{E}\{|x(n-k)|^2\}$ . If  $\{x(n)\}$  is a stationary sequence, the stability condition for LMS is easy to obtain: we need to choose  $\mu$  in the range

$$0 < \mu < \frac{2}{(\beta + 1) \times M \times (\text{average power of } x(n))}. \quad (12.218)$$

This concludes our stability analysis for LMS. We now show how to extend these results to NLMS and RLS, in a unified way. At the end of the next section, we give examples comparing the models to simulated results.

#### 1.12.4.4 A unified statistical analysis

In this section, we assume a nonstationary environment, where the optimal solution varies according to the random walk model (see Assumption 2). We start by obtaining a general update equation for the LMS, NLMS, and RLS algorithms, in order to provide a unified statistical analysis for these algorithms.

##### 1.12.4.4.1 A general update equation

In order to provide a unified analysis for the LMS, NLMS and RLS algorithms, we consider the following general update equation

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \rho(n) \mathbf{M}(n) \boldsymbol{\phi}(n) e^*(n), \quad (12.219)$$

where  $\rho(n)$  is a step-size,  $\mathbf{M}(n)$  is a non-singular matrix with hermitian symmetry ( $\mathbf{M}^H(n) = \mathbf{M}(n)$ ), and  $e(n) = d(n) - \mathbf{w}^H(n) \boldsymbol{\phi}(n)$  is the estimation error. Many adaptive algorithms can be written as in (12.219), by proper choices of  $\rho(n)$  and  $\mathbf{M}(n)$ . The LMS, NLMS, and RLS algorithms employ the step-sizes  $\rho(n)$  and the matrices  $\mathbf{M}(n)$  as in Table 12.6, where  $\mathbf{I}$  is the  $M \times M$  identity matrix,  $\mu$  and  $0 < \tilde{\mu} < 2$  are step-sizes, and  $0 \ll \lambda < 1$  is a forgetting factor. For RLS,  $\mathbf{M}(n) = (1 - \lambda)^{-1} \mathbf{P}(n)$  (see Eq. (12.162)). Note that, since  $(1 - \lambda)$  plays the role of step-size in RLS, we multiply the matrix  $\mathbf{P}(n)$  by  $(1 - \lambda)^{-1}$ , keeping the update equation of the RLS weights as in (12.167).

We must rewrite (12.219) in terms of the weight-error vector  $\tilde{\mathbf{w}}$ . Thus, subtracting both sides of (12.219) from  $\mathbf{w}_o(n+1) = \mathbf{w}_o(n) + \mathbf{q}(n)$  (Eq. (12.182)), we obtain

$$\begin{aligned} \mathbf{w}_o(n+1) - \mathbf{w}(n+1) &= \mathbf{w}_o(n) + \mathbf{q}(n) - \mathbf{w}(n) - \rho(n) \mathbf{M}(n) \boldsymbol{\phi}(n) e^*(n), \\ \tilde{\mathbf{w}}(n+1) &= \tilde{\mathbf{w}}(n) - \rho(n) \mathbf{M}(n) \boldsymbol{\phi}(n) e^*(n) + \mathbf{q}(n). \end{aligned} \quad (12.220)$$

Replacing  $e^*(n) = e_a^*(n) + v^*(n) = \boldsymbol{\phi}^H(n)\tilde{\mathbf{w}}(n) + v^*(n)$  in (12.220), we arrive at

$$\tilde{\mathbf{w}}(n+1) = \left[ \mathbf{I} - \rho(n)\mathbf{M}(n)\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n) \right] \tilde{\mathbf{w}}(n) - \rho(n)\mathbf{M}(n)\boldsymbol{\phi}(n)v^*(n) + \mathbf{q}(n). \quad (12.221)$$

We next use recursion (12.221) to evaluate the EMSE and the MSD of LMS, NLMS, and RLS.

#### 1.12.4.4.2 Alternative analysis methods

Closed-form expressions for the EMSE and MSD can be obtained in two ways:

- i. Finding a recursion for the correlation matrix of the weight-error vector, denoted by  $\mathbf{S}(n)$ . The EMSE and MSE can be easily computed given  $\mathbf{S}(n)$ , as we saw in Section 1.12.4.2. This is the earlier approach in the literature [1,2] and therefore, we refer to it here as *traditional* method.
- ii. Using the energy conservation analysis of [3,76–78]. The energy conservation approach relies on an energy conservation relation that holds for a large class of adaptive filters. The energy conservation method presents some advantages when compared to the traditional one. For example, using energy conservation, one can obtain steady-state results directly, bypassing several of the difficulties encountered in obtaining them as a limiting case of a transient analysis. This method can also be used in transient analyses of adaptive algorithms (see, e.g., Part V of [3]).

It is important to understand that both methods involve different sets of approximations (assumptions), so they do not always arrive at the same expressions. Generally, they tend to be equal in the limit of slow adaptation (i.e., if  $\rho(n) \rightarrow 0$  in (12.221)). For larger values of  $\rho(n)$ , which method will provide the best approximation depends on the algorithm. As for the work necessary to arrive at the final answers, the energy conservation approach is almost always easier to use if one is interested only in steady-state results. If one is interested in transient performance and stability studies, then the traditional method may be simpler, depending on the algorithm under study.

To give an overview of both methods, we first use the traditional method to obtain the transient and steady-state performance of the LMS, NLMS, and RLS algorithms. This is an extension of what we did for LMS in Section 1.12.4.3, taking advantage of the similarities between the algorithms. In the sequel, we employ the energy conservation method to obtain closed-form expressions for the steady-state EMSE of the same algorithms.

In this section we allow the environment to be nonstationary, i.e., the optimal solution  $\mathbf{w}_o$  is time-variant. The results for stationary environments can be obtained simply by making  $\mathbf{Q} = \mathbf{0}$  in the equations below. In order to simplify the arguments, we require that the regressor sequence  $\{\boldsymbol{\phi}(n)\}$  be wide-sense stationary, that is, we assume that  $\mathbf{R}_{\boldsymbol{\phi}}$  is constant.

#### 1.12.4.4.3 Analysis with the traditional method

The traditional analysis method consists of finding a recursion for  $\mathbf{S}(n)$ , and using (12.184) and (12.185) to evaluate the MSD and the EMSE, respectively. A recursion for  $\mathbf{S}(n)$  can be obtained by multiplying (12.221) by its Hermitian, taking the expectations of both sides, and recalling that Assumption 1 implies that  $\tilde{\mathbf{w}}(n)$  is independent of  $\boldsymbol{\phi}(n)$  and of the zero-mean  $v_o(n)$ . Furthermore, using the fact that

$E\{\tilde{w}(n)q^H(n)\} = \mathbf{0}$  since the sequence  $\{q(n)\}$  is iid (Assumption 2), we arrive at

$$\begin{aligned}
 E\{\tilde{w}(n+1)\tilde{w}^H(n+1)\} &= E\{q(n)q^H(n)\} + E\{\tilde{w}(n)\tilde{w}^H(n)\} \\
 &\quad - \overbrace{E\{\rho(n)\tilde{w}(n)\tilde{w}^H(n)\phi(n)\phi^H(n)M(n)\}}^{\mathcal{A}} \\
 &\quad - \overbrace{E\{\rho(n)M(n)\phi(n)\phi^H(n)\tilde{w}(n)\tilde{w}^H(n)\}}^{\mathcal{B}} \\
 &\quad + \overbrace{E\{\rho^2(n)M(n)\phi(n)\phi^H(n)\tilde{w}(n)\tilde{w}^H(n)\phi(n)\phi^H(n)M(n)\}}^{\mathcal{C}} \\
 &\quad + \overbrace{E\{\rho^2(n)|v_o(n)|^2M(n)\phi(n)\phi^H(n)M(n)\}}^{\mathcal{D}}. \tag{12.222}
 \end{aligned}$$

To simplify (12.222), we also need two additional assumptions:

**Assumption 3.** Matrix  $M(n)$  varies slowly in relation to  $\tilde{w}(n)$ . Thus, when  $M(n)$  appears inside the expectations of (12.222), we simply replace it by its mean. For LMS and NLMS, this assumption is not necessary, since  $M(n) = I$ . For RLS,  $M(n) = (1 - \lambda)^{-1}P(n) \approx (1 - \lambda)^{-1}E\{P(n)\}$  if  $\lambda \approx 1$ ;

**Assumption 4.** The input regressor vector  $\phi(n)$  is assumed to be Gaussian. This assumption makes the analysis more tractable, since Gaussianity facilitates the computation of the expected values in  $\mathcal{C}$ .

Now, using the Assumptions 1–4, we can evaluate the terms  $\mathcal{A}$ – $\mathcal{D}$  of (12.222):

**A-** The term  $\mathcal{A}$  can be approximated by

$$\begin{aligned}
 \mathcal{A} &= E\left\{E\left\{\rho(n)\tilde{w}(n)\tilde{w}^H(n)\phi(n)\phi^H(n)M(n)|\phi(n)\right\}\right\} \\
 &\approx E\left\{\rho(n)E\left\{\tilde{w}(n)\tilde{w}^H(n)\right\}\phi(n)\phi^H(n)\right\}E\{M(n)\} \\
 &= S(n)E\left\{\rho(n)\phi(n)\phi^H(n)\right\}E\{M(n)\}. \tag{12.223}
 \end{aligned}$$

For LMS, this term reduces to

$$\mathcal{A}^{\text{LMS}} \approx \mu S(n)R_\phi. \tag{12.224}$$

For NLMS, we must obtain an approximation for

$$E\left\{\rho(n)\phi(n)\phi^H(n)\right\} = E\left\{\frac{\tilde{\mu}\phi(n)\phi^H(n)}{\varepsilon + \phi^H(n)\phi(n)}\right\}. \tag{12.225}$$

It is possible to evaluate this expected value exactly (see [61, 63, 64]), but our intention here is to obtain the simplest possible model. Therefore, we will use two additional approximations:

**Assumption 5.** The number of coefficients  $M$  is large enough for each element  $\phi(n)\phi^H(n)$  in the numerator to be approximately independent from the denominator  $\sum_{k=0}^{M-1} |\phi(n-k)|^2$ .

This is equivalent to apply the averaging principle proposed in [79], since for large  $M$ ,  $\|\boldsymbol{\phi}(n)\|^2$  tends to vary slowly compared to the individual entries of  $\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)$ .

Thus, (12.225) can be approximated as

$$E \left\{ \frac{\boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n)}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} \right\} \approx E \left\{ \frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} \right\} E \left\{ \boldsymbol{\phi}(n)\boldsymbol{\phi}^H(n) \right\}. \quad (12.226)$$

Under Assumption 5, the first expected value in (12.226) can be approximated by:

$$E \left\{ \frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} \right\} \approx \frac{1}{E\{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)\}} = \frac{1}{\varepsilon + \text{Tr}(\mathbf{R}_\phi)}. \quad (12.227)$$

An alternative, more accurate approximation is obtained when  $\boldsymbol{\phi}(n)$  is Gaussian and a tapped-delay line:

**Assumption 6.** The regressor  $\boldsymbol{\phi}(n)$  is formed by a tapped-delay line with Gaussian entries as in (12.113), and  $\varepsilon = 0$ .

Under Assumptions 4, 5, and 6 it can be shown (see [65] for details) that

$$E \left\{ \frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} \right\} \approx \frac{1}{\sigma_x^2(M-2)}. \quad (12.228)$$

Both approximations are only valid for long filters. However, (12.228) shows that NLMS needs a large regularization constant if  $M < 3$ , which is verified in practice [63]. We will present the final results for NLMS in terms of Assumption 6, but recall that a less precise, but more general option is available using only Assumption 4 and approximations such as (12.227).

Thus, the term  $\mathcal{A}$  for NLMS reduces to (using (12.228))

$$\mathcal{A}^{\text{NLMS}} \approx \frac{\tilde{\mu}}{\sigma_x^2(M-2)} \mathbf{S}(n) \mathbf{R}_\phi. \quad (12.229)$$

Finally, for RLS, we obtain

$$\mathcal{A}^{\text{RLS}} \approx \mathbf{S}(n) \mathbf{R}_\phi \bar{\mathbf{P}}(n), \quad (12.230)$$

where  $\bar{\mathbf{P}}(n) \triangleq E\{\mathbf{R}_\phi^{-1}(n)\}$ , which we will evaluate later on.

**B-** Analogously, we obtain for **B**

$$\mathcal{B} \approx E\{\mathbf{M}(n)\} E \left\{ \rho(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\} \mathbf{S}(n). \quad (12.231)$$

Particularizing for each algorithm, we have

$$\mathcal{B}^{\text{LMS}} \approx \mu \mathbf{R}_\phi \mathbf{S}(n), \quad (12.232)$$

$$\mathcal{B}^{\text{NLMS}} \approx \frac{\tilde{\mu}}{\sigma_x^2(M-2)} \mathbf{R}_\phi \mathbf{S}(n), \quad (12.233)$$

and

$$\mathcal{B}^{\text{RLS}} \approx \bar{\mathbf{P}}(n) \mathbf{R}_\phi \mathbf{S}(n). \quad (12.234)$$

**C-** The term  $\mathcal{C}$  can be approximated by

$$\begin{aligned}\mathcal{C} &= \mathbb{E} \left\{ \rho^2(n) \mathbf{M}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \tilde{\mathbf{w}}(n) \tilde{\mathbf{w}}^H(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{M}(n) \right\} \\ &\approx \mathbb{E}\{\mathbf{M}(n)\} \mathbb{E} \left\{ \rho^2(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{S}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\} \mathbb{E}\{\mathbf{M}(n)\}.\end{aligned}\quad (12.235)$$

Under the Gaussianity assumption 4, it holds that (see Box 7)

$$\mathbb{E} \left\{ \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{S}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\} = \mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n) \mathbf{R}_\phi, \quad (12.236)$$

where  $\beta = 1$  (resp.,  $\beta = 2$ ) for complex (resp. real) data. Thus, for LMS we have

$$\mathcal{C}^{\text{LMS}} \approx \mu^2 [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n) \mathbf{R}_\phi]. \quad (12.237)$$

For NLMS, using the same arguments to get (12.226), the following approximation holds for large  $M$ :

$$\begin{aligned}\mathbb{E} \left\{ \rho^2(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{S}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\} \\ \approx \tilde{\mu}^2 \mathbb{E} \left\{ \frac{1}{[\boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)]^2} \right\} \mathbb{E} \left\{ \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{S}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\}.\end{aligned}\quad (12.238)$$

Under Assumption 5, the first expectation on the r.h.s. of (12.238) is given by

$$\mathbb{E} \left\{ \frac{1}{[\boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)]^2} \right\} = \frac{1}{\varepsilon + \text{Tr}(\mathbf{R}_\phi) + \beta \text{Tr}(\mathbf{R}_\phi^2)}. \quad (12.239)$$

On the other hand, under Assumption 6, we obtain [65]

$$\mathbb{E} \left\{ \frac{1}{[\boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)]^2} \right\} = \frac{1}{\sigma_x^4(M-2)(M-4)}. \quad (12.240)$$

Replacing (12.240) and (12.236) in (12.238), we arrive at

$$\mathcal{C}^{\text{NLMS}} \approx \frac{\tilde{\mu}^2}{\sigma_x^4(M-2)(M-4)} [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n) \mathbf{R}_\phi]. \quad (12.241)$$

Finally, for RLS we obtain

$$\mathcal{C}^{\text{RLS}} \approx \overline{\mathbf{P}}(n) [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n) \mathbf{R}_\phi] \overline{\mathbf{P}}(n). \quad (12.242)$$

**D-** Using the fact that  $v_o(n)$  is independent of  $\boldsymbol{\phi}(n)$  and  $\tilde{\mathbf{w}}(n)$ ,  $\mathcal{D}$  reduces to

$$\begin{aligned}\mathcal{D} &= \mathbb{E}\{\rho^2(n)|v_o(n)|^2 \mathbf{M}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{M}(n)\} \\ &\approx \sigma_v^2 \mathbb{E}\{\mathbf{M}(n)\} \mathbb{E}\{\rho^2(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n)\} \mathbb{E}\{\mathbf{M}(n)\}.\end{aligned}\quad (12.243)$$

Particularizing for each algorithm, we obtain

$$\mathcal{D}^{\text{LMS}} \approx \mu^2 \sigma_v^2 \mathbf{R}_\phi(n), \quad (12.244)$$

$$\mathcal{D}^{\text{NLMS}} \approx \frac{\tilde{\mu}^2}{\sigma_x^4(M-2)(M-4)} \sigma_v^2 \mathbf{R}_\phi(n), \quad (12.245)$$

where we could also have used (12.239), and

$$\mathcal{D}^{\text{RLS}} \approx \sigma_v^2 \bar{\mathbf{P}}(n) \mathbf{R}_\phi \bar{\mathbf{P}}(n). \quad (12.246)$$

From the previous results, (12.222) reduces to

$$\begin{aligned} \mathbf{S}(n+1) &\approx \mathbf{S}(n) - \mathbb{E}\{\rho(n)\} [\mathbf{S}(n) \mathbf{R}_\phi \mathbb{E}\{\mathbf{M}(n)\} + \mathbb{E}\{\mathbf{M}(n)\} \mathbf{R}_\phi \mathbf{S}(n)] \\ &\quad + \mathbb{E}\{\rho^2(n)\} \mathbb{E}\{\mathbf{M}(n)\} [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n) \mathbf{R}_\phi] \mathbb{E}\{\mathbf{M}(n)\} \\ &\quad + \sigma_v^2 \mathbb{E}\{\rho^2(n)\} \mathbb{E}\{\mathbf{M}(n)\} \mathbf{R}_\phi \mathbb{E}\{\mathbf{M}(n)\} + \mathbf{Q} \end{aligned} \quad (12.247)$$

with initialization  $\mathbf{S}(0) = [\mathbf{w}_o(0) - \mathbf{w}(0)] [\mathbf{w}_o(0) - \mathbf{w}(0)]^H$ .

For RLS, we still need an approximation for  $\bar{\mathbf{P}}(n) = \mathbb{E}\{\mathbf{P}(n)\}$ . Initializing the estimate of the autocorrelation matrix with  $\widehat{\mathbf{R}}_\phi(-1) = \delta \mathbf{I}$ , we will have at iteration  $n$

$$\widehat{\mathbf{R}}_\phi(n) = \lambda^n \delta \mathbf{I} + \sum_{\ell=1}^n \lambda^{n-\ell} \boldsymbol{\phi}(\ell) \boldsymbol{\phi}^H(\ell). \quad (12.248)$$

Taking the expectations of both sides of (12.248), we obtain

$$\mathbb{E}\{\widehat{\mathbf{R}}_\phi(n)\} = \lambda^n \delta \mathbf{I} + \mathbf{R}_\phi \frac{1 - \lambda^n}{1 - \lambda}. \quad (12.249)$$

For  $\lambda \approx 1$ , the variance of  $\widehat{\mathbf{R}}_\phi(n)$  is small, and we can approximate  $\bar{\mathbf{P}}(n) \approx [\mathbb{E}\{\widehat{\mathbf{R}}_\phi(n)\}]^{-1}$ , so that

$$\bar{\mathbf{P}}(n) \approx \left[ \lambda^n \delta \mathbf{I} + \mathbf{R}_\phi \frac{1 - \lambda^n}{1 - \lambda} \right]^{-1}. \quad (12.250)$$

This approximation is good for large  $n$  and in steady-state. During the initial phases of the transient, however, the approximation will only be reasonable if  $\delta$  is large enough to make the last term in (12.249) small compared to  $\lambda^n \delta \mathbf{I}$ . This is because for  $n < M$ , the term  $\sum_{\ell=1}^n \lambda^{n-\ell} \boldsymbol{\phi}(\ell) \boldsymbol{\phi}^H(\ell)$  in (12.248) will be singular, and thus will not be well approximated by  $(1 - \lambda^n) \mathbf{R}_\phi / (1 - \lambda)$ .

Using the previous approximations, (12.247) can be particularized for the LMS, NLMS, and RLS algorithms as shown in Table 12.7. Since we can obtain the EMSE and the MSD from  $\mathbf{S}(n)$  and  $\mathbf{R}_\phi$  through (12.185) and (12.184), we should use the recursions of Table 12.7 to compute  $\mathbf{S}(n)$  and then evaluate  $\zeta(n) = \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi)$  and  $\chi(n) = \text{Tr}(\mathbf{S}(n))$ .

**Example 3.** To verify the accuracy of the expressions of Table 12.7, we consider a system identification application in a stationary environment ( $\mathbf{Q} = \mathbf{0}$ ). The optimal solution is a lowpass FIR filter with linear

**Table 12.7** Recursions for Covariance Matrix  $\mathbf{S}(n+1)$ . The Expressions for NLMS Assume that the Regressor is a Tapped-Delay Line. Alternative Expressions for NLMS are Available Using (12.227) and (12.239)

Algorithm	$\mathbf{S}(n+1)$
LMS	$\mathbf{S}(n+1) \approx \mathbf{S}(n) - \mu [\mathbf{S}(n)\mathbf{R}_\phi + \mathbf{R}_\phi\mathbf{S}(n)]$ $+ \mu^2 [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n)\mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n)\mathbf{R}_\phi + \sigma_v^2 \mathbf{R}_\phi] + \mathbf{Q}$
NLMS	$\mathbf{S}(n+1) \approx \mathbf{S}(n) - \frac{\tilde{\mu}}{\sigma_x^2(M-2)} [\mathbf{S}(n)\mathbf{R}_\phi + \mathbf{R}_\phi\mathbf{S}(n)]$ $+ \frac{\tilde{\mu}^2}{\sigma_x^4(M-2)(M-4)} [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n)\mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n)\mathbf{R}_\phi + \sigma_v^2 \mathbf{R}_\phi] + \mathbf{Q}$
RLS	$\mathbf{S}(n+1) \approx \mathbf{S}(n) - [\mathbf{S}(n)\mathbf{R}_\phi \bar{\mathbf{P}}(n) + \bar{\mathbf{P}}(n)\mathbf{R}_\phi \mathbf{S}(n)]$ $+ \bar{\mathbf{P}}(n) [\mathbf{R}_\phi \text{Tr}(\mathbf{S}(n)\mathbf{R}_\phi) + \beta \mathbf{R}_\phi \mathbf{S}(n)\mathbf{R}_\phi + \sigma_v^2 \mathbf{R}_\phi] \bar{\mathbf{P}}(n) + \mathbf{Q}$

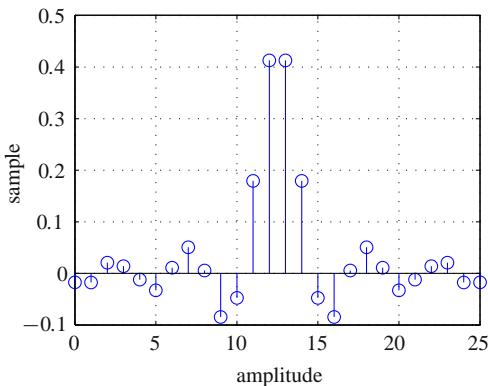
phase, whose coefficients are shown in Figure 12.29. The regressor  $\phi(n)$  is obtained from a process  $x(n)$  generated with a first-order autoregressive model, whose transfer function is  $\sqrt{1-b^2}/(1-bz^{-1})$ . Note that the sequence  $\{\phi(n)\}$  is not iid. This model is fed with an iid Gaussian random process, whose variance is such that  $\text{Tr}(\mathbf{R}_\phi) = 1$ . Moreover, additive iid noise  $v_o(n)$  with variance  $\sigma_v^2 = 0.01$  is added to form the desired signal.

Figure 12.30 shows the EMSE for RLS ( $\lambda = 0.995$ ,  $\delta = 1$ ), NLMS ( $\tilde{\mu} = 0.1$ ,  $\varepsilon = 10^{-5}$ ), and LMS ( $\mu = 0.05$ ), estimated from the ensemble-average of 2000 independent runs. The dashed lines represent the theoretical EMSE computed as  $\zeta(n) = \text{Tr}(\mathbf{S}(n)\mathbf{R}_\phi)$ . We can observe a good agreement between theory and simulation.

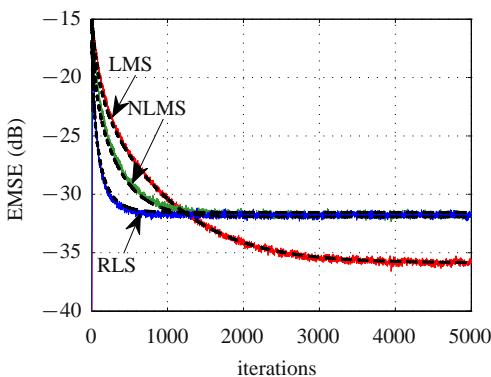
Figure 12.31 shows the EMSE of RLS ( $\lambda = 0.995$ ), assuming different values for  $\delta$ . We can observe that the theoretical EMSE of RLS presents a good agreement with simulation during the initial phases of the transient, but only for large values of  $\delta$  due to the approximation (12.250).

#### 1.12.4.4.4 Steady-state analysis with the energy conservation method

This method (also known as *feedback* approach [78, 80]) is based on an energy conservation relation that holds for a large class of adaptive filters. To obtain this relation for the algorithms of the form (12.219), we first assume a stationary environment ( $\mathbf{q}(n) \equiv \mathbf{0}$ ).

**FIGURE 12.29**

Impulse response  $w_o$  of the unknown lowpass filter.

**FIGURE 12.30**

EMSE for RLS ( $\lambda = 0.995, \delta = 1$ ), NLMS ( $\tilde{\mu} = 0.1, \varepsilon = 10^{-5}$ ), and LMS ( $\mu = 0.05$ );  $b = 0.8, \sigma_v^2 = 10^{-2}, \sigma_q^2 = 0$ ; mean of 2000 independent runs. The dashed lines represent the predicted values of  $\xi(n)$  for each algorithm.

Recall that the excess a posteriori error is given by

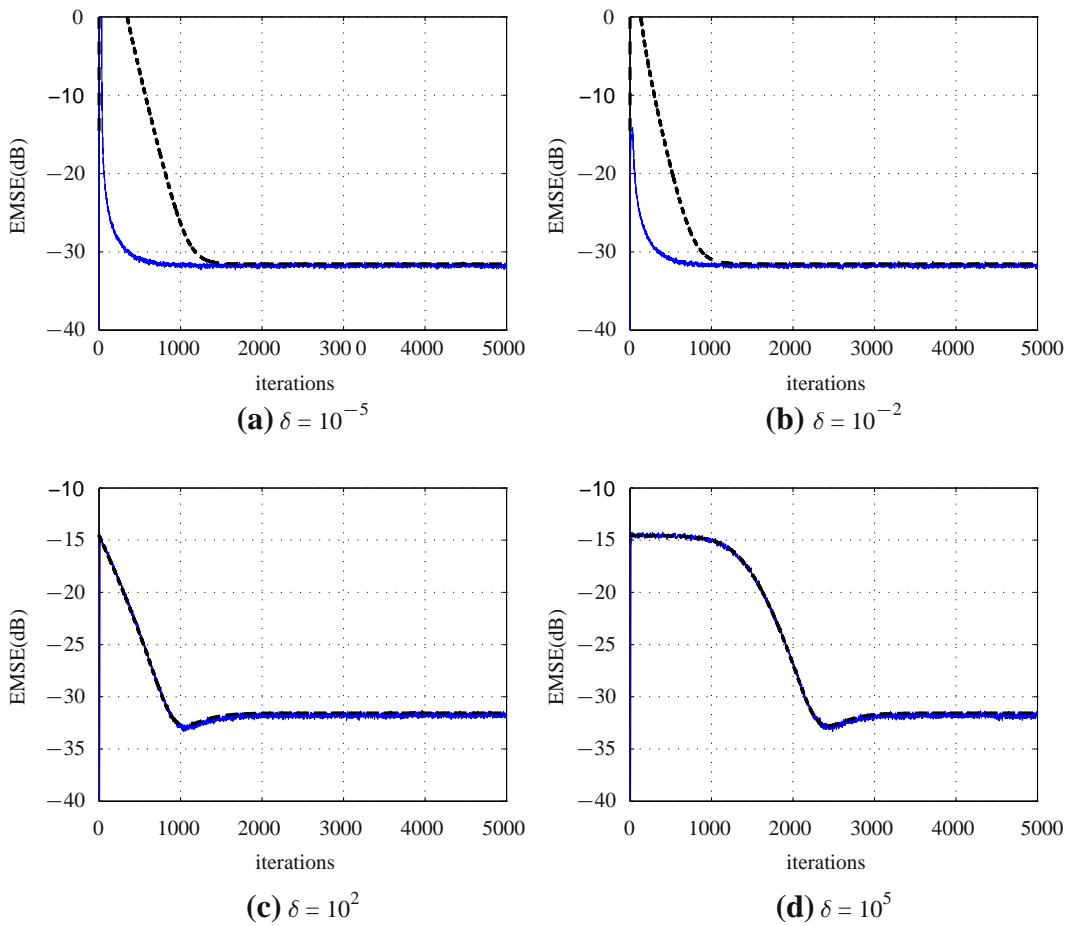
$$e_p(n) = \tilde{w}^H(n+1)\phi(n). \quad (12.251)$$

Then, we multiply both sides of the Hermitian of (12.220) (with  $q(n) \equiv \mathbf{0}$ ) from the right by  $\phi(n)$  to obtain

$$\tilde{w}^H(n+1)\phi(n) = \tilde{w}^H(n)\phi(n) - \rho(n)\phi^H(n)M(n)\phi(n)e(n). \quad (12.252)$$

Using (12.172) and (12.251), (12.252) reduces to

$$e_p(n) = e_a(n) - \rho(n)\|\phi(n)\|_{M(n)}^2 e(n), \quad (12.253)$$



## **FIGURE 12.31**

EMSE for RLS considering  $\lambda = 0.995$  and different values of  $\delta$ ;  $b = 0.8$ ,  $\sigma_v^2 = 10^{-2}$ ,  $\sigma_q^2 = 0$ ; mean of 2000 independent runs. The dashed lines represent the predicted values of  $\zeta(n)$  for each algorithm.

where  $\|\mathbf{u}\|_A^2 = \mathbf{u}^H \mathbf{A} \mathbf{u}$  stands for the Euclidean norm of  $\mathbf{u}$  weighted by the matrix  $\mathbf{A}$ . Using (12.253) to eliminate  $e(n)$  in (12.220) (with  $\mathbf{q}(n) \equiv \mathbf{0}$ ) and assuming that  $\phi(n) \neq \mathbf{0}$ , we get

$$\tilde{\mathbf{w}}(n+1) + \frac{\mathbf{M}(n)\phi(n)}{\|\phi(n)\|_{\mathbf{M}(n)}^2} e_a^*(n) = \tilde{\mathbf{w}}(n) + \frac{\mathbf{M}(n)\phi(n)}{\|\phi(n)\|_{\mathbf{M}(n)}^2} e_p^*(n). \quad (12.254)$$

Defining

$$\bar{\rho}(n) \triangleq \begin{cases} 1/\|\phi(n)\|_{M(n)}^2, & \text{if } \phi(n) \neq \mathbf{0} \\ 0, & \text{otherwise} \end{cases} \quad (12.255)$$

to include the case of zero regressor and equating the squared weighted norms on both sides of (12.254) with  $\mathbf{M}^{-1}(n)$  as a weighting matrix, the cross-terms cancel and we obtain

$$\|\tilde{\mathbf{w}}(n+1)\|_{\mathbf{M}^{-1}(n)}^2 + \bar{\rho}(n)|e_a(n)|^2 = \|\tilde{\mathbf{w}}(n)\|_{\mathbf{M}^{-1}(n)}^2 + \bar{\rho}(n)|e_p(n)|^2. \quad (12.256)$$

This relation shows how the weighted energies of the weight-error vectors at two successive time instants are related to the energies of the excess *a priori* and excess *a posteriori* errors. This relation is the reason why this approach is known as the energy conservation method—(12.256) relates the (weighted) coefficient error energies at instants  $n$  and  $n+1$  to the *a priori* and *a posteriori* errors. In [3], this energy conservation relation is used extensively to find models for a wide class of adaptive filters. We follow here a shorter route, however, and derive in a more direct way the relations necessary for the study of just LMS, NLMS and RLS.

We can obtain the steady-state EMSE by computing the squared weighted norms on both sides of (12.220) with  $\mathbf{M}^{-1}(n)$  as a weighting matrix, which leads to

$$\begin{aligned} \|\tilde{\mathbf{w}}(n+1)\|_{\mathbf{M}^{-1}(n)}^2 &= \|\tilde{\mathbf{w}}(n)\|_{\mathbf{M}^{-1}(n)}^2 + \rho^2(n)|e(n)|^2\|\phi(n)\|_{\mathbf{M}(n)}^2 \\ &\quad - \rho(n)[e_a(n)e^*(n) + e_a^*(n)e(n)] + \|\mathbf{q}(n)\|_{\mathbf{M}^{-1}(n)}. \end{aligned} \quad (12.257)$$

By taking expectations on both sides of (12.257), we arrive at (we already used Assumption 2 to eliminate cross-terms with  $\mathbf{q}(n)$  on the right-hand side)

$$\begin{aligned} \overbrace{\mathbb{E}\left\{\|\tilde{\mathbf{w}}(n+1)\|_{\mathbf{M}^{-1}(n)}^2\right\}}^{\mathcal{A}} &= \overbrace{\mathbb{E}\left\{\|\tilde{\mathbf{w}}(n)\|_{\mathbf{M}^{-1}(n)}^2\right\}}^{\mathcal{B}} \\ &\quad + \overbrace{\mathbb{E}\left\{\rho^2(n)|e(n)|^2\|\phi(n)\|_{\mathbf{M}(n)}^2\right\}}^{\mathcal{C}} \\ &\quad - \overbrace{\mathbb{E}\left\{\rho(n)[e_a(n)e^*(n) + e_a^*(n)e(n)]\right\}}^{\mathcal{D}} \\ &\quad + \overbrace{\|\mathbf{q}(n)\|_{\mathbf{M}^{-1}(n)}}^{\mathcal{E}}. \end{aligned} \quad (12.258)$$

To proceed, we have to assume again that matrix  $\mathbf{M}^{-1}(n)$  varies slowly in relation to  $\tilde{\mathbf{w}}(n)$  and  $\mathbf{q}(n)$ , as in Assumption 3. Since now we are dealing only with the steady-state, this approximation is in general good. Instead of Assumption 1, in energy conservation analysis the following assumptions are used:

**Assumption 7.**  $\|\phi(n)\|_{\mathbf{M}(n)}^2$  is independent of  $e_a(n)$  at the steady-state.

**Assumption 8.** The noise sequence  $\{\mathbf{v}_o(n)\}$  is iid and independent of the input sequence  $\{\phi(n)\}$ .

Assumption 7 is referred to in the literature as *separation principle* and is reasonable in steady-state since for large  $n$ ,  $e_a(n)$  tends to be less sensitive to the regressor data, in particular for long filters. Assumption 8 is a weakened version of Assumption 1, which is necessary to relate the MSE with the EMSE through (12.181).

Under model 2 and Assumptions 3 and 7, we can now evaluate the terms  $\mathcal{A} - \mathcal{E}$ :

**A-** Using 3, we obtain

$$\mathcal{A} \approx E \left\{ \| \tilde{\mathbf{w}}(n+1) \|_{E\{\mathbf{M}^{-1}(n)\}}^2 \right\}. \quad (12.259)$$

For LMS and NLMS, we have

$$\mathcal{A}^{LMS} = \mathcal{A}^{NLMS} \approx E \left\{ \| \tilde{\mathbf{w}}(n+1) \|^2 \right\}. \quad (12.260)$$

For RLS,

$$\mathcal{A}^{RLS} \approx E \left\{ \| \tilde{\mathbf{w}}(n+1) \|_{\mathbf{R}_\phi}^2 \right\}. \quad (12.261)$$

**B-** Assuming that the filter is stable and reaches a steady-state, we have

$$\mathcal{B} = E \left\{ \| \tilde{\mathbf{w}}(n) \|_{\mathbf{M}^{-1}(n)}^2 \right\} \approx E \left\{ \| \tilde{\mathbf{w}}(n+1) \|_{\mathbf{M}^{-1}(n)}^2 \right\}. \quad (12.262)$$

**C-** Given that  $e(n) = e_a(n) + v_o(n)$  and since  $v_o(n)$  is assumed independent of  $e_a(n)$  (Assumption 8), we obtain

$$E\{|e(n)|^2\} \approx E\{|e_a(n)|^2\} + \sigma_v^2.$$

Then, using Assumptions 3 and 8, we arrive at

$$\mathcal{C} \approx E \left\{ \rho^2(n) \boldsymbol{\phi}^H(n) E\{\mathbf{M}(n)\} \boldsymbol{\phi}(n) \right\} \left( E\{|e_a(n)|^2\} + \sigma_v^2 \right). \quad (12.263)$$

For LMS, this term reduces to

$$\mathcal{C}^{LMS} \approx \mu^2 \text{Tr}(\mathbf{R}_\phi) \left( E\{|e_a(n)|^2\} + \sigma_v^2 \right). \quad (12.264)$$

For NLMS,

$$\mathcal{C}^{NLMS} \approx \tilde{\mu}^2 E \left\{ \frac{1}{\varepsilon + \boldsymbol{\phi}^H(n) \boldsymbol{\phi}(n)} \right\} \left( E\{|e_a(n)|^2\} + \sigma_v^2 \right), \quad (12.265)$$

which for long filters and Gaussian tapped-delay line regressors can be approximated as (see Eq. (12.228), and the alternative (12.227) for general regressors)

$$\mathcal{C}^{NLMS} \approx \frac{\tilde{\mu}^2}{\sigma_x^2(M-2)} \left( E\{|e_a(n)|^2\} + \sigma_v^2 \right). \quad (12.266)$$

Finally, for RLS we obtain

$$\mathcal{C}^{RLS} \approx (1-\lambda)^2 M \left( E\{|e_a(n)|^2\} + \sigma_v^2 \right). \quad (12.267)$$

**D-** Using the same arguments,  $\mathcal{D}$  reduces to

$$\mathcal{D} \approx 2E\{\rho(n)\}E\{|e_a(n)|^2\}. \quad (12.268)$$

For LMS,  $\mathcal{D}^{LMS} \approx 2\mu E\{|e_a(n)|^2\}$ . Using (Eq. (12.228)) for NLMS, we get

$$\mathcal{D}^{NLMS} \approx \frac{2\tilde{\mu}^2}{\sigma_x^2(M-2)} E\{|e_a(n)|^2\}.$$

For RLS, this term is given by  $\mathcal{D}^{RLS} \approx 2(1-\lambda)E\{|e_a(n)|^2\}$ .

**Table 12.8** Expressions for the Steady-State EMSE Obtained from the Energy Conservation Method for LMS, NLMS, and RLS

Algorithm	$\xi(\infty)$	$\xi(\infty)$
LMS	for a wider range of $\mu$ , $\tilde{\mu}$ , and $\lambda$ $\frac{\mu\sigma_V^2 \text{Tr}(\mathbf{R}_\phi) + \mu^{-1} \text{Tr}(\mathbf{Q})}{2 - \mu \text{Tr}(\mathbf{R}_\phi)}$	for small $\mu$ and $\tilde{\mu}$ , and $\lambda \approx 1$ $\frac{\mu\sigma_V^2 \text{Tr}(\mathbf{R}_\phi) + \mu^{-1} \text{Tr}(\mathbf{Q})}{2}$
NLMS	$\frac{\tilde{\mu}\sigma_V^2 + \tilde{\mu}^{-1}\sigma_X^2(M-2)\text{Tr}(\mathbf{Q})}{2 - \tilde{\mu}}$	$\frac{\tilde{\mu}\sigma_V^2 + \tilde{\mu}^{-1}\sigma_X^2(M-2)\text{Tr}(\mathbf{Q})}{2}$
RLS	$\frac{\sigma_V^2(1-\lambda)M + \frac{\text{Tr}(\mathbf{Q}\mathbf{R}_\phi)}{1-\lambda}}{2 - (1-\lambda)M}$	$\frac{\sigma_V^2(1-\lambda)M + \frac{\text{Tr}(\mathbf{Q}\mathbf{R}_\phi)}{1-\lambda}}{2}$

$\mathcal{E}$ - Using Assumption 3 and recalling that  $\text{Tr}(\mathbf{AB}) = \text{Tr}(\mathbf{BA})$ , we obtain

$$\mathcal{E} \approx \mathbb{E} \left\{ \mathbf{q}^H(n) \mathbb{E}\{\mathbf{M}^{-1}(n)\} \mathbf{q}(n) \right\} = \text{Tr} \left( \mathbb{E} \left\{ \mathbf{M}^{-1}(n) \right\} \mathbb{E} \{ \mathbf{q}(n) \mathbf{q}^H(n) \} \right).$$

For LMS and NLMS this reduces to

$$\mathcal{E}^{\text{LMS}} = \mathcal{E}^{\text{NLMS}} \approx \text{Tr}(\mathbf{Q}). \quad (12.269)$$

For RLS,

$$\mathcal{E}^{\text{RLS}} \approx \text{Tr}(\mathbf{R}_\phi \mathbf{Q}). \quad (12.270)$$

Using these approximations in (12.258), we arrive at the following steady-state approximation

$$\mathbb{E}\{|e_a(n)|^2\} \approx \frac{\sigma_v^2 \mathbb{E}\{\rho^2(n) \boldsymbol{\phi}^H(n) \mathbb{E}\{\mathbf{M}(n)\} \boldsymbol{\phi}(n)\} + \text{Tr}(\mathbf{Q} \mathbb{E}\{\mathbf{M}^{-1}(n)\})}{2\mathbb{E}\{\rho(n)\} - \mathbb{E}\{\rho^2(n) \boldsymbol{\phi}^H(n) \mathbb{E}\{\mathbf{M}(n)\} \boldsymbol{\phi}(n)\}}. \quad (12.271)$$

Particularizing (12.271) for LMS, NLMS, and RLS, we obtain the results of the second column of Table 12.8, which hold over a wide range of step-sizes and forgetting factors.

#### 1.12.4.4.5 Relation between the results obtained with both analysis methods

As we explained before, each method is based on a different set of approximations. We now show how they are related.

The energy conservation method is capable of obtaining closed-form expressions for the EMSE using less restrictive assumptions, in particular  $\boldsymbol{\phi}(n)$  is not required to be Gaussian. These results are equivalent to those obtained for LMS and RLS using the traditional analysis, taking the recursions of Table 12.7 to the limit as  $n \rightarrow \infty$ , and assuming  $\beta = 0$  (recall that the values of  $\beta$  used for the traditional analysis are only valid if  $\boldsymbol{\phi}(n)$  is Gaussian). The assumption of  $\beta = 0$  implies the following approximation:

$$\mathbb{E} \left\{ \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \mathbf{S}(n) \boldsymbol{\phi}(n) \boldsymbol{\phi}^H(n) \right\} \approx \mathbf{R}_\phi \text{Tr}(\mathbf{S}(n) \mathbf{R}_\phi).$$

**Table 12.9** Expressions for the Optimal Adaptation Parameters ( $\rho_o$ ) and Minimum Steady-State EMSE ( $\zeta_{\min}(\infty)$ ) Obtained from the Expressions of the Third Column of Table 12.8

Algorithm	$\rho_o$	$\zeta_{\min}(\infty)$
LMS	$\mu_o = \sqrt{\frac{\text{Tr}(\mathbf{Q})}{\sigma_v^2 \text{Tr}(\mathbf{R}_\phi)}}$	$\sqrt{\sigma_v^2 \text{Tr}(\mathbf{R}_\phi) \text{Tr}(\mathbf{Q})}$
NLMS	$\tilde{\mu}_o = \sqrt{\frac{(M-2)\sigma_x^2 \text{Tr}(\mathbf{Q})}{\sigma_v^2}}$	$\sqrt{\sigma_v^2 \sigma_x^2 (M-2) \text{Tr}(\mathbf{Q})}$
RLS	$1 - \lambda_o = \sqrt{\frac{\text{Tr}(\mathbf{Q}\mathbf{R}_\phi)}{M\sigma_v^2}}$	$\sqrt{\sigma_v^2 M \text{Tr}(\mathbf{Q}\mathbf{R}_\phi)}$

For small step-sizes, this approximation does not have a large impact, since we are disregarding a term of  $\mathcal{O}(\mu^2)$ , that is small compared to other  $\mathcal{O}(\mu)$  terms. For NLMS, in order to recover (12.271) using the traditional method, we would choose  $\beta = 0$  and make

$$\mathbb{E} \left\{ \frac{1}{[\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)]^2} \right\} \approx \left[ \mathbb{E} \left\{ \frac{1}{\varepsilon + \boldsymbol{\phi}^H(n)\boldsymbol{\phi}(n)} \right\} \right]^2,$$

which leads to reasonable results at the steady-state for long filters.

For small step-sizes (or forgetting factors close to one), the denominator of (12.271) can be approximated by  $2\mathbb{E}\{\rho(n)\}$ . In this case, the expressions of the second column reduce to those of the third column of Table 12.8. Again, these results can be obtained from the recursions of Table 12.7, disregarding the term  $\mathcal{C}$  (Eq. (12.235)), which will be negligible compared to  $\mathcal{B}$  in (12.222) if  $\rho \approx 0$ .

The results for steady-state analysis using the traditional method are commonly obtained as the limiting case of a transient analysis. Although this procedure is adequate for understanding both the steady-state and the transient behavior of an adaptive algorithm, it tends to be more laborious than using the energy conservation method to analyze the steady-state behavior. This is one of the largest advantages of this last method.

#### 1.12.4.4.6 Optimal step-size for tracking

From the results of the third column of Table 12.8, we can observe that the expressions for the steady-state EMSE have two terms: one for the stationary environment, that increases as the step-size increases, and another that appears in the nonstationary case, and increases as the step-size decreases. Therefore, there exist optimal adaptation parameters  $\rho_o$ , that minimize the EMSE. These parameters are  $\mu_o$ ,  $\tilde{\mu}_o$ , and  $1 - \lambda_o$  for LMS, NLMS, and RLS, respectively. The corresponding minima, denoted by  $\zeta_{\min}(\infty)$ , can also be evaluated. All these results are summarized in Table 12.9. The results for a wider range of adaptation parameters lead to more complicated expressions for  $\rho_o$  and  $\zeta_o(\infty)$  (see, e.g., [3]).

We can use the results of Table 12.9 to compare the tracking performance of the algorithms. For instance, comparing the minimum EMSE for LMS to that of RLS, we obtain the ratio

$$\frac{\zeta_{\min}^{\text{RLS}}}{\zeta_{\min}^{\text{LMS}}} = \sqrt{\frac{M \text{Tr}(\mathbf{Q}\mathbf{R}_\phi)}{\text{Tr}(\mathbf{R}_\phi) \text{Tr}(\mathbf{Q})}}. \quad (12.272)$$

Clearly, the results of this comparison depend on the environment. There are situations where RLS has superior tracking capability compared to LMS, and vice versa. This is highlighted considering three different choices for matrix  $\mathbf{Q}$ . It can be shown [81] that

- i. If  $\mathbf{Q}$  is a multiple of  $\mathbf{I}$ : the performance of LMS is similar to that of RLS;
- ii. If  $\mathbf{Q}$  is a multiple of  $\mathbf{R}_\phi$ : LMS is superior; and
- iii. If  $\mathbf{Q}$  is a multiple of  $\mathbf{R}_\phi^{-1}$ : RLS is superior.

These choices for  $\mathbf{Q}$  do not model practical situations, but they highlight that, even though the convergence rate of RLS is in general much higher than that of LMS, this advantage, unexpectedly, does not necessarily follow to the problem of tracking a slowly-varying parameter vector.

**Example 4.** To verify the accuracy of the expressions of Tables 12.8 and 12.9, we use the LMS, NLMS, and RLS filters to identify the same lowpass FIR system of Figure 12.29. Now, the environment is assumed nonstationary with  $\mathbf{Q} = 10^{-6}\mathbf{I}$ . The input signal and minimum MSE ( $\sigma_v^2$ ) are the same of Example 3.

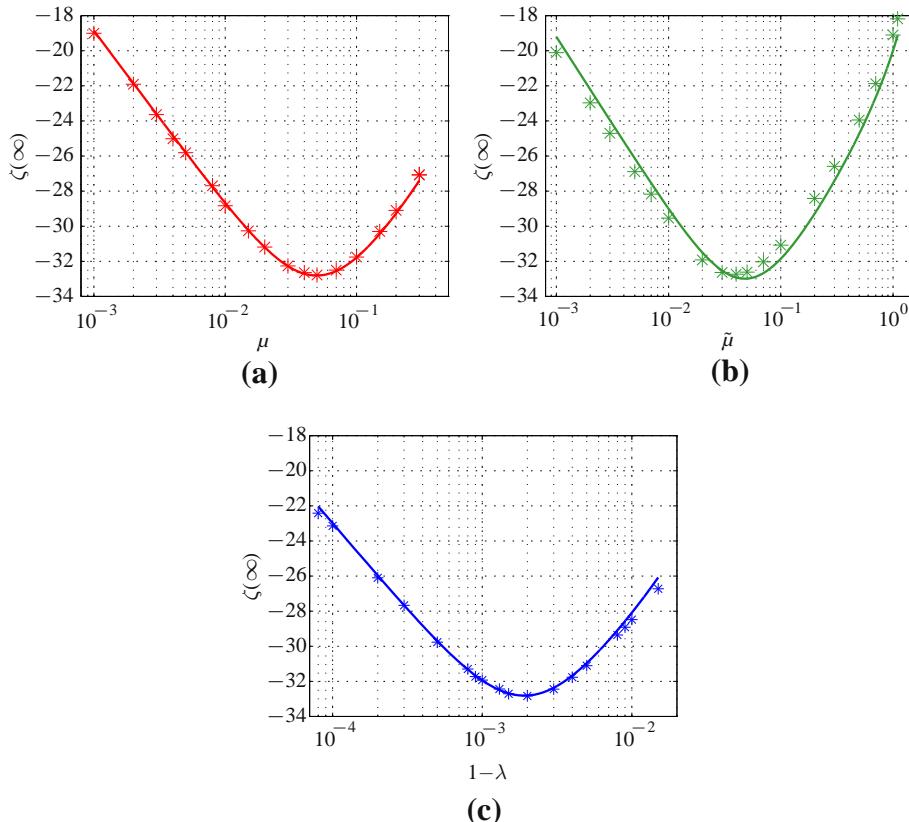
Figure 12.32 shows the measured steady-state EMSE with varying adaptation factors considering the theoretical and experimental results for LMS, NLMS, and RLS. Each value of experimental EMSE was obtained from the ensemble-average of 100 independent runs. The theoretical minimum EMSE predicted by the expressions of Table 12.9 are  $\xi_o^{\text{LMS}}(\infty) = \xi_o^{\text{RLS}}(\infty) = -32.9251$  dB and  $\xi_o^{\text{NLMS}}(\infty) = -33.0989$  dB, which correspond to the optimal adaptation parameters  $\mu_o = 5.1 \times 10^{-2}$ ,  $\tilde{\mu}_o = 4.9 \times 10^{-2}$  and  $1 - \lambda_o = 2 \times 10^{-3}$  ( $\lambda_o = 0.998$ ). The experimental values for  $\xi_{\min}(\infty)$  and  $\rho_o$  are close to the predicted by the expressions of Table 12.9 as shown in Figure 12.32.

**Example 5.** Again, we consider a system identification application but with the initial optimal solution given by

$$\mathbf{w}_o^T(0) = [0.5349 \ 0.9527 \ -0.9620 \ -0.0158 \ -0.1254].$$

The input signal is assumed to be colored Gaussian noise with variance  $\sigma_x^2 = 0.2$ . This signal is obtained from the filtering of a white Gaussian noise by a first-order autoregressive model ( $b = 0.8$ ). We also assume that the minimum MSE is  $\sigma_v^2 = 0.01$ .

As predicted by (12.272), the performance of RLS and LMS are similar when  $\mathbf{Q} = \sigma_q^2\mathbf{I}$ . Thus, assuming  $\mathbf{Q} = 10^{-6}\mathbf{I}$ , we computed  $\mu_o$  and  $\lambda_o$  from the expressions of Table 12.9 and used these parameters in the adaptation, such that the same minimum EMSE could be achieved by both algorithms in this situation. Figure 12.33 shows the EMSE estimated from the ensemble-average of 5000 independent runs for RLS ( $\lambda = \lambda_o = 0.9955$ ,  $\delta = 4.5 \times 10^{-3}$ ) and LMS ( $\mu = \mu_o = 0.0224$ ). At every  $7 \times 10^4$  iterations, the nonstationary environment is changed. During the first  $7 \times 10^4$  iterations,  $\mathbf{Q} = 10^{-6}\mathbf{R}_\phi$  and LMS presents a slightly better tracking performance than that of RLS. When  $\mathbf{Q} = 10^{-6}\mathbf{R}_\phi^{-1}$ , this behavior changes: RLS becomes better than LMS. Finally for  $\mathbf{Q} = 10^{-6}\mathbf{I}$ , RLS and LMS present similar performance. The dashed lines represent the theoretical values of  $\zeta(\infty)$  for each algorithm, predicted from the expressions of the second column of Table 12.8.

**FIGURE 12.32**

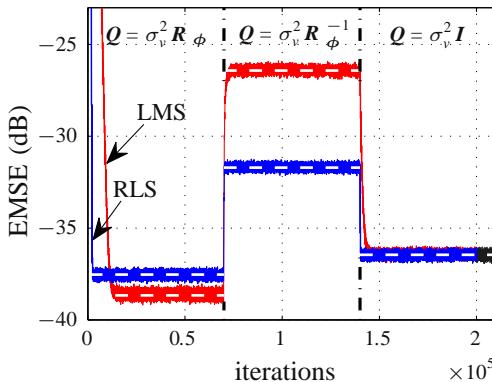
Theoretical and experimental EMSE as a function of the adaptation parameter for a) LMS, b) NLMS, and c) RLS. The asterisks represent simulation results and the solid lines represent the theoretical models.

### Box 7: Fourth-order moments of Gaussian vectors

The second-order moments of a random sequence are related to its average power (of course, this is an approximation unless the sequence is stationary and ergodic.) This makes second-order moments intuitive to work with and relatively easy to evaluate in practice. Therefore, when studying an adaptive filter, it is common to try to describe its performance in terms of the autocorrelation of the input sequence  $\mathbf{x}(n)$ .

During the analysis of LMS, one encounters a term that contains fourth-order powers of the input vector  $\mathbf{x}(n)$ :

$$\mathbf{F} \stackrel{\Delta}{=} \mathbb{E}\{\mathbf{x}(n)\mathbf{x}^H(n)\tilde{\mathbf{w}}(n)\tilde{\mathbf{w}}^H(n)\mathbf{x}(n)\mathbf{x}^H(n)\}. \quad (12.273)$$

**FIGURE 12.33**

EMSE for RLS ( $\lambda = 0.9955$ ,  $\delta = 4.5 \times 10^{-3}$ ), and LMS ( $\mu = 0.0224$ );  $b = 0.8$ ,  $\sigma_v^2 = 10^{-2}$ ,  $\sigma_q^2 = 10^{-6}$ ; mean of 5000 independent runs. The dashed lines represent the predicted values of  $\zeta(\infty)$  for each algorithm.

In order to evaluate a term such as this, one needs to know much more information about the statistics of  $x(n)$  than is provided simply by its mean and autocorrelation. However, in general this additional information is not available, and one assumes that the signals are Gaussian in order to proceed.

Why not any other distribution? First, because many processes in Nature are indeed approximately Gaussian. Second, because uncorrelated Gaussian variables are also independent, which is of great help when evaluating fourth-order terms, such as  $E\{x^2(n)x^2(n-1)\}$  or  $E\{x^3(n)x(n-2)\}$ .

Although in many cases the input sequence is not Gaussian (for example, speech does not follow a Gaussian distribution), this approximation describes the most important features of adaptive filter learning with reasonable accuracy, and leads to a relatively simple model.

In order to simplify the notation, we will write simply  $x$  and  $\tilde{w}$  instead of  $x(n)$  and  $\tilde{w}(n)$  in the following. Assume then that  $x$  is a Gaussian random vector, with autocorrelation  $E\{xx^H\} = R$ , and that  $\tilde{w}$  is another vector independent of  $x$ , and with autocorrelation  $E\{\tilde{w}\tilde{w}^H\} = S$  (we do not need to know the exact distribution of  $\tilde{w}$ .) Assume also that all variables have zero mean. Our goal is to evaluate

$$F = E\{xx^H\tilde{w}\tilde{w}^Hxx^H\}.$$

Since  $\tilde{w}$  is independent of  $x$ , we can write

$$F = E\{xx^H\tilde{w}\tilde{w}^Hxx^H\} = E\{xx^H E\{\tilde{w}\tilde{w}^H\} xx^H\} = E\{xx^H S xx^H\}. \quad (12.274)$$

The autocorrelation of  $x$  is in general a full matrix, that is,

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M} \\ r_{12}^* & r_{22} & \dots & r_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1M}^* & r_{2M}^* & \dots & r_{MM} \end{bmatrix},$$

that is, in general the elements of  $\mathbf{x}$  are correlated.  $\mathbf{R}$  has hermitian symmetry and is non-negative definite, which implies that there exists a unitary matrix  $\mathbf{U}$  such that

$$\mathbf{U}^H \mathbf{R} \mathbf{U} = \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{bmatrix},$$

where  $\lambda_i \geq 0$  for  $i = 1, \dots, M$ . Define  $\mathbf{x}' = \mathbf{U}^H \mathbf{x}$ . Then  $E\{\mathbf{x}' \mathbf{x}'^H\} = E\{\mathbf{U}^H \mathbf{x} \mathbf{x}^H \mathbf{U}\} = \mathbf{U}^H E\{\mathbf{x} \mathbf{x}^H\} \mathbf{U} = \mathbf{\Lambda}$ , and we see that the entries of  $\mathbf{x}'$  are uncorrelated. Since  $\mathbf{x}$  is Gaussian, and a linear transformation of a Gaussian vector is also Gaussian, this means that the entries of  $\mathbf{x}'$  are independent of each other. Multiplying (12.274) by  $\mathbf{U}^H$  on the left and  $\mathbf{U}$  on the right, and recalling that  $\mathbf{U}^H \mathbf{U} = \mathbf{U} \mathbf{U}^H = \mathbf{I}$ , we obtain

$$\mathbf{F}' \stackrel{\Delta}{=} \mathbf{U}^H \mathbf{F} \mathbf{U} = E\{\mathbf{U}^H \mathbf{x} \mathbf{x}^H \underbrace{\mathbf{U} \mathbf{U}^H}_{=\mathbf{I}} \mathbf{S} \underbrace{\mathbf{U} \mathbf{U}^H}_{=\mathbf{I}} \mathbf{x} \mathbf{x}^H \mathbf{U}\} = E\{\mathbf{x}' \mathbf{x}'^H \mathbf{U}^H \mathbf{S} \mathbf{U} \mathbf{x}' \mathbf{x}'^H \mathbf{U}\}.$$

Defining  $\bar{\mathbf{S}} = \mathbf{U}^H \mathbf{S} \mathbf{U}$ , we can expand  $\mathbf{F}'$  in terms of the elements  $s'_{ij}$  of  $\bar{\mathbf{S}}$  and  $x'_k$  of  $\mathbf{x}'$ . Each element of  $\mathbf{F}'$  will have terms of the form

$$E\{x'_{k_1} x'_{k_2} x'^{*}_{k_3} x'^{*}_{k_4}\} s'_{ij}.$$

If the signals are real, then, as  $x'_{k_1}$  is independent of  $x'_{k_2}$  if  $k_1 \neq k_2$ , these expected values will be nonzero only if  $k_1 = k_2 = k_3 = k_4$  or if  $k_1 = k_3$  and  $k_2 = k_4$ , or  $k_1 = k_4$  and  $k_2 = k_3$ , or if  $k_1 = k_2$  and  $k_3 = k_4$ . Evaluating all such terms that appear in the expression for  $\mathbf{F}'$  will result in

$$\mathbf{F}'|_{\text{real signals}} = \mathbf{R} \text{Tr}(\mathbf{S} \mathbf{R}) + 2 \mathbf{R} \mathbf{S} \mathbf{R}. \quad (12.275)$$

If, on the other hand, the signals are complex-valued, then we need to know how the real and imaginary parts of each entry of  $\mathbf{x}$  relate to each other. One common assumption is that  $\mathbf{x}$  is *circularly*-Gaussian. We explain more about this in Section 1.12.2.1. For now, we only remark that if this is the case, the real and imaginary parts of each entry of  $\mathbf{x}$  and of  $\mathbf{x}'$  are such that  $E\{x_k^2\} = E\{x_k'^2\} = 0$  for all  $k$  (note that  $E\{|x_k|^2\} = r_{kk}$ ,  $E\{|x'_k|^2\} = \lambda_k$  are still nonzero.) This will result in a different expression for  $\mathbf{F}'$ :

$$\mathbf{F}'|_{\text{complex signals}} = \mathbf{R} \text{Tr}(\mathbf{S} \mathbf{R}) + \mathbf{R} \mathbf{S} \mathbf{R}. \quad (12.276)$$

## 1.12.5 Extensions and current research

In this section we briefly describe some important extensions to the basic adaptive filtering algorithms, as well as some promising current research topics. The intention is not to describe in any detail all these techniques, but rather to give a list of key references where the interested reader may find more information.

It is usual when starting such lists to put a disclaimer, that the authors do not claim to have included all the important contributions, nor that their list is really of the most important contributions to an area.

This is not intended for legal purposes, nor false modesty: the size of the literature is indeed so large (a search on “adaptive filter” on Google gives over one million hits) that we cannot claim to know with certainty the best of it all. We included therefore the techniques that we found useful during our own research and experience, which is perforce limited.

### 1.12.5.1 Finite-precision arithmetic

Any operation made using finite-precision arithmetic may involve errors. For example, the product of  $a = 0.957$  and  $b = 0.542$  would result in  $c = 0.518694$ . However, if all numbers were stored keeping only three digits, the final result would be  $\bar{c} = 0.519$ , with an error of  $c - \bar{c} = -3.06 \times 10^{-4}$  [82, 83]. These *quantization* errors will accumulate and modify the behavior of an adaptive filter in important ways. A precise analysis of the effect of quantization errors is quite complicated, since these errors are highly nonlinear functions of the variables.

When finite-precision arithmetic effects are studied, it is usually important to know exactly how and in which order the arithmetic operations are performed. In particular, the exact numerical representation used is important: fixed-point vs. floating-point, truncation vs. rounding, etc. For this reason, it is more difficult to perform analyses that are equally valid for a large class of filters. The most important differences are between gradient-based algorithms (i.e., algorithms similar to LMS and NLMS) and Hessian-based algorithms (i.e., algorithms similar to RLS). We will give references to works treating both cases, starting with LMS.

#### 1.12.5.1.1 Finite-precision effects in LMS

The simplest models for adaptive filters in finite-precision arithmetic treat the quantization error as random noise, uniformly distributed and independent of all variables [84–87]. These models show that the quantization errors will add another term to the EMSE. However, this term does not converge to zero when the step-size is reduced to zero, even in a stationary environment: quite the opposite, it is inversely proportional to the step-size (similar to the EMSE in a nonstationary environment).

Another important undesirable effect is that the product  $\mu e(n)$  may be rounded down to zero when  $e(n)$  becomes small (underflow), virtually stopping the adaptation. When the step-size is small, this may happen when the filter is rather far from the optimum solution. Therefore, there is an optimum value for the step-size—neither too large, to avoid increasing the misadjustment, neither too small, to avoid underflow and excessive growth of the EMSE. A similar effect happens in RLS [1, p. 595].

This underflow problem, which is usually known as *stopping phenomenon*, was studied in more detail in [88–91], using a nonlinear model. These works show that the adaptation does not really stop, but rather convergence is reduced to a very low rate. Note that this phenomenon is less important in floating-point arithmetic, since floating-point is able to represent much smaller numbers without underflow than fixed-point arithmetic.

Finite precision effects may have worse consequences, in particular they may make the adaptive filter unstable. In the case of LMS, this phenomenon is rare, and will only happen when the regressor sequence  $\{\phi\}$  has a very ill-conditioned covariance matrix, and the noise  $v_o$  has a nonzero mean (this nonzero mean may itself appear due to quantization errors) [92–94]. This problem is not difficult to solve, using regularization as described in Section 1.12.5.2. The problem of numerical stability in RLS is much more serious and harder to solve, and has received much attention, as we see next.

In addition to studying the effect of quantization errors in the performance of an adaptive filter, another important topic is how to reduce the computational cost of a filter, taking advantage of implementation details. One such approach is to replace the error  $e(n)$  in the LMS recursion by its sign, i.e., to use the recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \text{sign}(e(n)) \boldsymbol{\phi}(n), \quad (12.277)$$

where  $\text{sign}(\cdot)$  is the sign function, defined by

$$\text{sign}(e) = \begin{cases} 1, & \text{if } e > 0, \\ 0, & \text{if } e = 0, \\ -1, & \text{if } e < 0. \end{cases} \quad (12.278)$$

The resulting algorithm, known as *sign-LMS*, has a smaller computational cost, if  $\mu$  is restricted to be a power of two, that is, if  $\mu = 2^k$  for an integer (usually negative)  $k$ . In this case,  $\mu \text{sign}(e(n))$  is always equal to  $\pm 2^k$  (or zero), and the product  $[\mu \text{sign}(e(n))] \boldsymbol{\phi}(n)$  can be implemented in fixed-point arithmetic as  $M$  shifts, instead of  $M$  multiplications. In floating-point arithmetic, the multiplications are replaced by sums, since we would need to add  $k$  to the exponent terms of all entries of  $\boldsymbol{\phi}(n)$ . This algorithm is very robust (see also Section 1.12.5.4), although its convergence rate under Gaussian inputs is considerably slower than that of LMS. Sign-LMS was thoroughly studied in [81, 95–98].

Another possibility, that works surprisingly well, reducing the computational cost with only a slight decrease in convergence speed is the *power-of-two LMS* algorithm [99]. In it, the error is rounded to the nearest power of two, using the function:

$$f(e) = \begin{cases} \text{sign}(e), & |e| \geq 1, \\ \text{sign}(e)2^{\lfloor \ln(|e|) \rfloor}, & 2^{-B+1} \leq |e| < 1, \\ 0, & |e| < 2^{-B+1}, \end{cases}$$

where  $B$  is the number of bits used to represent the error, and  $\lfloor x \rfloor$  returns the largest integer smaller than  $x$ . In this case,  $\mu$  should also be a power of two.

### 1.12.5.1.2 Finite-precision effects in RLS

As we briefly described in Section 1.12.3.3.1, the conventional RLS is numerically unstable in finite precision arithmetic and has  $\mathcal{O}(M^2)$  complexity. To solve these problems, different versions of RLS were proposed in the literature. Many of these versions are based on coordinate transformations of the state-space representation of the conventional RLS algorithm. This representation can be obtained by replacing  $\mathbf{k}(n)$  by  $\mathbf{P}(n)\boldsymbol{\phi}(n)$  in (12.166), i.e.,

$$\mathbf{w}(n+1) = \lambda \mathbf{P}(n) \hat{\mathbf{r}}_{d\boldsymbol{\phi}}(n-1) + \mathbf{P}(n) \boldsymbol{\phi}(n) d^*(n). \quad (12.279)$$

Recalling that  $\mathbf{w}(n) = \mathbf{P}(n-1) \hat{\mathbf{r}}_{d\boldsymbol{\phi}}(n-1)$ , we can replace  $\hat{\mathbf{r}}_{d\boldsymbol{\phi}}(n-1)$  in (12.279) as a function of  $\mathbf{P}^{-1}(n-1)$  and  $\mathbf{w}(n)$ , which leads to

$$\mathbf{w}(n+1) = \lambda \mathbf{P}(n) \mathbf{P}^{-1}(n-1) \mathbf{w}(n) + \mathbf{P}(n) \boldsymbol{\phi}(n) d^*(n). \quad (12.280)$$

This equation in conjunction with the definition of the *a priori* error  $e(n)$  characterize the adaptation and filtering operations of the RLS algorithm and constitute its state-space representation, i.e.,

$$\begin{bmatrix} \mathbf{w}(n+1) \\ e^*(n) \end{bmatrix} = \begin{bmatrix} \lambda \mathbf{P}(n) \mathbf{P}^{-1}(n-1) & \mathbf{P}(n) \phi(n) \\ -\phi^H(n) & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w}(n) \\ d^*(n) \end{bmatrix}. \quad (12.281)$$

In this representation,  $\mathbf{w}(n)$  is the state,  $e^*(n)$  is the output, and  $d^*(n)$  is the input.

Performing a coordinate transformation, (12.281) can be transformed to an unlimited number of systems with the same input-output relation, and hence solving the same least-squares problem (that is, the output error is always the same). Although all these realizations are equivalent in infinite precision arithmetic, the numerical behavior will vary from one coordinate system to another. Thus, the numerical instability of the conventional RLS can be avoided by choosing a convenient transformation of (12.281) [40, 100].

An alternative method of implementing RLS is based on the QR decomposition (QRD) [14] to triangularize the input data matrix. The main advantages of QRD-RLS-based algorithms are the possibility of implementation in systolic arrays (e.g., [101]) and the improved numerical behavior in finite precision arithmetic. Some versions require  $\mathcal{O}(M^2)$  operations per iteration, but others have a reduced computational cost of  $\mathcal{O}(M)$ . The low-cost versions of QRD-RLS algorithms are known as *fast-QR* algorithms. Some important references on this subject are [38, 101–116]. A good tutorial on QRD-RLS-based algorithms can be found in [43], where the most recent developments as well as the basic concepts are covered.

Another important class of fast algorithms solves the least-squares problem in a recursive form based on the lattice realization. These algorithms are very attractive due to the possible modular implementation and low cost ( $\mathcal{O}(M)$ ). Lattice-RLS-based algorithms are derived by solving the forward and backward linear prediction problems and require time-update and order-update recursions. They explore the time-shift property of the input signal vector (such as in (12.113)) and do not compute explicitly the filter weights. Recall that many applications do not require the computation of the filter weights and have regressors that satisfy the time-shift property as, for example, channel equalization and interference cancellation (see Section 1.12.1.3). In these applications, a fast RLS algorithm can be a reasonable choice since it presents a good tradeoff between convergence rate and computational cost. One of these algorithms is the modified EF-LSL (error-feedback least-squares lattice algorithm) that presents reliable numerical properties when implemented in finite-precision arithmetic [5]. This algorithm was proposed for echo-cancellation applications and was used in the example of Section 1.12.1.1. Some lattice-RLS-based algorithms can be found in the references [5, 117–124].

More recently, lattice-RLS algorithms have been generalized to filter structures other than tapped-delay lines, in particular, to Laguerre filters [125–127].

Other fast algorithms that solve the least-squares problem in a recursive form are the fast transversal RLS (FTRL) algorithms. Unlike the lattice-based algorithms, the FTRL algorithms require only time-recursive equations and therefore, compute explicitly the filter weights. The main drawback of these algorithms is that they are very sensitive to quantization effects and become unstable if certain actions are not taken [1]. The list of fast transversal RLS algorithms is vast. Some of the most important versions can be found in the references [128–134].

### 1.12.5.1.3 DCD-RLS

The DCD-RLS algorithm, proposed in [44], is a low-complexity alternative to the RLS algorithm, based on the dichotomous coordinate-descent (DCD) algorithm for function minimization proposed in [135]. The DCD algorithm is designed to be easily implementable in hardware (such as FPGAs, for example), and thus avoids multiplications, divisions and other operations that are costly to implement in hardware—most of its operations are additions and comparisons.

The RLS weight vector estimates are given by the solution of the following set of linear equations, which we repeat from Section 1.12.3.3:

$$\widehat{\mathbf{R}}_{\phi}(n)\mathbf{w}(n+1) = \widehat{\mathbf{r}}_{d\phi}(n), \quad (12.282)$$

where  $\widehat{\mathbf{R}}_{\phi}(n)$  and  $\widehat{\mathbf{r}}_{d\phi}(n)$  are given by the recursions

$$\widehat{\mathbf{R}}_{\phi}(n) = \lambda \widehat{\mathbf{R}}_{\phi}(n-1) + \phi(n)\phi^H(n), \quad (12.283)$$

and

$$\widehat{\mathbf{r}}_{d\phi}(n) = \lambda \widehat{\mathbf{r}}_{d\phi}(n-1) + d^*(n)\phi(n). \quad (12.284)$$

The difficulty in solving these expressions is that a general solution for (12.282) involves a number of operations of the order of  $M^3(\mathcal{O}(M^3))$  for a filter with  $M$  coefficients. This is usually too costly for practical applications, except perhaps for very short filters. The classical RLS algorithm described in Section 1.12.3.3 solves this problem partially by using the matrix inversion lemma to compute the inverse of  $\widehat{\mathbf{R}}_{\phi}(n)$  recursively using the already computed inverse of  $\widehat{\mathbf{R}}_{\phi}(n-1)$ . This reduces the total number of computations to  $\mathcal{O}(M^2)$ . However, the resulting algorithm is still difficult to implement in practice, because numerical errors may easily cause divergence.

The DCD-RLS algorithm avoids both these problems by using a couple of clever tricks. First, assume that you have a good approximation  $\hat{\mathbf{w}}(n)$  for  $\mathbf{w}(n)$ . Then you could use a recursive algorithm to find an approximation  $\hat{\mathbf{w}}(n+1)$  to the solution of (12.282), using your current approximation  $\hat{\mathbf{w}}(n)$  as an initial condition. Since the initial condition is already close to the solution, you would need only a few iterations of your recursive algorithm to obtain  $\hat{\mathbf{w}}(n+1)$ . However, this idea only helps if each iteration of the recursive algorithm is very cheap to compute. We explain next how this can be done using DCD.

#### DCD minimization of quadratic functions

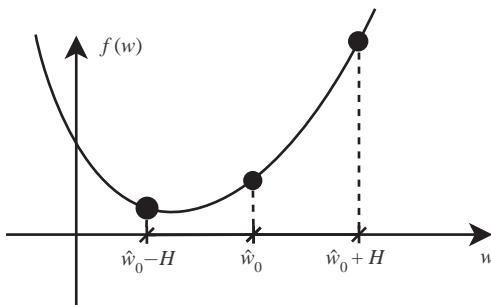
Let us consider first a one-dimensional quadratic problem

$$\min_w \left\{ f(w) = \frac{1}{2}aw^2 - bw + c \right\},$$

where  $a > 0$ ,  $b$ , and  $c$  are constants. Assume in addition that you want to implement your algorithm in hardware, using fixed-point arithmetic.

In order to solve this problem, you could proceed as follows. Assume you have an initial approximation  $\hat{w}(0)$ . Choose a step-size  $H > 0$  and compute

$$\Delta f_+ = f(\hat{w}(0) + H) - f(\hat{w}(0)), \quad \Delta f_- = f(\hat{w}(0) - H) - f(\hat{w}(0)).$$

**FIGURE 12.34**

DCD applied to one-dimensional minimization.

Then, if  $\Delta f_+ < 0$ , choose  $\hat{w}(1) = \hat{w}(0) + H$ , and if  $\Delta f_- < 0$ , choose  $\hat{w}(1) = \hat{w}(0) - H$ . If both are positive, choose  $\hat{w}(1) = \hat{w}(0)$  and reduce the step-size  $H \leftarrow H/2$  (note that if  $f(\cdot)$  is convex, the case of both  $\Delta f_+ < 0$  and  $\Delta f_- < 0$  can never occur). With the new estimate  $\hat{w}(1)$ , the procedure can be repeated to find a new improved estimate  $\hat{w}(2)$ , and so on. It can be shown that this algorithm will converge to the minimum of  $f(\cdot)$  whenever  $f(\cdot)$  is a convex function—see Figure 12.34.

Although the description given so far explains the general working of the DCD algorithm, it does not show how it can be efficiently implemented, using as few operations as possible. We will turn to this point now, but already extending the algorithm to minimize a quadratic function of several variables. Consider then the problem of finding the solution  $\mathbf{w}_o$  to

$$\min_{\mathbf{w}} \left\{ f(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{R} \mathbf{w} - \mathbf{b}^T \mathbf{w} + c \right\},$$

where  $\mathbf{R}$  is a positive-definite matrix,  $\mathbf{b}$  is a vector, and  $c$  is a constant (note that the minimum  $\mathbf{w}_o$  of  $f(\mathbf{w})$  is also the solution  $\mathbf{w}_o$  to  $\mathbf{R}\mathbf{w}_o = \mathbf{b}$ , that is, any algorithm to find  $\mathbf{w}_o$  can be also used to solve the normal equations (12.282)).

Assume that we have an initial approximation  $\hat{\mathbf{w}}(0)$  to the solution, and we want to find an improved approximation, changing one entry of  $\hat{\mathbf{w}}(0)$  at a time. In order to differentiate each new approximation, we use the following notation. Define  $\hat{\mathbf{w}}^{(0)}(0) = \hat{\mathbf{w}}(0)$ . We shall first seek an improved estimate to the first entry of  $\hat{\mathbf{w}}^{(0)}(0)$ . The resulting vector will be denoted by  $\hat{\mathbf{w}}^{(1)}(0)$ . Continuing, we find an improved estimate to the second entry of  $\hat{\mathbf{w}}^{(1)}(0)$ , resulting in vector  $\hat{\mathbf{w}}^{(2)}(0)$  and so on. After updating all entries of the initial estimate, we have  $\hat{\mathbf{w}}^{(M)}(0)$ . We then let  $\hat{\mathbf{w}}^{(0)}(1) = \hat{\mathbf{w}}^{(M)}(0)$ , and repeat the procedure.

Then we need to check if  $f(\hat{\mathbf{w}}^{(0)}(0) \pm H\mathbf{e}_1) < f(\hat{\mathbf{w}}(0))$  if we add or subtract  $H$  to the first entry of  $\hat{\mathbf{w}}^{(0)}(0)$  ( $\mathbf{e}_1$  is the vector  $[1 \ 0 \ \dots \ 0]^T$ ). The change  $\Delta f_+$  in the cost-function obtained by adding  $H$  to the first entry of  $\hat{\mathbf{w}}^{(0)}(0)$  is

$$\begin{aligned} \Delta f_+ &= \frac{1}{2} \left( \hat{\mathbf{w}}^{(0)}(0) + H\mathbf{e}_1 \right)^T \mathbf{R} \left( \hat{\mathbf{w}}^{(0)}(0) + H\mathbf{e}_1 \right) - \mathbf{b}^T \left( \hat{\mathbf{w}}^{(0)}(0) + H\mathbf{e}_1 \right) + c \quad (12.285) \\ &\quad - \left( \frac{1}{2} \hat{\mathbf{w}}^{(0)T}(0) \mathbf{R} \hat{\mathbf{w}}^{(0)}(0) - \mathbf{b}^T \hat{\mathbf{w}}^{(0)}(0) + c \right) \end{aligned}$$

$$= \mathbf{H}\mathbf{e}_1^T \mathbf{R}\hat{\mathbf{w}}^{(0)}(0) + \frac{1}{2}H^2\mathbf{e}_1^T \mathbf{R}\mathbf{e}_1 - H\mathbf{b}^T \mathbf{e}_1. \quad (12.286)$$

Similarly, the variation  $\Delta f_-$  obtained from subtracting  $H$  from the first entry of  $\hat{\mathbf{w}}(0)$  is

$$\Delta f_- = -\mathbf{H}\mathbf{e}_1^T \mathbf{R}\hat{\mathbf{w}}^{(0)}(0) + \frac{1}{2}H^2\mathbf{e}_1^T \mathbf{R}\mathbf{e}_1 + H\mathbf{b}^T \mathbf{e}_1. \quad (12.287)$$

Therefore, we should make

$$\hat{\mathbf{w}}^{(1)}(0) = \begin{cases} \hat{\mathbf{w}}^{(0)}(0) + H\mathbf{e}_1, & \text{if } \Delta f_+ < 0, \\ \hat{\mathbf{w}}^{(0)}(0) - H\mathbf{e}_1, & \text{if } \Delta f_- < 0, \\ \hat{\mathbf{w}}^{(0)}(0), & \text{otherwise.} \end{cases} \quad (12.288)$$

The same steps can be repeated for the second entry of  $\hat{\mathbf{w}}^{(1)}(0)$  and so on, that is, we can update the  $(m+1)$ th entry of  $\hat{\mathbf{w}}^{(m)}(0)$  to obtain a new approximation  $\hat{\mathbf{w}}^{(m+1)}(0)$  for  $m = 0, \dots, M-1$ . After obtaining  $\hat{\mathbf{w}}^{(M)}(0)$ , we have updated all entries of  $\hat{\mathbf{w}}(0)$ . We can then make  $\hat{\mathbf{w}}^{(0)}(1) = \hat{\mathbf{w}}^{(M)}(0)$  and repeat the procedure. However, if no update is made for  $m = 1, \dots, M$ , before repeating we decrease the step  $H$  by a factor of two (i.e.,  $H \leftarrow H/2$ ). By reducing  $H$  only when no update is necessary for any entry of the vector, we avoid problems that would appear if the initial value of  $H$  were too small.

Implemented as described so far, the algorithm has a high computational cost. For example, for the evaluation of  $\Delta f_+$  at each step we must

- Evaluate  $\mathbf{e}_m^T \mathbf{R}\hat{\mathbf{w}}^{(m-1)}(i)$ . This is equivalent to multiplying the  $m$ th row of  $\mathbf{R}$  by  $\hat{\mathbf{w}}^{(m-1)}(i)$ , involving  $M$  multiplications and  $M-1$  additions.
- Multiply  $-\mathbf{e}_m^T \mathbf{R}\hat{\mathbf{w}}^{(m-1)}(i) + \mathbf{b}^T \mathbf{e}_m$  by  $H$  and add the result to  $(H^2/2)R_{m,m}$  ( $R_{m,m}$  is the element  $(m, m)$  of  $\mathbf{R}$ ). This requires 2 multiplications and 2 additions, assuming  $H^2/2$  is pre-computed.

The same steps would be necessary for the computation of  $\Delta f_-$ . Note that all these computations would have to be repeated at each step, therefore the update of all entries of  $\hat{\mathbf{w}}^{(0)}(i)$  would require more than  $M^2$  multiplications.

However, the number of computations can be reduced substantially, if we take advantage of some properties of hardware implementations of arithmetic operations, as we describe next. Assume in the following that all variables are fixed-point binary numbers (see Box 8).

First, we can avoid the computation of  $\mathbf{R}\hat{\mathbf{w}}^{(m-1)}(i)$  by introducing a residue vector that is updated at each step. Let  $\mathbf{r}^{(0)}(0) = \mathbf{b} - \mathbf{R}\hat{\mathbf{w}}^{(0)}(0)$  be the initial residue vector. Note that this is the residue of solving  $\mathbf{R}\mathbf{w} = \mathbf{b}$  using  $\hat{\mathbf{w}}^{(0)}(0)$  as an approximation to the solution.

At the end of the first step,  $\hat{\mathbf{w}}^{(0)}(0)$  is updated as in (12.288). Since only one entry of  $\hat{\mathbf{w}}^{(0)}(0)$  is changed, the modification in the residue is simple to evaluate

$$\mathbf{r}^{(1)}(0) = \mathbf{b} - \mathbf{R}\hat{\mathbf{w}}^{(1)}(0) = \begin{cases} \mathbf{b} - \mathbf{R}(\hat{\mathbf{w}}^{(0)}(0) + H\mathbf{e}_1) = \mathbf{r}^{(0)}(0) - H\mathbf{R}\mathbf{e}_1, & \text{if } \Delta f_+ < 0, \\ \mathbf{b} - \mathbf{R}(\hat{\mathbf{w}}^{(0)}(0) - H\mathbf{e}_1) = \mathbf{r}^{(0)}(0) + H\mathbf{R}\mathbf{e}_1, & \text{if } \Delta f_- < 0, \\ \mathbf{r}^{(0)}(0), & \text{otherwise.} \end{cases} \quad (12.289)$$

This still requires  $M$  multiplications ( $H$  multiplied by the first column of  $\mathbf{R}$ ). However, if  $H$  is a power of two, that is, if  $H = 2^k$  for some integer  $k$ , then the computation of  $\mathbf{r}^{(1)}(0)$  in fixed-point

arithmetic requires only  $M$  shifts and  $M$  additions, which are much easier to implement in hardware than multiplications. Moreover, if both  $\Delta f_+$  and  $\Delta f_-$  are positive, no update is necessary.

Next, we note that it is not really necessary to compute both  $\Delta f_+$  and  $\Delta f_-$ : from (12.286) and (12.287), when updating the  $m$ th element of  $\hat{\mathbf{w}}^{(m-1)}(i)$  at step  $i$ , we have

$$\Delta f_+ = -H\mathbf{e}_m^T \mathbf{r}^{(m-1)}(i) + \frac{1}{2}H^2 \mathbf{e}_m^T R \mathbf{e}_m = -H r_m^{(m-1)}(i) + \frac{1}{2}H^2 R_{m,m}, \quad (12.290)$$

where  $r_m^{(m-1)}(i)$  is the  $m$ th entry of the residue vector  $\mathbf{r}^{(m-1)}(i)$ ,  $\mathbf{r}^{(0)}(i+1) \stackrel{\Delta}{=} \mathbf{r}^{(M)}(i)$ , and  $\mathbf{e}_m$  is the  $m$ th column of the  $M \times M$  identity matrix. Similarly,

$$\Delta f_- = H r_m^{(m-1)}(i) + \frac{1}{2}H^2 R_{m,m}. \quad (12.291)$$

Then, recalling that  $H$  and  $R_{m,m}$  are positive ( $R$  is positive-definite),  $\Delta f_+ < 0$  only if

$$r_m^{(m-1)}(i) > \frac{1}{2}H R_{m,m} \geq 0.$$

Repeating the arguments for  $\Delta f_-$ , we see that the  $m$ th entry of  $\hat{\mathbf{w}}^{(m-1)}(i)$  will be updated if and only if

$$|r_m^{(m-1)}(i)| > \frac{1}{2}H R_{m,m}, \quad (12.292)$$

with updates

$$\hat{\mathbf{w}}^{(m)}(i) = \hat{\mathbf{w}}^{(m-1)}(i) + H \text{sign}(r_m^{(m-1)}(i)) \mathbf{e}_m, \quad (12.293)$$

$$\mathbf{r}^{(m)}(i) = \mathbf{r}^{(m-1)}(i) - H \text{sign}(r_m^{(m-1)}(i)) R \mathbf{e}_m. \quad (12.294)$$

Equations 12.292–12.294 require one comparison,  $M + 1$  shifts, and  $M + 1$  additions. Table 12.10 summarizes the DCD algorithm for minimization of quadratic cost-functions. Note that the algorithm will update the entries of  $\hat{\mathbf{w}}(i)$  bit by bit. It therefore makes sense to choose the initial step-size  $H$  the power-of-two value represented by the most-significant bit in the fixed-point representation being used. The conditional jump in step 10 prevents problems in case  $H$  is chosen too small.

As seen in Table 12.10, the total number of operations never exceeds  $(2M + 2)(B - 1) + N_u(4M + 3)$  (note that here we are counting additions, comparisons and shifts as operations—the algorithm does not involve multiplications). In DCD-RLS, the maximum number of updates  $N_u$  can be chosen between 1 and 4 with good performance, while keeping the cost low.

When applied to RLS, one final simplification can be used when implementing DCD: instead of updating all entries of vector  $\hat{\mathbf{w}}(i)$ , we can only update the entry corresponding to the largest residue. As explained in [44], the resulting algorithm is still guaranteed to converge to the optimum solution.

### DCD-RLS

As we mentioned before, DCD-RLS uses the DCD algorithm just described to solve the normal equations as each new sample arrives. Since DCD is an iterative algorithm, we can use the current solution  $\mathbf{w}(n)$  as the initial value for computation of  $\mathbf{w}(n + 1)$ . In this way, the number of updates  $N_u$  of DCD can be

**Table 12.10** Summary of DCD Algorithm for Minimization of a Quadratic Cost-Function

Step	Initialization: Choose step $H = 2^k$ , number of bits $B$ , and maximum update count $N_u$ . Let $\hat{w} = 0$ , $r = b$ , $h = H$ , and $c = 0$ .	Number of	
		Additions, Shifts, Comparisons	Multiplications
1	for $b = 0 : B - 1$		
2	$h \leftarrow h/2$	1	0
3	flag $\leftarrow 0$	0	0
4	for $m = 1 : M$		
5	if $ r_m  > \frac{1}{2} h R_{m,m}$	2	0
6	$\hat{w}_m \leftarrow \hat{w}_m + h \text{sign}(r_m)$	1	0
7	$r \leftarrow r - h \text{sign}(r_m) R e_m$	$2M$	0
8	$c \leftarrow c + 1$ , flag $\leftarrow 1$	1	0
9	if $c > N_u$ , stop algorithm.	1	0
10	if flag = 1, go to step 2	1	0
Total: (worst case)		$\leq (2M + 2)(B - 1)$	0
		$N_u(2M + 3)$	

restricted to a small number (in many situations from 1 to 4 updates is enough for performance close to that of exact RLS). This further reduces the computational cost of the algorithm.

In the following, we will use  $w(n+1)$  to denote the exact solution of (12.282) (i.e., the exact RLS estimate), and  $\hat{w}(n+1)$  to denote the approximation computed by DCD-RLS (both would be equal if we let  $N_u \rightarrow \infty$ ). Assume that at time instant  $n$ , we have available both  $\hat{w}(n)$  and the residue

$$r(n) = \hat{r}_{d\phi}(n-1) - \hat{R}_\phi(n-1)\hat{w}(n), \quad (12.295)$$

In order to apply the DCD algorithm to solve for  $\hat{w}(n+1)$ , it is convenient to update the difference

$$\Delta w(n+1) = w(n+1) - \hat{w}(n). \quad (12.296)$$

Define also

$$\Delta \hat{R}_\phi(n) = \hat{R}_\phi(n) - \hat{R}_\phi(n-1), \quad (12.297)$$

$$\Delta \hat{r}_{d\phi}(n) = \hat{r}_{d\phi}(n) - \hat{r}_{d\phi}(n-1). \quad (12.298)$$

The normal Eq. (12.282) then become

$$\hat{R}_\phi(n) [\hat{w}(n) + \Delta w(n+1)] = \hat{r}_{d\phi}(n),$$

and thus

$$\begin{aligned} \hat{R}_\phi(n) \Delta w(n+1) &= \hat{r}_{d\phi}(n-1) + \Delta \hat{r}_{d\phi}(n) - \hat{R}_\phi(n-1)\hat{w}(n) - \Delta \hat{R}_\phi(n)\hat{w}(n) \\ &= r(n) + \Delta \hat{r}_{d\phi}(n) - \Delta \hat{R}_\phi(n)\hat{w}(n) \stackrel{\Delta}{=} \beta(n). \end{aligned} \quad (12.299)$$

Given an approximate solution  $\Delta\hat{\mathbf{w}}(n+1)$  to these equations, we obtain an updated approximate solution  $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \Delta\hat{\mathbf{w}}(n+1)$ . Note that the residue of solving (12.282) using  $\hat{\mathbf{w}}(n+1)$ ,

$$\begin{aligned}\mathbf{r}(n+1) &= \hat{\mathbf{r}}_{d\phi}(n) - \hat{\mathbf{R}}_\phi(n)\hat{\mathbf{w}}(n+1) \\ &= \hat{\mathbf{r}}_{d\phi}(n-1) + \Delta\hat{\mathbf{r}}_{d\phi}(n) - \hat{\mathbf{R}}_\phi(n)[\hat{\mathbf{w}}(n) + \Delta\hat{\mathbf{w}}(n+1)] \\ &= \hat{\mathbf{r}}_{d\phi}(n-1) + \Delta\hat{\mathbf{r}}_{d\phi}(n) - \hat{\mathbf{R}}_\phi(n)\Delta\hat{\mathbf{w}}(n+1) - [\hat{\mathbf{R}}_\phi(n-1) + \Delta\hat{\mathbf{R}}_\phi(n)]\hat{\mathbf{w}}(n) \\ &= \mathbf{r}(n) + \Delta\hat{\mathbf{r}}_{d\phi}(n) - \hat{\mathbf{R}}_\phi(n)\Delta\hat{\mathbf{w}}(n+1) - \Delta\hat{\mathbf{R}}_\phi(n)\hat{\mathbf{w}}(n) \\ &= \boldsymbol{\beta}(n) - \hat{\mathbf{R}}_\phi(n)\Delta\hat{\mathbf{w}}(n+1),\end{aligned}\tag{12.300}$$

is exactly the residue of approximately solving (12.299) by  $\Delta\hat{\mathbf{w}}(n+1)$ .

To complete, we only need to evaluate  $\Delta\hat{\mathbf{R}}_\phi(n)$  and  $\Delta\hat{\mathbf{r}}_{d\phi}(n)$ . From (12.297), (12.298), (12.283) and (12.284), we obtain

$$\Delta\hat{\mathbf{R}}_\phi(n) = (\lambda - 1)\hat{\mathbf{R}}_\phi(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n),\tag{12.301}$$

$$\Delta\hat{\mathbf{r}}_{d\phi}(n) = (\lambda - 1)\hat{\mathbf{d}}_{d\phi}(n-1) + d(n)\boldsymbol{\phi}(n).\tag{12.302}$$

Substituting these results in the definition of  $\boldsymbol{\beta}(n)$ , we obtain

$$\begin{aligned}\boldsymbol{\beta}(n) &= \mathbf{r}(n) + (\lambda - 1)\hat{\mathbf{r}}_{d\phi}(n-1) + d(n)\boldsymbol{\phi}(n) - [(\lambda - 1)\hat{\mathbf{R}}_\phi(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)]\hat{\mathbf{w}}(n) \\ &= \mathbf{r}(n) + (\lambda - 1)\mathbf{r}(n) + e(n)\boldsymbol{\phi}(n) = \lambda\mathbf{r}(n) + e(n)\boldsymbol{\phi}(n).\end{aligned}\tag{12.303}$$

We can therefore use the DCD algorithm to compute an approximate solution to (12.299) and update the residue defined by (12.300), as described in Table 12.11.

As we mentioned before, the number of operations of the DCD algorithm can be reduced if we modify the algorithm in Table 12.10 to update only the entries of the weight vector corresponding to the largest residue entry, as described in Table 12.12.

The total number of operations of DCD-RLS as described in Tables 12.11 and 12.12 is  $M^2 + 4M$  multiplications and  $M^2/2 + 3.5M + (2M + 1)N_u + B$  additions. Most of these operations are due to the updating of  $\hat{\mathbf{R}}_\phi$  in Table 12.11, which is responsible for the terms in  $M^2$ . While this cannot be helped for general regressors  $\boldsymbol{\phi}(n)$ , the operation count can be substantially reduced when our filter is modeling an FIR relation as in (12.41) (i.e.,  $\boldsymbol{\phi}(n)$  is a tap-delay line), as we show next.

If

$$\boldsymbol{\phi}(n) = [x(n) \ x(n-1) \ \dots \ x(n-M+1)]^T,\tag{12.304}$$

**Table 12.11** Summary of the DCD RLS Algorithm for General Regressors

Initialization:

$$\hat{\mathbf{w}}(0) = \mathbf{0}, \mathbf{r}(0) = \mathbf{0}, \hat{\mathbf{R}}_\phi(-1) = \Pi > 0$$

for  $n = 0, 1, 2, \dots$

$$\hat{\mathbf{R}}_\phi(n) = \lambda \hat{\mathbf{R}}(n-1) + \phi(n)\phi^T(n) \text{ (use (12.306) for tap-delay lines)}$$

$$\hat{y}(n) = \hat{\mathbf{w}}^T(n)\phi(n),$$

$$\mathbf{e}(n) = d(n) - \hat{y}(n),$$

$$\beta(n) = \lambda \mathbf{r}(n) + \mathbf{e}(n)\phi(n),$$

Use DCD algorithm to compute new  $\Delta \hat{\mathbf{w}}(n+1)$  and  $\mathbf{r}(n+1)$  from  $\hat{\mathbf{R}}_\phi(n)\Delta \hat{\mathbf{w}}(n+1) = \beta(n)$ ,

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \Delta \hat{\mathbf{w}}(n+1).$$

**Table 12.12** Summary of Leading-Element DCD Algorithm for Use in DCD-RLS, from [44]

Step	<b>Initialization: Choose initial condition step <math>H = 2^k</math>, number of bits <math>B</math>, and maximum update count <math>N_u</math>. Let <math>\Delta \hat{\mathbf{w}} = \mathbf{0}</math>, <math>\mathbf{r} = \beta(n)</math>, <math>h = H/2</math>, <math>b = 1</math>.</b>
	for $k = 1 : N_u$
1	$p = \arg \max_{1 \leq m \leq M} \{ r_m \}$ , go to step 4
2	$b \leftarrow b + 1, h \leftarrow h/2$
3	Stop if $b > B$
4	Go to step 2 if $ r_p  \leq (h/2)R_{p,p}$
5	$\Delta \hat{\mathbf{w}}_p = \Delta \hat{\mathbf{w}}_p + \text{sign}(r_p)h$
6	$\mathbf{r} = \mathbf{r} - \text{sign}(r_p)h\hat{\mathbf{R}}_\phi \mathbf{e}_p$

then  $\hat{\mathbf{R}}(n)$  and  $\hat{\mathbf{R}}(n-1)$  share a common structure. Assume that the initial condition  $\hat{\mathbf{R}}(-1) = \Pi = \text{diag}(\lambda^{M-1}, \lambda^{M-2}, \dots, 1)$ . Then

$$\begin{aligned} \hat{\mathbf{R}}(0) &= \lambda \begin{bmatrix} \lambda^{M-1} & 0 & \dots & 0 \\ 0 & \lambda^{M-2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} + \begin{bmatrix} x(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} x(0) & 0 & \dots & 0 \end{bmatrix}^T \\ &= \begin{bmatrix} \lambda^M + x^2(0) & 0 & \dots & 0 \\ 0 & \lambda^{M-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda \end{bmatrix} = \begin{bmatrix} \lambda^M + x^2(0) & \mathbf{0}^T \\ \mathbf{0} & [\hat{\mathbf{R}}_\phi(-1)]_{1:M-1, 1:M-1} \end{bmatrix}, \end{aligned}$$

where  $[\hat{\mathbf{R}}_\phi(-1)]_{1:M-1, 1:M-1}$  denotes the top-left  $(M-1) \times (M-1)$  block of  $\hat{\mathbf{R}}_\phi(-1)$ . We note that, except for its first row and column,  $\hat{\mathbf{R}}(0)$  can be obtained directly from  $\hat{\mathbf{R}}(-1)$ .

Following this observation, we claim that for tapped-delay line regressors, the computation of  $\widehat{\mathbf{R}}_\phi(n)$  in Table 12.11 can be replaced by an update of only its first column  $[\widehat{\mathbf{R}}_\phi(n)]_1$ , that is,

$$[\widehat{\mathbf{R}}_\phi(n)]_1 = \lambda[\widehat{\mathbf{R}}_\phi(n-1)]_1 + x(n)\phi(n). \quad (12.305)$$

Since  $\widehat{\mathbf{R}}_\phi(n)$  is symmetric, its first row is the transpose of (12.306), and does not need to be evaluated again.

We show next by induction that this pattern holds for all  $n$ . We just showed that it is true for  $n = 0$ . Assume then that for a certain  $n$ , it holds that

$$\widehat{\mathbf{R}}_\phi(n) = \left[ \begin{array}{c|c} \lambda[\widehat{\mathbf{R}}_\phi(n-1)]_1 + x(n)\phi(n) & (\cdot)^T \\ \hline [\widehat{\mathbf{R}}_\phi(n-1)]_{1:M-1,1:M-1} \end{array} \right], \quad (12.306)$$

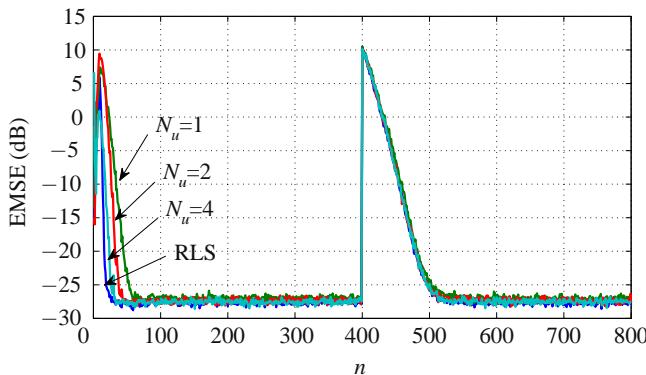
where we use  $(\cdot)^T$  to indicate that the entries of the first row are the same as the entries of the first column. We show next that  $\widehat{\mathbf{R}}_\phi(n+1)$  must follow the same pattern. Indeed,

$$\begin{aligned} \widehat{\mathbf{R}}_\phi(n+1) &= \lambda\widehat{\mathbf{R}}_\phi(n) + \begin{bmatrix} x(n+1) \\ x(n) \\ \vdots \\ n(n-M+2) \end{bmatrix} \begin{bmatrix} x(n+1) & x(n) & \dots & x(n-M+2) \end{bmatrix}^T \\ &= \left[ \begin{array}{c|c} \lambda[\widehat{\mathbf{R}}_\phi(n)]_1 & (\cdot)^T \\ \hline \lambda[\widehat{\mathbf{R}}_\phi(n-1)]_{1:M-1,1:M-1} \end{array} \right] \\ &\quad + \begin{bmatrix} x(n+1)\phi(n+1) \\ \begin{bmatrix} x^2(n) & \dots & x(n)x(n-M+2) \\ \vdots & \ddots & \vdots \\ x(n)x(n-M+2) & \dots & x^2(n-m+2) \end{bmatrix}^T \end{bmatrix} \\ &= \left[ \begin{array}{c|c} \lambda[\widehat{\mathbf{R}}_\phi(n)]_1 + x(n+1)\phi(n+1) & (\cdot)^T \\ \hline [\widehat{\mathbf{R}}_\phi(n)]_{1:M-1,1:M-1} \end{array} \right], \end{aligned}$$

where in the last step we identified the  $(2, 2)$  block of the intermediate result with the first  $M - 1$  rows and columns of  $\widehat{\mathbf{R}}_\phi(n)$ , from (12.283).

Note that in practice we should not actually move the entries of  $\widehat{\mathbf{R}}_\phi(n-1)$  to their new position; rather, we should use an indexing system to access the entries of  $\widehat{\mathbf{R}}_\phi(n)$  to avoid this costly moving operation. The total number of operations of DCD-RLS using (12.306) is  $3M$  multiplications and  $2MN_u + 6M$  additions. This is not much larger than the complexity of NLMS. The DCD-RLS algorithm has been implemented in FPGAs and run for long periods of time without observation of divergence [44].

Figure 12.35 shows a comparison of RLS and DCD-RLS in a system identification problem. We are trying to identify an  $M = 10$  FIR filter, so we use (12.306) to update  $\widehat{\mathbf{R}}_\phi(n)$  with lower cost. The input

**FIGURE 12.35**

EMSE for RLS and DCD-RLS for different values of  $N_u$ , with  $\lambda = 0.95$ ,  $M = 10$ , and for DCD,  $H = 2$  and  $B = 16$ . Average of  $L = 500$  realizations.

signal  $x(n)$  is white Gaussian noise with zero mean and unit variance. The optimum coefficient vector  $\mathbf{w}_o(n)$  follows a random-walk model as in (12.182) with  $\mathbf{Q} = 10^{-5} \mathbf{I}$  (the initial value,  $\mathbf{w}_o(0)$  is random, taken from a Gaussian distribution with covariance matrix equal to  $\mathbf{I}$ ). Both RLS and RLS-DCD use a forgetting factor of  $\lambda = 0.95$ . RLS-DCD is implemented with leading DCD (Table 12.12),  $B = 16$  and  $H = 2$ . The maximum number of updates  $N_u$  is varied from 1 to 8. The optimum vector  $\mathbf{w}_o(n)$  suffers an abrupt change at  $n = 400$ . The learning curves were obtained from the average of  $L = 500$  realizations. As the figure shows, even for  $N_u = 1$ , the performance of RLS-DCD is very close to that of RLS. The only difference is a smaller convergence rate at the start of the algorithm. However, after the abrupt modification of  $\mathbf{w}_o(n)$  at instant  $n = 400$ , all filters converge equally quickly to the new vector. This shows that RLS-DCD does not lose in tracking capability when compared to standard RLS.

### 1.12.5.2 Regularization

There are applications in which the autocorrelation matrix  $\mathbf{R}_\phi$  may become very ill-conditioned (i.e., nearly singular). In these cases, it can be shown that the LMS weight estimates may slowly drift to quite large values, even though the error remains small. The problem arises when the filter is implemented in finite-precision arithmetic, because the variables holding the filter weights may overflow, thereby making the error grow suddenly very large.

This can happen, for example, for certain kinds of equalizers in communications [87, 92, 93]. The mechanism through which the LMS coefficients may slowly drift is explained in [94, 136].

The usual solution to this problem is to modify the cost function to add a term proportional to  $\|\mathbf{w}\|^2$ , so that large values of the weight vector are penalized:

$$J_{\text{leaky}} = E \left\{ |d(n) - \mathbf{w}^H \boldsymbol{\phi}(n)|^2 \right\} + \alpha \|\mathbf{w}\|^2, \quad (12.307)$$

where  $\|\cdot\|$  is the Euclidean norm and  $\alpha > 0$  is a small constant. With this cost function, the corresponding algorithm is

$$\mathbf{w}(n+1) = (1 - \mu\alpha)\mathbf{w}(n) + \mu e^*(n)\boldsymbol{\phi}(n). \quad (12.308)$$

This is known as *leaky-LMS* algorithm [4]. It can be shown that the introduction of leakage will bias the solution away from the Wiener solution, but will prevent any excessive growth of the weight coefficients. A detailed analysis of the leaky-LMS algorithm is available in [137]. Reference [94] proposes a modified version of the algorithm without the bias.

Difficulties with singular or near-singular autocorrelation matrices also appears with RLS. For example, if the input is a periodic signal with little noise, the autocorrelation matrix will become nearly singular, and matrix  $\mathbf{P}(n)$  in RLS will diverge. This problem, as well as a solution to it using variable forgetting-factor, is described in [138]. A constant regularization method based on DCD can be found in [139]. Another approach, valid only for short filters, but which allows constant regularization, is described in [140].

The leaky-LMS algorithm has found other important applications, most particularly in beamforming, in which it is used to turn the beamformer robust against array imperfections [141]. The algorithms used in beamforming are examples of constrained adaptive filters, in which the cost function incorporates equality conditions that must be satisfied at all times—see Section 1.12.5.11.

Another algorithm closely related to leaky-LMS, in which the extra term in the cost function penalizes large values of the filter output (that is,  $\alpha|\mathbf{w}^H(n)\boldsymbol{\phi}(n)|^2$  instead of  $\alpha\|\mathbf{w}(n)\|^2$  in (12.307)), was recently employed to improve the performance of FIR adaptive filters in the presence of saturation nonlinearities [142].

More recent applications of regularization involve the use of the  $\ell_1$ -norm,

$$\|\mathbf{w}\|_1 = |w_0| + |w_1| + \dots + |w_{M-1}|,$$

instead of the Euclidean norm. This is because the  $\ell_1$  norm promotes solutions that are sparse, that is, a filter that minimizes the cost function

$$J_{\text{sparse}} = E \left\{ \left| d(n) - \mathbf{w}^H \boldsymbol{\phi}(n) \right|^2 \right\} + \alpha \|\mathbf{w}\|_1, \quad (12.309)$$

will favor solutions in which the vector  $\mathbf{w}$  has only a few nonzero entries (an intuitive explanation for this can be found in [143], which is a good introduction to compressive sensing). Algorithms for promoting sparsity have been receiving much attention lately, due to the interest in compressive sensing applications [144–148]. See also the references for the proportionate NLMS (PNLMS) algorithm in Section 1.12.5.3.

### 1.12.5.3 Variable step-size

The steady-state analysis of Section 1.12.4.4 shows that the step-size  $\mu$  plays an import role in controlling the performance of the LMS algorithm since it is one of the main factors affecting the convergence rate and the misadjustment. To obtain a better tradeoff between convergence speed and steady-sate EMSE, several authors proposed different forms of adjusting the step-size of LMS. Variable step-size algorithms allow the filters to dynamically adjust their performance in response to conditions in the input data and error signals.

In this context, one of the most popular algorithms is the variable step-size LMS (VSLMS) proposed in [149]. In the VSLMS algorithm, each coefficient  $w_k(n)$ ,  $k = 0, \dots, M - 1$  is updated with a time-varying step-size  $\mu_k(n)$ , whose adjustment is given by

$$\mu_k(n) = \mu_k(n - 1) + \rho \text{sign}[e(n)\phi_k(n)] \text{sign}[e(n - 1)\phi_k(n - 1)], \quad (12.310)$$

where  $\rho$  is a small positive constant. This algorithm operates in a very intuitive way. When the algorithm has not yet converged, the gradient term  $e(n)\phi_k(n)$  shows a positive or negative direction in successive iterations and the step-size  $\mu_k(n)$  increases up to an allowed maximum value. On the other hand, near steady-state  $e(n)\phi_k(n)$  approaches zero and its sign changes in successive iterations, which reduces  $\mu_k(n)$  (the step-size is lower-bounded by a small minimum value). Therefore, during the initial convergence VSLMS uses a large step-size, which leads to a high convergence rate. When its coefficients are close to the optimum solution, the algorithm is updated with a small step-size, which allows VSLMS to achieve a small EMSE. A variant of this algorithm is obtained by dropping the sign functions in (12.310), i.e.,

$$\mu_k(n) = \mu_k(n - 1) + \rho e(n)e(n - 1)\phi_k(n)\phi_k(n - 1). \quad (12.311)$$

The choice between (12.310) and (12.311) depends on the application [150].

The NLMS algorithm, proposed independently by Nagumo and Noda [151] and Albert and Gardner [152], can be interpreted a variable step-size LMS with a particular choice for  $\mu(n)$  (see Section 1.12.3.2). An important variant of NLMS is the so-called *power-normalized* LMS (PN-LMS) algorithm, which uses the following variable step-size

$$\mu(n) = \frac{\tilde{\mu}}{\varepsilon + p(n)}, \quad (12.312)$$

where

$$p(n) = \gamma p(n - 1) + (1 - \gamma)|\phi_0(n)|^2, \quad (12.313)$$

with  $p(-1) = 0$  and  $\gamma$  being a positive scalar chosen from within the interval  $0 < \gamma \leq 1$ . This algorithm is useful when the regressor is a tapped-delay line, that is, when  $\phi_i(n) = x(n - i)$ , so that the scalar  $p(n)$  is an estimate for the power of the input sequence  $\{x(n)\}$ . The range of  $\tilde{\mu}$  in (12.312) is  $0 \leq \tilde{\mu} \leq \frac{2}{M}$  [3].

Over the years, several variable step-size algorithms were proposed in the literature. Some of these algorithms are in the papers [153–167] and in their references. Besides the variable step-size approaches, an alternative scheme to improve the tradeoff between convergence speed and steady-state EMSE is constituted by combinations of adaptive algorithms, described in Section 1.12.5.8.1.

Another class of variable step-size algorithms was developed specifically to accelerate the convergence of filters whose optimum weight vector  $\mathbf{w}_o$  is sparse, that is, has only a small fraction of nonzero elements. The primary application for this is echo cancellation, mainly for echo due to reflections in the telephone connection. The algorithms are known as *proportionate* NLMS (PNLMS) algorithms, and have usually very good performance for approximating sparse weight vectors [168–172].

**Table 12.13** Fourth- and Sixth-Order Moments of Different Distributions

Distribution	$\frac{E\{(x-\mu_x)^4\}}{\sigma_x^4}$	$\frac{E\{(x-\mu_x)^6\}}{\sigma_x^6}$
Binary	1	1
Uniform	9/5	27/7
Gaussian	3	15
Exponential	6	90

#### 1.12.5.4 Non-Gaussian noise and robust filters

Consider a problem in which we know that the regressor  $\phi(n)$  and the desired signal  $d(n)$  are related through a model (we assume for simplicity that all variables are real)

$$d(n) = \mathbf{w}_o^T \phi(n) + v(n), \quad (12.314)$$

in which  $v(n)$  is independent of  $\phi(n)$ . It is shown in [173] that minimization of the *fourth* power of the error may lead to an adaptive filter with better compromise between convergence rate and excess mean-square error than LMS, if  $v(n)$  is *sub-Gaussian*. Let us explain what this means.

Recall that for any random variable  $x$ , it holds that  $0 \leq \sigma_x^2 = E\{x^2\} - (E\{x\})^2$ . Similarly, if we define  $\mu_x = E\{x\}$ ,

$$E\{(x - \mu_x)^4\} \geq \left(E\{(x - \mu_x)^2\}\right)^2 = \sigma_x^4, \quad E\{(x - \mu_x)^6\} \geq \left(E\{(x - \mu_x)^2\}\right)^3 = \sigma_x^6.$$

In general, we can write  $E\{(x - \mu_x)^4\} = \alpha \sigma_x^4$ , with  $\alpha \geq 1$ . For example, consider the distributions in Table 12.13. Distributions for which  $\frac{E\{(x - \mu_x)^4\}}{\sigma_x^4} < 3$ , such as binary and uniform, are called *sub-Gaussian*, and those for which  $\frac{E\{x^4\}}{(E\{(x - \mu_x)^2\})^2} > 3$ , such as the exponential, are called *super-Gaussian*. These relations are usually given in terms of the *kurtosis*  $\gamma_4$  of a distribution

$$\gamma_4 = \frac{E\{(x - \mu_x)^4\}}{\sigma_x^4} - 3.$$

Therefore, a distribution is sub-Gaussian if its kurtosis is negative, and super-Gaussian if its kurtosis is positive. A distribution with positive kurtosis is such that its probability density function (pdf)  $f(x)$  decays more slowly than the Gaussian as  $|x| \rightarrow \infty$ .

The authors of [173] show that when the noise  $v(n)$  is sub-Gaussian, an algorithm based on the minimization of

$$J_{LMP} = E\{|e(n)|^p\}, \quad (12.315)$$

with  $p > 2$  results in lower steady-state excess mean-square error (EMSE) than LMS. They study particularly the case  $p = 4$ , for which the stochastic gradient algorithm is of the form

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e^3(n) \phi(n). \quad (12.316)$$

This algorithm is known as *least-mean fourth (LMF)* algorithm. Analyses of general algorithms with error nonlinearities can be found in [174], and a unified analysis is available in [175].

The LMF algorithm can be understood intuitively as a variable-step version of LMS:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + (\mu e^2(n)) e(n) \phi(n).$$

When the error is large, the “equivalent” step-size  $\mu e^2(n)$  is large, and the filter converges quickly. On the other hand, when the error is small, the “equivalent” step-size is reduced, which consequently reduces the EMSE.

This idea works well if the distributions of  $\phi(n)$  and  $v(n)$  are indeed sub-Gaussian, but it was soon noticed that the algorithm becomes sensitive to outliers (large but rare values of the input signals). This is also easy to understand: if the error is too large, the “equivalent” step-size may be such that the recursion becomes unstable. References [176, 177] proposed changing the cost function to a mixture of quadratic and quartic, in an attempt to reduce this sensitivity. In [178, p. 313], a different modification of the cost function is proposed, with the same purpose. The new cost function is the following

$$J_{\text{LMF, mod}} = E \{ f(e(n)) \}, \quad f(e) = \begin{cases} e^4, & \text{if } |e| \leq \gamma, \\ e^2, & \text{if } |e| > \gamma, \end{cases} \quad (12.317)$$

in which  $\gamma$  is a constant. References [45, 59] show that when the regressor is Gaussian, LMF has a non-zero probability of diverging. From these results one can see that the solution proposed in [178] indeed stabilizes the algorithm.

Since the LMF recursion contains a cubic nonlinearity, it can be used as a simpler platform to understand the behavior of blind-equalization algorithms such as CMA (see Section 1.12.5.5), which also contains a cubic nonlinearity.

*Set-membership* algorithms [1], which are explained in more detail in Section 1.12.5.10, can be seen as an extreme version of (12.317), in which  $f(e)$  is reduced to simply 0 when  $|e| \leq \gamma$ . The corresponding cost function is

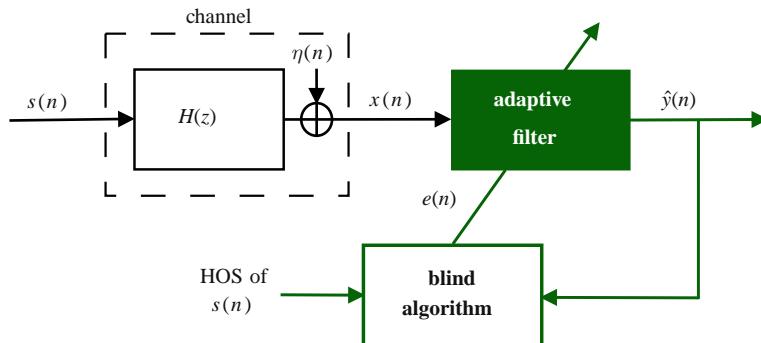
$$J_{\text{SM}} = E \{ f(e(n)) \}, \quad f(e) = \begin{cases} 0, & \text{if } |e| \leq \gamma, \\ e^2, & \text{if } |e| > \gamma. \end{cases} \quad (12.318)$$

Set-membership algorithms are particularly well-suited to bounded noise, i.e., to the case in which one knows that  $|v| \leq \gamma$ .

All these methods are suited to the case in which the noise  $v(n)$  is sub-Gaussian. When the noise is super-Gaussian, there is a small but significant probability that  $v(n)$  will assume a very large value. This may make the performance of even LMS become poor. The solution is to give less weight to large errors, and one choice is to use  $p < 2$  in (12.315). This will give rise to several *robust* adaptive filters, that is, algorithms that are insensitive to outliers. It should be noticed that LMS itself is quite robust in terms of energy (quadratic) relations: in fact, [179] shows that LMS is optimal in the  $\mathcal{H}_\infty$  sense (see also [178, ch. 17]). However, for noise distributions with higher kurtosis, it is convenient to use a smaller value of  $p$  [180–183].

One important algorithm of this class is the *sign-LMS* algorithm [98, 184], which is obtained from (12.315) with  $p = 1$ , and which we already saw in Section 1.12.5.1.1.

For very impulsive signals, methods based on *order statistics* (such as the median, for example), may produce better results, although their complexity tends to be higher [185].

**FIGURE 12.36**

Simplified communications system with a blind adaptive equalizer.

### 1.12.5.5 Blind equalization

As described in Section 1.12.1.3, the objective of equalization is to mitigate the intersymbol interference (ISI) introduced by dispersive channels in order to recover the transmitted sequence. Assuming that the equalizer is an adaptive FIR filter, its coefficients may be updated in two different ways:

- i. using a supervised algorithm in the training and decision-directed modes, as explained in Section 1.12.1.3; or
- ii. using a blind equalization algorithm, which uses higher-order statistics (HOS) of the transmitted signal instead of a training sequence, as shown in Figure 12.36. In this case, the available bandwidth is used in a more efficient manner due to the absence of a training sequence.

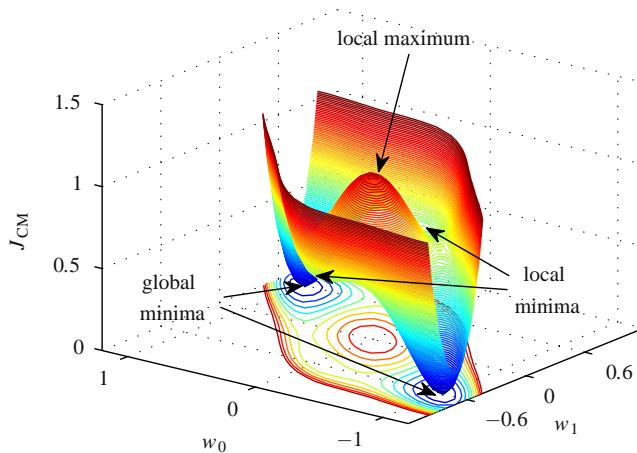
The literature contains many blind equalization algorithms based on HOS, but the most popular is the constant modulus algorithm (CMA), proposed independently by Godard [186] and Treichler and Agee [187] in the 1980s. CMA seeks to minimize the constant-modulus cost function defined as

$$J_{CM} = E \left\{ (r - |\hat{y}(n)|^2)^2 \right\}, \quad (12.319)$$

where  $r = E\{|s(n)|^4\}/E\{|s(n)|^2\}$  is a dispersion constant, which contains information about higher-order statistics of the transmitted signal  $s(n)$  and  $\hat{y}(n) = \mathbf{w}^H \mathbf{x}(n)$  is the equalizer output. This function penalizes deviations in the modulus of the equalized signal away from the dispersion constant  $r$ . Different from the mean-square-error cost function used in supervised adaptive filtering,  $J_{CM}$  is not convex in relation to the coefficients of the equalizer (see the definition of convex functions in Box 5). It has local minima, and constant-modulus-based algorithms can get stuck at these suboptimal solutions.

CMA is obtained from an instantaneous approximation for the gradient of  $J_{CM}$  in relation to  $\mathbf{w}$ . Defining the estimation error

$$e(n) = (r - |\hat{y}(n)|^2)\hat{y}(n), \quad (12.320)$$

**FIGURE 12.37**

Constant-modulus cost function as a function of the equalizer weights.

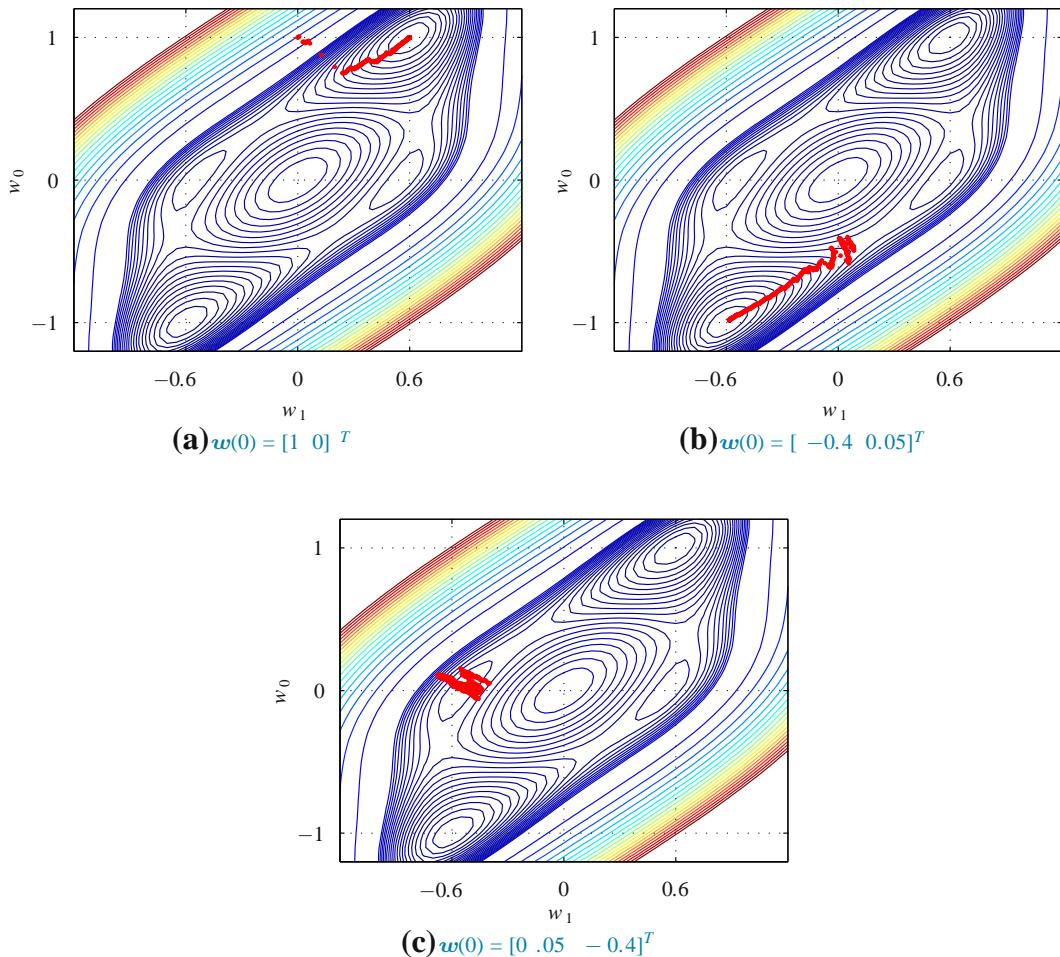
the update equation of CMA can be written as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e^*(n) \mathbf{x}(n), \quad (12.321)$$

where  $\mu$  stands for a step-size. Since CMA is a stochastic-gradient type algorithm, the similarity of (12.321) to the LMS update equation is not surprising. Due to this similarity, CMA is sometimes interpreted as a blind version of the LMS algorithm [188]. Thus, as in LMS, small step-sizes lead to a small misadjustment and slow convergence. However, the similarity to LMS stops here. The multimodality of the CM cost function makes the prediction of the behavior of CMA a hard task.

To illustrate the constant-modulus cost function, we assume the transmission of a binary signal  $\{-1, +1\}$  through the IIR channel  $H(z) = 1/(1 + 0.6z^{-1})$  in the absence of noise [189]. Considering an FIR equalizer with  $M = 2$  coefficients, we can obtain a 3-D plot of  $J_{CM}$  as a function of  $w_0$  and  $w_1$  as shown in Figure 12.37. As indicated in the figure,  $J_{CM}$  contains two global minima at  $\pm[1 \ 0.6]^T$  and two local minima. Note that if  $\mathbf{w}$  converges to one of the global minima, the transmitted sequence or its inverse will be recovered. These two possibilities occur due to the fact that CMA seeks to recover only the modulus of the transmitted signal and does not solve phase ambiguities introduced by the channel. The phase ambiguities can be corrected by using differential code, which is based on the change of sign between two successive samples.

The animations in Figure 12.38 illustrate the behavior of CMA initialized at the different points. Initializing at  $\mathbf{w}(0) = [1 \ 0]^T$ , the weight vector converges to  $[+1 \ +0.6]^T$ . Now, considering  $\mathbf{w}(0) = [-0.4 \ 0.05]^T$ , we get  $\mathbf{w}(n) \approx [-1 \ -0.6]^T$  at the steady-state. On the other hand, if CMA is initialized at  $[0.05 \ -0.4]^T$ , the algorithm gets stuck at a local minimum. We should notice that in the presence of noise or depending on the step-size, CMA also can escape from a minimum and drift to another solution.

**FIGURE 12.38**

CMA initialized at different points,  $\mu = 0.016$ , channel  $H(z) = 1/(1 + 0.6z^{-1})$  in the absence of noise.  
[Click](#) on each caption to see animations on your browser.

Despite the name of the algorithm, CMA also works for non-constant modulus signals. However, as shown analytically in [80, 190], its best performance occurs for constant-modulus signals since the variability in the transmitted signal modulus plays a role similar to the measurement noise in supervised algorithms.

The stochastic analysis of constant-modulus based algorithms is a hard task. Due to the nonlinearity and multimodality of the CM cost function, additional assumptions are necessary. A good review of the main results in the analysis of CMA can be found in [191], but many results were obtained after its

publication. These results address different issues of constant-modulus based algorithms as the steady-state and transient behavior [77, 80, 190, 192–196], convergence and stability [58, 60, 197–200], phase rotation [201–203], solutions for non-constant modulus signals [204–207], among others.

### 1.12.5.6 Subband and transform-domain adaptive filters

In transform-domain adaptive filters, the input signals are transformed (using the FFT, the DCT or other convenient transform), and the adaptation is performed in the transform domain. The gains are twofold: first, using block processing, one can obtain large reductions in computational cost. Second, by using the decorrelation properties of the transforms, it is possible to use different step-sizes for each tap in the transformed domain, thereby reducing the equivalent eigenvalue spread and increasing the convergence rate [208–216].

A related approach is that of subband adaptive filters, which are also useful to both reduce the computational cost and increase the convergence speed of long filters, in particular for acoustic echo. The main idea is to split the input signal in several parts using an *analysis filter bank*. Each part, relative to a particular band of the full spectrum, is processed separately, and finally all parts are joined together again by a *synthesis filter bank* [217]. Since each subsignal has a smaller bandwidth, the sampling rate can be reduced. The whole process must be made carefully to avoid as much as possible a deterioration of the signals due to aliasing, while still gaining in computational cost, in comparison with standard full-band filters [218–224]. It is important to point out that it is possible to avoid the introduction of extra delay (compared with a full-band filter) [225–228].

Good introductions to the subject can be found in [1, 3, 150].

### 1.12.5.7 Affine projections algorithm

The *affine projections* algorithm (APA) is an extension of NLMS, designed to obtain a convergence rate closer to that of RLS, but with a low computational cost, as close as possible to that of NLMS. We saw in Section 1.12.3.2 that NLMS can be derived as a method for choosing the step-size so that the current a posteriori estimation error is zeroed (12.130). The APA extends this idea, by canceling a vector of a posteriori errors, obtained from data from times  $n - K$  to  $n$ , and the weight vector at time  $n + 1$ :

$$\xi(n) = \mathbf{d}(n) - \Phi(n)\mathbf{w}^*(n+1), \quad (12.322)$$

in which

$$\mathbf{d}(n) = \begin{bmatrix} d(n) \\ d(n-1) \\ \vdots \\ d(n-K-1) \end{bmatrix}, \quad \Phi(n) = \begin{bmatrix} \phi^T(n) \\ \phi^T(n-1) \\ \vdots \\ \phi^T(n-K-1) \end{bmatrix}. \quad (12.323)$$

The affine projections algorithm chooses as next weight estimate the vector  $\mathbf{w}(n+1)$  closest to the current estimate  $\mathbf{w}(n)$  such that  $\xi(n) = \mathbf{0}$  [229–234]:

$$\mathbf{w}(n+1) = \arg \min_{\substack{\mathbf{w} \\ \text{s.t. } \xi(n)=\mathbf{0}}} |\mathbf{w} - \mathbf{w}(n)|^2. \quad (12.324)$$

The solution to this problem is a recursion

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Phi^T(n) \left( \Phi^*(n) \Phi^T(n) \right)^{-1} \mathbf{e}(n), \quad (12.325)$$

where

$$\mathbf{e}(n) = \mathbf{d}(n) - \Phi(n) \mathbf{w}^*(n). \quad (12.326)$$

Note that (12.325) reduces to NLMS when  $K = 1$ . Similarly to what is done for NLMS, it is convenient to introduce a regularization term  $\varepsilon \mathbf{I}$ , with  $\varepsilon > 0$  and a step-size  $0 < \mu \leq 1$  in (12.325), leading to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \Phi^T(n) \left( \varepsilon \mathbf{I} + \Phi^*(n) \Phi^T(n) \right)^{-1} \mathbf{e}(n). \quad (12.327)$$

The advantage of this algorithm is that  $\Phi^*(n) \Phi^T(n)$  is  $K \times K$  (as opposed to the  $M \times M$  matrix that must be recursively inverted for RLS), and the value of  $K$  does not need to be large, which keeps the computational cost low. In general, a value of two or three already gives a good improvement to the convergence rate even for regressors whose covariance matrix has large spreading factor.

In order to implement (12.327), it is useful to remember that computing the inverse explicitly is never a good idea: it is much better to solve the linear system

$$\left( \varepsilon \mathbf{I} + \Phi^*(n) \Phi^T(n) \right) \mathbf{a} = \mathbf{e}(n), \quad (12.328)$$

and then compute  $\Phi^T(n) \mathbf{a}$ . From a numerical point of view, the best option would be to use the QR decomposition of  $\Phi(n)$  and avoid forming  $\Phi^*(n) \Phi^T(n)$ , but the computational cost would be larger. When the regressor is a tapped-delay line and  $K$  is large, one can use a fast version of APA, in which the solution of (12.328) is obtained in  $\mathcal{O}(K)$  operations, instead of the usual  $\mathcal{O}(K^3)$  [235–237].

Recent works focusing in the analysis of the APA are [238–241]. The problem of regularization (choice of  $\varepsilon$ ) is considered in [242, 243]. Applications to echo cancellation can be found in [244, 245].

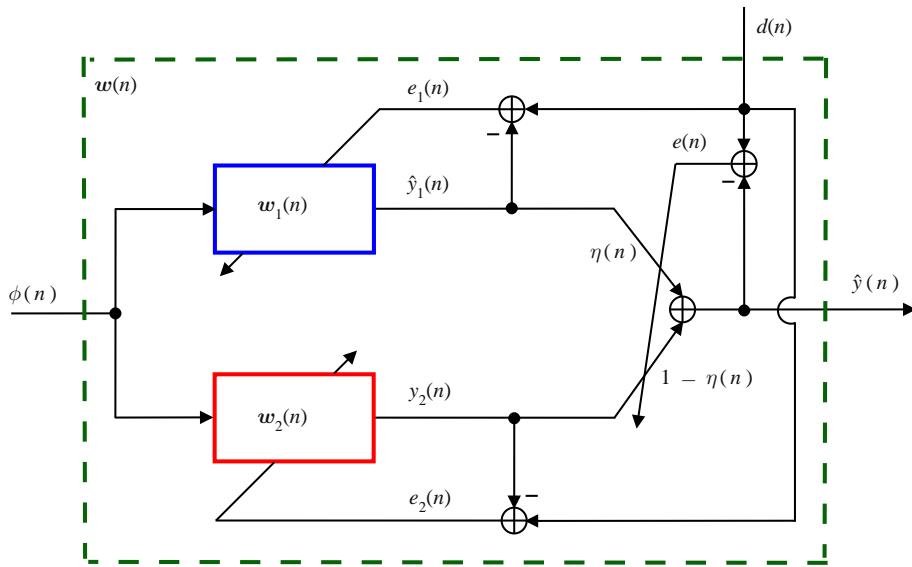
A related approach is the sliding-window RLS algorithm, a version of RLS in which only a finite memory is considered, instead of a forgetting factor as used in Section 1.12.3.3 [246, 247].

### 1.12.5.8 Cooperative estimation

In this section, we briefly describe important advances in adaptive filtering through cooperative estimation. We first describe combinations of adaptive filters and in the sequel, we focus on distributed adaptive filtering.

#### 1.12.5.8.1 Combinations of adaptive filters

Combinations of adaptive filters have received considerable attention lately, since they decrease the sensitivity of the filter to choices of parameters such as the step-size, forgetting factor or filter length. The idea is to combine the outputs of two (or several) different independently-run adaptive algorithms to achieve better performance than that of a single filter. In general, this approach is more robust than variable parameter schemes [190].

**FIGURE 12.39**

Convex combination of two transversal adaptive filters.

The first combined scheme that attracted attention was the convex combination of adaptive filters due to its relative simplicity and the proof that the optimum combination is universal, i.e., the combined estimate is at least as good as the best of the component filters in steady-state, for stationary inputs [248]. This scheme was proposed in [249] and further extended and analyzed in [248, 250]. The original idea was to combine one fast and one slow LMS filter to obtain an overall filter with fast convergence and low misadjustment, but it was extended to other algorithms to take advantages of different characteristics, as explained below.

Figure 12.39 shows the convex combination of two adaptive filters, in which the output of the overall filter is computed as

$$\hat{y}(n) = \eta(n)\hat{y}_1(n) + [1 - \eta(n)]\hat{y}_2(n), \quad (12.329)$$

where  $\eta(n) \in [0, 1]$  is the mixing parameter,  $\hat{y}_i(n) = \mathbf{w}_i^H(n)\phi(n)$ ,  $i = 1, 2$ , are the outputs of the transversal filters,  $\phi(n)$  is the common regressor vector, and  $\mathbf{w}_i(n)$  are the weight vectors of the component filters. Note that the weight vector and the estimation error of the overall filter are given respectively by

$$\mathbf{w}(n) = \eta(n)\mathbf{w}_1(n) + [1 - \eta(n)]\mathbf{w}_2(n) \quad (12.330)$$

and

$$e(n) = d(n) - \hat{y}(n) = \eta(n)e_1(n) + [1 - \eta(n)]e_2(n), \quad (12.331)$$

where  $e_i(n) = d(n) - \hat{y}_i(n)$  are the estimation errors of each component filter.

In order to restrict the mixing parameter to the interval  $[0, 1]$  and to reduce gradient noise when  $\eta(n) \approx 0$  or  $\eta(n) \approx 1$ , a nonlinear transformation and an auxiliary variable  $a(n)$  are used, i.e.,

$$\eta(n) = \text{sgm}[a(n)] \triangleq \frac{1}{1 + e^{-a(n)}}, \quad (12.332)$$

where  $a(n)$  is updated to minimize the overall squared error  $e^2(n)$ , i.e.,

$$a(n+1) = a(n) - \frac{\mu_a}{2} \nabla_a e^2(n) = a(n) + \mu_a [\hat{y}_1(n) - \hat{y}_2(n)] e(n) \eta(n) [1 - \eta(n)]. \quad (12.333)$$

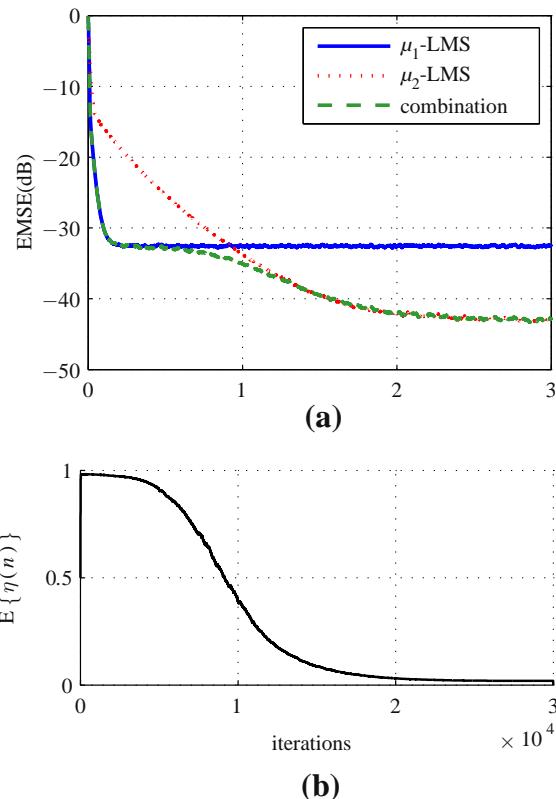
In practice,  $a(n)$  must be restricted by saturation of (12.333) to an interval  $[-a_+, a_+]$ , since the factor  $\eta(n)[1 - \eta(n)]$  would virtually stop adaptation if  $a(n)$  were allowed to grow too much. The correct adjustment of the step-size  $\mu_a$  depends on the input signal and additive noise powers and also on the step-sizes of the component filters. To make the choice of  $\mu_a$  independent of the filtering scenario, a normalized scheme was proposed in [251].

The combination of one fast and one slow LMS (with  $\mu_1 > \mu_2$ ) operates in a very intuitive way. When fast changes appear, the fast filter outperforms the slow one, making  $\eta(n) \approx 1$ . In stationary situations, the slow filter gets a lower quadratic error and  $\eta(n) \approx 0$ , which allows the overall filter to achieve the misadjustment of the slow LMS filter. There are situations in which the combination outperforms each component filter and in these cases,  $0 < \eta(n) < 1$ . This behavior can be observed in the simulation results shown in Figure 12.40, where the convex combination of two LMS filters with different step-sizes ( $\mu_1 = 0.1$  and  $\mu_2 = 0.01$ ) was used to identify the system (from [248])

$$[0.9003 \ -0.5377 \ 0.2137 \ -0.0280 \ 0.7826 \ 0.5242 \ -0.0871].$$

The regressor  $\phi(n)$  is obtained from a process  $x(n)$  generated with a first-order autoregressive model, whose transfer function is  $\sqrt{1 - b^2}/(1 - bz^{-1})$ . This model is fed with an iid Gaussian random process, whose variance is such that  $\text{Tr}(\mathbf{R}_\phi) = 1$ . Moreover, additive iid noise  $v(n)$  with variance  $\sigma_v^2 = 0.01$  is added to form the desired signal. Figure 12.40a shows the EMSE curves estimated from the ensemble-average of 500 independent runs and filtered by a moving-average filter with 128 coefficients to facilitate the visualization. The combined filter acquires the faster convergence of  $\mu_1$ -LMS and attains the EMSE of  $\mu_2$ -LMS. Figure 12.40b shows the time evolution of the mixing parameter. We can observe that it rapidly changes toward  $\text{sgm}[a_+]$  during the initial convergence, while its steady-state value is  $1 - \text{sgm}[a_+]$ .

After the publication of [249], many papers on combinations of adaptive filters appeared in the literature. Apparently, the idea of combining the outputs of several different independently-run adaptive algorithms was first proposed in [252], and later improved in [253, 254]. The algorithms proposed in [252–254] are based on a Bayesian argument, and construct an overall (combined) filter through a linear combination of the outputs of several independent adaptive filters. The weights are the *a posteriori* probabilities that the underlying models used to describe each individual algorithm are “true.” Since the weights add up to one, in a sense these first papers also proposed “convex” combinations of algorithms. Unconstrained linear combinations of adaptive filters were also proposed in [255]. However, the method of [248, 249] has received more attention due to its relative simplicity and the proof that the optimum combination is universal. By this, we mean that if the combination uses at every instant the optimum

**FIGURE 12.40**

(a) EMSE for  $\mu_1$ -LMS,  $\mu_2$ -LMS, and their convex combination; (b) ensemble-average of  $\eta(n)$ ;  $\mu_1 = 0.1$ ,  $\mu_2 = 0.01$ ,  $\mu_a = 100$ ,  $a^+ = 4$ ,  $b = 0.8$ ; mean of 500 independent runs.

value of  $\eta$ , then the combined filter is always at least as good as the best component filter. However, in practice  $\eta$  must also be estimated, so the performance of the combination may be slightly worse than that of the best filter.

In the sequel, we briefly describe some of the most important recent contributions in this area.

In [250], the convergence of the overall filter is greatly improved by transferring a part of the fast filter  $\mathbf{w}_1$  to the slow filter  $\mathbf{w}_2$  when  $\lambda(n) \geq \gamma$ , that is,

$$\mathbf{w}_2(n+1) = \alpha [\mathbf{w}_2(n) + \mu_2 e_2(n) \mathbf{x}(n)] + (1 - \alpha) \mathbf{w}_1(n), \quad (12.334)$$

where  $\alpha$  is a parameter close to 1 and  $\gamma$  is a threshold close to the maximum value that can be reached by  $\lambda(n)$ .

A steady-state analysis of the combined filter was presented in [248]. This analysis shows that the convex combination is universal if the optimum  $\eta$  is used, a property that was exploited to design filters

with improved tracking performance. Due to the nonlinear function (12.332), a transient analysis of the convex combination needs additional simplifications. This issue was addressed in [256] and [257], which provide models for the transient of the convex combination based on first and second-order Taylor series approximations.

The convex combination was used in [258] to improve the performance of a variable tap-length LMS algorithm in a low signal-to-noise environment ( $\text{SNR} \leq 0$  dB). The adaptation of the tap-length in the variable tap-length LMS algorithm is highly affected by the parameter choice and the noise level. Combination approaches improve such adaptation by exploiting advantages of parallel adaptive filters with different parameters.

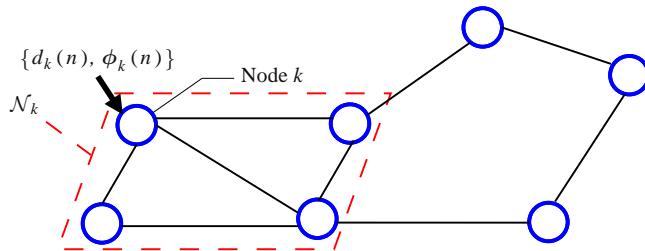
The convex combination was exploited in [190] to improve the tracking performance of adaptive filters by combining filters with different tracking capabilities, as are the cases of LMS and RLS (see Section 1.12.4.4). The combination of algorithms of different families was also addressed in [259], where it was shown that a combination of two filters from the same family (i.e., two LMS or two RLS filters) cannot improve the performance over that of a single filter of the same type with optimal selection of the step-size (or forgetting factor). However, combining LMS and RLS filters, it is possible to simultaneously outperform the optimum LMS and RLS filters. In other words, combination schemes can achieve smaller errors than optimally adjusted individual filters.

Convex combinations were used for sparse echo cancellation in [260], considering the combination of two improved proportionate normalized least-mean-square (IPNLMS) filters. The combined scheme increases the IPNLMS robustness to channels with different degrees of sparsity and also alleviates the tradeoff between rate of convergence versus steady-state misadjustment imposed by the selection of the step-size. Still considering the echo cancellation application, [261] proposes a combination of adaptive Volterra kernels that presents a similar behavior to that of the complete Volterra filters, but with a reduction in the computational load. This scheme is robust regardless of the level of nonlinear distortion, which is a desired property for nonlinear echo cancellation.

In [262], the convex combination was also used as a scheme for adaptively biasing the weights of adaptive filters using an output multiplicative factor. Interpreting the biased estimator as the combination of the original filter and a filter with constant output equal to zero, practical schemes to adaptively adjust the multiplicative factor were proposed. This scheme provides a convenient bias versus variance tradeoff, leading to reductions in the filter mean-square error, especially in situations with a low signal-to-noise ratio.

Extending [248], the authors of [263] proposed an affine combination of two LMS algorithms, where the condition on the mixing parameter is relaxed, allowing it to be negative. Thus, this scheme can be interpreted as a generalization of the convex combination since the mixing parameter is not restricted to the interval  $[0, 1]$ . This approach allows for smaller EMSE in theory, but suffers from larger gradient noise in some situations. Under certain conditions, the optimum mixing parameter was proved to be negative in steady-state. Although the optimal linear combiner is unrealizable, two realizable algorithms were introduced. One is based on a stochastic gradient search and the other is based on the ratio of the average error powers from each individual adaptive filter. Under some circumstances, both algorithms present performance close to the optimum.

Similarly to the convex combination, the correct adjustment of the step-size for the updating of the mixing parameter in the affine combination (using the stochastic algorithm of [263]) depends on characteristics of the filtering scenario. This issue was addressed in [193], where a transient analysis for

**FIGURE 12.41**

Diffusion network with  $N = 7$  nodes: at time  $n$ , every node  $k$  takes a measurement  $\{d_k(n), \phi_k(n)\}$ .

the affine combination of two adaptive filters was presented. The results of this analysis were used to facilitate the adjustment of the free parameters of the scheme and to propose two normalized algorithms to update the mixing parameter. Recently, the scheme based on the ratio of error powers of the two filters proposed in [263] to update the mixing parameter was analyzed in [264].

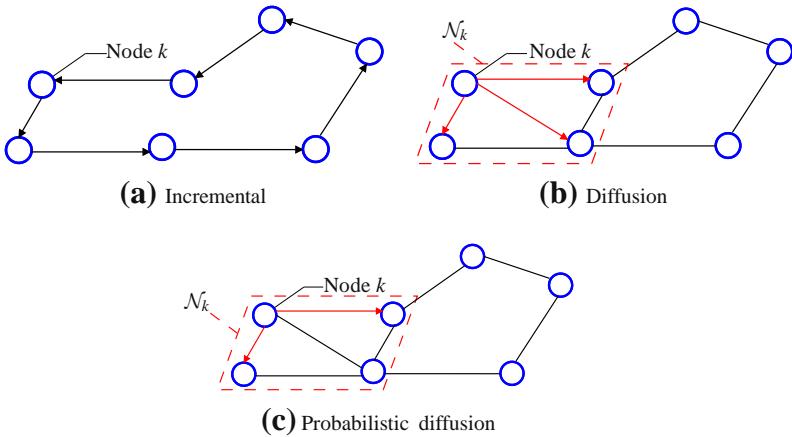
Finally, Kozat et al. [265] studied different mixing strategies in which the overall output is formed as the weighted linear combination (not necessarily constrained to convex or affine, as before) of the outputs of several component algorithms for stationary and certain nonstationary data.

### 1.12.5.8.2 Distributed adaptive filtering

Adaptive networks use the information from data collected at nodes distributed over a certain region. Each node collects noisy observations related to a parameter of interest and interacts with its neighbors considering a certain network topology. The objective is to obtain an estimate of the parameter of interest as accurate as the one that would be obtained if each node had access to the information across the entire network [266, 267]. Distributed estimation algorithms are useful in several contexts, including wireless and sensor networks, where scalability, robustness, and low power consumption are desirable. They also find applications in precision agriculture and environmental monitoring and transportation [268].

Figure 12.41 shows a network composed by  $N$  nodes distributed over some geographical area. At time instant  $n$ , every node  $k$  takes a measurement  $\{d_k(n), \phi_k(n)\}$  to estimate some parameter vector  $\mathbf{w}_o$ . The scalar measurement  $d_k(n)$  represents the desired signal and  $\phi_k(n)$  denotes the length- $M$  input regressor vector for node  $k$ . There are different solutions to the problem of estimating  $\mathbf{w}_o$ . In the *centralized* solution, every node transmits its data  $\{d_k(n), \phi_k(n)\}$  to a fusion center for processing. This solution is non-robust to failure in the fusion center. In *distributed* solutions, every node exchanges information with a subset of its neighboring nodes, and the processing is distributed among all nodes in the network. The set of nodes connected to node  $k$  (including  $k$  itself) is denoted by  $\mathcal{N}_k$  and is called the neighborhood of node  $k$ .

There are three main network topologies used in distributed estimation: incremental, diffusion, and probabilistic diffusion [267]. In incremental networks, the nodes are connected so that information flows sequentially from node to node, in a cyclic manner (Figure 12.42a). In diffusion networks, each node communicates with its entire neighborhood at each instant (Figure 12.42b). Finally, in probabilistic



**FIGURE 12.42**

## Different network topologies.

diffusion networks each node communicates with a (possibly random) subset of its neighbors at each instant (Figure 12.42c).

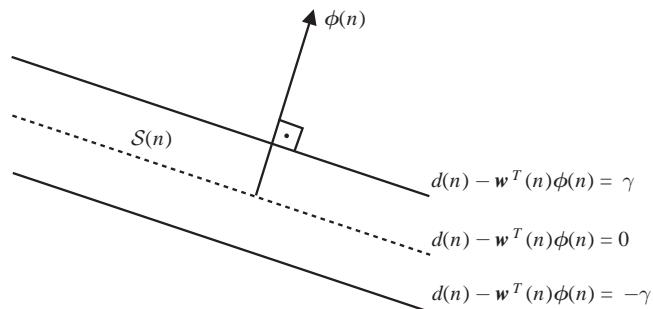
Several estimation algorithms have been proposed in the context of distributed adaptive filtering, such as incremental LMS, incremental RLS, diffusion LMS, diffusion RLS, diffusion Kalman filtering, and smoothing algorithms. In these algorithms, the nodes share information with their neighbors, and perform local adaptation and merging of information using convex combinations of available estimates. Distributed algorithms only require local communications between neighboring nodes, and can attain good estimation performance compared to centralized solutions. This is a current area of intense research [268–280].

Recently, adaptive distributed algorithms have also been used to model biological systems due to their self-organization property. Examples include fish joining together in schools, bacterial motility, and bird flight formations [281–286].

### 1.12.5.9 Adaptive IIR filters

Adaptive infinite impulse response (IIR) filters represent an advantageous alternative in relation to adaptive FIR filters, due to their capacity of providing long impulse responses with a small number of coefficients. Their use may be desirable in applications requiring hundreds or thousands of taps, such as satellite-channel and mobile-radio equalizers. The advantages of adaptive IIR filters have a price: slow convergence, possible instability, error surface with local minima or biased global minimum [1,287–290]. In addition, not all long impulse responses will necessarily be well approximated by an IIR filter with low order.

The nonlinear relation between the adaptive-filter coefficients and the internal signals makes the gradient computation and convergence analysis more complicated than those of adaptive FIR filters [1]. Over the years, problems related to local minima, stability, and the effect of poles close to the unit

**FIGURE 12.43**

Constraint imposed by data at time  $n$ .

circle have been addressed by several authors leading to different adaptive algorithms and realization structures [291–300].

Another difficulty with adaptive IIR filters is that the performance of simple constant-gain algorithms may rapidly degrade as the order of the filter grows. Even in the absence of local minima, the adaptation of filters with order greater than two can remain almost stopped in regions where the mean square error is relatively high. This issue was addressed in [287], which was able to obtain a large acceleration in the convergence rate of IIR filters.

### 1.12.5.10 Set-membership and projections on convex sets

When the desired signal and the regressor are related as

$$d(n) = \mathbf{w}_o^H \boldsymbol{\phi}(n) + v(n),$$

where  $v(n)$  is *bounded*, that is,  $|v(n)| \leq \gamma$  for all  $n$ , it is natural to consider that each pair  $(d(n), \boldsymbol{\phi}(n))$  defines a region in which the true solution  $\mathbf{w}_*$  must lie:

$$\mathbf{w}_o \in \mathcal{S}(n) = \left\{ \mathbf{w} \in \mathcal{C}^M : |d(n) - \mathbf{w}^H \boldsymbol{\phi}(n)| \leq \gamma \right\}. \quad (12.335)$$

For example, if all vectors are two-dimensional and real, the region  $\mathcal{S}(n)$  would be as in Figure 12.43. *Set-membership* algorithms aim to find the set of possible solutions, by tracking the intersection

$$\mathcal{T}(n) = \bigcap_{k=0}^n \mathcal{S}(k) \quad (12.336)$$

of all  $\mathcal{S}(n)$  [301]. In the control literature and earlier algorithms, this is made by approximating  $\mathcal{T}(n)$  by ellipsoids [301–303]. However, very good performance is obtained if one uses just any point contained in  $\mathcal{T}(n)$  as a point estimate of  $\mathbf{w}_o$  at each instant. In fact, it is more interesting to restrict the memory,

so that the filter can track variations of  $\mathbf{w}_o$ . In this case, only the last  $K$  constraint sets  $\mathcal{S}(n)$  are used:

$$\mathcal{T}_K(n) = \bigcap_{k=n-K+1}^n \mathcal{S}(k). \quad (12.337)$$

This approach has a very nice property: close to steady-state, if the weight estimates are already inside  $\mathcal{T}_K(n)$ , it is not necessary to perform any update. This may reduce considerably the average complexity of the filter [304–310].

More recently, the approach has been extended to a more general setting, using projections onto convex sets and subgradients. This allows the extension of the results to non-differentiable cost functions, with different descriptions for the constraints (not restricted to (12.335)), also allowing the design of kernel adaptive filters [6,311–313] (an introduction to kernel adaptive filters can be found in [314]).

### 1.12.5.11 Adaptive filters with constraints

In some applications, one wishes to minimize a cost function given certain constraints: a typical case is beamforming. In the simplest example, an array of antennas or microphones receives a signal coming from a certain (known) direction, and interference from other (unknown) directions. The output of the array is a linear combination of the signals  $\phi_i(n)$  received at each antenna, e.g.,

$$\hat{y}(n) = w_0^* \phi_0(n) + w_1^* \phi_1(n) + \cdots + w_{M-1}^* \phi_{M-1}(n).$$

One way of improving the signal-to-noise ratio at the output is to minimize the output power, under the restriction that the gain of the array in the direction of the desired signal is one. This is described by a constraint of the kind

$$\mathbf{c}^H \mathbf{w} = 1,$$

where  $\mathbf{w} = [w_0 \dots w_{M-1}]$  is the weight vector, and  $\mathbf{c} \in \mathcal{C}^M$  is a vector describing the constraint. The goal of the filter would then be to find the optimum  $\mathbf{w}_o$  that is the solution of

$$\mathbf{w}_o = \arg \min_{\substack{\mathbf{w} \\ \text{s.t. } \mathbf{c}^H \mathbf{w} = 1}} |\hat{y}(n)|^2.$$

This solution is known as *minimum-variance distortionless response* (MVDR), or Capon beamformer, since it was originally proposed in [315]. There is an extensive literature on this subject [312,316–325]. Introductions to the topic can be found in [326–328].

### 1.12.5.12 Reduced-rank adaptive filters

In applications in which large filters are necessary, such as acoustic echo cancellation and nonlinear models using Volterra expansions, it may be useful to reduce the order of the problem, by working in a smaller subspace of the original problem. This reduces the number of parameters to be estimated, increasing the convergence speed and reducing the excess mean-square error [329]. Several methods have been proposed, such as [330,331], based on eigen-decompositions, the multi-stage Wiener filter of [332–334], the auxiliary vector filtering algorithm [335], and the joint iterative optimization method of [336–339]. This last method has the advantage of having low computational cost.

---

## Box 8: [Fixed-point arithmetic]

---

In fixed-point arithmetic, all variables are restricted to lie in a fixed interval  $[-a, a)$ , and are stored as a sequence of  $B$  bits  $b_0 b_1 \dots b_{B-1}$ . Assuming that two's complement arithmetic is used and that  $a = 1$ , then  $b_0$  represents the signal of the variable and the sequence of bits represents the number

$$-b_0 + \sum_{m=1}^{B-1} b_m 2^{-m}.$$


---

---

## Acknowledgment

The authors thank Prof. Wallace A. Martins (Federal University of Rio de Janeiro) for his careful reading and comments on the text.

*Relevant Theory:* Signal Processing Theory and Machine Learning

See this Volume, [Chapter 3](#) Discrete-Time Signals and Systems

See this Volume, [Chapter 4](#) Random Signals and Stochastic Processes

See this Volume, [Chapter 6](#) Digital Filter Structures and Their Implementation

See this Volume, [Chapter 11](#) Parametric Estimation

See this Volume, [Chapter 25](#) A Tutorial on Model Selection

---

## References

- [1] P.S.R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation, third ed., Springer, 2008.
- [2] S. Haykin, Adaptive Filter Theory, fourth ed., Prentice Hall, 2001.
- [3] A.H. Sayed, Adaptive Filters, Wiley-IEEE Press, 2008.
- [4] B. Widrow, S.D. Stearns, Adaptive Signal Processing, Prentice Hall, Englewood Cliffs, 1985.
- [5] M.D. Miranda, M. Gerken, M.T.M. Silva, Efficient implementation of error-feedback LSL algorithm, Electron. Lett. 35 (16) (1999) 1308–1309.
- [6] S. Theodoridis, K. Slavakis, I. Yamada, Adaptive learning in a world of projections, IEEE Signal Process. Mag. 28 (1) (2011) 97–123.
- [7] B. Widrow, M. Hoff, Adaptive switching circuits, in: IRE WESCON Conv. Rec., pt. 4, 1960, pp. 96–104.
- [8] B. Widrow, Thinking about thinking: the discovery of the LMS algorithm, IEEE Signal Process. Mag. 22 (1) (2005) 100–106.
- [9] A.V. Oppenheim, A.S. Willsky, Signals and Systems, Prentice-Hall, 1996.
- [10] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, 1987.
- [11] R.A. Horn, C.R. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge, MA, 1991.
- [12] A.J. Laub, Matrix Analysis for Scientists and Engineers, Society for Industrial and Applied Mathematics, PA, December 2005.
- [13] C.D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia, USA, 2000.
- [14] G.H. Golub, C.F.V. Loan, Matrix Computations, third ed., Johns Hopkins, 1996.

- [15] K.J. Åström, B. Wittenmark, *Adaptive Control*, Addison-Wesley, 1995.
- [16] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*, third ed., Prentice Hall, January 2008.
- [17] T. Kailath, A.H. Sayed, B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, 2000.
- [18] P. Ioannou, B. Fidan, *Adaptive Control Tutorial*, SIAM, Society for Industrial and Applied Mathematics, December 2006.
- [19] T. Gansler, J. Benesty, New insights into the stereophonic acoustic echo cancellation problem and an adaptive nonlinearity solution, *IEEE Trans. Speech Audio Process.* 10 (5) (2002) 257–267.
- [20] H.I.K. Rao, B. Farhang-Boroujeny, Fast LMS/Newton algorithms for stereophonic acoustic echo cancelation, *IEEE Trans. Signal Process.* 57 (8) (2009) 2919–2930.
- [21] C.H. Hansen, S.D. Snyder, *Active Control of Noise and Vibration*, Taylor & Francis, 1997.
- [22] S.M. Kuo, D.R. Morgan, *Active Noise Control Systems – Algorithms and DSP Implementations*, Wiley Series in Telecommunications and Signal Processing, Wiley-Interscience, 1996.
- [23] P.A. Nelson, S.J. Elliot, *Active Control of Sound*, Academic Press, 1992.
- [24] S.D. Snyder, *Active Noise Control Primer*, Springer-Verlag, 2000.
- [25] F.G. de Almeida Neto, V.H. Nascimento, M.T.M. Silva, Reduced-complexity widely linear adaptive estimation, in: 7th International Symposium on Wireless Communication Systems, September 2010, pp. 399–403.
- [26] B. Picinbono, Wide-sense linear mean square estimation and prediction, in: International Conference on Acoustics, Speech, and Signal Processing, 1995, ICASSP-95, vol. 3, May 1995, pp. 2032–2035.
- [27] B. Picinbono, P. Bondon, Second-order statistics of complex signals, *IEEE Trans. Signal Process.* 45 (2) (1997) 411–420.
- [28] T. Adali, H. Li, R. Aloysius, On properties of the widely linear MSE filter and its LMS implementation, in: 43rd Annual Conference on Information Sciences and Systems, March 2009, pp. 876–881.
- [29] A. Chinatto, C. Junqueira, J.M.T. Romano, Interference mitigation using widely linear arrays, in: 17th European Signal Processing Conference, September 2009, pp. 353–357.
- [30] S.C. Douglas, D.P. Mandic, Performance analysis of the conventional complex LMS and augmented complex LMS algorithms, in: Proc. IEEE Int. Acoustics Speech and Signal Processing (ICASSP) Conf., 2010, pp. 3794–3797.
- [31] D.P. Mandic, S. Still, S.C. Douglas, Duality between widely linear and dual channel adaptive filtering, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, ICASSP 2009, 2009, pp. 1729–1732.
- [32] P.M. Djuric, J.H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M.F. Bugallo, J. Miguez, Particle filtering, *IEEE Signal Process. Mag.* 20 (5) (2003) 19–38.
- [33] W.R. LePage, *Complex Variables and the Laplace Transform for Engineers*, Dover, 1980.
- [34] K. Kreutz-Delgado, The complex gradient operator and the CR-Calculus, 0906.4835, June 2009.
- [35] W.A. Sethares, The least mean square family, in: N. Kalouptsidis, S. Theodoridis (Eds.), *Adaptive System Identification and Signal Processing Algorithms*, Prentice Hall, 1993.
- [36] V. Solo, Averaging analysis of the LMS algorithm, in: C.T. Leondes (Ed.), *Control and Dynamic Systems, Stochastic Techniques in Digital Signal Processing Systems*, Part 2 of 2, vol. 65, Academic Press, 1994, pp. 379–397.
- [37] B.D.O. Anderson, R.R. Bitmead, C.R.J. Johnson Jr., P.V. Kokotovic, R.L. Kosut, I.M.Y. Mareels, L. Praly, B.D. Riedle, Stability of adaptive systems: passivity and averaging analysis, MIT Press, Cambridge, MA, 1986.
- [38] A.H. Sayed, T. Kailath, A state-space approach to adaptive RLS filtering, *IEEE Signal Process. Mag.* 11 (3) (1994) 18–60.
- [39] H.J. Kushner, G.G. Yin, *Stochastic Approximation Algorithms and Applications*, Applications of Mathematics, Springer, 1997.

- [40] P.A. Regalia, Numerical stability properties of a QR-based fast least squares algorithm, *IEEE Trans. Signal Process.* 41 (6) (1993) 2096–2109.
- [41] D.T.M. Slock, The backward consistency concept and round-off error propagation dynamics in recursive least-squares filtering algorithms, *Opt. Eng.* 31 (1992) 1153–1169.
- [42] G.E. Bottomley, S.T. Alexander, A novel approach for stabilizing recursive least squares filters, *IEEE Trans. Signal Process.* 39 (8) (1991) 1770–1779.
- [43] J.A. Apolinário Jr. (Ed.), *QRD-RLS Adaptive Filtering*, Springer, NY, 2009.
- [44] Y. Zakharov, G. White, J. Liu, Low-complexity RLS algorithms using dichotomous coordinate descent iterations, *IEEE Trans. Signal Process.* 56 (7) (2008) 3150–3161.
- [45] V.H. Nascimento, J.C.M. Bermudez, Probability of divergence for the least-mean fourth (LMF) algorithm, *IEEE Trans. Signal Process.* 54 (4) (2006) 1376–1385.
- [46] V.H. Nascimento, A.H. Sayed, On the learning mechanism of adaptive filters, *IEEE Trans. Signal Process.* 48 (6) (2000) 1609–1625.
- [47] S. Chen, Low complexity concurrent constant modulus algorithm and soft directed scheme for blind equalization, *IEE Proc. – Vis. Image Signal Process.* 150 (2003) 312–320.
- [48] H.J. Butterweck, Iterative analysis of the steady-state weight fluctuations in LMS-type adaptive filters, Technical Report EUT 96-E-299, Eindhoven University of Technology, Netherlands, June 1996.
- [49] H.J. Butterweck, A wave theory of long adaptive filters, *IEEE Trans. Circ. Syst. I* 48 (6) (2001) 739–747.
- [50] E. Eweda, O. Macchi, Convergence of an adaptive linear estimation algorithm, *IEEE Trans. Autom. Control* AC-19 (2) (1984) 119–127.
- [51] O. Macchi, E. Eweda, Convergence analysis of self-adaptive equalizers, *IEEE Trans. Inform. Theory* IT-30 (2) (1984) 161–176.
- [52] J.E. Mazo, On the independence theory of equalizer convergence, *Bell Syst. Tech. J.* 58 (1979) 963–993.
- [53] V. Solo, X. Kong, *Adaptive Signal Processing Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1995.
- [54] S.C. Douglas, W. Pan, Exact expectation analysis of the LMS adaptive filter, *IEEE Trans. Signal Process.* 43 (12) (1995) 2863–2871.
- [55] A. Feuer, E. Weinstein, Convergence analysis of LMS filters with uncorrelated Gaussian data, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-33 (1) (1985) 222–229.
- [56] S. Florian, A. Feuer, Performance analysis of the LMS algorithm with a tapped delay line (two-dimensional case), *IEEE Trans. Acoust. Speech Signal Process.* ASSP-34 (6) (1986) 1542–1549.
- [57] A.H. Sayed, V.H. Nascimento, Energy conservation and the learning ability of LMS adaptive filters, in: S. Haykin, B. Widrow (Eds.), *Least-Mean-Square Adaptive Filters*, Wiley, 2003.
- [58] O. Dabeer, E. Masry, Convergence analysis of the constant modulus algorithm, *IEEE Trans. Inform. Theory* 49 (6) (2003) 1447–1464.
- [59] P.I. Hübscher, J.C.M. Bermudez, V.H. Nascimento, A mean-square stability analysis of the least mean fourth (LMF) adaptive algorithm, *IEEE Trans. Signal Process.* 55 (8) (2007) 4018–4028.
- [60] V.H. Nascimento, M.T.M. Silva, Stochastic stability analysis for the constant-modulus algorithm, *IEEE Trans. Signal Process.* 56 (10) (2008) 4984–4989.
- [61] T.Y. Al-Naffouri, M. Moinuddin, Exact performance analysis of the  $\epsilon$ -NLMS algorithm for colored circular Gaussian inputs, *IEEE Trans. Signal Process.* 58 (10) (2010) 5080–5090.
- [62] T.Y. Al-Naffouri, M. Moinuddin, M.S. Sohail, Mean weight behavior of the NLMS algorithm for correlated Gaussian inputs, *IEEE Signal Process. Lett.* 18 (1) (2011) 7–10.
- [63] N.J. Bershad, Analysis of the normalized LMS algorithm with Gaussian inputs, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-34 (1986) 793–806.
- [64] N.J. Bershad, Behavior of the  $\epsilon$ -normalized LMS algorithm with Gaussian inputs, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-35, May 1987, pp. 636–644.

- [65] M.H. Costa, J.C.M. Bermudez, An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients, in: Proc. ICASSP 2002, vol. 2, pp. 1385–1388.
- [66] V.H. Nascimento, A simple model for the effect of normalization on the convergence rate of adaptive filters, in: Proc. of the 2004 IEEE International Conference on Acoustics, Speech and, Signal Processing, 2004, pp. II-453–II-456.
- [67] D.T.M. Slock, On the convergence behavior of the LMS and the normalized LMS algorithms, *IEEE Trans. Signal Process.* 41 (9) (1993) 2811–2825.
- [68] M. Tarrab, A. Feuer, Convergence and performance analysis of the normalized LMS algorithm with uncorrelated Gaussian data, *IEEE Trans. Inform. Theory* 34 (4) (1988) 680–691.
- [69] A. Carini, E. Mumolo, G.L. Sicuranza, V-vector algebra and its application to Volterra-adaptive filtering, *IEEE Trans. Circ. Syst. II* 46 (5) (1999) 585–598.
- [70] F. Kuech, W. Kellermann, Orthogonalized power filters for nonlinear acoustic echo cancellation, *Signal Process.* 86 (6) (2006) 1168–1181.
- [71] V.J. Mathews, G.L. Sicuranza, *Polynomial Signal Processing*, Wiley-Interscience, New York, 2000.
- [72] A. Stenger, W. Kellermann, Adaptation of a memoryless preprocessor for nonlinear acoustic echo cancelling, *Signal Process.* 80 (9) (2000) 1747–1760.
- [73] G.L. Sicuranza, A. Carini, A. Fermo, Nonlinear adaptive filters for acoustic echo cancellation in mobile terminals, in: K.E. Barner, G.R. Arce (Eds.), *Nonlinear Signal and Image Processing: Theory, Methods, and Applications*, The Electrical Engineering and Applied Signal Processing Series, CRC Press, 2003, pp. 223–255 (Chapter 7).
- [74] E. Eleftheriou, D. Falconer, Tracking properties and steady-state performance of RLS adaptive filter algorithms, *IEEE Trans. Acoust. Speech Signal Process.* 34 (5) (1986) 1097–1110.
- [75] G.V. Moustakides, Study of the transient phase of the forgetting factor RLS, *IEEE Trans. Signal Process.* 45 (10) (1997) 2468–2476.
- [76] A.H. Sayed, M. Rupp, Error-energy bounds for adaptive gradient algorithms, *IEEE Trans. Signal Process.* 44 (8) (1996) 1982–1989.
- [77] N. Yousef, A.H. Sayed, A unified approach to the steady-state and tracking analysis of adaptive filters, *IEEE Trans. Signal Process.* 49 (2) (2001) 314–324.
- [78] M. Rupp, A.H. Sayed, A time-domain feedback analysis of filtered-error adaptive gradient algorithms, *IEEE Trans. Signal Process.* 44 (1996) 1428–1439.
- [79] C. Samsom, V.U. Reddy, Fixed point error analysis of the normalized ladder algorithm, *IEEE Trans. Acoust. Speech, Signal Process.* 31 (5) (1983) 1177–1191.
- [80] J.Y. Mai, A.H. Sayed, A feedback approach to the steady-state performance of fractionally spaced blind adaptive equalizers, *IEEE Trans. Signal Process.* 48 (1) (2000) 80–91.
- [81] E. Eweda, Comparison of RLS, LMS, and sign algorithms for tracking randomly time-varying channels, *IEEE Trans. Signal Process.* 42 (11) (1994) 2937–2944.
- [82] J.H. Wilkinson, *Rounding Errors in Algebraic Processes*, Her Majesty's Stationery Office, 1963.
- [83] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, UK, 1965.
- [84] S.T. Alexander, Transient weight misadjustment properties for the finite precision LMS algorithm, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-35 (9) (1987) 1250–1258.
- [85] S. Ardalan, Floating-point error analysis of recursive least-squares and least-mean-squares adaptive filters, *IEEE Trans. Circ. Syst.* 33 (12) (1986) 1192–1208.
- [86] C. Caraiscos, B. Liu, A roundoff error analysis of the LMS adaptive algorithm, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-32 (1) (1984) 34–41.
- [87] R.D. Gitlin, J.E. Mazo, M.G. Taylor, On the design of gradient algorithms for digitally implemented adaptive filters, *IEEE Trans. Circ. Theory*, CT-20 (2) (1973) 125–136.

- [88] J.C.M. Bermudez, N.J. Bershad, A nonlinear analytical model for the quantized LMS algorithm the arbitrary step size case, *IEEE Trans. Signal Process.* 44 (5) (1996) 1175–1183.
- [89] J.C.M. Bermudez, N.J. Bershad, Transient and tracking performance analysis of the quantized LMS algorithm for time-varying system identification, *IEEE Trans. Signal Process.* 44 (8) (1996) 1990–1997.
- [90] N.J. Bershad, J.C. Bermudez, New insights on the transient and steady-state behavior of the quantized LMS algorithm, *IEEE Trans. Signal Process.* 44 (10) (1996) 2623–2625.
- [91] N.J. Bershad, J.C.M. Bermudez, A nonlinear analytical model for the quantized LMS algorithm-the power-of-two step size case, *IEEE Trans. Signal Process.* 44 (11) (1996) 2895–2900.
- [92] J.M. Cioffi, J.J. Werner, The tap-drifting problem in digitally implemented data-driven echo cancellers, *Bell Syst. Tech. J.* 64 (1) (1985) 115–138.
- [93] R.D. Gitlin, H.C. Meadors Jr., S.B. Weinstein, The tap-leakage algorithm: an algorithm for the stable operation of a digitally implemented, fractionally spaced adaptive equalizer, *Bell Syst. Tech. J.* 61 (8) (1982) 1817–1839.
- [94] V.H. Nascimento, A.H. Sayed, Unbiased and stable leakage-based adaptive filters, *IEEE Trans. Signal Process.* 47 (12) (1999) 3261–3276.
- [95] E. Eweda, Almost sure convergence of a decreasing gain sign algorithm for adaptive filtering, *IEEE Trans. Acoust. Speech Signal Process.* 36 (10) (1988) 1669–1671.
- [96] E. Eweda, A tight upper bound of the average absolute error in a constant step-size sign algorithm, *IEEE Trans. Acoust. Speech Signal Process.* 37 (11) (1989) 1774–1776.
- [97] E. Eweda, Convergence analysis of the sign algorithm without the independence and Gaussian assumptions, *IEEE Trans. Signal Process.* 48 (9) (2000) 2535–2544.
- [98] A. Gersho, Adaptive filtering with binary reinforcement, *IEEE Trans. Inform. Theory* 30 (2) (1984) 191–199.
- [99] P. Xue, B. Liu, Adaptive equalizer using finite-bit power-of-two quantizer, *IEEE Trans. Acoust. Speech Signal Process.* 34 (6) (1986) 1603–1611.
- [100] S. Ljung, L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms, *Automatica* 21 (2) (1985) 157–167.
- [101] J. Cioffi, The fast adaptive ROTOR's RLS algorithm, *IEEE Trans. Acoust. Speech Signal Process.* 38 (4) (1990) 631–653.
- [102] S. Alexander, A. Ghimikar, A method for recursive least squares filtering based upon an inverse QR decomposition, *IEEE Trans. Signal Process.* 41 (1) (1993) 20.
- [103] S.T. Alexander, A.L. Ghimikar, A method for recursive least squares filtering based upon an inverse QR decomposition, *IEEE Trans. Signal Process.* 41 (1) (1993).
- [104] J.A. Apolinário Jr., P. Diniz, A new fast QR algorithm based on a priori errors, *IEEE Signal Process. Lett.* 4 (11) (1997) 307–309.
- [105] M.D. Miranda, L. Aguayo, M. Gerken, Performance of the a priori and a posteriori QR-LSL algorithms in a limited precision environment, in: Proc. IEEE Int Acoustics, Speech, and Signal Processing ICASSP-97 Conf., vol. 3, 1997, pp. 2337–2340.
- [106] M.D. Miranda, M. Gerken, A hybrid QR-lattice least squares algorithm using a priori errors, in: Proc. 38th Midwest Symp. Circuits and Systems, Proceedings, vol. 2, 1995, pp. 983–986.
- [107] M.D. Miranda, M. Gerken, A hybrid least squares QR-lattice algorithm using a priori errors, *IEEE Trans. Signal Process.* 45 (12) (1997) 2900–2911.
- [108] I.K. Proudler, J.G. McWhirter, T.J. Shepherd, Computationally efficient QR decomposition approach to least squares adaptive filtering, *IEE Proc. F Radar Signal Process.* 138 (4) (1991) 341–353.
- [109] P.A. Regalia, M.G. Bellanger, On the duality between fast QR methods and lattice methods in least-squares adaptive filtering, *IEEE Trans. Signal Process.* 39 (1991) 879–891.
- [110] A. Rontogiannis, S. Theodoridis, On inverse factorization adaptive LS algorithms, *Signal Process.* 52 (1997) 35–47.

- [111] A.A. Rontogiannis, S. Theodoridis, New fast inverse QR least squares adaptive algorithms, in: Proc. Int. Acoustics, Speech, and Signal Processing, vol. 2, 1995, pp. 1412–1415.
- [112] A.A. Rontogiannis, S. Theodoridis, New multichannel fast QRD-LS adaptive algorithms, in: Proc. 13th Int. Conf. Digital Signal Processing, vol. 1, 1997, pp. 45–48.
- [113] A.A. Rontogiannis, S. Theodoridis, Multichannel fast QRD-LS adaptive filtering: new technique and algorithms, IEEE Trans. Signal Process. 46 (11) (1998) 2862–2876.
- [114] A.A. Rontogiannis, S. Theodoridis, New fast QR decomposition least squares adaptive algorithms, IEEE Trans. Signal Process. 46 (8) (1998) 2113–2121.
- [115] M. Shoaib, S. Werner, J.A. Apolinário, Multichannel fast QR-decomposition algorithms: weight extraction method and its applications, IEEE Trans. Signal Process. 58 (1) (2010) 175–188.
- [116] M. Shoaib, S. Werner, J.A. Apolinário, T.I. Laakso, Solution to the weight extraction problem in fast QR-decomposition rls algorithms, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing Conf., ICASSP 2006, vol. 3, 2006.
- [117] B. Friedlander, Lattice filters for adaptive processing, IEEE Trans. Signal Process. 70 (8) (1982) 829–867.
- [118] D. Lee, M. Morf, B. Friedlander, Recursive least squares ladder estimation algorithms, IEEE Trans. Circ. Syst. 28 (6) (1981) 467–481.
- [119] H. Lev-Ari, T. Kailath, J. Cioffi, Least-squares adaptive lattice and transversal filters: a unified geometric theory, IEEE Trans. Inform. Theory 30 (2) (1984) 222–236.
- [120] F. Ling, D. Manolakis, J. Proakis, Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients, IEEE Trans. Acoust. Speech Signal Process. 34 (4) (1986) 837–845.
- [121] V.J. Mathews, Z. Xie, Fixed-point error analysis of stochastic gradient adaptive lattice filters, IEEE Trans. Acoust. Speech Signal Process. 38 (1) (1990) 70–80.
- [122] R.C. North, J.R. Zeidler, W.H. Ku, T.R. Albert, A floating-point arithmetic error analysis of direct and indirect coefficient updating techniques for adaptive lattice filters, IEEE Trans. Signal Process. 41 (5) (1993) 1809–1823.
- [123] M.J. Reed, B. Liu, Analysis of simplified gradient adaptive lattice algorithms using power-of-two quantization, in: Proc. IEEE Int. Circuits and Systems Symp., 1990, pp. 792–795.
- [124] C. Samson, V. Reddy, Fixed point error analysis of the normalized ladder algorithm, IEEE Trans. Acoust. Speech Signal Process. 31 (5) (1983) 1177–1191.
- [125] R. Merched, Extended RLS lattice adaptive filters, IEEE Trans. Signal Process. 51 (9) (2003) 2294–2309.
- [126] R. Merched, A.H. Sayed, Order-recursive RLS laguerre adaptive filtering, IEEE Trans. Signal Process. 48 (11) (2000) 3000–3010.
- [127] R. Merched, A.H. Sayed, RLS-Laguerre lattice adaptive filtering: error-feedback, normalized, and array-based algorithms, IEEE Trans. Signal Process. 49 (11) (2001) 2565–2576.
- [128] J.-L. Botto, G.V. Moustakides, Stabilizing the fast Kalman algorithms, IEEE Trans. Acoust. Speech Signal Process. 37 (9) (1989) 1342–1348.
- [129] J. Cioffi, T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering, IEEE Trans. Acoust. Speech Signal Process. 32 (2) (1984) 304–337.
- [130] J. Cioffi, T. Kailath, Windowed fast transversal filters adaptive algorithms with normalization, IEEE Trans. Acoust. Speech Signal Process. 33 (3) (1985) 607–625.
- [131] D. Falconer, L. Ljung, Application of fast Kalman estimation to adaptive equalization, IEEE Trans. Commun. 26 (10) (1978) 1439–1446.
- [132] G.V. Moustakides, S. Theodorides, Fast Newton transversal filters a new class of adaptive estimation algorithms, IEEE Trans. Signal Process. 39 (10) (1991) 2184–2193.
- [133] D.T.M. Slock, T. Kailath, Fast transversal filters with data sequence weighting, IEEE Trans. Acoust., Speech Signal Process. 37 (3) (1989) 346–359.

- [134] D.T.M. Slock, T. Kailath, Numerically stable fast transversal filters for recursive least squares adaptive filtering, *IEEE Trans. Signal Process.* 39 (1) (1991) 92–114.
- [135] Y. Zakharov, T. Tozer, Multiplication-free iterative algorithm for LS problem, *Electron. Lett.* 40 (9) (2004) 567–569.
- [136] W.A. Sethares, D.A. Lawrence, C.R. Johnson Jr., R.R. Bitmead, Parameter drift in LMS adaptive filters, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-34 (1986) 868–878.
- [137] K. Mayyas, T. Aboulnasr, The leaky LMS algorithm: MSE analysis for Gaussian data, *IEEE Trans. Signal Process.* 45 (4) (1997) 927–934.
- [138] C.S. Ludovico, J.C.M. Bermudez, A recursive least squares algorithm robust to low-power excitation, in: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, Proceedings (ICASSP'04), 2004.
- [139] J. Liu, Y. Zakharov, Low complexity dynamically regularised RLS algorithm, *Electron. Lett.* 44 (14) (2008) 885–886.
- [140] M.C. Tsakiris, C.G. Lopes, V.H. Nascimento, An array recursive least-squares algorithm with generic non-fading regularization matrix, *IEEE Signal Process. Lett.* 17 (12) (2010) 1001–1004.
- [141] N. Jablon, Adaptive beamforming with the generalized sidelobe canceller in the presence of array imperfections, *IEEE Trans. Antennas Propag.* 34 (8) (1986) 996–1012.
- [142] J.C.M. Bermudez, M.H. Costa, Optimum leakage factor for the MOV-LMS algorithm in nonlinear modeling and control systems, in: *Proc. IEEE Int. Acoustics, Speech, and Signal Processing (ICASSP) Conf.*, vol. 2, 2002.
- [143] E.J. Candès, M.B. Wakin, People hearing without listening: an introduction to compressive sampling, *IEEE Signal Process. Mag.* 25 (2) (2008) 21–30.
- [144] D. Angelosante, J.A. Bazerque, G.B. Giannakis, Online adaptive estimation of sparse signals: where RLS meets the  $\ell_1$ -norm, *IEEE Trans. Signal Process.* 58 (7) (2010) 3436–3447.
- [145] D. Angelosante, G.B. Giannakis, RLS-weighted Lasso for adaptive estimation of sparse signals, in: *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, ICASSP 2009*, 2009, pp. 3245–3248.
- [146] Y. Chen, Y. Gu, A.O. Hero, Sparse LMS for system identification, in: *Proc. ICASSP 2009*, April 2009, pp. 3125–3128.
- [147] Y. Kopsinis, K. Slavakis, S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted  $\ell_1$  balls, *IEEE Trans. Signal Process.* 59 (3) (2011) 936–952.
- [148] G. Mileounis, B. Babadi, N. Kalouptsidis, V. Tarokh, An adaptive greedy algorithm with application to nonlinear communications, *IEEE Trans. Signal Process.* 58 (6) (2010) 2998–3007.
- [149] R. Harris, D. Chabries, F. Bishop, A variable step (VS) adaptive filter algorithm, *IEEE Trans. Acoust. Speech Signal Process.* 34 (2) (1986) 309–316.
- [150] B. Farhang-Boroujeny, *Adaptive Filters – Theory and Applications*, John Wiley & Sons, West Sussex, 1998.
- [151] J. Nagumo, A. Noda, A learning method for system identification, *IEEE Trans. Autom. Control* 12 (3) (1967) 282–287.
- [152] A.E. Albert, L.S. Gardner Jr., *Stochastic Approximation and Nonlinear Regression*, MIT Press, 1967.
- [153] T. Aboulnasr, K. Mayyas, A robust variable step-size LMS-type algorithm, *IEEE Trans. Signal Process.* 45 (3) (1997) 631–639.
- [154] W.-P. Ang, B. Farhang-Boroujeny, A new class of gradient adaptive step-size LMS algorithms, *IEEE Trans. Signal Process.* 49 (4) (2001) 805–810.
- [155] J. Benesty, H. Rey, L. Vega, S. Tressens, A nonparametric VSS NLMS algorithm, *IEEE Signal Process. Lett.* 13 (10) (2006) 581–584.
- [156] N. Bershad, On the optimum gain parameter in LMS adaptation, *IEEE Trans. Acoust. Speech Signal Process.* 35 (7) (1987) 1065–1068.

- [157] R.C. Bilcu, P. Kuosmanen, K. Egiazarian, A transform domain LMS adaptive filter with variable step-size, *IEEE Signal Process. Lett.* 9 (2) (2002) 51–53.
- [158] J.B. Evans, P. Xue, B. Liu, Analysis and implementation of variable step size adaptive algorithms, *IEEE Trans. Signal Process.* 41 (8) (1993) 2517–2535.
- [159] B. Farhang-Boroujeny, Variable-step-size LMS algorithm: new developments and experiments, *IEE Proc. Vis. Image Signal Process.* 141 (5) (1994) 311–317.
- [160] A. Gupta, S. Joshi, Variable step-size LMS algorithm for fractal signals, *IEEE Trans. Signal Process.* 56 (4) (2008) 1411–1420.
- [161] R.H. Kwong, E.W. Johnston, A variable step size LMS algorithm, *IEEE Trans. Signal Process.* 40 (7) (1992) 1633–1642.
- [162] V.J. Mathews, Z. Xie, A stochastic gradient adaptive filter with gradient adaptive step size, *IEEE Trans. Signal Process.* 41 (6) (1993) 2075–2087.
- [163] C. Paleologu, J. Benesty, S. Ciochina, A robust variable forgetting factor recursive least-squares algorithm for system identification, *IEEE Signal Process. Lett.* 15 (2008) 597–600.
- [164] C. Paleologu, S. Ciochina, J. Benesty, Variable step-size NLMS algorithm for under-modeling acoustic echo cancellation, *IEEE Signal Process. Lett.* 15 (2008) 5–8.
- [165] D.I. Pazaritis, A.G. Constantinides, A novel kurtosis driven variable step-size adaptive algorithm, *IEEE Trans. Signal Process.* 47 (3) (1999) 864–872.
- [166] H.-C. Shin, A.H. Sayed, W.-J. Song, Variable step-size NLMS and affine projection algorithms, *IEEE Signal Process. Lett.* 11 (2) (2004) 132–135.
- [167] L. Vega, H. Rey, J. Benesty, S. Tressens, A new robust variable step-size NLMS algorithm, *IEEE Trans. Signal Process.* 56 (5) (2008) 1878–1893.
- [168] J. Benesty, S.L. Gay, An improved PNLMS algorithm, in: International Conference on Acoustics, Speech and Signal Processing, vol. 2, IEEE, 2002, pp. 1881–1884.
- [169] F. das Chagas de Souza, O.J. Tobias, R. Seara, D.R. Morgan, A PNLMS algorithm with individual activation factors, *IEEE Trans. Signal Process.* 58 (4) (2010) 2036–2047.
- [170] F. das Chagas de Souza, O.J. Tobias, R. Seara, D.R. Morgan, Stochastic model for the mean weight evolution of the IAF-PNLMS algorithm, *IEEE Trans. Signal Process.* 58 (11) (2010) 5895–5901.
- [171] D.L. Duttweiler, Proportionate normalized least-mean-squares adaptation in echo cancellers, *IEEE Trans. Speech Audio Process.* 8 (5) (2000) 508–518.
- [172] T. Gansler, S.L. Gay, M.M. Sondhi, J. Benesty, Double-talk robust fast converging algorithms for network echo cancellation, *IEEE Trans. Speech Audio Process.* 8 (6) (2000) 656–663.
- [173] E. Walach, B. Widrow, The least mean fourth (LMF) adaptive algorithm and its family, *IEEE Trans. Inform. Theory* IT-30 (2) (1984) 275–283.
- [174] R. Sharma, W.A. Sethares, J.A. Bucklew, Asymptotic analysis of stochastic gradient-based adaptive filtering algorithms with general cost functions, *IEEE Trans. Signal Process.* 44 (9) (1996) 2186–2194.
- [175] T.Y. Al-Naffouri, A.H. Sayed, Transient analysis of adaptive filters with error nonlinearities, *IEEE Trans. Signal Process.* 51 (3) (2003) 653–663.
- [176] J.A. Chambers, O. Tanrikulu, A.G. Constantinides, Least mean mixed-norm adaptive filtering, *Electron. Lett.* 30 (19) (1994).
- [177] O. Tanrikulu, J.A. Chambers, Convergence and steady-state properties of the least-mean mixed-norm (LMMN) adaptive algorithm, *IEEE Trans. Signal Process.* 44 (3) (1996) 137–142.
- [178] A.H. Sayed, Fundamentals of Adaptive Filtering, Wiley-Interscience, 2003.
- [179] B. Hassibi, A.H. Sayed, T. Kailath, LMS is  $H^\infty$ -optimal, in: Proc. Conference on Decision and Control, San Antonio, TX, vol. 1, December 1993, pp. 74–79.
- [180] G. Aydin, O. Arikan, A.E. Cetin, Robust adaptive filtering algorithms for  $\alpha$ -stable random processes, *IEEE Trans. Circ. Syst. II* 46 (2) (1999) 198–202.

- [181] S.R. Kim, A. Efron, Adaptive robust impulse noise filtering, *IEEE Trans. Signal Process.* 43 (8) (1995) 1855–1866.
- [182] E. Masry, Alpha-stable signals and adaptive filtering, *IEEE Trans. Signal Process.* 48 (11) (2000) 3011–3016.
- [183] M. Shao, C.L. Nikias, Signal processing with fractional lower order moments: stable processes and their applications, *Proc. IEEE* 81 (7) (1993) 986–1010.
- [184] D. Hirsch, W. Wolf, A simple adaptive equalizer for efficient data transmission, *IEEE Trans. Commun. Technol.* 18 (1) (1970) 5–12.
- [185] A. Flaig, G.R. Arce, K.E. Barner, Affine order-statistic filters: medianization of linear FIR filters, *IEEE Trans. Signal Process.* 46 (8) (1998) 2101–2112.
- [186] D.N. Godard, Self-recovering equalization and carrier tracking in two-dimensional data communication systems, *IEEE Trans. Commun.* 28 (11) (1980) 1867–1875.
- [187] J.R. Treichler, B. Agee, A new approach to multipath correction of constant modulus signals, *IEEE Trans. Acoust. Speech Signal Process. ASSP-28* (1983) 334–358.
- [188] C.B. Papadias, D.T.M. Slock, Normalized sliding window constant modulus and decision-direct algorithms: a link between blind equalization and classical adaptive filtering, *IEEE Trans. Signal Process.* 45 (1997) 231–235.
- [189] Z. Ding, Y. Li, *Blind Equalization and Identification*, Marcel Dekke, 2001.
- [190] M.T.M. Silva, V.H. Nascimento, Improving the tracking capability of adaptive filters via convex combination, *IEEE Trans. Signal Process.* 56 (7) (2008) 3137–3149.
- [191] C.R. Johnson Jr., P. Schniter, T.J. Endres, J.D. Behm, D.R. Brown, R.A. Casas, Blind equalization using the constant modulus criterion: a review, *Proc. IEEE* 86 (10) (1998) 1927–1950.
- [192] R. Candido, M.T.M. Silva, M.D. Miranda, V.H. Nascimento, A statistical analysis of the dual-mode CMA, in: *Proc. IEEE Int. Circuits and Systems (ISCAS) Symp.*, 2010, pp. 2510–2513.
- [193] R. Candido, M.T.M. Silva, V.H. Nascimento, Transient and steady-state analysis of the affine combination of two adaptive filters, *IEEE Trans. Signal Process.* 58 (8) (2010) 4064–4078.
- [194] I. Fijalkow, C.E. Manlove, C.R. Johnson Jr., Adaptive fractionally spaced blind CMA equalization: excess MSE, *IEEE Trans. Signal Process.* 46 (1) (1998) 227–231.
- [195] M.T.M. Silva, M.D. Miranda, Tracking issues of some blind equalization algorithms, *IEEE Signal Process. Lett.* 11 (9) (2004) 760–763.
- [196] M.T.M. Silva, V.H. Nascimento, Tracking analysis of the constant modulus algorithm, in: *Proc., ICASSP 2008*, Las Vegas, NV, USA, pp. 3561–3564.
- [197] M.D. Miranda, M.T.M. Silva, V.H. Nascimento, Avoiding divergence in the constant modulus algorithm, in: *Proc., ICASSP 2008*, Las Vegas, NV, USA, 2008, pp. 3565–3568.
- [198] M.D. Miranda, M.T.M. Silva, V.H. Nascimento, Avoiding divergence in the Shalvi-Weinstein algorithm, *IEEE Trans. Signal Process.* 56 (11) (2008) 5403–5413.
- [199] P.A. Regalia, M. Mboup, Undermodeled equalization: a characterization of stationary points for a family of blind criteria, *IEEE Trans. Signal Process.* 47 (1999) 760–770.
- [200] R. Suyama, R.R.F. Attux, J.M.T. Romano, M. Bellanger, On the relationship between least squares and constant modulus criteria for adaptive filtering, in: *The 37th Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2003, pp. 1293–1297.
- [201] J. Yang, J.-J. Werner, G.A. Dumont, The multimodulus blind equalization and its generalized algorithms, *IEEE J. Sel. Areas Commun.* 20 (2002) 997–1015.
- [202] J.-T. Yuan, T.-C. Lin, Equalization and carrier phase recovery of CMA and MMA in blind adaptive receivers, *IEEE Trans. Signal Process.* 58 (6) (2010) 3206–3217.
- [203] J.-T. Yuan, K.-D. Tsai, Analysis of the multimodulus blind equalization algorithm in QAM communication systems, *IEEE Trans. Commun.* 53 (2005) 1427–1431.

- [204] S. Abrar, A. Nandi, Blind equalization of square-QAM signals: a multimodulus approach, *IEEE Trans. Commun.* 58 (6) (2010) 1674–1685.
- [205] K. Banovic, E. Abdel-Raheem, M.A.S. Khalid, A novel radius-adjusted approach for blind adaptive equalization, *IEEE Signal Process. Lett.* 13 (2006) 37–40.
- [206] J. Mendes Filho, M.T.M. Silva, M.D. Miranda, A family of algorithms for blind equalization of QAM signals, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process, Prague, Czech Republic, 2011, pp. 3388–3391.
- [207] J. Mendes Filho, M.T.M. Silva, M.D. Miranda, V.H. Nascimento, A region-based algorithm for blind equalization of QAM signals, in: Proc. of the IEEE/SP 15th Workshop on Statistical Signal Processing, Cardiff, UK, 2009, pp. 685–688.
- [208] F. Beaufays, Transform-domain adaptive filters: an analytical approach, *IEEE Trans. Signal Process.* 43 (2) (1995) 422–431.
- [209] G. Clark, S. Mitra, S. Parker, Block implementation of adaptive digital filters, *IEEE Trans. Acoust. Speech Signal Process.* 29 (3) (1981) 744–752.
- [210] B. Farhang-Boroujeny, Analysis and efficient implementation of partitioned block LMS adaptive filters, *IEEE Trans. Signal Process.* 44 (11) (1996) 2865–2868.
- [211] E. Ferrara, Fast implementations of LMS adaptive filters, *IEEE Trans. Acoust. Speech Signal Process.* 28 (4) (1980) 474–475.
- [212] R. Merched, A.H. Sayed, An embedding approach to frequency-domain and subband adaptive filtering, *IEEE Trans. Signal Process.* 48 (9) (2000) 2607–2619.
- [213] S. Narayan, A. Peterson, M. Narasimha, Transform domain LMS algorithm, *IEEE Trans. Acoust. Speech Signal Process.* 31 (3) (1983) 609–615.
- [214] S.S. Narayan, A.M. Peterson, Frequency domain least-mean-square algorithm, *Proc. IEEE* 69 (1) (1981) 124–126.
- [215] J.J. Shynk, Frequency-domain and multirate adaptive filtering, *IEEE Signal Process. Mag.* 9 (1) (1992) 14–37.
- [216] J.-S. Soo, K.K. Pang, Multidelay block frequency domain adaptive filter, *IEEE Trans. Acoust. Speech Signal Process.* 38 (2) (1990) 373–376.
- [217] P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [218] I. Furukawa, A design of canceller of broadband acoustic echo, in: Proc. Int. Teleconference Symposium, Tokyo, Japan, 1984, pp. 1–8.
- [219] A. Gilloire, M. Vetterli, Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancellation, *IEEE Trans. Signal Process.* 40 (8) (1992) 1862–1875.
- [220] W. Kellermann, Kompensation akustischer Echos in Frequenzteilbändern, *Frequenz* 39 (1985) 209–215.
- [221] M.R. Petraglia, R.G. Alves, P.S.R. Diniz, New structures for adaptive filtering in subbands with critical sampling, *IEEE Trans. Signal Process.* 48 (12) (2000) 3316–3327.
- [222] M.R. Petraglia, P.B. Batalheiro, Nonuniform subband adaptive filtering with critical sampling, *IEEE Trans. Signal Process.* 56 (2) (2008) 565–575.
- [223] M.R. Petraglia, S.K. Mitra, Adaptive fir filter structure based on the generalized subband decomposition of fir filters, *IEEE Trans. Circ. Syst. II* 40 (6) (1993) 354–362.
- [224] S. Weiss, R.W. Stewart, R. Rabenstein, Steady-state performance limitations of subband adaptive filters, *IEEE Trans. Signal Process.* 49 (9) (2001) 1982–1991.
- [225] N. Hirayama, H. Sakai, S. Miyagi, Delayless subband adaptive filtering using the Hadamard transform, *IEEE Trans. Signal Process.* 47 (6) (1999) 1731–1734.
- [226] R. Merched, M. Petraglia, P.S.R. Diniz, A delayless alias-free subband adaptive filter structure, *IEEE Trans. Signal Process.* 47 (6) (1999) 1580–1591.

- [227] D.R. Morgan, J.C. Thi, A delayless subband adaptive filter architecture, *IEEE Trans. Signal Process.* 43 (8) (1995) 1819–1830.
- [228] K. Nishikawa, H. Kiya, Conditions for convergence of a delayless subband adaptive filter and its efficient implementation, *IEEE Trans. Signal Process.* 46 (4) (1998) 1158–1167.
- [229] J. Apolinário Jr., M.L.R. Campos, P.S.R. Diniz, Convergence analysis of the binormalized data-reusing LMS algorithm, *IEEE Trans. Signal Process.* 48 (11) (2000) 3235–3242.
- [230] S.G. Kratzer, D.R. Morgan, The partial-rank algorithm for adaptive beamforming, in: Proc. SPIE, vol. 564, 1985, pp. 9–14.
- [231] D.R. Morgan, S.G. Kratzer, On a class of computationally efficient, rapidly converging, generalized NLMS algorithms, *IEEE Signal Process. Lett.* 3 (8) (1996) 245–247.
- [232] K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties, *Electron. Commun. Jpn.* 67-A (1984) 19–27.
- [233] S. Roy, J.J. Shynk, Analysis of the data-reusing LMS algorithm, in: Proc. 32nd Midwest Symp. Circuits and Systems, 1989, pp. 1127–1130.
- [234] M. Rupp, A family of adaptive filter algorithms with decorrelating properties, *IEEE Trans. Signal Process.* 46 (3) (1998) 771–775.
- [235] S.L. Gay, S. Tavathia, The fast affine projection algorithm, in: Proc. Int. Acoustics, Speech, and Signal Processing ICASSP-95 Conf., vol. 5, 1995, pp. 3023–3026.
- [236] G. Rombouts, M. Moonen, A fast exact frequency domain implementation of the exponentially windowed affine projection algorithm, in: Proc. and Control Symp Adaptive Systems for Signal Processing, Communications 2000, AS-SPCC, The IEEE 2000, 2000, pp. 342–346.
- [237] M. Tanaka, S. Makino, J. Kojima, A block exact fast affine projection algorithm, *IEEE Trans. Speech Audio Process.* 7 (1) (1999) 79–86.
- [238] S.J.M. de Almeida, J.C.M. Bermudez, N.J. Bershad, A stochastic model for a pseudo affine projection algorithm, *IEEE Trans. Signal Process.* 57 (1) (2009) 107–118.
- [239] S.J.M. de Almeida, J.C.M. Bermudez, N.J. Bershad, M.H. Costa, A statistical analysis of the affine projection algorithm for unity step size and autoregressive inputs, *IEEE Trans. Circ. Syst. I* 52 (7) (2005) 1394–1405.
- [240] S.G. Sankaran, A.A.L. Beex, Convergence behavior of affine projection algorithms, *IEEE Trans. Signal Process.* 48 (4) (2000) 1086–1096.
- [241] H.-C. Shin, A.H. Sayed, Mean-square performance of a family of affine projection algorithms, *IEEE Trans. Signal Process.* 52 (1) (2004) 90–102.
- [242] C. Paleologu, J. Benesty, S. Ciochina, Regularization of the affine projection algorithm, *IEEE Trans. Circ. Syst. II* 58 (6) (2011) 366–370.
- [243] H. Rey, L.R. Vega, S. Tressens, J. Benesty, Variable explicit regularization in affine projection algorithm: robustness issues and optimal choice, *IEEE Trans. Signal Process.* 55 (5) (2007) 2096–2109.
- [244] J. Benesty, P. Duhamel, Y. Grenier, A multichannel affine projection algorithm with applications to multi-channel acoustic echo cancellation, *IEEE Signal Process. Lett.* 3 (2) (1996) 35–37.
- [245] C. Paleologu, J. Benesty, S. Ciochina, A variable step-size affine projection algorithm designed for acoustic echo cancellation, *IEEE Trans. Audio Speech Lang. Process.* 16 (8) (2008) 1466–1478.
- [246] D. Manolakis, F. Ling, J. Proakis, Efficient time-recursive least-squares algorithms for finite-memory adaptive filtering, *IEEE Trans. Circ. Syst.* 34 (4) (1987) 400–408.
- [247] K. Maouche, D.T.M. Slock, Performance analysis and FTF version of the generalized sliding window recursive least-squares (GSWRLS) algorithm, in: Proc. Conf. Signals, Systems and Computers Record of the 29th Asilomar Conf., vol. 1, 1995, pp. 685–689.
- [248] J. Arenas-García, A.R. Figueiras-Vidal, A.H. Sayed, Mean-square performance of a convex combination of two adaptive filters, *IEEE Trans. Signal Process.* 54 (3) (2006) 1078–1090.

- [249] M. Martínez-Ramon, J. Arenas-García, A. Navia-Vázquez, A.R. Figueiras-Vidal, An adaptive combination of adaptive filters for plant identification, in: 14th International Conference on Digital Signal Processing (DSP 2002), 2002, pp. 1195–1198.
- [250] J. Arenas-García, M. Martínez-Ramón, A. Navia-Vázquez, A.R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, *Signal Process.* 86 (2006) 2430–2438.
- [251] L.A. Azpicueta-Ruiz, A.R. Figueiras-Vidal, J. Arenas-García, A normalized adaptation scheme for the convex combination of two adaptive filters, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., Las Vegas, NV, 2008, pp. 3301–3304.
- [252] P. Andersson, Adaptive forgetting in recursive identification through multiple models, *Int. J. Control.* 42 (1985) 1175–1193.
- [253] M. Niedźwiecki, Identification of nonstationary stochastic systems using parallel estimation schemes, *IEEE Trans. Autom. Control* 35 (3) (1990) 329–334.
- [254] M. Niedźwiecki, Multiple-model approach to finite memory adaptive filtering, *IEEE Trans. Signal Process.* 40 (2) (1992) 470–473.
- [255] S.S. Kozat, A.C. Singer, Multi-stage adaptive signal processing algorithms, in: Proc. SAM Signal Process. Workshop, 2000, pp. 380–384.
- [256] V.H. Nascimento, M.T. Silva, R. Cândido, J. Arenas-García, A transient analysis for the convex combination of adaptive filters, in: Proc. IEEE Workshop on Statistical Signal Process., Cardiff, UK, 2009, pp. 53–56.
- [257] M.T.M. Silva, V.H. Nascimento, J. Arenas-García, A transient analysis for the convex combination of two adaptive filters with transfer of coefficients, in: Proc., ICASSP 2010, Dallas, USA, pp. 3842–3845.
- [258] Y. Zhang, J. Chambers, Convex combination of adaptive filters for a variable tap-length LMS algorithm, *IEEE Signal Process. Lett.* 13 (10) (2006) 628–631.
- [259] V.H. Nascimento, M.T.M. Silva, L.A. Azpicueta-Ruiz, J. Arenas-García, On the tracking performance of combinations of least mean squares and recursive least squares adaptive filters, in: Proc. IEEE Int. Acoustics Speech and Signal Processing (ICASSP) Conf., 2010, pp. 3710–3713.
- [260] J. Arenas-García, A.R. Figueiras-Vidal, Adaptive combination of proportionate filters for sparse echo cancellation, *IEEE Trans. Audio Speech Lang. Process.* 17 (2009) 1087–1098.
- [261] L.A. Azpicueta-Ruiz, M. Zeller, A.R. Figueiras-Vidal, J. Arenas-García, W. Kellermann, Adaptive combination of Volterra kernels and its application to nonlinear acoustic echo cancellation, *IEEE Trans. Acoust. Speech Signal Process.* 19 (2011) 97–110.
- [262] M. Lazaro-Gredilla, L.A. Azpicueta-Ruiz, A.R. Figueiras-Vidal, J. Arenas-García, Adaptively biasing the weights of adaptive filters, *IEEE Trans. Signal Process.* 58 (7) (2010) 3890–3895.
- [263] N.J. Bershad, J.C.M. Bermudez, J.-Y. Tourneret, An affine combination of two LMS adaptive filters—transient mean-square analysis, *IEEE Trans. Signal Process.* 56 (5) (2008) 1853–1864.
- [264] J.C.M. Bermudez, N.J. Bershad, J.-Y. Tourneret, Stochastic analysis of an error power ratio scheme applied to the affine combination of two LMS adaptive filters, *Signal Process.* 91 (11) (2011) 2615–2622.
- [265] S.S. Kozat, A.T. Erdogan, A.C. Singer, A.H. Sayed, Steady-state MSE performance analysis of mixture approaches to adaptive filtering, *IEEE Trans. Signal Process.* 58 (8) (2010) 4050–4063.
- [266] C.G. Lopes, A.H. Sayed, Incremental adaptive strategies over distributed networks, *IEEE Trans. Signal Process.* 55 (2007) 4064–4077.
- [267] C.G. Lopes, A.H. Sayed, Diffusion least-mean squares over adaptive networks: formulation and performance analysis, *IEEE Trans. Signal Process.* 56 (7) (2008) 3122–3136.
- [268] F.S. Cattivelli, A.H. Sayed, Diffusion LMS strategies for distributed estimation, *IEEE Trans. Signal Process.* 58 (3) (2010) 1035–1048.
- [269] F.S. Cattivelli, C.G. Lopes, A.H. Sayed, Diffusion recursive least-squares for distributed estimation over adaptive networks, *IEEE Trans. Signal Process.* 56 (5) (2008) 1865–1877.

- [270] F.S. Cattivelli, A.H. Sayed, Diffusion strategies for distributed Kalman filtering and smoothing, *IEEE Trans. Autom. Control* 55 (9) (2010) 2069–2084.
- [271] F.S. Cattivelli, A.H. Sayed, Analysis of spatial and incremental LMS processing for distributed estimation, *IEEE Trans. Signal Process.* 59 (4) (2011) 1465–1480.
- [272] F.S. Cattivelli, A.H. Sayed, Distributed detection over adaptive networks using diffusion adaptation, *IEEE Trans. Signal Process.* 59 (5) (2011) 1917–1932.
- [273] R.L.G. Cavalcante, I. Yamada, B. Mulgrew, An adaptive projected subgradient approach to learning in diffusion networks, *IEEE Trans. Signal Process.* 57 (7) (2009) 2762–2774.
- [274] S. Chouvardas, K. Slavakis, S. Theodoridis, Adaptive robust distributed learning in diffusion sensor networks, *IEEE Trans. Signal Process.* 59 (10) (2011) 4692–4707.
- [275] L. Li, J. Chambers, A new incremental affine projection-based adaptive algorithm for distributed networks, *Signal Process.* 88 (10) (2008) 2599–2603.
- [276] G. Mateos, I.D. Schizas, G.B. Giannakis, Closed-form MSE performance of the distributed LMS algorithm, in: Proc. IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop DSP/SPE 2009, 2009, pp. 66–71.
- [277] G. Mateos, I.D. Schizas, G.B. Giannakis, Distributed recursive least-squares for consensus-based in-network adaptive estimation, *IEEE Trans. Signal Process.* 57 (11) (2009) 4583–4588.
- [278] V.H. Nascimento, A.H. Sayed, Continuous-time distributed estimation, in: Proc. of the 45th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, November 2011, pp. 1–5.
- [279] I.D. Schizas, G. Mateos, G.B. Giannakis, Distributed LMS for consensus-based in-network adaptive processing, *IEEE Trans. Signal Process.* 57 (6) (2009) 2365–2382.
- [280] N. Takahashi, I. Yamada, A.H. Sayed, Diffusion least-mean squares with adaptive combiners: formulation and performance analysis, *IEEE Trans. Signal Process.* 58 (9) (2010) 4795–4810.
- [281] F.S. Cattivelli, A.H. Sayed, Modeling bird flight formations using diffusion adaptation, *IEEE Trans. Signal Process.* 59 (5) (2011) 2038–2051.
- [282] J. Chen, A.H. Sayed, Bio-inspired cooperative optimization with application to bacteria motility, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing (ICASSP) Conf., 2011, pp. 5788–5791.
- [283] J. Chen, X. Zhao, A.H. Sayed, Bacterial motility via diffusion adaptation, in: Proc. Conf. Signals, Systems and Computers (ASILOMAR) Record of the 44th Asilomar Conf., 2010, pp. 1930–1934.
- [284] S.-Y. Tu, A.H. Sayed, Foraging behavior of fish schools via diffusion adaptation, in: Proc. 2nd Int. Cognitive Information Processing (CIP), Workshop, 2010, pp. 63–68.
- [285] S.-Y. Tu, A.H. Sayed, Tracking behavior of mobile adaptive networks, in: Proc. Conf. Signals, Systems and Computers (ASILOMAR) Record of the 44th Asilomar Conf., 2010, pp. 698–702.
- [286] S.-Y. Tu, A.H. Sayed, Cooperative prey herding based on diffusion adaptation, in: Proc. IEEE Int. Acoustics, Speech and Signal Processing (ICASSP) Conf., 2011, pp. 3752–3755.
- [287] P.M.S. Burt, P. Regalia, A new framework for convergence analysis and algorithm development of adaptive IIR filters, *IEEE Trans. Signal Process.* 53 (2005) 3129–3140.
- [288] S.L. Netto, P.S.R. Diniz, P. Agathoklis, Adaptive IIR filtering algorithms for system identification: a general framework, *IEEE Trans. Educ.* 38 (1) (1995) 54–66.
- [289] P.A. Regalia, Adaptive IIR Filtering in Signal Processing and Control, Marcel Dekker, 1995.
- [290] J.J. Shynk, Adaptive IIR filtering, *IEEE ASSP Mag.* 6 (2) (1989) 4–21.
- [291] P.M.S. Burt, M. Gerken, A polyphase IIR adaptive filter: error surface analysis and application, in: Proc. IEEE Int. Conf. Acoustics, Speech, and, Signal Process., vol. 3, April 1997, pp. 2285–2288.
- [292] P.M.S. Burt, M. Gerken, A polyphase IIR adaptive filter, *IEEE Trans. Circ. Syst. II* 49 (5) (2002) 356–359.
- [293] J.E. Cousseau, P.S. Diniz, A general consistent equation-error algorithm for adaptive IIR filtering, *Signal Process.* 56 (2) (1997) 121–134.

- [294] J.E. Cousseau, P.S.R. Diniz, New adaptive IIR filtering algorithms based on the Steiglitz-McBride method, *IEEE Trans. Signal Process.* 45 (5) (1997) 1367–1371.
- [295] P.S.R. Diniz, J.E. Cousseau, A. Antoniou, Fast parallel realization of IIR adaptive filters, *IEEE Trans. Circ. Syst. II* 41 (8) (1994) 561–567.
- [296] H. Fan, A structural view of asymptotic convergence speed of adaptive IIR filtering algorithms. I. Infinite precision implementation, *IEEE Trans. Signal Process.* 41 (4) (1993) 1493–1517.
- [297] C.R. Johnson Jr., Adaptive IIR filtering: current results and open issues, *IEEE Trans. Inform. Theory* 30 (2) (1984) 237–250.
- [298] R. Lopez-Valcarce, F. Perez-Gonzalez, Adaptive lattice IIR filtering revisited: convergence issues and new algorithms with improved stability properties, *IEEE Trans. Signal Process.* 49 (4) (2001) 811–821.
- [299] P.A. Regalia, Stable and efficient lattice algorithms for adaptive IIR filtering, *IEEE Trans. Signal Process.* 40 (2) (1992) 375–388.
- [300] S. Stearns, Error surfaces of recursive adaptive filters, *IEEE Trans. Acoust. Speech Signal Process.* 29 (3) (1981) 763–766.
- [301] D.P. Bertsekas, I.B. Rhodes, Recursive state estimation for a set-membership description of uncertainty, *IEEE Trans. Autom. Control* AC-16 (2) (1971) 117–128.
- [302] S. Dasgupta, Y.-F. Huang, Asymptotically convergent modified recursive least-squares with data-dependent updating and forgetting factor for systems with bounded noise, *IEEE Trans. Inform. Theory* 33 (3) (1987) 383–392.
- [303] J.R. Deller Jr., M. Nayeri, S.F. Odeh, Least-square identification with error bounds for real-time signal processing and control, *Proc. IEEE* 81 (6) (1993) 815–849.
- [304] T. Aboulhasr, K. Mayyas, Complexity reduction of the NLMS algorithm via selective coefficient update, *IEEE Trans. Signal Process.* 47 (5) (1999) 1421–1424.
- [305] R.C. de Lamare, P. Diniz, Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression, *IEEE Trans. Veh. Technol.* 58 (2) (2009) 644–654.
- [306] P.S.R. Diniz, S. Werner, Set-membership binormalized data-reusing LMS algorithms, *IEEE Trans. Signal Process.* 51 (1) (2003) 124–134.
- [307] K. Dogancay, O. Tanrikulu, Adaptive filtering algorithms with selective partial updates, *IEEE Trans. Circ. Syst. II* 48 (8) (2001) 762–769.
- [308] S.C. Douglas, Adaptive filters employing partial updates, *IEEE Trans. Circ. Syst. II* 44 (3) (1997) 209–216.
- [309] S. Werner, J.A. Apolinário, M.L. de Campos, P.S. Diniz, Low-complexity constrained affine-projection algorithms, *IEEE Trans. Signal Process.* 53 (12) (2005) 4545–4555.
- [310] S. Werner, M.L.R. de Campos, P.S.R. Diniz, Partial-update NLMS algorithms with data-selective updating, *IEEE Trans. Signal Process.* 52 (4) (2004) 938–949.
- [311] K. Slavakis, S. Theodoridis, I. Yamada, Online kernel-based classification using adaptive projection algorithms, *IEEE Trans. Signal Process.* 56 (7) (2008) 2781–2796.
- [312] K. Slavakis, S. Theodoridis, I. Yamada, Adaptive constrained learning in reproducing kernel hilbert spaces: the robust beamforming case, *IEEE Trans. Signal Process.* 57 (12) (2009) 4744–4764.
- [313] I. Yamada, K. Slavakis, K. Yamada, An efficient robust adaptive filtering algorithm based on parallel sub-gradient projection techniques, *IEEE Trans. Signal Process.* 50 (5) (2002) 1091–1101.
- [314] W. Liu, J. Principe, S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley Publishing, 2010.
- [315] J. Capon, High-resolution frequency-wavenumber spectrum analysis, *Proc. IEEE* 57 (8) (1969) 1408–1418.
- [316] S. Applebaum, Adaptive arrays, *IEEE Trans. Antennas Propag.* 24 (5) (1976) 585–598.
- [317] N. Bornhorst, M. Pesavento, A.B. Gershman, Distributed beamforming for multi-group multicasting relay networks, *IEEE Trans. Signal Process.* 60 (1) (2012) 221–232.

- [318] H. Cox, R. Zeskind, M. Owen, Robust adaptive beamforming, *IEEE Trans. Acoust. Speech, Signal Process.* 35 (10) (1987) 1365–1376.
- [319] O.L. Frost III, An algorithm for linearly constrained adaptive array processing, *Proc. IEEE* 60 (8) (1972) 926–935.
- [320] G.-O. Glentis, K. Berberidis, S. Theodoridis, Efficient least squares adaptive algorithms for FIR transversal filtering, *IEEE Signal Process. Mag.* 16 (4) (1999) 13–41.
- [321] L. Lei, J.P. Lie, A.B. Gershman, C.M.S. See, Robust adaptive beamforming in partly calibrated sparse sensor arrays, *IEEE Trans. Signal Process.* 58 (3) (2010) 1661–1667.
- [322] B.D. Van Veen, An analysis of several partially adaptive beamformer designs, *IEEE Trans. Acoust. Speech Signal Process.* 37 (2) (1989) 192–203.
- [323] B.D. van Veen, Minimum variance beamforming with soft response constraints, *IEEE Trans. Signal Process.* 39 (9) (1991) 1964–1972.
- [324] S.A. Vorobyov, H. Chen, A.B. Gershman, On the relationship between robust minimum variance beamformers with probabilistic and worst-case distortionless response constraints, *IEEE Trans. Signal Process.* 56 (11) (2008) 5719–5724.
- [325] S.A. Vorobyov, A.B. Gershman, Z.-Q. Luo, N. Ma, Adaptive beamforming with joint robustness against mismatched signal steering vector and interference nonstationarity, *IEEE Signal Process. Lett.* 11 (2) (2004) 108–111.
- [326] H. Krim, M. Viberg, Two decades of array signal processing research: the parametric approach, *IEEE Signal Process. Mag.* 13 (4) (1996) 67–94.
- [327] H.L.V. Trees, *Optimum Array Processing*, Wiley, 2002.
- [328] B.D. Van Veen, K.M. Buckley, Beamforming: a versatile approach to spatial filtering, *IEEE ASSP Mag.* 5 (2) (1988) 4–24.
- [329] L. Scharf, D. Tufts, Rank reduction for modeling stationary signals, *IEEE Trans. Acoust. Speech Signal Process.* 35 (3) (1987) 350–355.
- [330] A.M. Haimovich, Y. Bar-Ness, An eigenanalysis interference canceler, *IEEE Trans. Signal Process.* 39 (1) (1991) 76–84.
- [331] Y. Song, S. Roy, Blind adaptive reduced-rank detection for DS-CDMA signals in multipath channels, *IEEE J. Sel. Areas Commun.* 17 (11) (1999) 1960–1970.
- [332] J.S. Goldstein, I.S. Reed, L.L. Scharf, A multistage representation of the Wiener filter based on orthogonal projections, *IEEE Trans. Inform. Theory* 44 (7) (1998) 2943–2959.
- [333] M.L. Honig, J.S. Goldstein, Adaptive reduced-rank interference suppression based on the multistage Wiener filter, *IEEE Trans. Commun.* 50 (6) (2002) 986–994.
- [334] L.L. Scharf, E.K.P. Chong, M.D. Zoltowski, J.S. Goldstein, I.S. Reed, Subspace expansion and the equivalence of conjugate direction and multistage Wiener filters, *IEEE Trans. Signal Process.* 56 (10) (2008) 5013–5019.
- [335] D.A. Pados, G.N. Karystinos, An iterative algorithm for the computation of the MVDR filter, *IEEE Trans. Signal Process.* 49 (2) (2001) 290–300.
- [336] R.C. de Lamare, R. Sampaio-Neto, Reduced-rank interference suppression for DS-CDMA based on interpolated fir filters, *IEEE Commun. Lett.* 9 (3) (2005) 213–215.
- [337] R.C. de Lamare, R. Sampaio-Neto, Reduced-rank adaptive filtering based on joint iterative optimization of adaptive filters, *IEEE Signal Process. Lett.* 14 (12) (2007) 980–983.
- [338] R.C. de Lamare, R. Sampaio-Neto, Adaptive reduced-rank processing based on joint and iterative interpolation, decimation, and filtering, *IEEE Trans. Signal Process.* 57 (7) (2009) 2503–2514.
- [339] M. Yukawa, R. de Lamare, R. Sampaio-Neto, Efficient acoustic echo cancellation with reduced-rank adaptive filtering based on selective decimation and adaptive interpolation, *IEEE Trans. Speech Audio Process.* 16 (4) (2008) 696–710.

# Introduction to Machine Learning

# 13

Johan A.K. Suykens

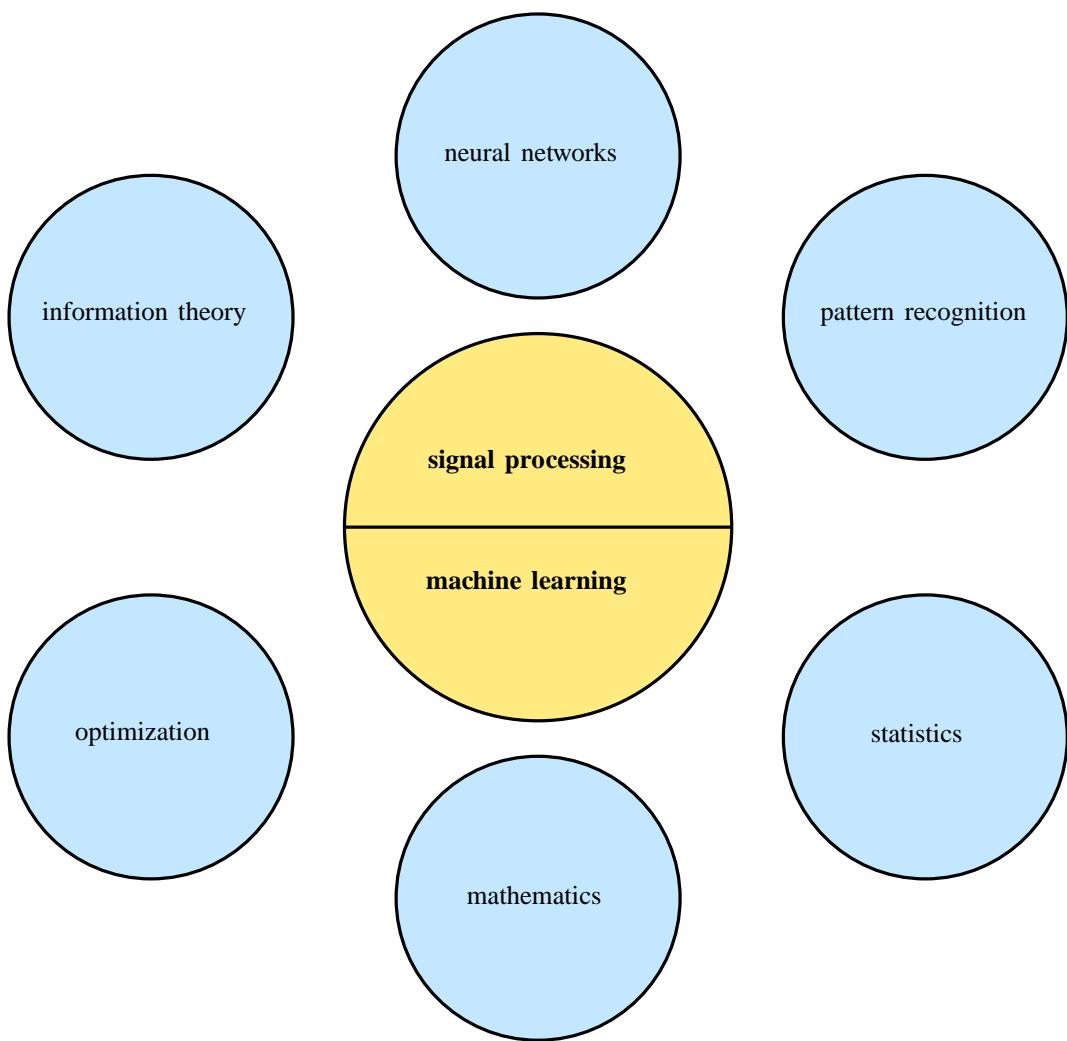
KU Leuven, ESAT-SCD/SISTA, Leuven (Heverlee), Belgium

## 1.13.1 Scope and context

Classically, within the signal processing community, linear parametric models have been a method of first choice in several applications. Historically, many computationally efficient algorithms have been developed for on-line and adaptive signal processing with e.g., LMS, recursive least squares and Kalman filtering type algorithms [1]. However, more recently considerable progress has been made also on the use of flexible nonlinear models, e.g., related to kernel methods, support vector machines [2–10] and probabilistic models [11–16], and the importance of regularization techniques has been realized both in the context of parametric models and non-parametric models. This is witnessed also by the progress in the area of compressed sensing and sparse models [17–22]. Moreover, many emerging applications in e.g., big data, networks applications, bioinformatics, brain-machine interfaces, are posing new challenges for predictive models towards handling large amounts of data in high dimensional input spaces. In this Machine Learning Section we therefore take a broad view on the subject of signal processing and machine learning in connection also to other related areas as pattern recognition and neural networks [23–25], mathematics and statistics [3,26,27], optimization [28,29], and information theory [30] (Figure 13.1).

In general one distinguishes between different types of learning models, such as supervised, unsupervised and semi-supervised learning [9,31,32], various tasks such as e.g., classification, regression, clustering and different types of models, including e.g., linear and nonlinear parametric models, kernel-based models, and probabilistic models (Figure 13.2). For many of the successful methods it is interesting to trace back to the original roots. For on-line learning of linear models in classification problems, the perceptron has originally served as a paradigm. However, soon one has encountered its limitations. In the neural networks area this led to introducing one or more hidden layers with multilayer perceptron neural networks. Backpropagation as the original learning algorithm for such feedforward networks, in its on-line learning form, could be interpreted as an extension of the LMS algorithm as used in adaptive signal processing [33]. On a different track, the perceptron has also been studied within the context of statistical learning theory [9,34,35]. Here one is interested in characterizing the generalization error of the model, which is typically expressed in terms of the error on the training data and a complexity term.

Multilayer perceptrons are universal approximators [36] which make them powerful tools to parameterize nonlinear functions. In order to overcome the problem of overfitting with flexible nonlinear models, an important technique to use is regularization [15,23]. In the objective function one not only

**FIGURE 13.1**

Signal processing and machine learning and several related areas.

minimizes then the error on the training data but one also keeps the estimated parameter values small. This leads to the notion of effective number of parameters which is relevant then to characterize the model complexity, instead of the number of parameters. The flexibility of the model is controlled by the regularization term. In a Bayesian inference and probabilistic modeling picture the regularization term corresponds then to the prior distribution on the unknown parameters. Classical regularization schemes minimize the  $\ell_2$  norm on the unknown parameters, which is known as ridge regression in statistics

Learning modes	Tasks	Models
supervised learning	regression	linear parametric
unsupervised learning	classification	non-linear parametric
semi-supervised learning	clustering	polynomial model
reinforcement learning	density estimation	multilayer perceptron
inductive learning	component analysis	radial basis function network
transductive learning	dimensionality reduction	splines
ensemble learning	data visualization	kernel-based model
transfer learning	manifold learning	support vector machines
	structure/feature selection	graphical models
	multi-task learning	probabilistic models
	dynamical systems modelling	
	time-series analysis	mixture models

**FIGURE 13.2**

Learning modes, learning tasks and examples of different possible models.

and dates back also to ill-posed problems and Tikhonov regularization [37]. In recent years there has been considerable interest in alternative regularization schemes based on  $\ell_0$ ,  $\ell_1$ , and  $\ell_p$  regularization to achieve sparsity in the solution vector (Figure 13.3), in connection also to compressed sensing [17–20,22].

Regularization also plays an important role in non-parametric and kernel-based models. The use of positive definite kernels and reproducing kernels dates back to the early work of Mercer [38], Moore [39], Aronszajn [40] and are key ingredients within methods of function estimation in reproducing kernel Hilbert spaces, the theory of splines and radial basis function networks [10,41]. Early use of reproducing kernel Hilbert spaces in signal processing is e.g., [42–44]. In Gaussian processes the kernel function relates to the correlation function [14,16]. An increasing and renewed interest in kernel-based methods appeared with the introduction of nonlinear support vector machines for classification and regression [9]. The use of a positive definite kernel is viewed here in connection to a feature map (often called the kernel trick, which relates to the Mercer theorem), where in the primal a constrained optimization problem formulation is given on the model that is expressed in terms of the feature map. The Lagrange dual problem results then into a kernel-based model representation. In standard support

Regularization	
Parametric	Kernel-based
$\ell_2$ , ridge regression	RKHS function estimation
$\ell_1$ , LASSO	splines
$\ell_p$ ( $0 < p \leq 1$ )	regularization networks
group LASSO	Gaussian processes
elastic net	support vector machines
spectral regularization	LS-SVMs
nuclear norm	

**FIGURE 13.3**

Regularization and its role in parametric and non-parametric modeling approaches.

vector machines a sparse kernel-based model is then achieved through the choice of the loss function, typically the hinge loss in classification and the epsilon-insensitive loss function in function estimation.

The kernel trick on its own has also been frequently employed to obtain nonlinear kernel versions of classically known linear estimation schemes, e.g., kernel principal component analysis [45] as an extension to the classical linear principal component analysis [46]. Special kernel functions have also been designed to handle specific data types or in specific applications area such as e.g., text mining or bioinformatics [5,6,47]. It is also possible to relate kernel functions to probabilistic graphical models and graphs. In least squares support vector machines one works with simple core models within the primal-dual setting for a wide range of problems in supervised and unsupervised learning and beyond [8,48]. The primal representation relates then to parametric picture, while the dual representation to a non-parametric. Depending on the nature of the given problem (large number of data versus dimensionality of the input space) this choice in representation can be exploited for developing efficient large scale algorithms [8,48].

An advantage of support vector machines for classification and regression is that the problem is recasted as a convex optimization problem, up to a small amount of tuning parameters of regularization constant(s) and kernel parameter(s). This has been viewed as a considerable advantage over other

nonlinear models such as multilayer perceptrons which suffer from the existence of many local minima solutions. Also towards sparse models and compressed sensing, convex optimization is playing an important role [29] (Figure 13.3). In many emerging applications one often has to cope with large amounts of data in often high-dimensional input spaces. This is posing new challenges for scalable optimization algorithms. In this direction efficient first order methods, on-line optimization, stochastic optimization or distributed optimization are suitable possible algorithms [49,50].

In the next section a brief overview is given on the chapter contributions that present introductory and tutorial contributions related to Signal Processing and Machine Learning.

### 1.13.2 Contributions

In [51] the authors present an overview of learning theory including statistical and computational aspects, with emphasis on classification and regression problems. Empirical risk minimization is discussed and concepts for characterizing the generalization performance of the model such as Rademacher complexity, covering numbers, Vapnik-Chervonenkis and fat shattering dimension. In connection to this, the problem of model selection is addressed.

In [52] an overview is presented on different types of neural networks for supervised and unsupervised learning. Starting from the perceptron, feedforward networks and backpropagation is explained. Next recurrent neural networks and recursive structure processing are discussed. Neural architectures for principal component analysis and topographic mapping for data mining and data visualization is outlined.

In [53] the authors give an introduction to the foundations and implementations of kernel methods, computational issues and recent developments. This includes the kernel trick, properties and types of kernels, kernel principal component analysis, kernel canonical correlation analysis, kernel Fisher discriminant analysis, support vector machines for classification and regression, and Gaussian processes.

In [54] on-line learning in reproducing kernel Hilbert spaces is presented. First parameter estimation is discussed in regression and classification tasks and how to overcome overfitting by applying regularization. It is explained how a nonlinear task can be mapped to a linear task. In this way kernel LMS and complex kernel LMS are extended to kernel versions of the well-known LMS algorithm in signal processing. For least squares learning algorithms extensions to kernel recursive least squares are discussed. Finally, convex analysis concepts for online learning are provided.

In [55] an introduction to probabilistic graphical models is given. It includes three representations of probabilistic graphical models: Markov networks (or undirected graphical models), Bayesian networks (or directed graphical models) and factor graphs. An overview about structure and parameter learning techniques is given on maximum likelihood and Bayesian learning, and generative and discriminative learning. Exact inference methods and approximate inference techniques are addressed. Applications for each of the three representations are given: Bayesian networks for expert systems, dynamic Bayesian networks for speech processing, Markov random fields for image processing, and factor graphs for decoding error-correcting codes.

In [56] a tutorial introduction to Monte Carlo Methods, Markov Chain Monte Carlo and Particle Filtering is given. Starting from the Monte Carlo principle and basic techniques for simulating and transforming random variables, Markov Chain Monte Carlo is explained. Other topics that are addressed are

rejection sampling, detailed balance, the Gibbs sampler, sequential Monte Carlo, importance sampling, resampling, and advanced Monte Carlo methods.

In [57] an introduction to clustering is given. Different clustering algorithms are discussed including hierarchical clustering, the K-means algorithm, fuzzy C-means algorithm, mixture density-based clustering, neural network-based clustering based on adaptive resonance theory, spectral clustering, subspace clustering and biclustering, and deep learning clustering.

In [58] unsupervised learning algorithms and latent variable models are presented. Basic linear and multilinear models for matrix and tensor factorizations and decompositions are discussed. Constrained matrix and tensor decompositions for sparse representation of data and their extensions are addressed. Various constraints such as orthogonality, statistical independence, nonnegativity, and/or sparsity are explained. The importance of matrix/tensor decompositions is given for blind source separation, dimensionality reduction, pattern recognition, object detection, classification, multiway clustering, sparse representation and coding, and data fusion.

In [59] an introduction is presented on semi-supervised learning. Discussed topics include transductive support vector machine and low density separation, co-training and multi-view, co-regularization and expectation-maximization for mixture models. Finally graph-based semi-supervised learning is addressed with graph Laplacian regularization, manifold regularization, measure-based regularization, and semi-supervised learning for structured outputs.

In [60] an overview is given on sparsity-aware learning and compressed sensing. The Least Absolute Shrinkage and Selection Operator (LASSO), sparse signal representation,  $\ell_2$ ,  $\ell_0$ ,  $\ell_1$  norm minimizers and their geometric interpretation are discussed. In view of conditions for equivalence of the  $\ell_0$  and  $\ell_1$  minimizer, mutual coherence and the Restricted Isometry Property (RIP) is explained. Robust sparse signal recovery from noisy measurements and compressed sensing is covered. Sparsity-promoting algorithms are discussed like Orthogonal Matching Pursuit, the Least Angle Regression (LARS) algorithm and Iterative Shrinkage Algorithms. A case study on time-frequency analysis is provided.

In [61] the authors present information based learning approaches. Starting from information theoretic descriptors as entropy, divergence and mutual information, a unifying information theoretic framework for machine learning is outlined. Filtering, classification, feature extraction and nonparametric information estimators are discussed. Next a reproducing kernel Hilbert space framework for information based learning is proposed. Illustrative examples are given on adaptive system training, classification, information cut for clustering, and independent component analysis.

In [62] model selection aspects are discussed. The Akaike information criterion and the Kullback information criterion are explained with linear regression as an example application. Then consistency and efficiency are addressed. Other topics that are included are Bayesian approaches to model selection, the Bayesian information criterion, Markov-Chain Monte-Carlo Bayesian methods, model selection by compression, minimum message length, model selection consistency, parameter estimation consistency, and sequential variants of minimum description length.

In [63] an overview is given on music mining. Topics that are addressed include ground truth acquisition and evaluation, audio feature extraction, extracting context information about music, content-based similarity retrieval, genre classification, emotion/mood classification, music clustering, automatic tag annotation, audio fingerprinting, and cover song detection.

## Acknowledgments

The author acknowledges support from KU Leuven, the Flemish Government, FWO, the Belgian Federal Science Policy Office, and the European Research Council (ERC AdG A-DATADRIVE-B, CoE EF/05/006, GOA MANET, IUAP DYSCO, FWO G.0377.12, POM II, Cost IntelliCIS, iMinds Future Health Department).

## References

- [1] S. Haykin, Adaptive Filter Theory, third Ed., Prentice-Hall, 1996.
- [2] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.
- [3] F. Cucker, D.-X. Zhou, Learning Theory: An Approximation Theory Viewpoint, Cambridge University Press, 2007.
- [4] W. Liu, J. Principe, S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, Wiley, Hoboken, New Jersey, 2010.
- [5] B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
- [6] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, June 2004.
- [7] I. Steinwart, A. Christmann, Support Vector Machines, Springer, New York, 2008.
- [8] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002.
- [9] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
- [10] G. Wahba, Spline Models for Observational Data, Series in Applied Mathematics, vol. 59, SIAM, Philadelphia, 1990.
- [11] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- [12] M.I. Jordan, Learning in Graphical Models, MIT Press, 1999.
- [13] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
- [14] D.J.C. MacKay, Introduction to Gaussian processes, in: C.M. Bishop (Ed.), Neural networks and machine learning, Springer NATO-ASI Series F: Computer and Systems Sciences, vol. 168, 1998, pp. 133–165.
- [15] D.J.C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
- [16] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.
- [17] R.G. Baraniuk, V. Cevher, M.F. Duarte, C. Hegde, Model-based compressive sensing, IEEE Trans. Inform. Theory 56 (4) (2010) 1982–2001.
- [18] A.M. Bruckstein, D.L. Donoho, M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, SIAM Rev. 51 (1) (2009) 34–81.
- [19] E.J. Candes, J. Romberg, T. Tao, Stable recovery from incomplete and inaccurate measurements, Commun. Pure Appl. Math. 59 (8) (2006) 1207–1223.
- [20] E.J. Candes, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
- [21] S. Chen, D.L. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.
- [22] D.L. Donoho, M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via  $\ell^1$  minimization, in: Proceedings of National Academy Sciences, 2003, pp. 2197–2202.
- [23] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

- [24] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, second ed., John Wiley & Sons, New York, 2001.
- [25] S. Theodoridis, K., *Pattern Recognition*, Academic Press, Koutroumbas, 2009.
- [26] F. Cucker, S. Smale, On the mathematical foundations of learning theory, *Bull. AMS* 39 (2002) 1–49.
- [27] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2001.
- [28] H.H. Bauschke, P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.
- [29] S. Boyd, L, *Convex Optimization*, Cambridge University Press, Vandenberghe, 2004.
- [30] J.C. Principe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*, Springer, 2010.
- [31] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, *J. Mach. Learn. Res.* 7 (2006) 2399–2434.
- [32] O. Chapelle, B. Schölkopf, A. Zien (Eds.), *Semi-Supervised Learning*, MIT Press, 2006.
- [33] B. Widrow, R.G. Winter, Neural nets for adaptive filtering and adaptive pattern recognition, *IEEE Comput. Mag.* 21 (3) (1988) 25–39.
- [34] V. Vapnik, A. Lerner, Pattern recognition using generalized portrait method, *Autom. Remote Control* 24 (1963) 774–780.
- [35] V. Vapnik, A. Chervonenkis, A note on one class of perceptrons, *Autom. Remote Control* 25 1964.
- [36] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.
- [37] A.N. Tikhonov, V.Y. Arsenin, *Solution of Ill-Posed Problems*, Winston, Washington, DC, 1977.
- [38] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, *Philos. Trans. Roy. Soc. Lond.* 209 (1909) 415–446.
- [39] E.H. Moore, On properly positive Hermitian matrices, *Bull. Am. Math. Soc.* 23 (1916) 59.
- [40] N. Aronszajn, Theory of reproducing kernels, *Trans. Am. Math. Soc.* 68 (1950) 337–404.
- [41] T. Poggio, F. Girosi, Networks for approximation and learning, *Proc. IEEE* 78 (9) (1990) 1481–1497.
- [42] T. Kailath, RKHS approach to detection and estimation problems: Part I: Deterministic signals in Gaussian noise, *IEEE Trans. Inform. Theory* 17 (5) (1971) 530–549.
- [43] T. Kailath, A view of three decades of linear filtering theory, *IEEE Trans. Inform. Theory* 20 (2) (1974) 146–181.
- [44] E. Parzen, Statistical inference on time series by RKHS methods, Department of Statistical Stanford University Technical Report 14, January 1970.
- [45] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Comput.* 10 (1998) 1299–1319.
- [46] I.T. Jolliffe, *Principal Component Analysis*, Springer Series in Statistics, Springer-Verlag, 1986.
- [47] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, S.V.N. Vishwanathan (Eds.), *Predicting Structured Data*, MIT Press, Cambridge, MA, 2007.
- [48] J.A.K. Suykens, C. Alzate, K. Pelckmans, Primal and dual model representations in Kernel-based learning, *Stat. Surv.* 4 (2010) 148–183.
- [49] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Mach. Learn.* 3 (1) (2011) 1–122.
- [50] Y.E. Nesterov, A method for solving the convex programming problem with convergence rate  $O(1/k^2)$ , *Dokl. Akad. Nauk SSSR* 269 (1983) 543–547 (in Russian).
- [51] A. Tewari, P.L. Bartlett, Learning theory, in: S. Theodoridis, R. Chellappa (Eds.), *Library in Signal Processing*, vol. 1, Academic Press, 2013.
- [52] B. Hammer, Neural networks, in: S. Theodoridis, R. Chellappa (Eds.), *Library in Signal Processing*, vol. 1, Academic Press, 2013.
- [53] J. Shawe-Taylor, S. Sun, Kernel methods and support vector machines, in: S. Theodoridis, R. Chellappa (Eds.), *Library in Signal Processing*, vol. 1, Academic Press, 2013.

- [54] K. Slavakis, P. Bouboulis, S. Theodoridis, Online learning in reproducing Kernel Hilbert spaces, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [55] F. Pernkopf, R. Peharz, S. Tschiatschek, Introduction to probabilistic graphical models, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [56] A.T. Cemgil, A tutorial introduction to Monte Carlo methods, Markov Chain Monte Carlo and particle filtering, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [57] D. Lam, D.C. Wunsch, Clustering, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [58] A. Cichocki, Unsupervised learning algorithms and latent variable models: PCA/SVD, CCA/PLS, ICA, NMF, etc., in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [59] X. Zhou, M. Belkin, Semi-supervised learning, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [60] S. Theodoridis, Y. Kopsinis, K. Slavakis, Sparsity-aware learning and compressed sensing: an overview, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [61] J.C. Principe, B. Chen, L.G. Sanchez Giraldo, Information based learning, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [62] E. Makalic, D.F. Schmidt, A.-K. Seghouane, A tutorial on model selection, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.
- [63] G. Tzanetakis, Music mining, in: S. Theodoridis, R. Chellappa (Eds.), Library in Signal Processing, vol. 1, Academic Press, 2013.

# Learning Theory

# 14

Ambuj Tewari\* and Peter L. Bartlett†

\*439 West Hall 1085 South University Ann Arbor, MI, USA

†387 Soda Hall #1776 Berkeley, CA, USA

## 1.14.1 Introduction

In a section on *Learning Machines* in one of his most famous papers [1], Turing wrote:

*Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationer's. Rather little mechanism, and lots of blank sheets. (Mechanism and writing are from our point of view almost synonymous.) Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.*

The year was 1950. Earlier, in 1943, McCulloch and Pitts [2] had already hit upon the idea that mind-like machines could be built by studying how biological nerve cells behave. This directly lead to neural networks. By the late 1950s, Samuels [3] had programmed a machine to play checkers and Rosenblatt [4] had proposed one of earliest models of a learning machine, the perceptron. The importance of the perceptron in the history of learning theory cannot be overemphasized. Vapnik says that the appearance of the perceptron was “when the mathematical analysis of learning processes truly began.” Novikoff’s perceptron theorem (we will prove it in Section 1.14.6.5.2) was also proved in 1962 [5].

Vapnik [6] points out the interesting fact that the 1960s saw four major developments that were to have lasting influence on learning theory. First, Tikhonov and other developed regularization theory for the solution of ill posed inverse problems. Regularization theory has had and continues to have tremendous impact on learning theory. Second, Rosenblatt, Parzen, and Chentsov pioneered nonparametric methods for density estimation. These beginnings were crucial for a distribution free theory of non-parametric classification and regression to emerge later. Third, Vapnik and Chervonenkis proved the uniform law of large numbers for indicators of sets. Fourth, Solomonoff, Kolmogorov, and Chaitin all independently discovered algorithmic complexity: the idea that the complexity of a string of 0's and 1's can be defined by the length of the shortest program that can generate that string. This later led Rissanen to propose his Minimum Description Length (MDL) principle.

In 1986, a major technique to learn weights in neural networks was discovered: backpropagation [7]. Around the same time, Valiant [8] published his paper introducing a computational model of learning that was to be called the PAC (probably approximately correct) model later. The 1990s saw the appearance of support vector machines [9] and AdaBoost [10]. This brings us to the brink of the second millennium which is where we end our short history tour.

As we can see, learning theory has absorbed influences from a variety of sources: cybernetics, neuroscience, nonparametric statistics, regularization theory of inverse problems, and the theory of computation and algorithms. Ongoing research is not only trying to overcome known limitations of classic models but is also grappling with new issues such as dealing with privacy and web-scale data.

Learning theory is a formal mathematical theory. Assumptions are made, beautiful lemmas are proved, and deep theorems are discovered. But developments in learning theory also lead to practical algorithms. SVMs and AdaBoost are prime examples of high impact learning algorithms evolving out of a principled and well-founded approach to learning. We remain hopeful that learning theory will help us discover even more exciting learning algorithms in the future.

Finally, a word about the mathematical background needed to read this overview of learning theory. Previous exposure to mathematical reasoning and proofs is a must. So is familiarity with probability theory. But measure theory is not needed as we will avoid all measure-theoretic details. It will also be helpful, but not essential, to be aware of the basics of computational complexity such as the notion of polynomial time computation and NP-completeness. Similarly, knowing a little bit of functional analysis will help.

### 1.14.2 Probabilistic formulation of learning problems

A basic problem in learning theory is that of learning a functional dependence  $f : \mathcal{X} \rightarrow \mathcal{Y}$  from some *input space*  $\mathcal{X}$  to some *output* or *label* space  $\mathcal{Y}$ . Some special choices for the output space deserve particular attention since they arise often in applications. In *multiclass classification*, we have  $\mathcal{Y} = \{1, \dots, K\}$  and the goal is to classify any input  $x \in \mathcal{X}$  into one of  $K$  given classes. A special case of this problem is when  $K = 2$ . Then, the problem is called *binary classification* and it is mathematically convenient to think of the output space as  $\mathcal{Y} = \{-1, +1\}$  rather than  $\mathcal{Y} = \{0, 1\}$ . In *regression*, the output space  $\mathcal{Y}$  is some subset of the set  $\mathbb{R}$  of real numbers. Usually  $\mathcal{Y}$  is assumed to be a bounded interval, say  $[-1, +1]$ .

We assume that there is an underlying distribution  $P$  over  $\mathcal{X} \times \mathcal{Y}$  and that the learner has access to  $n$  samples

$$(X_1, Y_1), \dots, (X_n, Y_n),$$

where the  $(X_i, Y_i)$  are independent and identically distributed (or *iid*) random variables with the same distribution  $P$ . Note that  $P$  itself is assumed to be unknown to the learner. A *learning rule* or *learning algorithm*  $\hat{f}_n$  is simply a mapping

$$\hat{f}_n : (\mathcal{X} \times \mathcal{Y})^n \times \mathcal{X} \rightarrow \mathcal{Y}.$$

That is, a learning rule maps the  $n$  samples to a function from  $\mathcal{X}$  to  $\mathcal{Y}$ . When the sample is clear from context, we will denote the function returned by a learning rule itself by  $\hat{f}_n$ .

### 1.14.2.1 Loss function and risk

In order to measure the predictive performance of a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , we use a *loss function*. A loss function  $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$  is a non-negative function that quantifies how bad the prediction  $f(x)$  is if the true label is  $y$ . We say that  $\ell(f(x), y)$  is the loss incurred by  $f$  on the pair  $(x, y)$ . In the classification case, binary or otherwise, a natural loss function is the 0–1 loss:

$$\ell(y', y) = \mathbf{1}[y' \neq y].$$

The 0–1 loss function, as the name suggests, is 1 if a misclassification happens and is 0 otherwise. For regression problems, some natural choices for the loss function are the *squared loss*:

$$\ell(y', y) = (y' - y)^2,$$

or the *absolute loss*:

$$\ell(y', y) = |y' - y|.$$

Given a loss function, we can define the *risk* of a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  as its expected loss under the true underlying distribution:

$$R(f) = \mathbb{E}_{(X, Y) \sim P}[\ell(f(X), Y)].$$

Note that the risk of any function  $f$  is not directly accessible to the learner who only sees the samples. But the samples can be used to calculate the *empirical risk* of  $f$ :

$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^n [\ell(f(X_i), Y_i)].$$

Minimizing the empirical risk over a fixed class  $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$  of functions leads to a very important learning rule, namely *empirical risk minimization* (ERM):

$$\hat{f}_n = \operatorname{argmin}_{f \in \mathcal{F}} \widehat{R}(f).$$

Under appropriate conditions on  $\mathcal{X}$ ,  $\mathcal{Y}$ , and  $\mathcal{F}$ , an empirical risk minimizer is guaranteed to exist though it will not be unique in general. If we knew the distribution  $P$  then the best function from  $\mathcal{F}$  would be

$$f_{\mathcal{F}}^* = \operatorname{argmin}_{f \in \mathcal{F}} R(f),$$

as  $f_{\mathcal{F}}^*$  minimizes the expected loss on a random  $(X, Y)$  pair drawn from  $P$ . Without restricting ourselves to the class  $\mathcal{F}$ , the best possible function to use is

$$f^* = \operatorname{argmin}_f R(f),$$

where the infimum is with respect to all<sup>1</sup> functions.

---

<sup>1</sup>We should say “all measurable functions” here. We will, however, ignore measurability issues in this overview article.

For classification with 0–1 loss,  $f^*$  is called the *Bayes classifier* and is given simply by

$$f^*(x) = \operatorname{argmax}_{y \in \mathcal{Y}} P(Y = y | X = x).$$

Note that  $f^*$  depends on the underlying distribution  $P$ . If we knew  $P$ , we could achieve minimum misclassification probability by always predicting, for a given  $x$ , the label  $y$  with the highest conditional probability (ties can be broken arbitrarily). The following result bounds the *excess risk*  $R(f) - R(f^*)$  of any function  $f$  in the binary classification case.

**Theorem 1.** *For binary classification with 0–1 loss, we have*

$$R(f) - R(f^*) = \mathbb{E}[\mathbf{1}[f(X) \neq f^*(X)]|2\eta(X) - 1|],$$

where  $\eta(x) = P(Y = +1 | X = x)$ .

For the regression case with squared loss,  $f^*$  is called the *regression function* and is given simply by the conditional expectation of the label given  $x$ :

$$f^*(x) = \mathbb{E}[Y | X = x].$$

In this case, the excess risk takes a particularly nice form.

**Theorem 2.** *For regression with squared loss, we have*

$$R(f) - R(f^*) = \mathbb{E}[|f(X) - f^*(X)|^2].$$

### 1.14.2.2 Universal consistency

A particularly nice property for a learning rule  $\hat{f}_n$  to have is that its (expected) excess risk converges to zero as the sample size  $n$  goes to infinity. That is,

$$\mathbb{E}[R(\hat{f}_n)] - R(f^*) \rightarrow 0.$$

Note  $R(\hat{f}_n)$  is a random variable since  $\hat{f}_n$  depends on a randomly drawn sample. A rule  $\hat{f}_n$  is said to be *universally consistent* if the above convergence holds irrespective of the true underlying distribution  $P$ . This notion of consistency is also sometimes called *weak universal consistency* in order to distinguish it from *strong universal consistency* that demands

$$R(\hat{f}_n) - R(f^*) \rightarrow 0$$

with probability 1.

The existence of universally consistent learning rules for classification and regression is a highly non-trivial fact which has been known since Stone's work in the 1970s [11]. Proving universal consistency typically requires addressing the *estimation error* versus *approximation error* trade-off. For instance, often the function  $\hat{f}_n$  is guaranteed to lie in a function class  $\mathcal{F}_n$  that does not depend on the sample. The classes  $\mathcal{F}_n$  typically grow with  $n$ . Then we can decompose the excess risk of  $\hat{f}_n$  as

$$R(\hat{f}_n) - R(f^*) = \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}_n}^*)}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}_n}^*) - R(f^*)}_{\text{approximation error}}.$$

This decomposition is useful as the estimation error is the only random part. The approximation error depends on how rich the function class  $\mathcal{F}_n$  is but does not depend on the sample. The reason we have to trade these two errors off is that the richer the function class  $\mathcal{F}_n$ , the smaller the approximation error will be. However, that leads to a large estimation error. This trade-off is also known as the *bias-variance trade-off*. The bias-variance terminology comes from the regression with squared error case but tends to be used in the general loss setting as well.

### 1.14.2.3 Learnability with respect to a fixed class

Unlike universal consistency, which requires convergence to the minimum possible risk over all functions, learnability with respect to a fixed function class  $\mathcal{F}$  only demands that

$$\mathbb{E}[R(\hat{f}_n)] - R(f_{\mathcal{F}}^*) \rightarrow 0.$$

We can additionally require quantitative bounds on the number of samples needed to reach, with high probability, a given upper bound on excess risk relative to  $f_{\mathcal{F}}^*$ . The *sample complexity* of a learning rule  $\hat{f}_n$  is the minimum number  $n_0(\varepsilon, \delta)$  such that for all  $n \geq n_0(\varepsilon, \delta)$ , we have

$$R(\hat{f}_n) - R(f_{\mathcal{F}}^*) \leq \varepsilon$$

with probability at least  $1 - \delta$ . We now see how we can arrive at sample complexity bounds for ERM via *uniform convergence* of empirical means to true expectations.

We mention some examples of functions classes that arise often in practice.

*Linear Functions:* This is one of the simplest functions classes. In any finite dimensional Euclidean space  $\mathbb{R}^d$ , define the class of real valued functions

$$\mathcal{F} = \{x \mapsto \langle w, x \rangle : w \in \mathcal{W} \subseteq \mathbb{R}^d\},$$

where  $\langle w, x \rangle = \sum_{j=1}^d w_j x_j$  denotes the standard *inner product*. The *weight vector* might be additionally constrained. For instance, we may require  $\|w\| \leq W$  for some norm  $\|\cdot\|$ . The set  $\mathcal{W}$  takes care of such constraints.

*Linear Threshold Functions or Halfspaces:* This is class of binary valued function obtained by thresholding linear functions at zero. This gives us the class

$$\mathcal{F} = \{x \mapsto \text{sign}(\langle w, x \rangle) : w \in \mathcal{W} \subseteq \mathbb{R}^d\}.$$

*Reproducing Kernel Hilbert Spaces:* Linear functions can be quite restricted in their approximation ability. However, we can get a significant improvement by first mapping the inputs  $x \in \mathcal{X}$  into a feature space through a *feature mapping*  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$  where  $\mathcal{H}$  is a very high dimensional Euclidean space (or an infinite dimensional *Hilbert space*) and then considering linear functions in the feature space:

$$\mathcal{F} = \{x \mapsto \langle w, \Phi(x) \rangle_{\mathcal{H}} : w \in \mathcal{W}\},$$

where the inner product now carries the subscript  $\mathcal{H}$  to emphasize the space where it operates. Many algorithms access the inputs only through pairwise inner products

$$\langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{H}}.$$

In such cases, we do not care what the feature mapping is or how complicated it is to evaluate, provided we can compute these inner products efficiently.

This idea of using only inner products leads us to reproducing kernel Hilbert space. We say that a function  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  is symmetric positive semidefinite (psd) if it satisfies the following two properties.

*Symmetry:* For any  $x, x' \in \mathcal{X}$ , we have  $K(x, x') = K(x', x)$ .

*Positive Semidefiniteness:* For any  $(x_1, \dots, x_n) \in \mathcal{X}^n$ , the symmetric matrix

$$\mathbf{K} = (K(x_i, x_j))_{i,j} \in \mathbb{R}^{n \times n}$$

is psd.<sup>2</sup> The matrix  $\mathbf{K}$  is often called the *Gram matrix*.

Note that, given a feature mapping  $\Phi$ , it is trivial to verify that

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$$

is a symmetric psd function. However, the much less trivial converse is also true. Every symmetric psd function arises out of a feature mapping. This is a consequence of the *Moore-Aronszajn theorem*. To state the theorem, we first need a definition. A *reproducing kernel Hilbert space or RKHS*  $\mathcal{H}$  is a Hilbert space of functions  $f : \mathcal{X} \rightarrow \mathbb{R}$  such that there exists a kernel  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  satisfying the properties:

- For any  $x \in \mathcal{X}$ , the function  $K_x = K(x, \cdot)$  is in  $\mathcal{H}$ .
- For any  $x \in \mathcal{X}$ ,  $f \in \mathcal{H}$ , the *reproducing property* holds:  $f(x) = \langle f, K_x \rangle_{\mathcal{H}}$ .

**Theorem 3 (Moore-Aronszajn).** *Given a symmetric psd function  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ , there is a unique RKHS  $\mathcal{H}$  with reproducing kernel  $K$ .*

This theorem gives us the feature mapping easily. Starting from a symmetric psd function  $K$ , we use the Moore-Aronszajn theorem to get an RKHS  $\mathcal{H}$  whose reproducing kernel is  $K$ . Now, by the reproducing property, for any  $x, x' \in \mathcal{X}$ ,

$$K(x, x') = K_x(x') = \langle K_x, K_{x'} \rangle_{\mathcal{H}}$$

which means that  $\Phi : \mathcal{X} \mapsto \mathcal{H}$  given by  $\Phi(x) = K_x$  is a feature mapping we were looking for. Given this equivalence between symmetric psd functions and RKHSes, we will use the word *kernel* for a symmetric psd function itself.

*Convex Hulls:* This example is not about a function class but rather about a means of obtaining a new function class from an old one. Say, we have class  $\mathcal{F}$  of binary valued functions  $f : \mathcal{X} \rightarrow \{\pm 1\}$ . We can consider functions that take a *majority vote* over a subset of functions in  $\mathcal{F}$ . That is,

$$f'(x) = \text{sign} \left( \sum_{\ell=1}^L f_{\ell} \right), \quad f_{\ell} \in \mathcal{F}.$$

---

<sup>2</sup>Recall that a symmetric matrix  $\mathbf{K} \in \mathbb{R}^{n \times n}$  is psd iff for all  $u \in \mathbb{R}^n$ ,  $\langle u, \mathbf{K}u \rangle \geq 0$ .

More generally, we can assign weights  $w_\ell \geq 0$  to  $f_\ell$  and consider a weighted majority

$$f'(x) = \text{sign} \left( \sum_{\ell=1}^L w_\ell f_\ell \right).$$

Constraining the weights to sum to no more than  $W$  gives us the scaled convex hull

$$W \cdot \text{conv}(\mathcal{F} \cup \{0\}) = \left\{ \sum_{\ell=1}^L w_\ell f_\ell : L \in \mathbb{N}, w_\ell \geq 0, \sum_\ell w_\ell \leq W \right\}.$$

The class above can, of course, be thresholded to produce binary valued functions.

#### 1.14.2.4 ERM and uniform convergence

The excess risk of ERM relative to  $f_{\mathcal{F}}^*$  can be decomposed as

$$R(\hat{f}_n) - R(f_{\mathcal{F}}^*) = (R(\hat{f}_n) - \hat{R}(\hat{f}_n)) + (\hat{R}(\hat{f}_n) - \hat{R}(f_{\mathcal{F}}^*)) + (\hat{R}(f_{\mathcal{F}}^*) - R(f_{\mathcal{F}}^*)).$$

By the definition of ERM, the difference  $\hat{R}(\hat{f}_n) - \hat{R}(f_{\mathcal{F}}^*)$  is non-positive. The difference  $\hat{R}(f_{\mathcal{F}}^*) - R(f_{\mathcal{F}}^*)$  is also easy to deal with since it deals with a fixed (i.e. non-random) function  $f_{\mathcal{F}}^*$ . Using Hoeffding's inequality (see Section 1.14.3.1), we have, with probability at least  $1 - \delta$ ,

$$\hat{R}(f_{\mathcal{F}}^*) - R(f_{\mathcal{F}}^*) \leq \sqrt{\frac{\log(1/\delta)}{2n}}.$$

This simply reflects the fact that as the sample size  $n$  grows, we expect the empirical average of  $f_{\mathcal{F}}^*$  to converge to its true expectation.

The matter with the difference  $R(\hat{f}_n) - \hat{R}(\hat{f}_n)$  is not so simple since  $\hat{f}_n$  is a random function. But we do know  $\hat{f}_n$  lies in  $\mathcal{F}$ . So we can clearly bound

$$R(\hat{f}_n) - \hat{R}(\hat{f}_n) \leq \sup_{f \in \mathcal{F}} (R(f) - \hat{R}(f)). \quad (14.1)$$

This provides a *generalization bound* (or, more properly, a *generalization error bound*) for ERM. A generalization bound is an upper bound on the true expectation  $R(\hat{f}_n)$  of a learning rule in terms of its empirical performance  $\hat{R}(\hat{f}_n)$  (or some other performance measure computed from data). A generalization bound typically involves additional terms that measure the statistical complexity of the class of functions that the learning rule uses. Note the important fact that the bound above is valid not just for ERM but for any learning rule that outputs a function  $\hat{f}_n \in \mathcal{F}$ .

The measure of complexity in the bound (14.1) above is the maximum deviation between true expectation and empirical average over functions in  $\mathcal{F}$ . For each *fixed*  $f \in \mathcal{F}$ , we clearly have

$$R(f) - \hat{R}(f) \rightarrow 0$$

both in probability (weak law of large numbers) and almost surely (strong law of large numbers). But in order for

$$\sup_{f \in \mathcal{F}} R(f) - \hat{R}(f) \rightarrow 0$$

the law of large numbers has to hold *uniformly* over the function class  $\mathcal{F}$ .

To summarize, we have shown in this section that, with probability at least  $1 - \delta$ ,

$$R(\hat{f}_n) - R(f_{\mathcal{F}}^*) \leq \sup_{f \in \mathcal{F}} (R(f) - \hat{R}(f)) + \sqrt{\frac{\log(1/\delta)}{2n}}. \quad (14.2)$$

Thus, we can bound the excess risk of ERM if we can prove uniform convergence of empirical means to true expectations for the function class  $\mathcal{F}$ . This excess risk bound, unlike the generalization error bound (14.1), is applicable only to ERM.

### 1.14.2.5 Bibliographic note

There are several book length treatments of learning theory that emphasize different aspects of the subject. The reader may wish to consult Anthony and Biggs [12], Anthony and Bartlett [13], Devroye et al. [14], Vapnik [15], Vapnik [6], Vapnik [16], Vidyasagar [17], Natarajan [18], Cucker and Zhou [19], Kearns and Vazirani [20], Cesa-Bianchi and Lugosi [21], Györfi et al. [22], and Hastie et al. [23].

## 1.14.3 Uniform convergence of empirical means

Let us consider the simplest case of a finite function class  $\mathcal{F}$ . In this case, Hoeffding's inequality (see Theorem 5) gives us for any  $f \in F$ , with probability at least  $1 - \delta/|\mathcal{F}|$ ,

$$R(f) - \hat{R}(f) \leq c \sqrt{\frac{\log(|\mathcal{F}|/\delta)}{n}}.$$

Therefore, denoting the event that the above inequality fails to hold for function  $f$  by  $A_f$ , we have

$$P(\exists f \in \mathcal{F}, A_f \text{ holds}) \leq \sum_{f \in \mathcal{F}} P(A_f) \leq |\mathcal{F}| \cdot \frac{\delta}{|\mathcal{F}|} = \delta.$$

The first inequality is called a *union bound* in probability theory. It is exact if and only if all the events it is applied to are pairwise disjoint, i.e.,  $A_f \cap A_{f'} = \emptyset$  for  $f \neq f'$ . Clearly, if there are functions that are “similar” to each other, the bound above will be loose. Nevertheless, the union bound is an important starting point for the analysis of uniform convergence of empirical means and for a finite class, it gives us the following result.

**Theorem 4.** *Consider a finite class  $\mathcal{F}$  and a loss function bounded by 1. Then, we have, with probability at least  $1 - \delta$ ,*

$$\sup_f (R(f) - \hat{R}(f)) \leq \sqrt{\frac{\log |\mathcal{F}| + \log(1/\delta)}{2n}}.$$

Therefore, using (14.2), we have the following result for ERM. With probability at least  $1 - 2\delta$ ,

$$R(\hat{f}_n) - R(f_{\mathcal{F}}^*) \leq \sqrt{\frac{\log |\mathcal{F}|}{2n}} + \sqrt{\frac{2 \log(1/\delta)}{n}}.$$

It is instructive to focus on the first term on the right hand side. First, it decreases as  $n$  increases. This means that ERM gets better as it works with more data. Second, it increases as the function class  $\mathcal{F}$  grows in size. But the dependence is logarithmic and hence mild.

However, the bound above is useless if  $|\mathcal{F}|$  is not finite. The goal of the theory we will build below (often called “VC theory” after its architects Vapnik and Chervonenkis) will be to replace  $\log |\mathcal{F}|$  with an appropriate measure of the *capacity* of an infinite function class  $\mathcal{F}$ .

### 1.14.3.1 Concentration inequalities

Learning theory uses *concentration inequalities* heavily. A typical concentration inequality will state that a certain random variable is tightly concentrated around its mean (or in some cases median). That is, the probability that they deviate far enough from the mean is small.

The following concentration inequality, called *Hoeffding’s inequality*, applies to sums of iid random variables.

**Theorem 5.** *Let  $Z_i \in [0, 1]$  be bounded iid random variables with common expectation  $\mu$ . Then we have, for  $\varepsilon > 0$*

$$\mathbb{P} \left[ \mu - \frac{1}{n} \sum_{i=1}^n Z_i > \varepsilon \right] \leq \exp(-2n\varepsilon^2).$$

This can be restated as follows. For any  $\delta \in (0, 1)$ ,

$$\mathbb{P} \left[ \mu - \frac{1}{n} \sum_{i=1}^n Z_i \leq \sqrt{\frac{\log(1/\delta)}{2n}} \right] > 1 - \delta.$$

A couple of remarks are in order. First, even though we have stated Hoeffding’s inequality as applying to bounded iid random variables, only the boundedness and independence are really needed. It is possible to let the  $Z_i$ ’s have different distributions. Second, the above result provides probability upper bounds for deviations below the mean  $\mu$  but a similar result holds for deviations of  $\frac{1}{n} \sum_{i=1}^n Z_i$  above  $\mu$  as well.

In fact, Hoeffding’s inequality can be considered a special case of the following inequality that is often called *McDiarmid’s inequality* or the *bounded differences inequality*.

**Theorem 6.** *Let  $F : \mathcal{X}^n \rightarrow \mathbb{R}$  be a function that satisfies the bounded differences property: for all  $i \in [n]$  and  $x_{1:n} \in \mathcal{X}^n$ ,  $x'_i \in \mathcal{X}$ ,*

$$|f(x_1, \dots, x_i, \dots, x_n) - f(x_1, \dots, x'_i, \dots, x_n)| \leq c_i$$

for some constants  $c_i$ . Then, for any independent  $X_1, \dots, X_n$ , the random variable  $Z = f(X_1, \dots, X_n)$  satisfies,

$$\mathbb{P}[Z - \mathbb{E}Z \geq \varepsilon] \leq \exp \left( -\frac{2\varepsilon^2}{\sum_{i=1}^n c_i^2} \right)$$

as well as

$$\mathbb{P}[Z - \mathbb{E}Z \leq -\varepsilon] \leq \exp \left( -\frac{2\varepsilon^2}{\sum_{i=1}^n c_i^2} \right)$$

for any  $\varepsilon > 0$ .

### 1.14.3.2 Rademacher complexity

Let us recall that we need to control the following quantity:

$$\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f).$$

Since this satisfies the conditions of McDiarmid's inequality with  $c_i = 1/n$ , we have, with probability at least  $1 - \delta$ ,

$$\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f) \leq \mathbb{E} [\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f)] + \sqrt{\frac{\log(1/\delta)}{2n}}.$$

Now, we can appeal to a powerful idea known as *symmetrization*. The basic idea is to replace

$$\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f)$$

with

$$\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (\mathbb{E}[\ell(f(X'_i), Y'_i)] - \ell(f(X_i), Y_i)),$$

where  $(X'_1, Y'_1), \dots, (X'_n, Y'_n)$  are samples drawn from the distribution  $P$  that are independent of each other and of the original sample. The new sample is often referred to as the *ghost sample*. The point of introducing the ghost sample is that, by an application of Jensen's inequality, we get

$$\begin{aligned} \mathbb{E} [\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f)] &= \mathbb{E} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (\mathbb{E}[\ell(f(X'_i), Y'_i)] - \ell(f(X_i), Y_i)) \right] \\ &\leq \mathbb{E} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (\ell(f(X'_i), Y'_i) - \ell(f(X_i), Y_i)) \right]. \end{aligned}$$

Now, the distribution of

$$\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (\ell(f(X'_i), Y'_i) - \ell(f(X_i), Y_i))$$

is invariant under exchanging any  $X_i, Y_i$  with an  $X'_i, Y'_i$ . That is, for any fixed choice of  $\pm 1$  signs  $\epsilon_1, \dots, \epsilon_n$ , the above quantity has the same distribution as

$$\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i (\ell(f(X'_i), Y'_i) - \ell(f(X_i), Y_i)).$$

This allows us introduce even more randomization by making the  $\epsilon_i$ 's themselves random. This gives,

$$\begin{aligned}
 \mathbb{E} [\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f)] &\leq \mathbb{E}_{\epsilon, X, Y, X', Y'} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i (\ell(f(X'_i), Y'_i) - \ell(f(X_i), Y_i)) \right] \\
 &\leq \mathbb{E}_{\epsilon, X', Y'} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i \ell(f(X'_i), Y'_i) \right] \\
 &\quad + \mathbb{E}_{\epsilon, X, Y} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (-\epsilon_i) \ell(f(X_i), Y_i) \right] \\
 &= 2 \cdot \mathbb{E}_{\epsilon, X, Y} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i \ell(f(X_i), Y_i) \right].
 \end{aligned}$$

This directly yields a bound in terms of the *Rademacher complexity*:

$$\mathfrak{R}_n(\ell \circ \mathcal{F}) = \mathbb{E}_{\epsilon, X, Y} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i \ell(f(X_i), Y_i) \right],$$

where  $\epsilon_i$  is a *Rademacher random variable* that is either -1 or +1, each with probability  $\frac{1}{2}$ . The subscript below the expectation serves as a reminder that the expectation is being taken with respect to the randomness in  $X_i$ ,  $Y_i$ 's, and  $\epsilon_i$ 's. We have thus proved the following theorem.

**Theorem 7 (Symmetrization).** *For a function class  $\mathcal{F}$  and loss  $\ell$ , define the loss class*

$$\ell \circ \mathcal{F} = \{(x, y) \mapsto \ell(f(x), y) : f \in \mathcal{F}\}.$$

We have,

$$\mathbb{E} [\sup_{f \in \mathcal{F}} R(f) - \widehat{R}(f)] \leq 2\mathfrak{R}_n(\ell \circ \mathcal{F}).$$

An interesting observation is that the sample dependent quantity

$$\widehat{\mathfrak{R}}_n(\ell \circ \mathcal{F}) = \mathbb{E}_{\epsilon} \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \epsilon_i \ell(f(X_i), Y_i) \right]$$

also satisfies the bounded differences condition with  $c_i = 1/n$ . This is the *empirical Rademacher average* and the expectation in its definition is only over the Rademacher random variables. Applying McDiarmid's inequality, we have,

$$\mathfrak{R}_n(\ell \circ \mathcal{F}) \leq \widehat{\mathfrak{R}}_n(\ell \circ \mathcal{F}) + \sqrt{\frac{\log(1/\delta)}{2n}}$$

with probability at least  $1 - \delta$ .

To prove a bound on the Rademacher complexity of a finite class, the following lemma, due to [24], is useful.

**Lemma 8 (Massart's finite class lemma).** Let  $\mathcal{A}$  be a finite subset of  $\mathbb{R}^n$  and  $\epsilon_1, \dots, \epsilon_n$  be independent Rademacher random variables. Let  $r = \max_{a \in \mathcal{A}} \|a\|_2$ . Then, we have,

$$\mathbb{E} \left[ \sup_{a \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^n \epsilon_i a_i \right] \leq \frac{r \sqrt{2 \log |\mathcal{A}|}}{n}.$$

Therefore, for any finite class  $\mathcal{F}$  consisting of functions bounded by 1,

$$\widehat{\mathfrak{R}}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log |\mathcal{F}|}{n}}.$$

Since the right hand side is not random, the same bound holds for  $\mathfrak{R}_n(\mathcal{F})$  as well.

The Rademacher complexity, and its empirical counterpart, satisfy a number of interesting structural properties.

**Theorem 9 (Structural Properties of Rademacher Complexity).** Let  $\mathcal{F}, \mathcal{F}_1, \dots, \mathcal{F}_k$  and  $\mathcal{H}$  be classes of real valued functions. Then  $\mathfrak{R}_n(\cdot)$  (as well as  $\widehat{\mathfrak{R}}_n(\cdot)$ ) satisfies the following properties:

*Monotonicity:* If  $\mathcal{F} \subseteq \mathcal{H}$ ,  $\mathfrak{R}_n(\mathcal{F}) \leq \mathfrak{R}_n(\mathcal{H})$ .

*Convex Hull:*  $\mathfrak{R}_n(\text{conv}(\mathcal{F})) = \mathfrak{R}_n(\mathcal{F})$ .

*Scaling:*  $\mathfrak{R}_n(c\mathcal{F}) = |c|\mathfrak{R}_n(\mathcal{F})$  for any  $c \in \mathbb{R}$ .

*Contraction:* If  $\phi : \mathbb{R} \rightarrow \mathbb{R}$  is Lipschitz with constant  $L_\phi$ , then  $\mathfrak{R}_n(\phi \circ \mathcal{F}) \leq L_\phi \mathfrak{R}_n(\mathcal{F})$ .

*Translation:* For any bounded function  $h$ ,  $\mathfrak{R}_n(\mathcal{F} + h) = \mathfrak{R}_n(\mathcal{F})$ .

*Subadditivity:*  $\mathfrak{R}_n\left(\sum_{i=1}^k \mathcal{F}_i\right) \leq \sum_{i=1}^k \mathfrak{R}_n(\mathcal{F}_i)$ .

For proofs of these properties, see [25]. However, note that the Rademacher complexity is defined there such that an extra absolute value is taken before the supremum over  $f \in \mathcal{F}$ . That is,

$$\mathfrak{R}_n^{\text{abs}}(\mathcal{F}) = \mathbb{E} \left[ \sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^n f(X_i) \right| \right].$$

Nevertheless, the proofs given there generalize quite easily to definition of Rademacher complexity that we are using. For example, see [26], where the Rademacher complexity as we have defined it here (without the absolute value) is called *free Rademacher complexity*.

Using these properties, it is possible to directly bound the Rademacher complexity of several interesting functions classes. See [25] for examples. The contraction property is a very useful one. It allows us to transition from the Rademacher complexity of the loss class to the Rademacher complexity of the function class itself (for Lipschitz losses).

The Rademacher complexity can also be bounded in terms of another measure of complexity of a function class: namely covering numbers.

### 1.14.3.3 Covering numbers

The idea underlying covering numbers is very intuitive. Let  $(\mathcal{D}, \rho)$  be any (pseudo-) metric space and fix a subset  $T \subseteq \mathcal{D}$ . A set  $T' \subseteq \mathcal{D}$  is said to be an  $\alpha$ -cover of  $T$  if

$$\forall f \in T, \exists g \in T' \text{ s.t. } \rho(f, g) \leq \alpha.$$

In other words “balls” of radius  $\alpha$  placed at elements of  $T'$  “cover” the set  $T$  entirely. The *covering number* (at scale  $\alpha$ ) of  $T$  is defined as the size of the smallest cover of  $T$  (at scale  $\alpha$ ).

$$\mathcal{N}(T, \rho, \alpha) = \min \{|T'| : T' \text{ is an } \alpha\text{-cover of } T\}.$$

The logarithm of covering number is called *entropy*.

Given a sample  $x_{1:n} = (x_1, \dots, x_n)$ , define the data dependent metric on  $\mathcal{F}$  as

$$\widehat{\rho}_p(f, g) = \left( \frac{1}{n} \sum_{i=1}^n |f(x_i) - g(x_i)|^p \right)^{1/p},$$

where  $p \in [1, \infty)$ . Taking the limit  $p \rightarrow \infty$  suggests the metric

$$\widehat{\rho}_\infty(f, g) = \max_{i \in [n]} |f(x_i) - g(x_i)|.$$

These metrics gives us the  $p$ -norm covering numbers

$$\mathcal{N}_p(\mathcal{F}, x_{1:n}, \alpha) = \mathcal{N}(\mathcal{F}, \widehat{\rho}, \alpha).$$

These covering numbers increase with  $p$ . That is, for  $p \in [1, \infty]$ ,

$$\mathcal{N}_1(\mathcal{F}, x_{1:n}, \alpha) \leq \mathcal{N}_p(\mathcal{F}, x_{1:n}, \alpha) \leq \mathcal{N}_\infty(\mathcal{F}, x_{1:n}, \alpha).$$

Finally, we can also define the worst case (over samples)  $p$ -norm covering number

$$\mathcal{N}_p(\mathcal{F}, n, \alpha) = \sup_{x_{1:n} \in \mathcal{X}^n} \mathcal{N}_p(\mathcal{F}, x_{1:n}, \alpha).$$

A major result connecting Rademacher complexity with covering numbers is due to [27].

**Theorem 10 (Dudley's entropy integral bound).** *For any  $\mathcal{F}$  consisting of real valued functions bounded by 1, we have*

$$\widehat{\mathfrak{R}}_n(\mathcal{F}) \leq \alpha + 12 \int_\alpha^1 \sqrt{\frac{\log \mathcal{N}_2(\mathcal{F}, X_{1:n}, \beta)}{n}} d\beta.$$

Therefore, using Jensen's inequality, we have

$$\mathfrak{R}_n(\mathcal{F}) \leq \alpha + 12 \int_\alpha^1 \sqrt{\frac{\log \mathbb{E} \mathcal{N}_2(\mathcal{F}, X_{1:n}, \beta)}{n}} d\beta.$$

#### 1.14.3.4 Binary valued functions: VC dimension

For binary function classes  $\mathcal{F} \subseteq \{\pm 1\}^{\mathcal{X}}$ , we can provide distribution free upper bounds on the Rademacher complexity or covering numbers in terms of a combinatorial parameter called the *Vapnik-Chervonenkis (VC) dimension*.

To define it, consider a sequence  $x_{1:n} = (x_1, \dots, x_n)$ . We say that  $x_{1:n}$  is *shattered* by  $\mathcal{F}$  if each of the  $2^n$  possible  $\pm 1$  labelings of these  $n$  points are realizable using some  $f \in \mathcal{F}$ . That is,  $x_{1:n}$  is shattered by  $\mathcal{F}$  iff

$$|\{(f(x_1), \dots, f(x_n)) : f \in \mathcal{F}\}| = 2^n.$$

The *VC dimension* of  $\mathcal{F}$  is the length of the longest sequence that can be shattered by  $\mathcal{F}$ :

$$\text{VCdim}(\mathcal{F}) = \max \{n : \exists x_{1:n} \in \mathcal{X}^n \text{ s.t. } x_{1:n} \text{ is shattered by } \mathcal{F}\}.$$

The size of the restriction of  $\mathcal{F}$  to  $x_{1:n}$  is called the *shatter coefficient*:

$$\mathbb{S}(\mathcal{F}, x_{1:n}) = |\{(f(x_1), \dots, f(x_n)) : f \in \mathcal{F}\}|.$$

Considering the worst sequence gives us the *growth function*

$$\mathbb{S}(\mathcal{F}, n) = \max_{x_{1:n} \in \mathcal{X}^n} \mathbb{S}(\mathcal{F}, x_{1:n}).$$

If  $\text{VCdim}(\mathcal{F}) = d$ , then we know that  $\mathbb{S}(\mathcal{F}, n) = 2^n$  for all  $n \leq d$ . One can ask: what is the behavior of the growth function for  $n > d$ ? The following combinatorial lemma proved independently by Vapnik and Chervonenkis [28], Sauer [29], and Shelah [30], gives the answer to this question.

**Lemma 11.** *If  $\text{VCdim}(\mathcal{F}) = d$ , then we have, for any  $n \in \mathbb{N}$ ,*

$$\mathbb{S}(\mathcal{F}, n) = \sum_{i=0}^d \binom{n}{i}.$$

In particular, if  $d > 2$ ,  $\mathbb{S}(\mathcal{F}, n) \leq n^d$ .

Thus, for a class with finite VC dimension, the Rademacher complexity can be bounded directly using Massart's lemma.

**Theorem 12.** *Suppose  $2 < \text{VCdim}(\mathcal{F}) = d < \infty$ . Then, we have,*

$$\widehat{\mathfrak{R}}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log \mathbb{S}(\mathcal{F}, n)}{n}} \leq \sqrt{\frac{2d \log n}{n}}.$$

It is possible to shave off the  $\log n$  factor in the bound above plugging an estimate of the entropy  $\log N_2(\mathcal{F}, n, \alpha)$  for VC classes due to [31] into Theorem 10.

**Theorem 13.** *Suppose  $\text{VCdim}(\mathcal{F}) = d < \infty$ . We have, for any  $n \in \mathbb{N}$ ,*

$$\mathcal{N}_2(\mathcal{F}, n, \alpha) \leq Cd \log \frac{1}{\alpha}$$

for a universal constant  $C$ .

These results show that for a class  $\mathcal{F}$  with finite VC dimension  $d$ , the sample complexity required by ERM to achieve  $\varepsilon$  excess risk relative to  $f_{\mathcal{F}}^*$  is, with probability at least  $1 - \delta$ ,  $O\left(\frac{d}{\varepsilon^2} + \log \frac{1}{\delta}\right)$ . Therefore, a finite VC dimension is *sufficient* for learnability with respect to a fixed functions class. In fact, a finite VC dimension also turns out to be necessary (see, for instance, [14, Chapter 14]). Thus, we have the following characterization.

**Theorem 14.** *In the binary classification setting with 0–1 loss, there is a learning algorithm with bounded sample complexity for every  $\varepsilon, \delta \in (0, 1)$  iff  $\text{VCdim}(\mathcal{F}) < \infty$ .*

Note that, when  $\text{VCdim}(\mathcal{F}) < \infty$ , the dependence of the sample complexity on  $1/\varepsilon$  is polynomial and on  $1/\delta$  is *logarithmic*. Moreover, such a sample complexity is achieved by ERM. On the other hand, when  $\text{VCdim}(\mathcal{F}) = \infty$ , then no learning algorithm, including ERM, can have bounded sample complexity for arbitrarily small  $\varepsilon, \delta$ .

### 1.14.3.5 Real valued functions: fat shattering dimension

For real valued functions, getting distribution free upper bounds on Rademacher complexity or covering numbers involves combinatorial parameters similar to the VC dimension. The appropriate notion for determining learnability using a finite number of samples turns out to be a *scale sensitive* combinatorial dimension known as the *fat shattering dimension*.

Fix a class  $\mathcal{F}$  consisting of bounded real valued functions  $f : \mathcal{X} \rightarrow [0, 1]$  and a scale  $\alpha > 0$ . We say that a sequence  $x_{1:n} = (x_1, \dots, x_n)$  is  $\alpha$ -shattered by  $\mathcal{F}$  if there exists a witness sequence  $s_{1:n} \in \mathbb{R}^n$  such that, for every  $\epsilon_{1:n} \in \{\pm 1\}^n$ , there is an  $f \in \mathcal{F}$  such that

$$\epsilon_i(f(x_i) - s_i) \geq \alpha .$$

In words, this means that, for any of the  $2^n$  possibilities for the sign pattern  $\epsilon_{1:n}$ , we have a function  $f \in \mathcal{F}$  whose values at the points  $x_i$  is above or below  $s_i$  depending on whether  $\epsilon_i$  is  $+1$  or  $-1$ . Moreover, the gap between  $f(x_i)$  and  $s_i$  is required to be at least  $\alpha$ .

The fat-shattering dimension of  $\mathcal{F}$  at scale  $\alpha$  is the size of the longest sequence that can be  $\alpha$ -shattered by  $\mathcal{F}$ :

$$\text{fat}_\alpha(\mathcal{F}) = \max\{n : \exists x_{1:n} \in \mathcal{X}^n \text{ s.t. } x_{1:n} \text{ is } \alpha\text{-shattered by } \mathcal{F}\}.$$

A bound on  $\infty$ -norm covering numbers in terms of the fat shattering dimension were first given by Alon et al. [32].

**Theorem 15.** *Fix a class  $\mathcal{F} \subseteq [0, 1]^{\mathcal{X}}$ . Then we have, for any  $\alpha > 0$ ,*

$$\mathcal{N}_\infty(\mathcal{F}, n, \alpha) \leq 2 \left( \frac{4n}{\alpha^2} \right)^{d \log(en/d\alpha)} ,$$

where  $d = \text{fat}_{\alpha/2}(\mathcal{F})$ .

Using this bound, the sample complexity of learning a real valued function class of bounded range using a Lipschitz loss<sup>3</sup>  $\ell$  is  $O(\frac{d}{\varepsilon^2} \log \frac{1}{\varepsilon} + \log \frac{1}{\delta})$  where  $d = \text{fat}_{ce}(\mathcal{F})$  for a universal constant  $c > 0$ . The importance of the fat shattering dimension lies in the fact that if  $\text{fat}_\alpha(\mathcal{F}) = \infty$  for some  $\alpha > 0$ , then no learning algorithm can have bounded sample complexity for learning the functions for arbitrarily small values of  $\varepsilon, \delta$ .

**Theorem 16.** *In the regression setting with either squared loss or absolute loss, there is a learning algorithm with bounded sample complexity for every  $\varepsilon, \delta \in (0, 1)$  iff  $\forall \alpha > 0, \text{fat}_\alpha(\mathcal{F}) < \infty$ .*

---

<sup>3</sup>By this we mean  $\ell(\cdot, y)$  is Lipschitz for all  $y$ .

Note that here, unlike the binary classification case, the dependence of the sample complexity on  $1/\varepsilon$  is not fixed. It depends on how fast  $\text{fat}_\alpha \mathcal{F}$  grows as  $\alpha \rightarrow 0$ . In particular, if  $\text{fat}_\alpha(\mathcal{F})$  grows as  $1/\alpha^p$  then the sample complexity is polynomial in  $1/\varepsilon$  (and logarithmic in  $1/\delta$ ).

### 1.14.4 Model selection

In general, model selection refers to the task of selecting an appropriate model from a given family based on available data. For ERM, the “model” is the class  $\mathcal{F}$  of functions  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that are intended to capture the functional dependence between the input variable  $X$  and the output  $Y$ . If we choose too small an  $\mathcal{F}$  then ERM will not have good performance if  $f^*$  is not even well approximated by any member of  $\mathcal{F}$ . On the other hand, if  $\mathcal{F}$  is too large (say all functions) then the estimation error will be so large that ERM will not perform well even if  $f^* \in \mathcal{F}$ .

#### 1.14.4.1 Structural risk minimization

Consider the classification setting with 0–1 loss. Suppose we have a countable sequence of nested function classes:

$$\mathcal{F}^{(1)} \subset \mathcal{F}^{(2)} \subset \dots \subset \mathcal{F}^{(k)} \dots ,$$

where  $\mathcal{F}^{(k)}$  has VC dimension  $d_k$ . The function  $\hat{f}_n^{(k)}$  chosen by ERM over  $\mathcal{F}_k$  satisfies the bound

$$R\left(\hat{f}_n^{(k)}\right) \leq \widehat{R}\left(\hat{f}_n^{(k)}\right) + O\left(\sqrt{\frac{d_k \log n}{n}}\right)$$

with high probability. As  $k$  increases, the classes become more complex. So, the empirical risk term will decrease with  $k$  whereas the VC dimension term will increase. The principle of *structural risk minimization (SRM)* chooses a value of  $k$  by minimizing the right hand side above.

Namely, we choose

$$\hat{k} = \operatorname{argmin}_k \widehat{R}\left(\hat{f}_n^{(k)}\right) + C \sqrt{\frac{d_k \log n}{n}}, \quad (14.3)$$

for some universal constant  $C$ .

It can be shown (see, for example, [14, Chapter 18]) that SRM leads to universally consistent rules provided appropriate conditions are placed on the functions classes  $\mathcal{F}_k$ .

**Theorem 17 (Universal Consistency of SRM).** *Let  $\mathcal{F}^{(1)}, \mathcal{F}^{(2)}, \dots$  be a sequence of classes such that for any distribution  $P$ ,*

$$\lim_{k \rightarrow \infty} R(f_{\mathcal{F}^{(k)}}^*) = R(f^*).$$

*Assume also that  $\text{VCdim}(\mathcal{F}^{(1)}) < \text{VCdim}(\mathcal{F}^{(2)}) < \dots$  are all finite. Then the learning rule  $\hat{f}_n^{(\hat{k})}$  with  $\hat{k}$  chosen as in (14.3) is strongly universally consistent.*

The penalty term in (14.3) depends on the VC dimension of  $\mathcal{F}^{(k)}$ . It turns out that it is not essential. All we need is an upper bound on the risk of ERM in terms of the empirical risk and some measure of the complexity of the function class. One could also imagine proving results for regression problems by

replacing the VC dimension with fat-shattering dimension. We can think of SRM as *penalizing* larger classes by their capacity to overfit the data. However, the penalty in SRM is fixed in advance and is not data dependent. We will now look into more general penalties.

#### 1.14.4.2 Model selection via penalization

Given a (not necessarily nested) countable or finite collection of models  $\{\mathcal{F}^{(k)}\}_k$  and a penalty function  $\text{pen} : \mathbb{N} \rightarrow \mathbb{R}_+$ , define  $\hat{k}$  as the minimizer of

$$\hat{R}\left(\hat{f}_n^{(k)}\right) + \text{pen}(k).$$

Recall that  $\hat{f}_n^{(k)}$  is the result of ERM over  $\mathcal{F}^{(k)}$ . Also note that the penalty can additionally depend on the data. Now define the *penalized estimator* as

$$\tilde{f}_n = \hat{f}_n^{(\hat{k})}.$$

The following theorem shows that any probabilistic upper bound  $U_{n,k}$  on the risk  $R(\hat{f}_n)$  can be used to design a penalty for which a performance bound can be given.

**Theorem 18.** *Assume that there are positive numbers  $c$  and  $m$  such that for each  $k$  we have the guarantee*

$$\forall \varepsilon > 0, \mathbb{P}\left[R\left(\hat{f}_n^{(k)}\right) > U_{n,k} + \varepsilon\right] \leq c \exp(-2m\varepsilon^2).$$

Then the penalized estimator  $\tilde{f}_n$  with

$$\text{pen}(k) = U_{n,k} - \hat{R}\left(\hat{f}_n^{(k)}\right) + \sqrt{\frac{\log k}{m}}$$

satisfies

$$\mathbb{E}[R(\tilde{f}_n)] - R(f^*) \leq \min_k \left\{ \mathbb{E}[\text{pen}(k)] + (R(f_{\mathcal{F}^{(k)}}^*) - R(f^*)) \right\} + \sqrt{\frac{\log (ce)}{2m}}.$$

Note that the result says that the penalized estimator achieves an almost optimal trade-off between the approximation error  $R(f_{\mathcal{F}^{(k)}}^*) - R(f^*)$  and the expected complexity  $\mathbb{E}[\text{pen}](k)$ . For a good upper bound  $U_{n,k}$ , the expected complexity will be a good upper bound on the (expected) estimation error. Therefore, the model selection properties of the penalized estimator depend on how good an upper bound  $U_{n,k}$  we can find.

Note that structural risk minimization corresponds to using a distribution free upper bound

$$U_{n,k} = \hat{R}\left(\hat{f}_n^{(k)}\right) + O\left(\sqrt{\frac{d_k \log n}{n}}\right).$$

The looseness of these distribution free upper bounds has prompted researchers to design data dependent penalties. We shall see an examples based on empirical Rademacher averages in the next section.

**Proof.** For any  $\varepsilon > 0$ ,

$$\begin{aligned}
 & \mathbb{P} \left[ R(\tilde{f}_n) - (\widehat{R}(\tilde{f}_n) + \text{pen}(\widehat{k})) > \varepsilon \right] \\
 & \leq \mathbb{P} \left[ \exists k \text{ s.t. } R(\widehat{f}_n^{(k)}) - (\widehat{R}(\widehat{f}_n^{(k)}) + \text{pen}(k)) > \varepsilon \right] \\
 & \leq \sum_{k=1}^{\infty} \mathbb{P} \left[ R(\widehat{f}_n^{(k)}) - (\widehat{R}(\widehat{f}_n^{(k)}) + \text{pen}(k)) > \varepsilon \right] \\
 & = \sum_{k=1}^{\infty} \mathbb{P} \left[ R(\widehat{f}_n^{(k)}) - U_{n,k} > \sqrt{\frac{\log k}{m}} + \varepsilon \right] \\
 & \leq \sum_{k=1}^{\infty} c \exp \left( -2m(\varepsilon + \sqrt{\log k/m})^2 \right) \\
 & \leq \sum_{k=1}^{\infty} c \exp \left( -2m(\varepsilon^2 + \log k/m) \right) \\
 & = c \exp(-2n\varepsilon^2) \sum_{k=1}^{\infty} \frac{1}{k^2} \leq 2c \exp(-2m\varepsilon^2).
 \end{aligned}$$

By integrating the final upper bound with respect to  $\varepsilon$ , we get the bound

$$\mathbb{E} \left[ R(\tilde{f}_n) - (\widehat{R}(\tilde{f}_n) + \text{pen}(\widehat{k})) \right] \leq \sqrt{\frac{\log(ce)}{2m}}. \quad (14.4)$$

Also, for any  $k$ , we have

$$\begin{aligned}
 \mathbb{E} \left[ \widehat{R}(\tilde{f}_n) + \text{pen}(\widehat{k}) - R(f_{\mathcal{F}^{(k)}}^*) \right] & \leq \mathbb{E} \left[ \widehat{R}(\widehat{f}_n^{(k)}) + \text{pen}(k) - R(f_{\mathcal{F}^{(k)}}^*) \right] \\
 & \leq \mathbb{E} \left[ \widehat{R}(f_{\mathcal{F}^{(k)}}^*) + \text{pen}(k) - R(f_{\mathcal{F}^{(k)}}^*) \right] \\
 & = \mathbb{E} [\text{pen}(k)].
 \end{aligned}$$

The first inequality above follows by definition of  $\widehat{k}$  and  $\tilde{f}_n$ . The second holds because  $\widehat{f}_n^{(k)}$  minimizes the empirical risk over  $\mathcal{F}^{(k)}$ . Summing this with (14.4), we get, for any  $k$ ,

$$\mathbb{E} \left[ R(\tilde{f}_n) \right] \leq \mathbb{E} [\text{pen}(k)] + R(f_{\mathcal{F}^{(k)}}^*) + \sqrt{\frac{\log(ce)}{2m}}.$$

Now subtracting  $R(f^*)$  from both sides and minimizing over  $k$  proves the theorem.  $\square$

#### 1.14.4.3 Data driven penalties using Rademacher averages

The results of Section 1.14.3.2 tell us that there exists universal constants  $C, c$  such that

$$\mathbb{P}[R(\widehat{f}_n) > \widehat{R}(\widehat{f}_n) + 2\widehat{\mathfrak{R}}_n(\mathcal{F}^{(k)}) + \varepsilon] < c \exp(-2Cn\varepsilon^2).$$

Thus, we can use the data dependent penalty

$$\text{pen}(k) = 2\widehat{\mathfrak{R}}_n(\mathcal{F}^{(k)}) + \sqrt{\frac{\log k}{Cn}}$$

in Theorem 18 to get the bound

$$\mathbb{E}[R(\tilde{f}_n)] - R(f^*) \leq \min_k \left\{ 2\mathfrak{R}_n(\mathcal{F}^{(k)}) + (R(f_{\mathcal{F}^{(k)}}^*) - R(f^*)) + \sqrt{\frac{\log k}{Cn}} \right\} + \sqrt{\frac{\log (ce)}{2Cn}}.$$

Note that the penalty here is data dependent. Moreover, except for the  $\sqrt{\frac{\log k}{Cn}}$  term, the penalized estimator makes the optimal trade-off between the Rademacher complexity, an upper bound on the estimation error, and the approximation error.

#### 1.14.4.4 Hold-out estimates

Suppose out of  $m + n$  iid samples we set aside  $m$  samples where  $m$  is much smaller than  $n$ . Since the hold-out set of size  $m$  is not used in the computation of  $\widehat{f}_n^{(k)}$ , we can estimate  $R(\widehat{f}_n^{(k)})$  by the hold-out estimate

$$U_{n,k} = \sum_{i=1}^m \ell\left(\widehat{f}_n^{(k)}(X_{n+i}), Y_{n+i}\right).$$

By Hoeffding's inequality, the assumption of Theorem 18 holds with  $c = 1$ . Moreover  $\mathbb{E}[U_{n,k}] = R(\widehat{f}_n^{(k)})$ . Therefore, we have

$$\mathbb{E}[R(\tilde{f}_n)] - R(f^*) \leq \min_k \left\{ \mathbb{E}\left[R(\widehat{f}_n^{(k)}) - \widehat{R}(\widehat{f}_n^{(k)})\right] + (R(f_{\mathcal{F}^{(k)}}^*) - R(f^*)) + \sqrt{\frac{\log k}{m}} \right\} + \sqrt{\frac{1}{2m}}.$$

### 1.14.5 Alternatives to uniform convergence

The approach of deriving generalization bounds using uniform convergence arguments is not the only one possible. In this section, we review three techniques for proving generalization bounds based on sample compression, algorithmic stability, and PAC-Bayesian analysis respectively.

#### 1.14.5.1 Sample compression

The sample compression approach to proving generalization bounds was introduced by [33]. It applies to algorithms whose output  $\widehat{f}_{n,S}$  (that is learned using some training set  $S$ ) can be reconstructed from a compressed subset  $S_{\mathbf{i}}$ . Here  $\mathbf{i}$  is a sequence of indices

$$\mathbf{i} = (i_1, i_2, \dots, i_k), \quad 1 \leq i_1 < i_2 < \dots < i_k \leq n,$$

and  $S_{\mathbf{i}}$  is simply the subsequence of  $S$  indexed by  $\mathbf{i}$ . For the index sequence  $\mathbf{i}$  above, we say that its length  $|\mathbf{i}|$  is  $k$ . To formalize the idea that the algorithms output can be reconstructed from a small number of examples, we will define a *compression function*

$$\mathcal{C} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \rightarrow \mathcal{I},$$

where  $\mathcal{I}$  is the set of all possible index sequences of finite length. It is assumed that  $|\mathcal{C}(S)| \leq n$  for an  $S$  is of size  $n$ . We also define a *reconstruction function*

$$\mathcal{R} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \rightarrow \mathcal{Y}^{\mathcal{X}}.$$

We assume that the learning rule satisfies, for any  $S$ ,

$$\widehat{f}_{n,S} = \mathcal{R}(S_{\mathcal{C}(S)}). \quad (14.5)$$

This formalizes our intuition that to reconstruct  $\widehat{f}_{n,S}$  it suffices to remember only those examples whose indices are given by  $\mathcal{C}(S)$ .

The following result gives a generalization bound for such an algorithm.

**Theorem 19.** *Let  $\widehat{f}_n$  be a learning rule for which there exists compression and reconstructions functions  $\mathcal{C}$  and  $\mathcal{R}$  respectively such that the equality (14.5) holds for all  $S$ . Then, we have, with probability at least  $1 - \delta$ ,*

$$R(\widehat{f}_n) \leq \frac{n}{n-k} \widehat{R}(\widehat{f}_n) + \sqrt{\frac{\log \binom{n}{k} + \log \frac{n}{\delta}}{2(n-k)}},$$

where  $k = |\mathcal{C}(S)|$ .

**Proof.** The proof uses the simple idea that for a fixed index set  $\mathbf{i}$ ,  $S_{\bar{\mathbf{i}}}$  is a sample of size  $n - |\mathbf{i}|$  that is independent of  $S_{\mathbf{i}}$ . Here  $\bar{\mathbf{i}}$  denotes the sequence  $[n] \setminus \mathbf{i}$ . By this independence, the risk of any function that depends only on  $S_{\mathbf{i}}$  is close to its empirical average on  $S_{\bar{\mathbf{i}}}$  by Hoeffding's inequality.

Denote the empirical risk of  $f$  calculated on a subset  $S_{\mathbf{i}}$  of  $S$  by  $\widehat{R}_{\mathbf{i}}(f)$ . We have, for any  $f$ ,

$$\widehat{R}_{\overline{\mathcal{C}(S)}}(f) \leq \frac{n}{n-k} \widehat{R}(f),$$

where  $k = |\mathcal{C}(S)|$ . Thus, all we need to show is

$$\mathbb{P}\left[R(\widehat{f}_n) - \widehat{R}_{\overline{\mathcal{C}(S)}}(\widehat{f}_n) \geq \varepsilon_{n,|\mathcal{C}(S)|}\right] \leq \delta,$$

where

$$\varepsilon_{n,k} = \sqrt{\frac{\log \binom{n}{k} + \log \frac{n}{\delta}}{2(n-k)}}$$

To show this, we proceed as follows,

$$\begin{aligned}
 & \mathbb{P} \left[ R(\widehat{f}_n) - \widehat{R}_{\overline{\mathcal{C}(S)}}(\widehat{f}_n) \geq \varepsilon_{n,|\mathcal{C}(S)|} \right] \\
 &= \mathbb{P} \left[ R(\mathcal{R}(S_{\mathcal{C}(S)})) - \widehat{R}_{\overline{\mathcal{C}(S)}}(\mathcal{R}(S_{\mathcal{C}(S)})) \geq \varepsilon_{n,|\mathcal{C}(S)|} \right] \\
 &\leq \mathbb{P} \left[ \exists \mathbf{i} \text{ s.t. } |\mathbf{i}| = k, R(\mathcal{R}(S_{\mathbf{i}})) - \widehat{R}_{\mathbf{i}}(\mathcal{R}(S_{\mathbf{i}})) \geq \varepsilon_{n,k} \right] \\
 &\leq \sum_{\mathbf{i}:|\mathbf{i}|=k} \mathbb{P} \left[ R(\mathcal{R}(S_{\mathbf{i}})) - \widehat{R}_{\mathbf{i}}(\mathcal{R}(S_{\mathbf{i}})) \geq \varepsilon_{n,k} \right] \\
 &\leq \sum_{k=1}^n \sum_{\mathbf{i}:|\mathbf{i}|=k} \exp(-2(n-k)\varepsilon_{n,k}^2) \\
 &= \sum_{k=1}^n \binom{n}{k} \exp(-2(n-k)\varepsilon_{n,k}^2) \\
 &\leq \sum_{k=1}^n \frac{\delta}{n} = \delta.
 \end{aligned}$$

The last step holds because, by our choice of  $\varepsilon_{n,k}$ ,

$$\binom{n}{k} \exp(-2(n-k)\varepsilon_{n,k}^2) = \frac{\delta}{n}.$$
 $\square$

### 1.14.5.2 Stability

*Stability* is a property of a learning algorithm that ensures that the output, or some aspect of the output, of the learning algorithm does not change much if the training data set changes very slightly. This is an intuitive notion whose roots in learning theory go back to the work of Tikhonov on regularization of inverse problems. To see how stability can be used to give generalization bounds, let us consider a result of Bousquet and Elisseeff [34] that avoids going through uniform convergence arguments by directly showing that any learning algorithm with *uniform stability* enjoys exponential tail generalization bounds in terms of both the training error and the leave-one-out error of the algorithm.

Let  $S^{\setminus i}$  denote the sample of size  $n-1$  obtained by removing a particular input  $(x_i, y_i)$  from a sample  $S$  of size  $n$ . A learning algorithm is said to have *uniform stability*  $\beta$  if for any  $S \in (\mathcal{X} \times \mathcal{Y})^n$  and any  $i \in [n]$ , we have,

$$\sup_{(x,y) \in \mathcal{X} \times \mathcal{Y}} |\ell(\widehat{f}_{n,S}(x), y) - \ell(\widehat{f}_{n-1,S^{\setminus i}}(x), y)| \leq \beta.$$

We have the following generalization bound for a uniformly stable learning algorithm.

**Theorem 20.** *Let  $\widehat{f}_n$  have uniform stability  $\beta$ . Then, with probability at least  $1 - \delta$ ,*

$$R(\widehat{f}_n) \leq \widehat{R}(\widehat{f}_n) + 2\beta + (4\beta n + 1) \sqrt{\frac{\log \frac{1}{\delta}}{2n}}.$$

Note that this theorem gives tight bound when  $\beta$  scales as  $1/n$ . As such it cannot be applied directly to the classification setting with 0–1 loss where  $\beta$  can only be 0 or 1. A good example of a learning algorithm that exhibits uniform stability is regularized ERM over a reproducing kernel Hilbert space using a convex loss function that has Lipschitz constant  $C_\ell$ :

$$\forall y, y', y'' \in \mathcal{Y} \subseteq \mathbb{R}, |\ell(y', y) - \ell(y'', y)| \leq C_\ell \cdot |y' - y''|.$$

A regularized ERM using kernel  $K$  is defined by

$$\widehat{f}_n = \underset{f \in \mathcal{F}_K}{\operatorname{argmin}} \widehat{R}(f) + \lambda \|f\|_K^2,$$

where  $\lambda$  is a regularization parameter. Regularized ERM has uniform stability  $\beta$  for  $\beta = \frac{C_\ell^2 \kappa^2}{2\lambda n}$ , where

$$\kappa = \sup_{x \in \mathcal{X}} \sqrt{K(x, x)}.$$

The literature on stability of learning algorithms is unfortunately thick with definitions. For instance, Kutin and Niyogi [35] alone consider over a dozen variants! We chose a simple (but quite strong) definition from Bousquet and Elisseeff above to illustrate the general point that stability can be used to derive generalization bounds. But we like to point the reader to Mukherjee et al. [36] where a notion of *leave-one-out (LOO) stability* is defined and shown sufficient for generalization for any learning algorithm. Moreover, for ERM they show it is necessary and sufficient for both generalization and learnability with respect to a fixed class. Shalev-Shwartz et al. [37] further examine the connections between stability, generalization and learnability in Vapnik’s general learning setting that includes supervised learning (learning from example, label pairs) as a special case.

### 1.14.5.3 PAC-Bayesian analysis

PAC-Bayesian analysis refers to a style of analysis of learning algorithms that output not just a single function  $f \in \mathcal{F}$  but rather a *distribution* (called a “posterior distribution”) over  $\mathcal{F}$ . Moreover, the complexity of the data-dependent posterior distribution is measured relative to a fixed distribution chosen in advance (that is, in a data independent way). The fixed distribution is called a “prior distribution.” Note that the terms “prior” and “posterior” are borrowed from the analysis of Bayesian algorithms that start with a prior distribution on some parameter space and, on seeing data, make updates to the distribution using Bayes’ rule to obtain a posterior distribution. Even though the terms are borrowed, their use is not required to be similar to their use in Bayesian analysis. In particular, the prior used in the PAC-Bayes theorem need not be “true” in any sense and the posterior need not be obtained using Bayesian updates: the bound holds for any choice of the prior and posterior.

Denote the space of probability distributions over  $\mathcal{F}$  by  $\Delta(\mathcal{F})$ . There might be measurability issues in defining this for general function classes. So, we can assume that  $\mathcal{F}$  is a countable class for the purpose of the theorem below. Note, however, that the cardinality of the class  $\mathcal{F}$  never enters the bound explicitly anywhere. For any  $\rho \in \Delta(\mathcal{F})$ , define

$$\begin{aligned} R(\rho) &= \mathbb{E}_{f \sim \rho}[R(f)], \\ \widehat{R}(\rho) &= \mathbb{E}_{f \sim \rho}[\widehat{R}(f)]. \end{aligned}$$

In the context of classification, a randomized classifier that, when asked for a prediction, samples a function from a distribution  $\rho$  and uses it for classification, is called a *Gibbs classifier*. Note that  $R(\rho)$  is the risk of such a Gibbs classifier.

Also recall the definition of *Kullback-Leibler divergence* or *relative entropy*:

$$D(\rho||\pi) = \int_{f \in \mathcal{F}} \log \frac{\rho(f)}{\pi(f)} d\rho(f).$$

For real numbers  $p, q \in [0, 1]$  we define (with some abuse of notation):

$$D(p||q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}.$$

With these definitions, we can now state the PAC Bayesian theorem [38].

**Theorem 21 (PAC-Bayesian Theorem).** *Let  $\pi$  be a fixed distribution over  $\mathcal{F}$ . Then, we have, with probability at least  $1 - \delta$ ,*

$$\forall \rho \in \Delta(\mathcal{F}), D(\widehat{R}(\rho)||R(\rho)) \leq \frac{D(\rho||\pi) + \log \frac{2n}{\delta}}{n-1}.$$

*In particular, for any learning algorithm that, instead of returning a single function  $\widehat{f}_n \in \mathcal{F}$ , returns a distribution  $\widehat{\rho}_n$  over  $\mathcal{F}$ , we have, with probability at least  $1 - \delta$ ,*

$$D(\widehat{R}(\widehat{\rho}_n)||R(\widehat{\rho}_n)) \leq \frac{D(\widehat{\rho}_n||\pi) + \log \frac{2n}{\delta}}{n-1}.$$

To get more interpretable bounds from this statement, the following inequality is useful for  $0 \leq p < q \leq 1$ :

$$\frac{(p-q)^2}{2} \leq \frac{(p-q)^2}{2q} \leq D(p||q).$$

This implies that if  $D(p||q) \leq x$  then two inequalities hold:

$$\begin{aligned} q &\leq p + \sqrt{2x}, \\ q &\leq p + \sqrt{2px} + 2x. \end{aligned}$$

The first gives us a version of the PAC-Bayesian bound that is often presented:

$$R(\widehat{\rho}_n) - \widehat{R}(\widehat{\rho}_n) \leq \sqrt{\frac{2(D(\widehat{\rho}_n||\pi) + \log \frac{2n}{\delta})}{n-1}}.$$

The second gives us the interesting bound

$$R(\widehat{\rho}_n) - \widehat{R}(\widehat{\rho}_n) \leq \sqrt{\frac{2\widehat{R}(\widehat{\rho}_n)(D(\widehat{\rho}_n||\pi) + \log \frac{2n}{\delta})}{n-1}} + 2 \frac{D(\widehat{\rho}_n||\pi) + \log \frac{2n}{\delta}}{n-1}.$$

If the loss  $\widehat{R}(\widehat{\rho}_n)$  is close to zero, then the dominating term is the second term that scales as  $\frac{1}{n}$ . If not, then the first term, which scales as  $\frac{1}{\sqrt{n}}$ , dominates.

## 1.14.6 Computational aspects

So far we have only talked about sample complexity issues ignoring considerations of *computational complexity*. To properly address the latter, we need a formal *model of computation* and need to talk about how functions are represented by the learning algorithm. The branch of machine learning that studies these questions is called *computational learning theory*. We will now provide a brief introduction to this area.

### 1.14.6.1 The PAC model

The basic model in computational learning theory is the Probably Approximately Correct (PAC) model. It applies to learning binary valued functions and uses the 0–1 loss. Since a  $\pm 1$  valued function is specified unambiguously by specifying the subset of  $\mathcal{X}$  where it is one, we have a bijection between binary valued functions and *concepts*, which are simply subsets of  $\mathcal{X}$ . Thus we will use (binary valued) function and concept interchangeably in this section. Moreover, the basic PAC model considers what is sometimes called the *realizable* case. That is, there is a “true” concept in  $\mathcal{F}$  generating the data. This means that  $y_i = f(x_i)$  for some  $f \in \mathcal{F}$ . Hence the minimal risk in the class is zero:  $R(f_{\mathcal{F}}^*) = 0$ . Note, however, that the distribution over  $\mathcal{X}$  is still assumed to be arbitrary.

To define the computational complexity of a learning algorithm, we assume that  $\mathcal{X}$  is either  $\{0, 1\}^d$  or  $\mathbb{R}^d$ . We assume a model of computation that can store a single real number in a memory location and can perform any basic arithmetic operation (addition, subtraction, multiplication, division) on two real numbers in one unit of computation time.

The distinction between a concept and its representation is also an important one when we study computational complexity. For instance, if our concepts are convex polytopes in  $\mathbb{R}^d$ , we may choose a representation scheme that specifies the vertices of the polytope. Alternatively, we may choose to specify the linear equalities describing the faces of the polytope. Their vertex-based and face-based representations can differ exponentially in size. Formally, a *representation scheme* is a mapping  $\text{rep} : (\Sigma \cup \mathbb{R})^* \rightarrow \mathcal{F}$  that maps a representation (consisting of symbols from a finite alphabet  $\Sigma$  and real numbers) to a concept. As noted above, there could be many  $\sigma$ ’s such that  $\text{rep}(\sigma) = f$  for a given concept  $f$ . The size of representation is assumed to be given by a function  $\text{size} : (\Sigma \cup \mathbb{R})^* \rightarrow \mathbb{N}$ . A simple choice of size could be simply the number of symbols and real numbers that appear in the representation. Finally, the size of a concept  $f$  is simply the size of its smallest representation:

$$\text{size}(f) = \min_{\sigma: \text{rep}(\sigma)=f} \text{size}(\sigma).$$

The last ingredient we need before can give the definition of efficient learnability in the PAC model is the representation class  $\mathcal{H}$  used by the algorithm for producing its output. We call  $\mathcal{H}$  the *hypothesis* class and it is assumed that there is a representation in  $\mathcal{H}$  for every function in  $\mathcal{F}$ . Moreover, it is required that for every  $x \in \mathcal{X}$  and any  $h \in \mathcal{H}$ , we can evaluate  $h(x)$  ( $= \text{rep}(h)(x)$ ) in time polynomial in  $d$  and  $\text{size}(h)$ .

Recall that we have assumed  $\mathcal{X} = \{0, 1\}^d$  or  $\mathbb{R}^d$ . We say that an algorithm *efficiently PAC learns*  $\mathcal{F}$  using hypothesis class  $\mathcal{H}$  if, for any  $f \in \mathcal{F}$ , given access to iid examples  $(x_i, f(x_i)) \in \mathcal{X} \times \{\pm 1\}$ ,

inputs  $\varepsilon \in (0, 1)$  (accuracy),  $\delta \in (0, 1)$  (confidence), and  $\text{size}(f)$ , the algorithm satisfies the following conditions:

1. With probability at least  $1 - \delta$ , it outputs a hypothesis  $h \in \mathcal{H}$  with  $R(h) \leq \varepsilon$ .
2. It runs in time polynomial in  $\frac{1}{\varepsilon}, \frac{1}{\delta}, \text{size}(f)$ , and  $d$ .

For examples of classes  $\mathcal{F}$  that are efficiently PAC learnable (using some  $\mathcal{H}$ ), see the texts [12, 18, 20].

### 1.14.6.2 Weak and strong learning

In the PAC learning definition, the algorithm must be able to achieve arbitrarily small values of  $\varepsilon$  and  $\delta$ . A weaker definition would require the learning algorithm to work only for fixed choices of  $\varepsilon$  and  $\delta$ . We can modify the definition of PAC learning by removing  $\varepsilon$  and  $\delta$  from the input to the algorithm. Instead, we assume that there are fixed polynomials  $p_\delta(\cdot, \cdot)$  and  $p_\varepsilon(\cdot, \cdot)$  such that the algorithm outputs  $h \in \mathcal{H}$  that satisfies

$$R(h) \leq \frac{1}{2} - \frac{1}{p_\varepsilon(d, \text{size}(f))}$$

with probability at least

$$\frac{1}{p_\delta(d, \text{size}(f))}.$$

Such an algorithm is called a *weak PAC learning* algorithm for  $\mathcal{F}$ . The original definition, in contrast, can be called a definition of *strong PAC learning*. The definition of weak PAC learning does appear quite weak. Recall that an accuracy of  $\frac{1}{2}$  can be trivially achieved by an algorithm that does not even look at the samples but simply outputs a hypothesis that outputs a random  $\pm 1$  sign (each with probability a half) for any input. The desired accuracy above is just an inverse polynomial away from the trivial accuracy guarantee of  $\frac{1}{2}$ . Moreover, the probability with which the weak accuracy is achieved is not required to be arbitrarily close to 1.

A natural question to ask is whether a class that is weakly PAC learnable using  $\mathcal{H}$  is also strongly learnable using  $\mathcal{H}$ ? The answer, somewhat surprisingly, is “Yes.” So, the weak PAC learning definition only appears to be a relaxed version of strong PAC learning. Moreover, a weak PAC learning algorithm, given as a subroutine or blackbox, can be converted into a strong PAC learning algorithm. Such a conversion is called *boosting*.

A boosting procedure has to start with a weak learning algorithm and boost two things: the confidence and the accuracy. It turns out that boosting the confidence is easier. The basic idea is to run the weak learning algorithm several times on independent samples. Therefore, the crux of the boosting procedure lies in boosting the accuracy. Boosting the accuracy seems hard until one realizes that the weak learning algorithm does have one strong property: it is guaranteed to work no matter what the underlying distribution of the examples is. Thus, if a first run of the weak learning only produces a hypothesis  $h_1$  with slightly better accuracy than  $\frac{1}{2}$ , we can make the second run focus only on those examples where  $h_1$  makes a mistake. Thus, we will focus the weak learning algorithm on “harder portions” of the input distribution. While this simple idea does not directly work, a variation on the same theme was shown to work by Schapire [39]. Freund [40] then presented a simpler boosting algorithm. Finally, a much more practical boosting procedure, called AdaBoost, was discovered by them jointly [10].

AdaBoost counts as one of the great practical success stories of computational learning theory. For more details on boosting, AdaBoost, and the intriguing connections of these ideas to game theory and statistics, we refer the reader to [41].

### 1.14.6.3 Random classification noise and the SQ model

Recall that the basic PAC model assumes that the labels are generated using a function  $f$  belonging to the class  $\mathcal{F}$  under consideration. Our earlier development of statistical tools for obtaining sampling complexity estimates did not require such an assumption (though the rates did depend on whether  $R(f_{\mathcal{F}}^*) = 0$  or not).

There are various *noise models* that extend the PAC model by relaxing the noise-free assumption and thus making the problem of learning potentially harder. Perhaps the simplest is the *random classification noise* model [42]. In this model, when the learning algorithm queries for the next labeled example of the form  $(x_i, f(x_i))$ , it receives it with probability  $1 - \eta$  but with probability  $\eta$ , it receives  $(x_i, -f(x_i))$ . That is, with some probability  $\eta < \frac{1}{2}$ , the label is flipped. The definition of efficient learnability remains the same except that now the learning algorithm has to run in time polynomial in  $\frac{1}{\epsilon}, \frac{1}{\delta}, d, \text{size}(f)$  and  $\frac{1}{1-2\eta_0}$  where  $\eta_0 < \frac{1}{2}$  is an input to the algorithm and is an upper bound on  $\eta$ .

Kearns [43] introduced the statistical query (SQ) model where learning algorithms do not directly access the labeled example but only make statistical queries about the underlying distribution. The queries take the form  $(\chi, \tau)$  where  $\chi : \mathcal{X} \times \{\pm 1\} \rightarrow \{\pm 1\}$  is a binary valued function of the labeled examples and  $\tau \in (0, 1]$  is a tolerance parameter. Given such a query, the oracle in the SQ model responds with an estimate of  $P(\chi(x, f(x)) = +1)$  that is accurate to within an additive error of  $\tau$ . It is easy to see, using Hoeffding's inequality, that given access to the usual PAC oracle that provides iid examples of the form  $(x_i, f(x_i))$ , we can simulate the SQ oracle by just drawing a sample of size polynomial in  $\frac{1}{\tau^2}$  and  $\log \frac{1}{\delta}$  and estimating the probability  $P(\chi(x, f(x)) = 1)$  based on it. The simulation will succeed with probability  $1 - \delta$ . Kearns gave a more complicated simulation that mimics the SQ oracle given access to only a PAC oracle with random misclassification noise.

**Lemma 22.** *Given a query  $(\chi, \tau)$ , the probability  $P(\chi(x, f(x)) = 1)$  can be estimated to within  $\tau$  error, with probability at least  $1 - \delta$ , using*

$$O\left(\frac{1}{\tau^2(1-2\eta)^2} \log \frac{1}{\delta}\right)$$

examples that have random misclassification noise at rate  $\eta < \frac{1}{2}$  in them.

This lemma means that learning algorithm that learns a concept class  $\mathcal{F}$  in the SQ model can also learn  $\mathcal{F}$  in the random classification noise model.

**Theorem 23.** *If there is an algorithm that efficiently learns  $\mathcal{F}$  from statistical queries using  $\mathcal{H}$ , then there is an algorithm that efficiently learns  $\mathcal{F}$  using  $\mathcal{H}$  in the random classification noise model.*

### 1.14.6.4 The agnostic PAC model

The agnostic PAC model uses the same definition of learning as the one we used in Section 1.14.2.3). That is, the distribution of examples  $(x, y)$  is arbitrary. In particular, it is not assumed that  $y = f(x)$  for

any  $f$ . The goal of the learning algorithm is to output a hypothesis  $h \in \mathcal{H}$  with  $R(h) - R(f_{\mathcal{F}}^*) \leq \varepsilon$  with probability at least  $1 - \delta$ . We say that a learning algorithm efficiently learns  $\mathcal{F}$  using  $\mathcal{H}$  in the agnostic PAC model if the algorithm not only outputs a probably approximately correct hypothesis  $h$  but also runs in time that is polynomial in  $\frac{1}{\varepsilon}$ ,  $\frac{1}{\delta}$ , and  $d$ .

The agnostic model has proved to be a very difficult one for exhibiting efficient learning algorithms. Note that the statistical issue is completely resolved: a class  $\mathcal{F}$  is learnable iff it has finite VC dimension. However, the “algorithm” implicit in this statement is ERM which, even for the class of halfspaces

$$\{x \mapsto \text{sign}(\langle w, x \rangle - \theta) : w \in \mathbb{R}^d, \theta \in \mathbb{R}\},$$

leads to NP-hard problems. For instance, see [44] for very strong negative results about agnostic learning of halfspaces. However, these results apply to *proper* agnostic learning only. That is, the algorithm outputs a hypothesis that is also a halfspace. An efficient algorithm for learning halfspaces in the agnostic PAC model using a general hypothesis class would be a major breakthrough.

### 1.14.6.5 The mistake bound model

The mistake bound model is quite different from the PAC model. Unlike the PAC model, no assumption is made on the distribution of the inputs. There is also no distinction between a training phase where the learner uses a batch of examples to learn a hypothesis and a test phase where the risk of the learned hypothesis determines the learner’s expected loss. Instead, learning happens in a series of trials (or rounds) in an *online* fashion. At any given round  $t \in \mathbb{N}$ , the order of events is as follows:

- Learner receives  $x_t \in \mathcal{X}$  where  $\mathcal{X} = \{0, 1\}^d$  or  $\mathbb{R}^d$ .
- Learner outputs a label  $\hat{y}_t \in \mathcal{Y}$ .
- Learner receives the correct label  $y_t = f(x_t)$ .

Note that the basic model assumes that  $f$  is a true function generating the labels (the case when such an  $f$  is assumed to exist is often called the *realizable case*). The identity of this function is, of course, hidden from the learner. The goal of the learner is to minimize the total number of *mistakes* it makes:

$$\sum_{t=1}^{\infty} \ell(\hat{y}_t, f(x_t)),$$

where  $\ell$  is the 0–1 loss. If a learning algorithm can guarantee, for any choice  $f \in \mathcal{F}$  of the true concept, that the total mistake bound will be a polynomial function of  $d$  and  $\text{size}(f)$  as well as the computation per round is polynomial in the same parameters, then we say that the algorithm *efficiently learns  $\mathcal{F}$  in the mistake bound model*.

Recall that, in the PAC model, VC dimension of  $\mathcal{F}$  characterizes learnability of  $\mathcal{F}$  if we ignore computational considerations. Moreover, VC dimension characterizes learnability in the agnostic PAC model as well. In the mistake bound model, it is the Littlestone dimension [45] whose finiteness provides a necessary and sufficient condition for learnability. There is also an *agnostic version* of the mistake bound model where we drop the assumption that  $y_t = f(x_t)$  for some  $f \in \mathcal{F}$ . It is also known that the Littlestone dimension characterizes learnability (ignoring efficiency) even in the agnostic online mistake bound model [46].

The Littlestone dimension of a class  $\mathcal{F} \subseteq \{\pm 1\}^{\mathcal{X}}$  is defined as follows. A  $\mathcal{X}$ -valued tree (or simply an  $\mathcal{X}$ -tree) of height  $t$  is a complete binary tree of height  $t$  whose *internal* nodes are labeled with instances from  $\mathcal{X}$ . A (complete binary) tree of height  $t$  has exactly  $2^t$  leaves which we identify, from left to right, with the  $2^t$  sequences length  $t$   $\pm 1$  sequences ordered lexicographically (with  $-1$  taking precedence over  $+1$  in determining the lexicographic order). For example, the leftmost leaf corresponds to all  $-1$ 's sequence whereas the rightmost leaf corresponds to the all  $+1$ 's sequences. If  $\lambda$  is a leaf, we denote its associated  $\pm 1$  sequence by  $\epsilon(\lambda)$ .

An  $\mathcal{X}$ -tree of height  $t$  is said to be *shattered* by  $\mathcal{F}$  if for each of the  $2^t$  leaves the following is true: there is a function  $f \in \mathcal{F}$  such that the sequence of values taken by  $f$  on the internal nodes on the path from the root to the leaf  $\lambda$  is exactly  $\epsilon(\lambda)$ . The *Littlestone dimension* of  $\mathcal{F}$  is the height of the tallest tree that is shattered by  $\mathcal{F}$ .

It is easy to see that if an  $\mathcal{X}$ -valued sequence  $(x_1, \dots, x_t)$  is shattered by  $\mathcal{F}$  then the  $\mathcal{X}$ -tree of height  $t$  obtained by repeating  $x_i$  across level  $i$  for  $i \in [t]$  is also shattered by  $\mathcal{F}$ . This proves the following relationship between  $\text{VCdim}(\mathcal{F})$  and  $\text{Ldim}(\mathcal{F})$ .

**Theorem 24.** *For any  $\mathcal{F} \subseteq \{\pm 1\}^{\mathcal{X}}$  we have*

$$\text{VCdim}(\mathcal{F}) \leq \text{Ldim}(\mathcal{F}).$$

A converse to the above theorem is not possible. Indeed, the class of threshold functions on  $\mathbb{R}$ :

$$\mathcal{F} = \{x \mapsto \text{sign}(x - \theta) : \theta \in \mathbb{R}\}$$

has  $\text{VCdim}(\mathcal{F}) = 1$  but  $\text{Ldim}(\mathcal{F}) = \infty$ . A finite class  $\mathcal{F}$  has a finite Littlestone dimension satisfying the upper bound  $\text{Ldim}(\mathcal{F}) \leq \log_2(|\mathcal{F}|)$ .

#### 1.14.6.5.1 Halving and weighted majority

Note that we proved risk bounds for a finite class in the probabilistic framework using Hoeffding's inequality along with a simple union bound. Here we show how to deal with a finite class in the online mistake bound model. In the realizable case, a simple algorithm called *halving* is guaranteed to make no more than  $\log_2(|\mathcal{F}|)$  mistakes.

At the beginning of round  $t$ , the halving algorithm considers a subset  $\mathcal{F}_t \subseteq \mathcal{F}$  of the original function class. Predictions are made by taking a majority vote over  $\mathcal{F}_{t-1}$ :

$$\hat{y}_t = \text{sign} \left( \sum_{f \in \mathcal{F}_{t-1}} f(x_t) \right).$$

Therefore, if a mistake is made, then  $|\mathcal{F}_{t-1}|$  reduces by at least a factor of 2. Moreover, the true function is never eliminated since it will always satisfy  $f(x_t) = y_t$ . Therefore, if  $M_t$  is the number of mistakes made by the halving algorithm in  $t$  rounds, then we have

$$1 \leq |\mathcal{F}_t| \leq \frac{|\mathcal{F}_0|}{2^{M_t}} = \frac{|\mathcal{F}|}{2^{M_t}}.$$

This implies that  $M_t \leq \log_2(|\mathcal{F}|)$ . Thus we have proved a mistake bound for halving.

---

**Algorithm 1.** Halving

---

```

Initialize  $\mathcal{F}_0 = \mathcal{F}$ 
for  $t = 1, 2, \dots$  do
  Receive  $x_t$ 
  Predict  $\hat{y}_t$  using a majority vote over  $\mathcal{F}_{t-1}$ 
  Receive true label  $y_t$ 
  if  $\hat{y}_t \neq y_t$  then
     $\mathcal{F}_t = \{f \in \mathcal{F}_{t-1} : f(x_t) = y_t\}$ 
  end if
end for

```

---

**Theorem 25.** *The halving algorithm when run on a finite class  $\mathcal{F}$  enjoys a mistake bound of  $\log_2(|\mathcal{F}|)$ .*

In the non-realizable case, halving does not work directly but a suitable modification does. Since no function in the class can be guaranteed to be correct all the time, it does not make sense to eliminate functions that make mistakes. The crucial idea here is to keep *weights* for different functions in the class. Then, when a function makes a mistake, its weight is multiplied by a factor of  $e^{-\eta} < 1$  where  $\eta > 0$  is a parameter. The prediction are now taken using a *weighted majority* vote over all  $\mathcal{F}$ :

$$\hat{y}_t = \text{sign} \left( \sum_{f \in \mathcal{F}} \rho_{t-1}(f) f(x_t) \right). \quad (14.6)$$

---

**Algorithm 2.** Weighted Majority

---

```

Initialize  $\rho_0(f) = \pi(f)$  for some distribution  $\pi$  over  $\mathcal{F}$ 
for  $t = 1, 2, \dots$  do
  Receive  $x_t$ 
  Predict  $\hat{y}_t$  using a  $\rho_{t-1}$ -weighted majority vote over  $\mathcal{F}_{t-1}$ 
  Receive true label  $y_t$ 
  for  $f \in \mathcal{F}$  do
     $\rho_t(f) = \rho_{t-1}(f) \exp(-\eta \ell(f(x_t), y_t))$ 
  end for
end for

```

---

**Theorem 26.** *At any the end of any given round, let  $M_f$  be the number of mistakes made by a function  $f \in \mathcal{F}$  so far. Then the number of mistakes  $M$  made by weighted majority so far is bounded as*

$$\forall f \in \mathcal{F}, M \leq 2 \cdot \frac{\eta M_f + \log \frac{1}{\pi(f)}}{1 - e^{-\eta}}.$$

In particular, using the uniform distribution  $\pi(f) = 1/|\mathcal{F}|$ , we get

$$\forall f \in \mathcal{F}, M \leq 2 \cdot \frac{\eta M_f + \log |\mathcal{F}|}{1 - e^{-\eta}}.$$

**Proof.** Fix a round index  $T \geq 1$  and let  $M$  and  $M_f$  be defined as the numbers of mistakes in the first  $T$  rounds. Let  $F_t$  be the fraction of weights on the functions that made a mistake on round  $t \leq T$ . Since we make a mistake if and only if  $F_t \geq \frac{1}{2}$ , we have

$$M = \sum_{t=1}^T \mathbf{1} \left[ F_t \geq \frac{1}{2} \right] \leq \sum_{t=1}^T 2F_t. \quad (14.7)$$

Define the total weight

$$W_t = \sum_{f \in \mathcal{F}} \rho_t(f).$$

At round  $t$ , the total weight changes as

$$W_{t+1} \leq (1 - (1 - e^{-\eta})F_t)W_t.$$

Hence the final total weight is

$$W_{T+1} = W_0 \prod_{t=1}^T (1 - (1 - e^{-\eta})F_t).$$

Note that  $W_0 = 1$ . If the function  $f$  made  $M_f$  mistakes, its weight is

$$\rho_{t+1}(f) = \pi(f)e^{-\eta M_f}.$$

Since  $\rho_{t+1}(f) \leq W_{t+1}$ , we have

$$\pi(f)e^{-\eta M_f} \leq \prod_{t=1}^T (1 - (1 - e^{-\eta})F_t).$$

Taking negative logs of both sides reverses the inequality giving

$$\log \frac{1}{\pi(f)} + \eta M_f \geq - \sum_{t=1}^T \log (1 - (1 - e^{-\eta})F_t).$$

Using the elementary but useful inequality  $-\log(1-x) > x$  gives

$$(1 - e^{-\eta}) \sum_{t=1}^T F_t \leq \eta M_f + \log \frac{1}{\pi(f)}.$$

Combining this with the bound (14.7) proves the theorem. □

It turns out we can shave off a factor of 2 in the above mistake bound by using a randomized version of the weighted majority algorithm. Instead of taking the majority vote with the weights  $\rho_{t-1}(f)$ , we can consider a Gibbs classifier that chooses a function  $f_t$  randomly from  $\mathcal{F}$  with probability proportional to  $\rho_{t-1}(f)$  and then outputs  $\hat{y}_t = f_t(x_t)$ . This makes the number of mistakes a random variable but the bound (14.7) turns into an equality without the factor of 2. That is, for the randomized weighted majority algorithm

$$\mathbb{E}[M] = \sum_{t=1}^T F_t.$$

Thus we have the following result.

**Theorem 27.** Suppose that we change the weighted majority step (14.6) to the randomized weighted majority step

$$f_t \sim \rho_{t-1}, \quad \hat{y}_t = \text{sign}(f_t(x_t)).$$

Then, this randomized version of weighted majority satisfies the expected mistake bound

$$\forall f \in \mathcal{F}, \quad \mathbb{E}[M] \leq \frac{\eta M_f + \log \frac{1}{\pi(f)}}{1 - e^{-\eta}}.$$

### 1.14.6.5.2 Perceptron and winnow

We now consider two online algorithms that learn a linear threshold function. *Perceptron* is a classic algorithm that dates back to the 1960s. The *Winnow* algorithm was proposed by Littlestone [45] as a better alternative when there are many irrelevant features in the inputs.

Both algorithms follow the general schema given in Algorithm 3. Perceptron uses the *additive* update rule:

$$\text{LTF-UPDATE}(w_{t-1}, \eta, x_t, y_t) = w_{t-1} - \eta y_t x_t,$$

whereas Winnow uses a *multiplicative* update rule

$$\text{LTF-UPDATE}(w_{t-1}, \eta, x_t, y_t) = \frac{w_{t-1} \odot \exp(-\eta y_t x_t)}{Z_t},$$

where  $\odot$  and  $\exp$  denote entry wise multiplication and exponentiation respectively. The normalization constant  $Z_t = \sum_{j=1}^d w_{t-1,j} \exp(-\eta y_t x_{t,j})$  ensures that the weight vector remains a probability distribution. Also note that in Perceptron, we use the initialization  $w_0 = \mathbf{0}$  whereas Winnow initializes  $w_0$  with the uniform distribution over the  $d$  features.

To give a mistake bound for Perceptron and Winnow, we will assume that we are in realizable case. That is, there is some weight vector  $w^*$  that can perfectly classify all the examples that the learner sees. Moreover, we can assume that there is a *margin* available in the correct classifications. For a correct classification, it should simply be the case that  $y_t \langle w^*, x_t \rangle > 0$ . The margin condition will ensure that there is a lower bound  $\gamma > 0$  such that

$$\forall t, \quad y_t \langle w^*, x_t \rangle \geq \gamma. \tag{14.8}$$

---

**Algorithm 3.** Online Learning of Linear Threshold Functions

---

```

Initialize  $w_0$ 
for  $t = 1, 2, \dots$  do
  Receive  $x_t$ 
  Predict  $\hat{y}_t = \text{sign}(\langle w, x_t \rangle)$ 
  Receive true label  $y_t$ 
  if  $\hat{y}_t \neq y_t$  then
     $w_t \leftarrow \text{LTF-UPDATE}(w_t, \eta, x_t, y_t)$ 
  else
     $w_t \leftarrow w_{t-1}$ 
  end if
end for

```

---

**Theorem 28.** Suppose there exists a  $w^*$  satisfying the margin condition (14.8). Then Perceptron with learning rate  $\eta = 1$  enjoys the mistake bound

$$M \leq \frac{\|w^*\|_2^2 \cdot B_2^2}{\gamma^2},$$

where  $B_2 = \max_t \|x_t\|_2$ .

**Proof.** Let  $T \geq 1$  and note that the number  $M = M_T$  of mistakes till time  $T$  is simply

$$M_T = \sum_{t=1}^T m_t$$

where  $m_t = \mathbf{1}[\hat{y} \neq y_t] = \mathbf{1}[y_t \langle w_t, x_t \rangle \leq 0]$ . Let us calculate how the inner product of  $w_t$  with the vector  $w^*$  evolves. Note that we can write

$$w_t = w_{t-1} + m_t y_t x_t.$$

Thus, we have,

$$\begin{aligned} \langle w^*, w_t \rangle &= \langle w^*, w_{t-1} + m_t y_t x_t \rangle \\ &= \langle w^*, w_{t-1} \rangle + m_t y_t \langle w^*, x_t \rangle \\ &\geq \langle w^*, w_{t-1} \rangle + m_t \gamma, \end{aligned}$$

where the last step is due to the margin assumption (14.8). Summing the above inequality for  $t = 1, \dots, T$  yields

$$\gamma M_T + \langle w^*, w_0 \rangle \leq \langle w^*, w_T \rangle.$$

Since  $w_0 = \mathbf{0}$  and  $\langle w^*, w_T \rangle \leq \|w^*\|_2 \|w_T\|_2$ , we get

$$\gamma M_T \leq \|w^*\|_2 \|w_T\|_2. \tag{14.9}$$

Let us now try to upper bound  $\|w_T\|_2$ . We have,

$$\begin{aligned}\|w_t\|_2^2 &= \|w_t + m_t y_t x_t\|_2^2 \\ &= \|w_t\|_2^2 + 2m_t y_t \langle w_t, x_t \rangle + m_t^2 y_t^2 \|x_t\|_2^2 \\ &\leq \|w_t\|_2^2 + m_t^2 y_t^2 \|x_t\|_2^2,\end{aligned}$$

where the last line is true because  $y_t \langle w_t, x_t \rangle \leq 0$  when  $m_t > 0$ . Also note that  $y_t^2 = 1$ ,  $m_t^2 = m_t$ , and  $\|x_t\|_2 \leq B_2$ . Therefore,

$$\|w_t\|_2^2 \leq \|w_t\|_2^2 + m_t B_2^2.$$

Summing this for  $t = 1, \dots, T$  gives

$$\|w_T\|_2^2 \leq \|w_0\|^2 + M_T B_2^2 = M_T B_2^2.$$

Combining this with (14.9) gives

$$\gamma M_T \leq \|w^*\|_2 \sqrt{M_T} B_2,$$

which gives us the final bound

$$M_T \leq \frac{\|w^*\|_2^2 \cdot B_2^2}{\gamma^2}. \quad \square$$

**Theorem 29.** Suppose there exists a  $w^*$  with positive entries that satisfies the margin condition (14.8). Let  $B_\infty = \max_t \|x_t\|_\infty$ . Then Winnow enjoys the mistake bound

$$M \leq \frac{\log d}{C(\eta)}$$

whenever

$$C(\eta) = \left( \frac{\eta \gamma}{\|w^*\|_1} - \log \left( \frac{e^{\eta B_\infty} + e^{-\eta B_\infty}}{2} \right) \right) > 0.$$

In particular, by setting  $\eta$  optimally, we get,

$$M \leq 2 \frac{\|w^*\|_1^2 \cdot B_\infty^2}{\gamma^2}.$$

**Proof.** Let  $u^* = w^*/\|w^*\|_1$ . Since  $w^* > 0$  (inequality is entrywise),  $u^*$  is a probability distribution. Moreover, Winnow maintains a probability distribution  $w_t$  at all times. We will track the progress of the algorithm using the relative entropy

$$D(u^* || w_t) = \sum_{j=1}^d u_j^* \log \frac{u_j^*}{w_{t,j}}$$

between  $u^*$  and  $w_t$ .

Suppose a mistake occurs at time  $t$ . Then we have,

$$\begin{aligned}
 D(u^{\star}||w_t) - D(u^{\star}||w_{t-1}) &= \sum_{j=1}^d u_j^{\star} \log \frac{w_{t-1,j}}{w_{t,j}} \\
 &= \sum_{j=1}^d u_j^{\star} \log \frac{Z_t}{\exp(\eta y_t x_{t,j})} \\
 &= \log Z_t \sum_{j=1}^d u_j^{\star} - \eta y_t \sum_{j=1}^d u_j^{\star} x_{t,j} \\
 &= \log Z_t - \eta y_t \langle u^{\star}, x_t \rangle \\
 &\leq \log Z_t - \frac{\eta \gamma}{\|w^{\star}\|_1},
 \end{aligned} \tag{14.10}$$

where the last step is due to the margin assumption (14.8). Note that  $y_t x_{t,j} \in [-B_{\infty}, B_{\infty}]$ .  $\square$

For any  $\alpha \in [-B_{\infty}, B_{\infty}]$  and  $\eta > 0$ , we have the inequality

$$e^{\eta\alpha} \leq \frac{1 + \alpha/B_{\infty}}{2} e^{\eta B_{\infty}} + \frac{1 - \alpha/B_{\infty}}{2} e^{-\eta B_{\infty}}.$$

To see this consider a random variable  $Z$  that takes value  $\eta L$  with probability  $(1 + \alpha/B_{\infty})/2$  and takes value  $-\eta L$  with probability  $(1 - \alpha/B_{\infty})/2$ . Then the inequality above is claims that  $\exp(\mathbb{E}[Z]) \leq \mathbb{E}[\exp(Z)]$  which is indeed true by Jensen's inequality. Using the inequality above, we have

$$\begin{aligned}
 Z_t &= \sum_{j=1}^d w_{t-1,j} e^{\eta y_t x_{t-1,j}} \\
 &\leq \sum_{j=1}^d w_{t-1,j} \left( \frac{1 + y_t x_{t-1,j}/B_{\infty}}{2} e^{\eta B_{\infty}} + \frac{1 - y_t x_{t-1,j}/B_{\infty}}{2} e^{-\eta B_{\infty}} \right) \\
 &= \left( \frac{e^{\eta B_{\infty}} + e^{-\eta B_{\infty}}}{2} \right) \sum_{j=1}^d w_{t-1,j} + \left( \frac{e^{\eta B_{\infty}} - e^{-\eta B_{\infty}}}{2B_{\infty}} \right) \sum_{j=1}^d y_t w_{t-1,j} x_{t,j} \\
 &= \left( \frac{e^{\eta B_{\infty}} + e^{-\eta B_{\infty}}}{2} \right) + \left( \frac{e^{\eta B_{\infty}} - e^{-\eta B_{\infty}}}{2B_{\infty}} \right) \sum_{j=1}^d y_t \langle w_{t-1}, x_t \rangle \\
 &\leq \left( \frac{e^{\eta B_{\infty}} + e^{-\eta B_{\infty}}}{2} \right),
 \end{aligned}$$

where the last step is true because, in a mistake round,  $y_t \langle w_{t-1}, x_t \rangle \leq 0$ . Combining this bound on  $Z_t$  with (14.10) gives us, on any mistake round,

$$D(u^{\star}||w_t) - D(u^{\star}||w_{t-1}) \leq -C(\eta),$$

where, we have defined

$$C(\eta) = \frac{\eta\gamma}{\|w^*\|_1} - \log\left(\frac{e^{\eta B_\infty} + e^{-\eta B_\infty}}{2}\right).$$

Thus, for any round,

$$D(u^*||w_t) - D(u^*||w_{t-1}) \leq -C(\eta)m_t,$$

where  $m_t = \mathbf{1}[\hat{y}_t \neq y_t]$ . Summing this over  $t = 1, \dots, T$  gives

$$D(u^*||w_T) - D(u^*||w_0) \leq -C(\eta) \sum_{t=1}^T m_t = -C(\eta)M_T.$$

Since relative entropy is always non-negative  $D(u^*||w_T) \geq 0$ . On the other hand,

$$D(u^*||w_0) = \sum_{j=1}^d u_j^* \log(u_j^* d) \leq \sum_{j=1}^d u_j^* \log d = \log d.$$

Thus, we have

$$-\log d \leq -C(\eta)M_T$$

which gives, whenever  $C(\eta) > 0$ ,

$$M_T \leq \frac{\log d}{C(\eta)}.$$

By choosing  $\eta$  to maximize  $C(\eta)$ , we get

$$\eta = \frac{1}{B_\infty} \log\left(\frac{L + \gamma/\|w^*\|_1}{L - \gamma/\|w^*\|_1}\right)$$

which yields the bound

$$M_T \leq \frac{\log d}{g\left(\frac{\gamma}{B_\infty \|w^*\|_1}\right)},$$

where

$$g(x) = \frac{1+x}{2} \log(1+x) + \frac{1-x}{2} \log(1-x).$$

The second statement of the theorem now follows because  $g(x) \geq x^2/2$  for  $x \in [-1, 1]$ .

## 1.14.7 Beyond the basic probabilistic framework

Our overview of learning theory emphasized the classical setting in which a learning algorithm sees fully labeled data that is an iid draw from an unknown distribution. There are many extensions possible

to this basic setting. We mention a few directions that have been explored by researchers. Many of these continue to evolve actively as new results appear.

*Richer Output Spaces:* We mostly talked about classification and regression. One can certainly consider richer output spaces. The field of *structured prediction* [47] deals with exponentially large output spaces consisting of graphs, trees, or other combinatorial objects. Another output space that arises in ranking applications is the space of permutations of a finite set of objects. There has also been work on learning vector valued functions [48].

*Learning with Dependent Data:* The iid assumption has also been relaxed. Aldous and Vazirani [49], Gamarnik [50] consider Markov chains. Irle [51], Meir [52], Vidyasagar [17], Lozano et al. [53], Mohri and Rostamizadeh [54], Steinwart et al. [55] consider mixing processes. There is also a negative result [56] showing that there is no universally consistent learning algorithm for stationary ergodic processes.

*Learning with Adversarially Generated Data:* As in the mistake bound model, it is possible to develop a theory of learning from online adversarially generated data provided one defines learnability in terms of low regret

$$\sum_{t=1}^T \ell(f_{t-1}(x_t), y_t) - \inf_{f \in \mathcal{F}} \ell(f(x_t), y_t)$$

which compares the performance of the learner's functions  $f_1, \dots, f_T$  to the best function chosen in hindsight (note that the learner has to commit to  $f_{t-1}$  seeing only examples till  $(x_{t-1}, y_{t-1})$ ). Analogues of Rademacher complexity, covering numbers, and fat shattering dimension have been developed [57]. This topic has fascinating connections to information theory and game theory. See [21] for a comprehensive overview.

*Active Learning:* In active learning, we consider situations where the learner has some control over the data it uses to learn. This is in contrast to the basic iid model where the learner passively sees iid data drawn from a distribution. In particularly useful model of active learning, we assume that the learning has access to *unlabeled* points  $X_1, \dots, X_n$  drawn iid from a distribution either all at once (a *pool*) or one at a time (a *stream*). The learner can only query for labels of examples in the pool or the stream. The *label complexity* of the learner is the number of label queries it needs to make in order to achieve a desired accuracy with a given level of confidence. In a lot of situations, unlabeled data is cheap and easy to get. So, it makes sense to charge the learner only for the number of labels it requires. The label complexity of different problems in an active learning has been studied (see, for example, [58] and references therein). Given the practical importance of reducing the number of labels required and the relatively unexplored nature of active learning, a lot remains to be done.

*Semisupervised Learning:* Semisupervised learning refers to learning with labeled data and large amounts of unlabeled data. A lot of work has been done in this area including methods that try to learn classifiers whose decision boundary passed through low density regions of the unlabeled data to graph based methods that use regularization to penalize functions that are nonsmooth along edges of a graph built from unlabeled data. For a good collection of articles on semisupervised learning, see the collection [59]. In the same collection, Balcan and Blum propose a PAC style framework for semisupervised learning.

*Learning Multiple Tasks:* In multitask learning, the learner is faced with learning multiple related tasks. Because of task relatedness, we expect the learner to require fewer samples when the tasks are learned together rather than separately. Many approaches try to encode task relatedness into a

regularization function that couples all the tasks together. Some important advances in our theoretical understanding of multitask learning have been made ([60–63]) but a comprehensive theory, if one exists, remains to be found.

## 1.14.8 Conclusions and future trends

Understanding intelligence and constructing intelligent machines is a great scientific challenge. An understanding of learning is central to an understanding of intelligence. Whether we talk about learning on an evolutionary time scale or learning within the lifespan of an organism, learning plays a crucial role in the development of intelligent behavior. What learning theory has attempted to do so far is provide a solid mathematical framework within which to pose and solve questions regarding the process of learning. Admittedly, the most mature part of learning theory is the theory of supervised learning including classification and regression problems. But this is just a beginning rather than the end. We hope that future research will make learning theory much richer and applicable to a broader variety of problems in both natural and artificial systems.

Learning theory is an interdisciplinary field. It draws ideas and inspiration from a variety of disciplines including biology, computer science, economics, philosophy, psychology, statistical physics, and statistics. It attempts to achieve clarity and rigor by using the language of mathematics to precisely state assumptions and prove results. We expect that interdisciplinary cross-fertilization of ideas will continue to enrich and give new directions to learning theory. For example, learning theory has much to contribute to problems arising in diverse areas such as control [17] and dynamical systems [64], learning in games [65], privacy [66], evolution [67], and mechanism design [68].

---

## Glossary

AdaBoost	Acronym for Adaptive Boosting, a popular boosting algorithm
Bayes classifier	In classification problems with 0–1 loss, a classifier with the least risk
Boosting	The process of converting a weak learning algorithm (that is, one that outputs classifiers whose performance is just a little bit better than random guessing) into a strong one
Concentration inequality	A probabilistic inequality stating that some random variable will not deviate too far from its mean or median
Excess risk	The difference between the risk of a given function and the minimum possible risk over a function class
Empirical risk	Same as risk except that the expectation under the unknown data distribution is replaced with an empirical average over the samples observed
Empirical risk minimization	A learning rule that minimizes the empirical risk to choose a prediction function

ERM	Empirical Risk Minimization
Fat shattering dimension	A scale sensitive generalization of the VC dimension.
Generalization error bound	An upper bound stating that the risk of a learning rule is not going to differ too much from its empirical risk
Kernel	A symmetric positive semidefinite function
Littlestone dimension	A combinatorial quantity associated with a class of binary valued functions. Plays a fundamental role in determining worst case bounds in the online mistake bound model
Loss function	A function used to measure the quality of a prediction given the true label.
Mistake bound model	A model of learning that, unlike the PAC model, does not assume anything about the way data is generated
Model selection	The task of selecting an appropriate statistical model from a family based on available data
PAC model	A computational model for analyzing the resources (time, memory, samples) required for performing learning tasks. Originally defined for the task of binary classification using the 0–1 loss. The acronym PAC stands for “Probably Approximately Correct”
Rademacher complexity	A quantity associated with a function class that measures its capacity to overfit by measuring how well the function class can fit randomly generated $\pm 1$ signs
Regression function	In regression problems with the squares loss, the prediction function with the least risk
Reproducing kernel Hilbert space	A Hilbert space of functions for which there exists a kernel with the reproducing property: point evaluations of functions can be written as linear functionals
Risk	The expected loss of a prediction function under the underlying unknown distribution generating input, output pair
Sample complexity	The number of samples requires to achieve a given level of accuracy (usually with high probability)
Sample compression	An approach to obtaining generalization error bounds for algorithms whose output can be reconstructed from a small number of examples belonging to the original sample
Stability	The property of a learning algorithm that ensures that its output does not change much if the training data set is changed slightly
Supervised learning	The task of learning a functional dependence between an input space and an output or label space using given examples of input, output pairs
Universal consistency	The property of a learning rule to converge to the minimum possible risk as the number of samples grows to infinity

Vapnik-Chervonenkis dimension	A combinatorial quantity associated with a binary valued function class that serves as a measure of the capacity of the function class to overfit the data
VC dimension	Vapnik-Chervonenkis dimension

### *Relevant Theory:* Machine Learning

See this Volume, Chapter 14 Learning Theory

See this Volume, [Chapter 15](#) Neural Networks

See this Volume, [Chapter 16](#) Kernel Methods and SVMs

See this Volume, [Chapter 17](#) Online Learning in Reproducing Kernel Hilbert Spaces

See this Volume, [Chapter 18](#) Introduction to Probabilistic Graphical Models

See this Volume, [Chapter 20](#) Clustering

See this Volume, [Chapter 22](#) Semi-supervised Learning

See this Volume, [Chapter 23](#) Sparsity-Aware Learning and Compressed Sensing: An Overview

See this Volume, [Chapter 25](#) A Tutorial on Model Selection

---

## Relevant websites

- Scholarpedia has a section devoted to machine learning containing links to articles written by leading researchers.  
[http://www.scholarpedia.org/article/Category:Machine\\_learning/](http://www.scholarpedia.org/article/Category:Machine_learning/)
- The Encyclopedia of Machine Learning is a valuable online resource available through subscription.  
<http://www.springerlink.com/content/978-0-387-30768-8/>
- The Journal of Machine Learning Research a leading open access journal publishing high quality articles in machine learning and related topics.  
<http://jmlr.csail.mit.edu/>
- Another high quality journal publishing articles on machine learning and related topics is Machine Learning.  
[www.springer.com/computer/ai/journal/10994](http://www.springer.com/computer/ai/journal/10994)
- There is a Google group for postings of possible interest to learning theory researchers.  
<http://groups.google.com/group/learning-theory/>
- The website of the International Machine Learning Society. The society organizes the annual International Conference on Machine Learning (ICML).  
<http://www.machinelearning.org/>
- The website of the Association for Computational Learning. The association organizes the annual Conference on Learning Theory (COLT).  
<http://seed.ucsd.edu/joomla/>

- The website of the Neural Information Processing Systems (NIPS) Foundation. The foundation organizes the annual NIPS conference.  
<http://nips.cc/>
- The arXiv is good way to find recent preprints in machine learning and learning theory.  
<http://arxiv.org/list/cs.LG/recent/>

## References

- [1] A.M. Turing, Computing machinery and intelligence, *Mind* 59 (1950) 433–460.
- [2] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in neural nets, *Bulletin of Mathematical Biophysics* 5 (1943) 115–137.
- [3] A.L. Samuels, Some studies in machine learning using the game of checkers, *IBM J. Res. Develop.* 3 (3) (1959) 210–233.
- [4] F. Rosenblatt, Two theorems of statistical separability in the perceptron, in: *Proceedings of a Symposium on the Mechanization of Thought Processes*, Her Majesty's Stationery Office, London, 1959, pp. 421–456.
- [5] A.B.J. Novikoff, On convergence proofs for perceptrons, in: *Proceedings of the Symposium on the Mathematical Theory of Automata* (New York, 1962), 1963, pp. 615–622.
- [6] V. Vapnik, Statistics for engineering and information science, *The Nature of Statistical Learning Theory*, second ed., Springer, 2000.
- [7] D.E. Rumelhart, G. Hinton, R.J. Williams, Learning representations by back-propagating errors, *Nature* 323 (1986) 533–536.
- [8] L.G. Valiant, A theory of the learnable, *J. ACM* 27 (11) (1984) 1134–1142.
- [9] C. Cortes, V.N. Vapnik, Support-vector networks, *Mach. Learn.* 20 (3) (1995) 273–295.
- [10] Y. Freund, R.E. Schapire, A decision-theoretic generalization of online-learning and an application to boosting, in: *Proceedings of the Second European Conference on Computational Learning Theory*, 1995, pp. 23–37.
- [11] C.J. Stone, Consistent nonparametric regression, *Ann. Stat.* 5 (4) (1977) 595–620.
- [12] M. Anthony, N. Biggs, *Computational Learning Theory: An Introduction*, Number 30 in Cambridge Tracts in Computer Science, Cambridge University Press, 1997.
- [13] M. Anthony, P.L. Bartlett, *Neural Network Learning: Theoretical Foundations*, Cambridge University Press, 1999.
- [14] L. Devroye, L. Györfi, G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Stochastic Modelling and Applied Probability, vol. 31, Springer, 1996.
- [15] V. Vapnik, *Estimation of Dependences Based on Empirical Data; Empirical Inference Science: Afterword of 2006*, second ed., Springer, 2006 (Reprint of 1982 edition with afterword of 2006).
- [16] V. Vapnik, *Adaptive and learning systems for signal processing, communications, and control, Statistical Learning Theory*, Wiley, 1998.
- [17] M. Vidyasagar, *Learning and Generalization: With Application to Neural Networks, Communications and control engineering*, second ed., Springer, 2003.
- [18] B.K. Natarajan, *Machine Learning: A Theoretical Approach*, Morgan Kauffmann, 1991.
- [19] F. Cucker, D.-X. Zhou, *Learning Theory: An Approximation Theory Viewpoint*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2007.
- [20] M.J. Kearns, U.V. Vazirani, *An Introduction to Computational Learning Theory*, MIT Press, 1994.
- [21] N. Cesa-Bianchi, G. Lugosi, *Prediction, Learning, and Games*, Cambridge University Press, 2006.

- [22] L. Györfi, M. Kohler, A. Krzyjak, H. Walk, *A Distribution Free Theory of Nonparametric Regression*, Springer Series in Statistics, Springer, 2002.
- [23] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, second ed., Springer, 2009.
- [24] P. Massart, Concentration inequalities and model selection, Lecture Notes in Mathematics, vol. 1896, Springer, 2007.
- [25] P.L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: Risk bounds and structural results, *J. Mach. Learn. Res.* 3 (2002) 463–482.
- [26] A. Ambroladze, E. Parrado-Hernández, J. Shawe-Taylor, Complexity of pattern classes and the lipschitz property, *Theor. Comp. Sci.* 382 (3) (2007) 232–246.
- [27] R.M. Dudley, The sizes of compact subsets of Hilbert space and continuity of Gaussian processes, *J. Funct. Anal.* 1 (3) (1967) 290–330.
- [28] V.N. Vapnik, A. Ya. Chervonenkis, On the uniform convergence of relative frequencies of event to their probabilities, *Soviet Mathematics Doklady* 9 (1968) 915–918.
- [29] N. Sauer, On the density of families of sets, *J. Comb. Theory A* 13 (1972) 145–147.
- [30] Saharon Shelah, A combinatorial problem; stability and order for models and theories in infinitary languages, *Pac. J. Math.* 41 (1972) 247–261.
- [31] D. Haussler, Sphere packing numbers for subsets of the boolean n-cube with bounded vapnik-chervonenkis dimension, *J. Comb. Theory A* 69 (2) (1995) 217–232.
- [32] N. Alon, S. Ben-David, N. Cesa-Bianchi, D. Haussler, Scale-sensitive dimensions, uniform convergence, and learnability, *J. ACM* 44 (4) (1997) 615–631.
- [33] M.K. Warmuth, Sample compression, learnability, and the vapnik-chervonenkis dimension, in: *Proceedings of the Third European Conference on Computational Learning Theory*, 1997, pp. 1–2.
- [34] O. Bousquet, A. Elisseeff, Stability and generalization, *J. Mach. Learn. Res.* 2 (2002) 499–526.
- [35] S. Kutin, P. Niyogi, Almost-everywhere algorithmic stability and generalization error, in: *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002, pp. 275–282.
- [36] S. Mukherjee, P. Niyogi, T. Poggio, R.M. Rifkin, Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization, *Adv. Comput. Math.* 25 (1–3) (2006) 161–193.
- [37] S. Shalev-Shwartz, O. Shamir, N. Srebro, K. Sridharan, Learnability, stability and uniform convergence, *J. Mach. Learn. Res.* 11 (2010) 2635–2670.
- [38] David A. McAllester, PAC-Bayesian stochastic model selection, *Mach. Learn.* 51 (1) (2003) 5–21.
- [39] Robert E. Schapire, The strength of weak learnability, *Mach. Learn.* 5 (1990) 197–227.
- [40] Y. Freund, Boosting a weak learning algorithm by majority, *Info. Comput.* 121 (2) (1995) 256–285.
- [41] Y. Freund, R.E. Schapire, Game theory, on-line prediction and boosting, in: *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, 1996, pp. 325–332.
- [42] D. Angluin, P.D. Laird, Learning from noisy examples, *Mach. Learn.* 2 (4) (1987) 343–370.
- [43] M. Kearns, Efficient noise-tolerant learning from statistical queries, *J. ACM* 45 (6) (1998) 983–1006.
- [44] V. Feldman, P. Gopalan, S. Khot, A.K. Ponnuswami, On agnostic learning of parities, monomials, and halfspaces, *SIAM J. Comput.* 39 (2) (2009) 606–645.
- [45] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm, *Mach. Learn.* 2 (4) (1987) 285–318.
- [46] S. Ben-David, D. Pál, S. Shalev-Shwartz, Agnostic online learning, in: *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.
- [47] G.H. Bakir, T. Hofmann, B. Schölkopf, A.J. Smola, Ben Taskar, S.V.N. Vishwanathan, *Predicting Structured Data*, MIT Press, 2007.
- [48] C.A. Micchelli, M. Pontil, On learning vector-valued functions, *Neural Comput.* 17 (1) (2005) 177–204.

- [49] D. Aldous, V. Vazirani, A Markovian extension of Valiant's learning model, in: Proceedings of the 31st Annual Symposium on Foundations of Computer Science, vol. 1, 1990, pp. 392–396.
- [50] D. Gamarnik, Extension of the PAC framework to finite and countable Markov chains, in: Proceedings of the twelfth annual conference on Computational learning theory, 1999, pp. 308–317.
- [51] A. Irle, On the consistency in nonparametric estimation under mixing assumptions, *J. Multivar. Anal.* 60 (1997) 123–147.
- [52] R. Meir, Nonparametric time series prediction through adaptive model selection, *Mach. Learn.* 39 (1) (2000) 5–34.
- [53] A. Lozano, S. Kulkarni, R. Schapire, Convergence and consistency of regularized boosting algorithms with stationary  $\beta$ -mixing observations, *Adv. Neural Inform. Process. Syst.* 18 (2006) 819–826.
- [54] M. Mohri, A. Rostamizadeh, Rademacher complexity bounds for non-i.i.d. processes, in: Advances in Neural Information Processing 21, 2009, pp. 1097–1104.
- [55] I. Steinwart, D. Hush, C. Scovel, Learning from dependent observations, *J. Multivar. Anal.* 100 (1) (2009) 175–194.
- [56] A. Nobel, Limits to classification and regression estimation from ergodic processes, *Ann. Stat.* 27 (1) (1999) 262–273.
- [57] A. Rakhlin, K. Sridharan, A. Tewari, Online learning: Random averages, combinatorial parameters, and learnability, *Adv. Neural Inform. Process. Syst.* 23 (2010) 1984–1992.
- [58] S. Hanneke, Rates of convergence in active learning, *Ann. Stat.* 39 (1) (2011) 333–361.
- [59] O. Chapelle, B. Schölkopf, A. Zien (Eds.), *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.
- [60] J. Baxter, A model of inductive bias learning, *J. Artif. Intell. Res.* 12 (2000) 149–198.
- [61] R. Caruana, Multitask learning, *Mach. Learn.* 28 (1) (1997) 41–75.
- [62] T. Evgeniou, C.A. Micchelli, M. Pontil, Learning multiple tasks with kernel methods, *J. Mach. Learn. Res.* 6 (2005) 615–637.
- [63] A. Maurer, Bounds for linear multi-task learning, *J. Mach. Learn. Res.* 7 (2006) 117–139.
- [64] M. Campi, P.R. Kumar, Learning dynamical systems in a stationary environment, *Syst. Control Lett.* 34 (3) (1998) 125–132.
- [65] H. Peyton Young, *Strategic Learning and Its Limits*, Oxford University Press, 2004.
- [66] M.-F. Balcan, A. Blum, S. Fine, Y. Mansour, Distributed learning, communication complexity and privacy, in: Proceedings of the 25th Annual Conference on Learning Theory, JMLR Workshop and Conference Proceedings, vol. 23, 2012, pp. 26.1–26.22.
- [67] L.G. Valiant, Evolvability, *J. ACM* 56 (1) 2009.
- [68] M.-F. Balcan, A. Blum, J.D. Hartline, Y. Mansour, Reducing mechanism design to algorithm design via machine learning, *J. Comp. Syst. Sci.* 74 (8) (2008) 1245–1270.

# Neural Networks

# 15

**Barbara Hammer**

*CITEC centre of excellence, Bielefeld University, D-33594 Bielefeld, Germany*

## 1.15.1 Introduction

Roughly speaking, Neuroinformatics is based on the paradigm to turn ideas from biological neural networks, which serve as very effective information processing units in real life, into efficient artificial machine learning methods in Computer Science. Thereby, the lines between biological motivation and statistical methodology are often blurred, and, today, a thorough mathematical background exists for most techniques from Neuroinformatics independent of their biological counterparts, i.e., Neuroinformatics has become a mature research field on its own independent of its biological roots. There exists a variety of excellent textbooks in this field covering mathematical background as well as modern training algorithms in this context, such as [1–6].

The early beginnings of Neuroinformatics date back to 1943, when McCulloch and Pitts showed the ability of circuits of simple neurons to realize every logical calculation [7]. In 1949, a training paradigm was proposed by Hebb [8], the famous Hebbian learning which states the principle to increase the strength of neurons if they are simultaneously active. This principle forms the base of many learning algorithms which are used today. Algorithmic realizations of Hebbian learning rules for simple neural networks without any intermediate connections have been realized by Rosenblatt [9] and, independently, by Widrow and Hoff [10]. Remarkably, perceptron learning as proposed by Rosenblatt is accompanied by a formal proof of its convergence to a solution if existing. However, research came to a rapid stop after the famous book of Minsky and Papert, “Perceptrons,” has been published. In this book, the authors proved the principled limitation of the Rosenblatt-Perceptron and other simple models [11].

A number of researchers continued to work on relevant topics related to Neuroscience including, e.g., associative memory [12] and the development of the visual system [13]. The breakthrough which brought Neuroscience back to the focus of a large number of researchers was the reinvention of back-propagation for training feedforward neural networks by Rumelhart et al. [14]. This algorithm has been proposed previously by Werbos [15] and it became widely accepted after it has been published a second time. It forms the base for most learning algorithms for multilayer neural networks which are used to date. Today a variety of different learning paradigms, neural architectures, and training algorithms exists which offer efficient tools for machine learning in diverse areas including pattern recognition, robotics, bioinformatics, or natural language processing.

Often, the biological background only plays a minor role in modern applications and neural learning techniques constitute special cases of more general statistical learners such as, e.g., general regression or classification models, generative statistical models, Bayesian techniques, and similar, see textbooks such as [1]. In this context, current problems typically center around the question of how to deal with larger and larger data sets, how to arrive at even more efficient learning techniques, how to integrate prior knowledge and structural constraints, how to make the decisions of neural networks accessible for humans, and similar. These research questions are no longer specific for neural networks, rather they occur in virtually all machine learning scenarios.

However, quite a few recent developments in the context of neural networks have been triggered by biological counterparts, again: deep learning architectures can be traced back to typical processing pipelines in the human visual system and its progressive representation of more and more abstract visual stimuli [16]. At present, deep learning architectures are discussed as one general principle to directly address complex input patterns in neural network architectures without the necessity of complex, domain-specific preprocessing of the given signals. Reservoir computing constitutes another example where biologically motivated paradigms offer a possible solution to long-standing problems in Neuroinformatics. Partially recurrent neural networks deal with time dependent signals as input data, and, thus, have potential impact on areas such as temporal signal processing, speech recognition, or motion generation. Training such networks, however, has been a problem due to inherent numeric difficulties of typical learning algorithms. Reservoir networks are based on a randomly wired recurrent network with very high dimensionality and a very simple readout, similar to intuitive recurrent processing as could happen in the brain [17,18]. They offer a partial answer to the complexity of recurrent neural network training by simply making it superfluous. These issues will be addressed in this chapter in more detail.

The main goal of this chapter is to give a representative picture of different relevant architectures of the “zoo” of neural networks. The emphasis lies on the principled idea behind the respective models and an understanding of their strengths and restrictions with links to further reading. We will give hints on principled training algorithms, and we will discuss mathematical issues which arise when training such networks using given examples. Naturally, this overview is necessarily limited to some of the most important neural architectures which are used today.

---

### 1.15.2 Learning with single neurons

Let us start with a simple problem: we would like to automatically classify handwritten digits, e.g., to automatically read the ZIP code hand written on letters and collected in a US postal office. For this purpose, we accumulate examples where we know the intended digit, assigned to by a person, the *training set*. Depending on the digitalization, we can assume that each example consists of an input vector in, say,  $\mathbb{R}^{64}$ , corresponding to a representation of the digit in a  $8 \times 8$  matrix with gray values. We represent the output class by a number. Generally, inputs stem from  $\mathbb{R}^n$ ,  $n$  being any fixed number, the input dimensionality. Outputs are class labels. For simplicity, we start with only two classes for the moment, 0 and 1. Actually, data sets of such type constitute typical benchmarks for neural classification algorithms even today, two popular data sets being the USPS (see <http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html> file called ZIP code) and the MNIST (see <http://yann.lecun.com/exdb/mnist/>) data sets.

Neural networks offer one possible mechanism to infer a classification describing an unknown regularity based on a finite number of examples. We start with the simplest neural network one can think of: a single neuron, more precisely, the perceptron.

### 1.15.2.1 The simple perceptron

The function of biological neural networks is well investigated and their dynamics can be described referring to the Hodgkin-Huxley model, for example [19]. Biological neurons can be interpreted as signal processing units based on electric currents. While the biological Hodgkin-Huxley model describes the dynamics of the most relevant ions regarding their spatio-temporal location in detail, abstractions in Neuroinformatics integrate over the spatial location of the electric current of all ions, or they even abstract from the precise temporal dynamics, referring to the mean firing rate of neurons.

The *perceptron* constitutes a particularly simple neuron which restricts information to a simple digital signal: 0 or 1 corresponding to active or inactive synapses. This abstraction allows to characterize a perceptron by its function:

$$\mathbb{R}^n \ni x \mapsto H(\mathbf{w}^t x - \theta) \in \mathbb{R},$$

whereby  $\mathbf{w} \in \mathbb{R}^n$  is the so-called *weight-vector* of the neuron, and  $\theta \in \mathbb{R}$  the *bias*. As usual,  $\mathbf{w}^t x$  denotes the dot product of the two vectors.  $H : \mathbb{R} \rightarrow \mathbb{R}$  denotes a nonlinear *activation function*, in the case of the perceptron it is the Heaviside function with

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

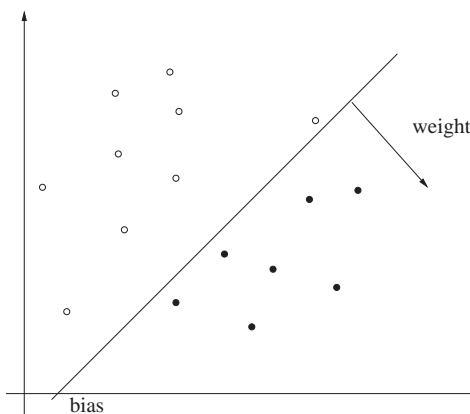
Thus, the neuron computes the weighted sum of inputs, compares the result with the bias, and outputs a signal depending on its *activation*  $\mathbf{w}^t x - \theta$ . This procedure resembles biological neurons which integrate incoming signals and fire depending on their action potential whereby, for the artificial counterpart, spatiotemporal aspects are neglected.

This function has an exact geometric counterpart: a perceptron defines a separating hyperplane of the data. The weight vector  $\mathbf{w}$  determines the orientation of the hyperplane, the bias  $\theta$  the offset. Thus, it defines a simple linear classification boundary, see Figure 15.1. For simple data sets or high dimensionality, a linear classifier is often sufficient to separate a given data set appropriately.

How can we get appropriate parameters  $w$  and  $\theta$  given a set of training data? The *perceptron training algorithm* is a very simple method which can be formulated in a few lines of code:

```
init  $\mathbf{w}, \theta$  randomly
repeat:
    choose a training example  $(x, y)$ 
    if  $H(\mathbf{w}^t x - \theta) \neq y$ 
         $\mathbf{w} := \mathbf{w} + x, \theta := \theta - 1$  if  $y = 1$ 
         $\mathbf{w} := \mathbf{w} - x, \theta := \theta + 1$  if  $y = 0$ 
```

The adaptation rule constitutes an instance of Hebbian learning: given an example which is wrongly classified as 0 but it should be 1, all weight components which receive a positive signal are reinforced by adding the input signal  $x$ . For a wrong classification as 1 which should be 0, the weight components

**FIGURE 15.1**

Two-class classification problem represented by points in the euclidean plane. Since the set is linearly separable, a simple perceptron can serve as classifier. It is uniquely characterized by its weight vector and bias.

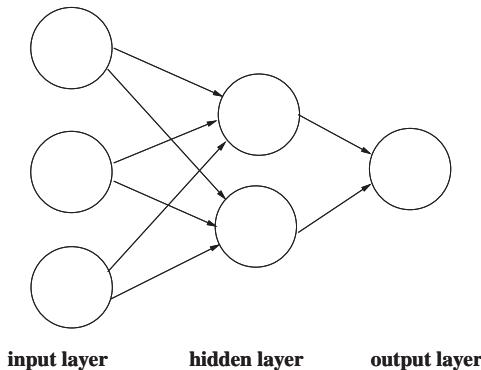
which receive a negative signal are increased. One can show that this learning rule converges towards a solution without errors, if it exists. It finds a solution with a minimum number of errors with high probability, provided the examples are randomly chosen and the best solution found so far is stored [9].

Because of the fact that a perceptron corresponds to a separating hyperplane, training sets which can correctly be classified by a perceptron are called *linearly separable*. Obviously, a perceptron can solve exactly those problems (up to a few errors), which are (almost) linearly separable. For very high dimensional data as present, e.g., in biomedical application dealing with mass spectra or microarrays, this is often the case. In general, decision boundaries are nonlinear in realistic problems. But even if we restrict to linear problems, a few questions arise at this point: how complex is perceptron learning, i.e., what is the number of iterations the algorithm requires until convergence? Can we guarantee, that perceptron learning finds the underlying regularity of the classification which we would like to learn and does not only map the given training set correctly? These two questions concern the *complexity of learning* and the *generalization ability*. We will consider these questions after having defined more general feed-forward neural networks for nonlinear problems.

### 1.15.2.2 Feedforward networks

A feedforward neural network consists of a number of neurons which are connected in an acyclic directed graph. Often, the neurons are arranged in layers, the neurons without predecessors forming the *input layer*, the neurons without successors forming the *output layer*, the ones in between forming the *hidden layer* or hidden layers, see Figure 15.2. Such a network computes a highly nonlinear function by combining the functions of single neurons according to the given connection graph.

Formally, this function can be defined as follows: Assume the number of input neurons is  $n$ , the number of output neurons is  $m$ ,  $w_{ji}$  constitutes the weight of neuron  $i$  pointing from neuron  $j$ , and  $\theta_i$  is the bias of neuron  $i$ . The activation function of every neuron is fixed, often the logistic function

**FIGURE 15.2**

A layered feedforward neural network with one hidden layer.

$\sigma(x) = (1 + \exp(-x))^{-1}$  or the hyperbolic tangent  $\tanh(x)$  are chosen. Such a network computes a function  $f_W : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $x \mapsto (o_{i_1}, \dots, o_{i_m})$ , which maps an input vector to the output values of the output neurons of the network. The function is parameterized by the weights and biases  $W$  in the network. The numbers  $i_j$  constitute the indices of the output neurons. The output  $o_i$  of a neuron  $i$  is recursively defined as follows:

$$o_i = \begin{cases} x_i & \text{if } i \text{ is an input neuron,} \\ \sigma(\sum_{j \mapsto i} w_{ji} o_j - \theta_i) & \text{otherwise.} \end{cases}$$

The activation of a single neuron is abbreviated by the term  $\text{net}_i = \sum_{j \mapsto i} w_{ji} o_j - \theta_i$ . Obviously, such a network can realize highly nonlinear functions. The perceptron activation function is substituted by the sigmoidal function which approximates the perceptron function for large inputs, but which is differentiable. This feature is used to derive training algorithms for multi-layer networks, as we will see below. A single sigmoidal unit can be interpreted as an approximation of a simple perceptron in its saturated regions. Often, however, sigmoidal neurons are activated in their linear area around 0. This feature can be motivated by another observation from biology: taking real values as average firing rate of neurons, a sigmoidal function mirrors the typical transfer dynamics of biological neurons.

A feedforward neural network can not only represent a simple binary classification such as the perceptron, but classification with more than two classes or real-valued functions. How can such a network be trained? Since the activation function  $\sigma$  is differentiable, the following mathematical principle can be taken: Given a set of training patterns of the form  $(x^p, y^p)$  the *mean square error* is defined as the quantity

$$E = 1/2 \cdot \sum_p (f_W(x^p) - y^p)^2.$$

This error measure constitutes a differentiable function which maps the weights  $W$  of the network to a real number indicating whether the current function approximately represents the given data set or not. Training consists in a minimization of this cost function by means of a numeric optimization

method, commonly a variation of a gradient descent. A simple *gradient descent* is given by the following pseudocode

```
init  $W$  randomly
repeat:
   $W := W - \eta \cdot \nabla_W E$ 
```

Thereby,  $\nabla_W E$  constitutes the vector of derivatives of  $E$  with respect to the weights  $w_{ij}$  and biases  $\theta$ .  $\eta > 0$  is a (small) learning rate. Initialization of the weights is usually done with small random values around 0.

A *stochastic gradient descent* decomposes the mean square error into the contributions of single patterns  $E = \sum_p E_p = 1/2 \cdot \sum_p (f_W(x^p) - y^p)^2$  and iteratively adapts according to a randomly chosen pattern:

```
init  $W$  randomly
repeat:
  choose  $p$  randomly,
   $W := W - \eta \cdot \nabla_W E_p$ 
```

This can also be applied in an online scenario where the patterns are not known beforehand but generated during training.

For feedforward networks, a particularly efficient way to compute the derivative  $\nabla_W E_p$  resp.  $\nabla_W E$  is used in the famous *back-propagation algorithm*. Note that the recursive definition of the outputs  $o_i$  allows to compute the derivative of  $o_i$  with respect to weights and biases directly in a recursive way, however, the complexity of this method is  $\mathcal{O}(|W|^2)$ ,  $|W|$  being the number of weights. Back-propagation defines the error terms  $\delta_j = \partial E_p / \partial \text{net}_j$ . It holds

$$\partial E_p / \partial w_{ij} = o_i \cdot \delta_j$$

and

$$\partial E_p / \partial \theta_j = -\delta_j.$$

The error terms  $\delta_j$  can be computed in linear time by back-propagation

$$\delta_j = \begin{cases} (o_j - y_j^p) \cdot \sigma'(\text{net}_j) & \text{if } j \text{ is an output neuron} \\ \sum_{j \rightarrow k} w_{jk} \delta_k \cdot \sigma'(\text{net}_j) & \text{otherwise} \end{cases}$$

propagating the error signals from the output neurons back to the hidden neurons. Note that numeric optimization techniques with a better convergence rate are usually applied today such as, e.g., approximate conjugate gradient techniques or alternative higher order methods. Still, these numeric techniques center around the partial derivatives and its efficient computation by means of back-propagation. Further, simple back-propagation has the benefit that it can be applied in classical batch mode as well as in online scenarios. Batch learning usually refers to the situation where all training examples are known in advance. In such cases, the full error function is available and numerical techniques which take into account, e.g., the curvature of the cost function can be used. In online settings such as, e.g., reinforcement learning or robotics, training data become available on the fly and adaptation has to take place immediately after a novel training example has been presented. In this case, a stochastic gradient descent can be used which corresponds to simple back-propagation applied separately for every given data point.

### 1.15.2.3 The training pipeline

A variety of questions arises in this context:

1. How do we choose the architecture?
2. What can we do if the training error does not decrease?
3. Does the network behave correctly for future patterns if it maps the training data correctly?

#### 1.15.2.3.1 How do we choose the architecture?

There is an easy answer to this question and a more complex one: the easy one—just try it and choose the best architecture you can find. Commonly, the suitability of an architecture is evaluated by means of cross-validation. That means, the architecture is trained on a part of the data set and its mean squared error evaluated on the remaining test set. To reduce statistical effects, this is repeated  $k$  times using disjoint test sets which decompose the given set of training data.

The complex answer to the question of how to choose the architecture puts some light on relevant aspects of feedforward networks: Unlike simple perceptrons, feedforward neural networks are *universal approximators*, i.e., they are capable of approximating every reasonable (e.g., continuous) function on a compact set up to any desired precision provided at least one hidden layer with a sufficient number of hidden neurons is available and an activation function such as the logistic function is used [20]. Hence, feedforward networks can, in principle, represent any regularity arbitrarily well. In consequence, an appropriate architecture can be found for any given training problem. In principle, the number of hidden layers can be restricted to one hidden layer, or, in practice, a small number of hidden layers.

How do we evaluate an architecture? One evaluation measure is the error on the training set. If this is large, the architecture is not expressive enough to represent the regularity which has to be learned. However, if the error is small, this does not necessarily guarantee that the underlying regularity has been learned, only the part represented by the training set is correct. One can think of a neural architecture as a parameterized function class, whereby the number of parameters depends on the number of neurons of the architecture. It has to be guaranteed, that a sufficient number of parameters is available to represent the training data correctly. At the same time it has to be guaranteed that no free parameters of the architecture are left which are not sufficiently specified given the current training set. If so, the output for data points not present in the training set could be random.

This problem is known as *bias-variance-dilemma*: architectures which are too small have a large bias, i.e., a large deviation from the desired outputs already on the training set. Architectures which are too large have a large variance, i.e., the output of patterns not contained in the training set is random because the parameters of the architecture are not determined by the training set.

#### 1.15.2.3.2 How to minimize the training error?

When applying simple back-propagation, training is often a very tedious task. This is due to several properties of the error surface when training neural networks. First of all, the error surface is multimodal and gradient descent methods are often trapped in local optima instead of global (or good) optima of the cost function. This problem is partially unavoidable because the problem of finding a global minimum of the mean squared error for neural networks is an NP-hard problem, as shown, e.g., in [21,22]. Remarkably, NP-hardness is already present when minimizing the mean squared error of a single neuron, or when training simple perceptrons for non-linearly separable data sets.

In practice, complex error surfaces and bad local optima can partially be avoided by simplifications of the training task using data preprocessing. This way, all relevant information is available for the network in explicit form and the dependency of the function value on the inputs is more simple. Standard preprocessing includes, for example, standard procedures such as normalization of the input variables as well as a variety of problem-dependent features, e.g., for image processing (such as filters) or time series analysis (such as Fourier or wavelet transform). Appropriate preprocessing of data can be time consuming and there is no recipe how to solve this problem in general.

Current research connected to Neuroinformatics centers around the question how complex preprocessing can be avoided. Alternatively, how can an appropriate representation of data autonomously be detected by the neural system itself? A very promising approach centers around deep learning, which deals with neural architectures with many hidden layers but a highly regularized and symmetric structure [16]. Intuitively, these architectures can extract complex features from the data by an iterative combination of simple features as represented by parts of those networks. High symmetry such as, e.g., weight sharing in different parts of the network make sure that globally meaningful and uniform local feature extractors result. However, training deep neural networks using back-propagation is usually very difficult if not impossible due to numeric problems, such that alternative iterative learning strategies are usually applied. Recent developments in this field can be found, e.g., at a NIPS 2010 workshop (see <http://deeplearningworkshopnips2010.wordpress.com/schedule/>) or the deep learning web site (see <http://deeplearning.net/>).

The problems which make a simple gradient technique unsuitable for training very deep architectures, also partially occur for standard feedforward networks of reasonable size. Because of the logistic activation function, the error surface of sigmoidal networks is composed of plateaus mixed with very steep parts. This causes oscillation or stagnation of the optimization procedure, depending on the curvature of the surface and the learning rate. Thereby, these effects can occur simultaneously with respect to different parameters which have to be optimized.

Various improvements of simple back-propagation try to overcome these problems such as higher order optimization methods [4]. One particularly powerful and, at the same time, very simple learning schemes which allows to train standard feedforward networks very efficiently is offered by *resilient propagation* (RProp) [23]. The idea is to use a separate adaptive learning rate for every parameter which determines the step size into this direction. The derivative does only contribute the direction of the step, but not its size. The step size is increased if the gradient points into the same direction in consecutive steps (i.e., likely a plateau is present) and it is decreased if the sign changes (i.e., oscillation takes place), whereby bad steps are retracted.

Another very interesting line of research centers around even more efficient training algorithms which do not rely on numeric optimization techniques. The *extreme learning machine* is based on the observation that feed-forward neural networks can approximate every given training set provided enough hidden neurons are available even if the connections of the inputs to the hidden neurons are random. In consequence, the extreme learning machine randomly generates a large hidden layer and only trains the linear readout. Interestingly, this very simple mechanism can yield good results for practical applications [24]. This principle is very similar to reservoir computing, a technique to deal efficiently with recurrent neural networks, which we will discuss below.

### 1.15.2.3.3 Does the network generalize to novel data points?

We have already discussed this issue in the context of how to choose an architecture: if the architecture is too flexible, it maps the training data correctly, but it can behave randomly on novel data points not used for training. Therefore, in practice, the *generalization ability* of a network is often tested by means of a hold out set which has not been used for training, the test set.

However, the question remains whether there exist any guarantees that a small training error also leads to a small test error provided the training set is large enough. Such guarantees can be derived in the framework of statistical learning theory: one can limit the “freedom” which is contained in a network with  $|W|$  parameters. One popular quantity which measures this capacity of a network is offered by the so-called *Vapnik-Chervonenkis dimension* of the function class [25]: The Vapnik-Chervonenkis dimension measures the combinatorial capacity of the architecture, i.e., the number of points for which every possible classification can be realized within the given neural architecture. The VC dimension of a sigmoidal feedforward network can be limited by a polynomial in the number of weights  $|W|$  [26]. Depending on this quantity, the deviation of the empirical error and the generalization error of the network for new examples can be limited with high probability regardless of the underlying distribution [25].

This formalization and test of the ability of a neural architecture to learn an underlying regularity is often referred to as the frequentist point of view, since the generalization error is limited by means of statistical counting arguments. Apart from an explicit posterior bound or estimation of the generalization ability obtained this way, it motivates another objective of neural networks training: a minimization of the training error is often accompanied by additional mechanisms which explicitly reduce the degrees of freedom of the network architecture. A popular example of this paradigm is given by a limitation of the weight sizes, a mechanism which can be linked to Tikhonov regularization in some cases. Another example refers to a pruning of connections, weight sharing, or some other form of regularizer incorporated in the cost function [1, 27]. One particularly popular form of regularization is included in training of the support vector machine or some forms of learning vector quantization, an optimization of the margin of the classifier, as we will explain below.

This frequentist point of view is complemented by a Bayesian treatment as explained, e.g., in the seminal work [27]. Here, instead of focussing on just one architecture which, from a statistical point of view often corresponds to a single parameter setting which is most likely given the current observations, a full probabilistic model is inferred for the data and the range of possible parameters. This includes an inherent regularization of the result such that overfitting does not occur. This benefit is paid for by a more complex training procedure where integrals corresponding to the infinite number of possible parameter settings have to be addressed. Here, a rich repertoire of techniques has been developed which make this problem feasible ranging closed form solutions for specific cases to efficient and general Markov chain Monte Carlo techniques [28].

---

## 1.15.3 Recurrent neural networks

Feedforward architectures are characterized by acyclic graphs. Biological networks possess back-connections. What happens, if we introduce cycles in a neural network? So-called recurrent neural networks consist of neurons which are connected in an arbitrary cyclic graph. Due to these cycles, the

activation of neurons depends on each other, and a simple iterative definition is no longer possible, rather, the dynamics of how the neurons are activated has to be taken into account.

There exist basically two different types of recurrent networks: fully recurrent networks which are used as associative memory, and partially recurrent networks which are used to model dynamic systems and to process time series.

### 1.15.3.1 A brief look at Hopfield networks

*Hopfield networks* are fully recurrent networks where the single neurons resemble perceptrons with Heaviside activation function. Original Hopfield networks are restricted to symmetric connections, i.e.,  $w_{ij} = w_{ji}$ , without self-repression, i.e.,  $w_{ii} \geq 0$ . Because of the recurrence, outputs of neurons are time dependent. Given a state  $o_j(t)$  of the system at a point in time  $t$ , every neuron computes its activation

$$\text{net}_i(t) = \sum_{j \rightarrow i} w_{ji} o_j(t) - \theta_j.$$

Typically, Hopfield networks are driven in asynchronous mode, i.e., at every time point a neuron is picked at random and changes its state according to

$$o_i(t+1) = H(\text{net}_i(t))$$

if discrete time steps are considered. The output of all other neurons is not changed in this step. Alternative to this dynamics, Hopfield networks can be driven in synchronous mode, in which all neurons change their state simultaneously according to the current input. This dynamics is particularly interesting if a generalization to continuous time is considered and the dynamics is described by corresponding differential equations.

It can be shown that, due to the symmetry of weights, Hopfield networks converge towards a *stable pattern* if driven in asynchronous mode, i.e., a state where no neuron changes its activation any more [5]. This way, Hopfield networks can be used as associative memory which assigns a completed pattern to an initial, possibly disrupted initialization. For a synchronous update mode, cycles of length two can occur in discrete time, but convergence is observed in continuous time.

Training Hopfield networks refers to the task to store a given set of patterns as stable states of a networks. A very fast training heuristic is offered by Hebbian learning: the weight  $w_{ij}$  is set proportional to the number of patterns for which the neurons  $i$  and  $j$  have identical activation. This learning rule works reliably for orthogonal or nearly orthogonal patterns. In general, cross-correlations occur and the stored patterns become disrupted.

A more general method consists in a reduction of Hopfield training to a perceptron learning problem [5]. This procedure yields to a solution if existing. It also demonstrates a principled restriction of Hopfield networks: the patterns which can be stored by Hopfield networks correspond to linearly separable data. More general networks which also possess hidden neurons have been proposed: Boltzmann machines substitute the crisp binary-valued activation by probabilistic neurons which states are distributed according to a Boltzmann distribution for a given temperature  $T$  [5]. Training refers to an optimization of this distribution such that it approximates the uniform distribution on a given set of pattern as much as possible as measured by the Kullback-Leibler divergence. Interestingly, Boltzmann

machines can be trained using hidden neurons, such that their storage capability is strictly enhanced as compared to simple Hopfield networks, and they can, in principle, represent every given set of patterns. It turned out that Boltzmann machines have a large potential for the unsupervised training of the layers in deep feedforward networks [16].

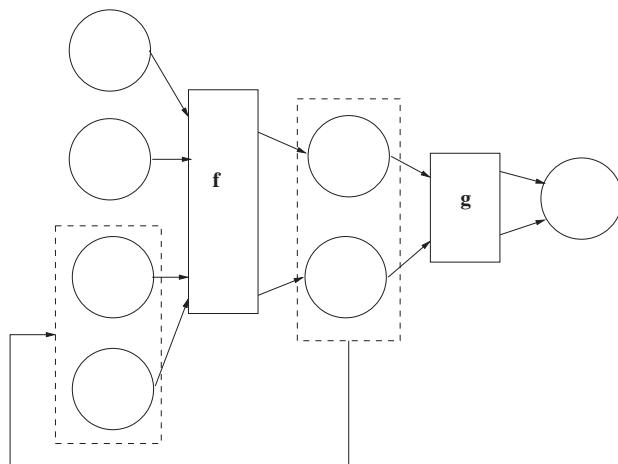
### 1.15.3.2 Recurrent neural networks

*Partially recurrent networks* have only a limited feedback which can be used to model temporal dependencies, e.g., in dynamic systems. This way, temporal sequences can be addressed. The dynamics is driven by the input sequence  $x(t)$  with temporal dependency  $t$ . A standardized dynamics is offered by the following equation

$$\begin{aligned} z(t) &= f(x(t), z(t-1)), \\ o(t) &= g(z(t)). \end{aligned}$$

$x(t)$  is the input at time step  $t$ ,  $z(t)$  describes the internal state of the recurrent network at time  $t$ , and  $f$  and  $g$  are computed by standard feedforward neural networks. See Figure 15.3 for an example. These dynamic equations can be generalized to a continuous dynamics by means of differential equations which characterize the state change over time.

If such a network is used for time series prediction, a sequence  $(x(t), y(t))$  of training patterns is given. Similar to feedforward networks, training can take place by minimization of the mean squared error  $E = 1/2 \cdot \sum_t (y(t) - o(t))^2$ . This can be done by a gradient descent, since the function is differentiable with respect to the weights. Thereby, there exist basically two different methods to compute the gradients within recurrent networks: back-propagation through time and real time recurrent learning.



**FIGURE 15.3**

Partially recurrent neural network in the Elman style, recurrent connections enable a processing of time dependent inputs in their temporal context.

*Back-propagation through time* directly transfers back-propagation to recurrent networks. The key observation consists in the fact that a recurrent network can be substituted by an equivalent feedforward network by unfolding in time. Thereby, the weights are shared between the copies. The gradient with respect to a weight  $w_{ij}$  results from the sum of gradients with respect to all copies of  $w_{ij}$  in this unfolded network. These latter terms can be computed by means of standard back-propagation. The time complexity is  $\mathcal{O}(|W| \cdot T)$ ,  $T$  being the length of the sequence, and the required space is  $\mathcal{O}(|W| \cdot T)$ .

As an alternative, one can compute the derivatives with respect to the weights directly from left to right in the unfolded network based on the recursive definition of the activation of neurons. This procedure has time complexity  $\mathcal{O}(|W|^2 \cdot T)$  but the required space is independent of the maximum time length  $T$ . In particular, the derivatives with respect to copies of  $w_{ij}$  for early time steps are available independent of the remaining time series. Hence, applying the changes according to early time steps directly, this method can also be used in an online scenario. This method is referred to as *real time recurrent learning*.

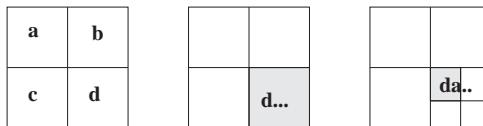
As for training of feedforward networks, several questions occur: how do we choose the architecture? In principle, partially recurrent networks are approximation complete in the sense that they can approximate any dynamic system with continuous transition function on a finite time horizon [29]. For a given training task, the architecture is chosen by cross-validation, as for feedforward networks.

What is the complexity of training recurrent neural networks? Since training feedforward networks constitutes a special case of training recurrent networks, the problem is NP-hard in the worst case. Besides this theoretical result, training recurrent networks turned out to be considerably more difficult than training feedforward networks. Already for small networks containing only two neurons, recurrent neural networks can induce a rich dynamics which includes the full dynamic spectrum ranging from stable states, periodic and quasiperiodic behavior, to chaotic systems [30]. Thereby stable behavior need not be preserved in the course of learning [31] and additional effort has to be taken to guarantee stability of the outcome [32]. Training using a gradient descent faces principled restrictions due to the so-called problem of *long-term dependencies* [33]. This refers to the fact that error signals either blow up within the neural network or vanish when back-propagated over time, such that they can hardly be stored over a long time context. Specific recurrent architectures which address this problem include long short term memory as a prominent example [34] where recurrence is restricted to linear neurons for which gradients can safely be back-propagated.

### 1.15.3.3 Learning without training recurrent connections

An alternative for recurrent neural network training has recently become popular: in reservoir computing, the recurrent part of the network is not trained according to given supervised signals. Instead, a sufficiently rich recurrent part is initialized, and a trainable readout is added to this state representation which can efficiently be trained because the mean squared error for a linear function gives rise to a convex cost function. Reservoir computing includes early approaches such as fractal prediction machines, and the recent popular echo state networks and liquid state machines [17, 35, 36]. Here we only explain the first two, since the latter is based on spiking neurons, which are biologically more plausible, but beyond the scope of this chapter.

Fractal prediction machines directly encode sequences over a finite alphabet by means of fractal codes. Assume the number of symbols is  $s$ . A fixed dimensionality of the internal state space is chosen and the unit hypersquare in this space is decomposed into  $s$  regular hypersquares. A function  $f_a$  is

**FIGURE 15.4**

Fractal encoding of sequences over a finite alphabet, fractal prediction machines encode symbolic sequences in a real vector space in a similar way as recurrent networks with small weights. In the limit, time series give rise to a fractal set in the encoding space.

associated with every symbol  $a$ . The function is the affine mapping which transforms the unit square to the square associated with  $a$ . Starting with a fixed initial value 0, the sequence  $a_1 \dots a_t$  can be encoded as  $f_{a_1} \circ \dots \circ f_{a_t}(0)$ . See Figure 15.4 for an example. Note that the set of points which are obtained by encoding all possible sequences constitutes a fractal set in the considered space. The precision of the location of a code determines the length of the sequence which can uniquely be decoded. On top of this code, a standard network or any alternative classifier can be trained. It has been shown in [36] that these simple models are often as powerful as standard recurrent neural networks. In particular, they exactly resemble the behavior of recurrent networks with small weights, i.e., recurrent neural networks during the initial phase of training.

Echo state networks use a large state space dimensionality such that a rich reservoir is available. The encoding function  $f$  of the recurrent network is given by a simple sigmoidal network with sparse random connections of the neurons and random weights. Thereby, the echo state property must be fulfilled, i.e., the internal state of the network is independent of the initial condition if a long enough recurrence is applied. This can be guaranteed, e.g., by a contractive transition function, which can easily be tested considering the spectral radius of the linear part of the transition. This way, the recurrence constitutes an encoding very similar to fractal encodings due to the echo state property, whereby a rich reservoir accounts for sufficient precision of the representation. On top of this code, a trainable (often even only linear) network is added. Despite their simplicity, echo state machines constitute quite powerful models with successful applications and biological plausibility [17].

It has been shown in [37] that the restriction of the recurrence to contractive mappings has beneficial effects with respect to the generalization ability of the network. For contractive networks, the generalization ability does not depend on the dimensionality of the state space, but on the contraction coefficient of the transition function. Generalization bounds can be obtained which depend on this contraction coefficient and the number of examples used for training. For general recurrent networks, it is not possible to obtain generalization bounds just in terms of the number of network parameters as shown in [38]. Unlike for feedforward networks, the concrete data distribution has to be taken into account to develop generalization bounds for general recurrent networks. Concrete bounds which take the data distribution (possibly as manifested in the training set) into account have been developed in [38], for example.

#### 1.15.3.4 Connection to the Chomsky hierarchy

Interestingly, a very strong link of different types of recurrent networks to classical computing mechanisms exists. The motivation behind this fact can, among other things, be traced back to the question

whether recurrent networks constitute suitable models to explain the perception of natural language by humans. Remarkable results on language processing with recurrent networks have been achieved already by Elman [39], the debate about the suitability of recurrent models for language processing still going on, see, e.g., [40–42].

It has rigorously been shown in [37] that recurrent neural networks with small weights, i.e., contractive transition function are equivalent to definite memory machines, i.e., models which have only a finite time horizon.

For larger weights, a strong connection to finite automata has been established, showing, on the one hand, the possibility to simulate every automaton on an infinite time horizon [43], on the other hand, establishing methods for automata insertion prior to training and automata extraction after training [44]. The capacity of recurrent networks with arbitrary weights, however, is larger. Several approaches succeeded in training networks on simple context free resp. context sensitive languages which include a counting mechanism [34, 45]. This is particularly interesting since natural language is believed to be somewhere between the class of context free and context sensitive languages.

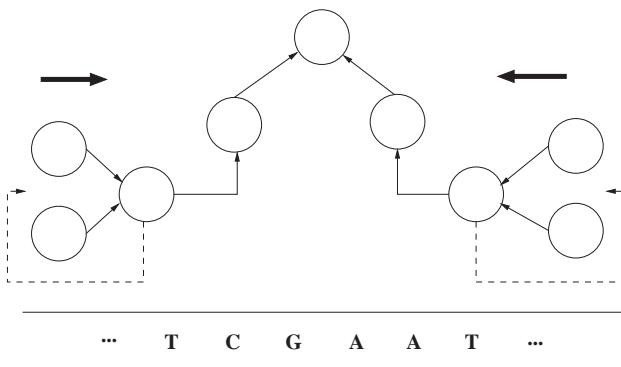
When neglecting learnability of the languages, one can even go a step further and connect recurrent networks to general Turing machines. The notion of simulation of possibly non halting computations with Turing machines in polynomial time has to be established. For this purpose, the Turing band is encoded in the activation of several specific neurons, and a specific neuron is identified which indicates the halting of a computation. Under this condition, it has been shown in [46] that neural networks with rational weights and the so-called semilinear activation function ( $\text{lin}(x) = 0$  for  $x \leq 0$ ,  $\text{lin}(x) = x$  for  $x \in [0, 1]$ ,  $\text{lin}(x) = 1$  for  $x \geq 1$ ) can simulate every Turing machine. A similar result (but requiring exponential time) has been shown for sigmoidal recurrent networks in [47]. Interestingly, recurrent networks with arbitrary (irrational) weights are even more powerful: sigmoidal networks can compute every function (in exponential time) on unary inputs [48]. Recurrent networks with arbitrary weights and the semilinear activation function are in fact equivalent to non-uniform Boolean circuits [46], a class which is beyond Turing computability. Of course, these functions cannot, in principle, be learned from a finite training set, such that the relevance of these results is more of theoretical interest.

### 1.15.3.5 Recurrent networks for structure processing

Partial recurrent neural networks can be used for time series processing, e.g., language, sensor data, financial time series, etc. Thereby, the structure of the data coincides with the dynamics of the network due to the time dependency of the series. A variety of generalizations of recurrent networks has been proposed which extend the applicability to more general data structures.

### 1.15.3.6 Spatial data

Spatial data such as DNA sequences differ from time series in their causality: for time series, strong causality holds in the sense that the future depends on the past but not vice versa. For spatial sequences, entries at the beginning or the end might influence each other without any restriction. Thus, if sequences are processed by recurrent networks, a causality assumption which might not be correct is assumed. In particular, the result of the processing can be different depending on the fact whether sequences are processed from the right or the left. Because of this fact, *bicausal neural networks* have been proposed in the approach [49]. These networks consist of two recurrent neural networks which process the sequence

**FIGURE 15.5**

Bicausal recurrent network for spatial sequence processing, e.g., DNA sequences. Due to two “directions” of the processing, spatial dependencies can be captured by these networks.

simultaneously from the left resp. from the right. The readout processes the output of both networks together such that, at this position, full information is available. The network is trained as a whole, i.e., full information is available also during training, see Figure 15.5. This method has successfully been used for splice site recognition and various tasks in protein structure prediction [49–51].

Another alternative model has been proposed in [52]. This is based on so-called recursive cascade correlation. *Cascade correlation* constitutes a method to simultaneously obtain the architecture and the weights of feedforward networks. It starts with a single output neuron without hidden neurons which is trained on the given training task. Then the following procedure is repeated until the classification result is satisfactory: a hidden neuron is added and connections from the input neuron and all existing hidden neurons to the new one are drawn. These connections are trained such that the correlation of the output of this hidden neuron and the error of the network as exists so far is maximized. The idea behind this procedure is that the hidden neuron can then be used for error correction. After training the connections to the hidden neuron are frozen. After introducing a hidden neuron, all connections to the output are trained such that the classification error is minimized. This procedure is repeated, adding the necessary number of hidden neurons while training the network.

The same procedure can be introduced for recurrent networks whereby a hidden neuron is pointed to by recurrent connections from the neuron itself and all previous hidden neurons. This way, we can introduce functional dependencies not only from the activation in the given time step, but also the previous time step. The principled training procedure is the same as for simple cascade correlation, whereby error signals are back-propagated through the recurrent connections as in back-propagation through time.

A *recurrent cascade correlation network* (RCC), unlike a simple recurrent network, possesses restricted recurrence since no recurrent connections from hidden neurons pointing to neurons introduced earlier are available. This restriction has the effect that RCC is strictly weaker than full recurrent networks since, as shown in [53], RCC networks cannot represent all finite automata, whereas recurrent networks can. However, this restriction has the effect that recurrent connections which refer to later time steps can be introduced as long as this is restricted to hidden neurons which are already frozen.

The resulting dynamic definition is acyclic. These networks, *contextual recurrent cascade correlation*, have been successfully introduced in [52]. They are trained in the same way as RCC networks, but integrate more context suitable for spatial data. It has been shown in [54] that, although these networks are weaker compared to recurrent networks when considering approximation for arbitrary size of the sequences, they are approximation complete when considering approximation in probability. Thus, they can approximate every possibly non-causal function on sequences up to sets of small probability.

### 1.15.3.7 Tree structures

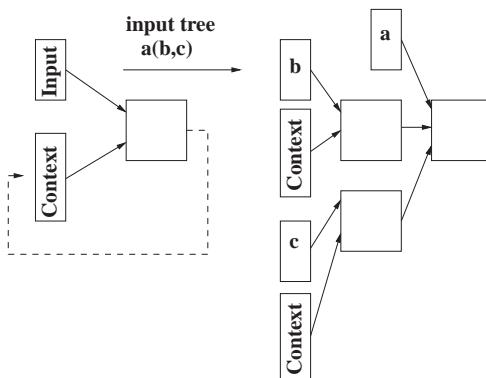
The idea of a time series driving a recurrent network can immediately be transferred to tree structures with limited fan-out since they possess a similar recursive structure. Assume  $v$  is a vertex of a tree with label  $l(v)$ , and  $l$  and  $r$  are the left and right subvertex, respectively. Then the state of a *recursive network* is defined as

$$z(v) = f(l(v), z(l), z(r)).$$

Thus, starting from the leaves of the tree, an output for every vertex of the tree can be obtained by recursive processing of the vertices within the context of their children, see Figure 15.6. These recursive networks have been proposed, e.g., in [55] and successfully been applied to various tasks including fingerprint recognition, logo recognition, classification chemical structures, etc. Interestingly, they constitute universal approximators for mappings on tree structures to real vectors as shown in [56].

### 1.15.3.8 Graph structures

RCC can be extended to *recursive cascade correlation* (RecCC) by using the same recurrence which takes the context of all children of a vertex into account, not just one single successor. It has been shown in [54] that RecCC is approximation complete for tree structures when considering approximation in



**FIGURE 15.6**

Unfolding of a recursive neural network for a given tree structure. The dynamics of standard recurrent networks is extended to tree structures with a fixed fan-out of the vertices.

probability, as beforehand. Interestingly, contextual processing can be added to RecCC in the same way as CRCC: each hidden neuron is equipped not only with recurrent connections referring to the children, but also recurrent connections referring to its parents pointing from all hidden neurons introduced previously. This way, we cannot only process tree structures, but also acyclic graphs with *contextual cascade correlation* (CRecCC). Interestingly, one can show that such networks are approximation complete for functions on tree structures as well as for acyclic graphs which can be pairwise distinguished by at least one path in the graph [54], thus the resulting architecture is quite powerful.

Another approach has been proposed in [57]. The graph neural network processes every vertex in the context of its neighbored vertices in the graph. That means, the activation of a neuron for a specific vertex in the graph is computed as function depending on the labeling of the vertex, and the activation of the neurons for the neighbors of the vertex. Since a graph is usually cyclic, hence activations of neighbored vertices influence each other, this definition depends on the ordering of the computation of the activations. In the graph neural network, the functions implemented by the neurons are restricted to contractive mappings. This way, the standard Banach fixed-point theorem guarantees that, in the limit, a unique activation is computed for every neuron and, thus, the overall dynamics is well defined. In [57], a training algorithm based on gradient descent is proposed which propagates the error signals of stable states of the network. This way, an efficient training procedure arises. Graph networks have been successfully applied, e.g., in the context of web page rankings and applications to chemistry.

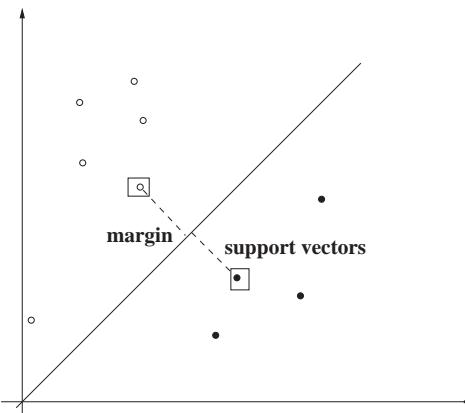
## 1.15.4 Learning by focussing on the generalization ability

So far, networks have been trained in the following way: a fixed architecture and training algorithm is chosen. Then, training aims at a minimization of the error on the training set. This method is referred to as *empirical risk minimization* [25] since, after a limitation of the principled function class by the architecture and training algorithm, the aim is to get an empirical error as small as possible. However, we are not interested in the empirical error, rather the generalization error, i.e., the behavior of the network for unknown examples should be considered. Thus, the goal is twofold: minimization of the empirical error and minimization of the structural freedom of the function beyond the training set.

There exist alternative training methods which explicitly address the so-called *structural risk* during training, i.e., the degrees of freedom which are not necessary to suit the training set.

### 1.15.4.1 Support vector machine

One of the most popular methods based on structural risk minimization is offered by the support vector machine (SVM). First, we consider a linear SVM, which is just a simple perceptron. For simplicity, we neglect the bias (which is often not necessary in practical applications). There are two different methods to limit the generalization ability of a perceptron: on the one hand, the number of parameters (the more parameters, the more data points are necessary to determine the parameters). On the other hand, we can consider the so-called *margin* which is the minimum distance of a training point from the separating hyperplane (see Figure 15.7). The larger the margin, the more tolerance with respect to noise in the training patterns can be observed [25].

**FIGURE 15.7**

A linear classifier with maximum margin; the support vector machine uses the margin trick to reach good generalization also in high dimensional feature space.

#### 1.15.4.1.1 Primal and dual problem

The margin can be parameterized as  $2/\|\mathbf{w}\|$ ,  $\mathbf{w}$  being the weight vector of the perceptron. Therefore, maximizing the margin is equivalent to a minimization of  $\|\mathbf{w}\|^2/2$ . Assume patterns have the form  $(\mathbf{x}^p, y^p)$  with  $y^p$  being in  $\{-1, 1\}$  (instead of  $\{0, 1\}$ ). Then  $\mathbf{x}^p$  is mapped correctly (w.l.o.g. we can assume that the activation of the perceptron is of absolute value at least one) if and only if  $y^p \cdot \mathbf{w}^t \mathbf{x}^p \geq 1$ . This yields to the *primal formulation* of SVM training for a linear SVM without bias:

$$\begin{aligned} &\text{minimize } \|\mathbf{w}\|^2/2 \\ &\text{such that } y^p \cdot \mathbf{w}^t \mathbf{x}^p \geq 1 \text{ for all } p. \end{aligned}$$

Usually, this problem is transferred to the equivalent Wolfe dual problem using the standard Karush-Kuhn-Tucker conditions of quadratic programming [25]:

$$\begin{aligned} &\text{maximize } \sum \alpha_i - 1/2 \cdot \sum \alpha_i \alpha_j y^i y^j (\mathbf{x}^i)^t \mathbf{x}^j \\ &\text{such that } \alpha_i \geq 0. \end{aligned}$$

The Lagrange parameter  $\alpha_i$  will become 0 for all points but the so-called *support vectors* which are vectors closest to the separating hyperplane. The hyperplane can be recovered from the Lagrange parameters by  $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}^i$ .

#### 1.15.4.1.2 Adatron

Usually, this optimization problem is solved using, e.g., efficient interior point methods. However, a very simple optimization scheme is offered by the so-called adatron algorithm [58], a gradient ascent of the dual cost function which takes the constraints into account:

```
init  $\alpha_i = 1$ 
repeat
```

$$\begin{aligned}\alpha_i &:= \alpha_i + \eta \cdot \left(1 - y^i \sum_j \alpha_j y^j (\mathbf{x}^j)^t \mathbf{x}^i\right) \\ \alpha_i &:= \max\{\alpha_i, 0\}\end{aligned}$$

This procedure finds a solution of the perceptron problem which optimizes the margin. Unlike the solution provided by the perceptron algorithm, the solution of a SVM is unique due to the regularization in terms of the margin. The result is robust with respect to its generalization ability. It has been shown in [25] that the VC dimension, i.e., a measure for the degree of freedom within the function class as introduced above, scales inverse with the margin. Thus, the larger the margin, the better theoretical generalization bounds. In addition, the solution with maximum margin turns perceptron training into an efficient polynomial problem: the dual optimization problem can be solved in time  $\mathcal{O}(n^3)$ ,  $n$  being the number of points.

#### 1.15.4.1.3 Alternative formulations

So far, the training algorithm finds a solution if and only if a separating hyperplane without errors exists. In general, this is not possible. For this case, the optimization problem for SVM training can be relaxed introducing *slack variables*  $\xi_p$ :

$$\begin{aligned}&\text{minimize } \|\mathbf{w}\|^2/2 + C \sum_p \xi_p \\ &\text{such that } y^p \cdot \mathbf{w}^t \mathbf{x}^p \geq 1 - \xi_p, \\ &\quad \xi_p \geq 0 \text{ for all } p.\end{aligned}$$

where  $C$  is a positive constant which regulates the number of allowed misclassifications. The Wolfe dual yields

$$\begin{aligned}&\text{maximize } \sum \alpha_i - 1/2 \cdot \sum \alpha_i \alpha_j y^i y^j (\mathbf{x}^i)^t \mathbf{x}^j \\ &\text{such that } C \geq \alpha_i \geq 0.\end{aligned}$$

That is, the size of the Lagrange variables is limited to prevent divergence. Support vectors are the vectors where the Lagrange parameter is non vanishing which includes points closest to the separating hyperplane ( $\alpha_p < C$ ) and misclassified points ( $\alpha_p = C$ ).

Note that, again, due to the regularization, the NP hard problem of training a perceptron in case of errors is substituted by a polynomial optimization problem. This complexity can even be improved. Working set techniques such as sequential minimal optimization [59] can severely speed up SVM training in practical applications, however, without formal guarantees concerning their computational complexity. Provably linear techniques rely on approximation algorithms by means of core set techniques, see, e.g., [60]. There exists a variety of further SVM training schemes which address different problems than simple classification such as regression problems, ranking problems, novelty detection, etc. [61].

#### 1.15.4.2 Kernels

Similar to the simple perceptron the linear SVM is appropriate only for those training sets which are (almost) linearly separable. To extend the applicability, the so-called kernel trick is used. The basic

idea is to include a fixed nonlinear preprocessing  $\Phi$  of the data points before separating the nonlinearly transformed points. If the dimensionality of the feature space is large enough and the preprocessing is sufficiently nonlinear, the image points  $\Phi(\mathbf{x}^p)$  become linearly separable [62, 63].

Two problems arise: on the one hand, the dimensionality becomes large such that the number of free parameters is large and the generalization ability might become weak. This problem is prevented by the fact that the hyperplane with maximum margin is chosen.

#### 1.15.4.2.1 The kernel trick

On the other hand scalar products have to be computed in the feature space. For high dimensionality, this becomes ineffective. To prevent this problem, the *kernel trick* is used: only those  $\Phi$  are chosen for which a mapping  $k$  exists with  $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^t \Phi(\mathbf{y})$ , i.e.,  $k$  provides an alternative efficient computation method for the scalar product of the nonlinear images of two vectors. Note that  $\Phi$  is used within SVM training and evaluation only within a scalar product such that the knowledge of  $k$  is sufficient for SVM training. Functions  $k$  which fulfill the equality  $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^t \Phi(\mathbf{y})$  are called *kernels*. Interestingly, one can check whether a function is a kernel without an explicit knowledge of  $\Phi$ , using, e.g., the so-called Mercer condition [61].

Popular kernels include

1. The *polynomial kernel*  $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^t \mathbf{y} + 1)^d$ ,  $d$  denoting the degree. This kernel corresponds to a map  $\Phi$  which builds all monomials of the input dimensions up to degree  $d$ , i.e., the dimensionality of the feature space scales exponentially in the input dimensionality.
2. The *Gaussian kernel*  $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\sigma^2)$ ,  $\sigma$  being the width of the Gaussians. This kernel corresponds to a map  $\Phi$  with infinite dimensional feature space.

Recently, a kernel inspired by extreme learning machines has been proposed in [64]. The *extreme learning machine kernel* relies on the idea to randomly map data into a feature space by means of a layer of randomly wired hidden neurons. Interestingly, under certain assumptions, the resulting dot product in feature space can be computed explicitly for the limit of an infinite number of neurons.

#### 1.15.4.2.2 Structure kernels

Kernels constitute the interface of the data to the SVM. Thereby, the data need not be restricted to real vectors, rather, every space for which a kernel can be defined is appropriate. A variety of kernels for structural inputs including sequences, trees, and graphs have been proposed. Mathematical closure properties as established in [65, 66] allow to design kernels based on a variety of principles including the reference to projections or substructures and the combination of such procedures. Two different approaches to design structure kernels are very popular: kernels which count the number of common occurrences of substructures, and kernels which base the comparison of structures on probabilistic models.

Substructure kernels include, for example, the *string kernel* and the *spectrum kernel* for sequences, which count common subsequences, whereby the variants differ in the fact whether subsequences need to be contiguous or not and whether exact matches are required [67, 68]. In principle, given two strings  $s_1$  and  $s_2$ , these kernels compute a term  $\sum_{s \subset_k s_1, s' \subset_k s_2} d(s, s')$  whereby  $s \subset_k s_1$  refers to some formalism which specifies that  $s$  is substring of  $s_1$  (contiguous or noncontiguous) and its length is limited to  $k$ .  $d(s, s')$  defines a simple kernel on strings of the same length. The main problem of this definition

consists in the task to design an efficient computation method for this kernel, since the number of possible substrings increases exponentially with the length  $k$ . Basically two ways have been proposed which allow a rather efficient computation: dynamic programming or suffix trees. Generalizations to tree structures are possible as described in [69, 70]. Similar procedures for graph structures face problems because of the complexity of graph isomorphism. Here, an alternative is given by just comparing paths in a graph. The *diffusion kernel* constitutes a concrete method to realize this idea [71].

One of the most popular kernels based on probabilistic models is the *Fisher kernel* [72]. The comparison of two structures is substituted by the comparison of relevant quantities of statistical models fitted to the data. In the case of the Fisher kernel, a manifold of parameterized generative models is considered, e.g., hidden Markov models in the case of sequences. Each data point stems from a most likely parameter setting. The Fisher kernel compares the tangent vectors in these points instead of the structures itself, whereby the natural metric to compare these tangents is based on the Fisher information.

### 1.15.4.3 Learning vector quantization

The SVM provides solutions in terms of support vectors which are points at the boundary or misclassified examples, i.e., the SVM expands solutions in terms of atypical training points. There exist alternatives such as the *relevance vector machine* which selects solutions in terms of typical points, based on a statistical formulation of the training problem [73]. However, we would like to consider an even simpler method which represents solutions in forms of typical points or prototypes: *learning vector quantization* (LVQ) [12]. An LVQ network consists of prototypes which are represented by prototypical locations  $w_i$  in the input space and their output class  $c(w_i)$  which is one of a finite number of possible output classes. An LVQ network computes a classification by means of a *winner takes all* mechanism:

$$\mathbf{x} \mapsto c(\mathbf{w}_i) \quad \text{for which } \|\mathbf{x} - \mathbf{w}_i\| \text{ is minimal,}$$

$\|\cdot\|$  denoting the euclidian metric, see Figure 15.8 for an example. A variety of different intuitive learning algorithms have been proposed which are commonly based on the principle of Hebbian learning and which constitute heuristics. The simplest one, LVQ1, can be described in a few lines:

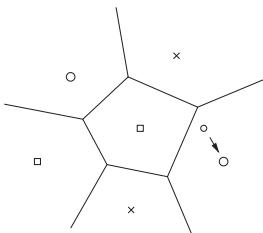
```

init  $w_i$ 
repeat
    choose a training pattern  $\mathbf{x}$  with output class  $y$ 
    determine the closest prototype  $\mathbf{w}_i$ 
    if  $y = c(\mathbf{w}_i)$  adapt  $\mathbf{w}_i = \mathbf{w}_i + \eta(\mathbf{x} - \mathbf{w}_i)$ 
    if  $y \neq c(\mathbf{w}_i)$  adapt  $\mathbf{w}_i = \mathbf{w}_i - \eta(\mathbf{x} - \mathbf{w}_i)$ 

```

Figure 15.9 depicts a learning step.

There exists a variety of alternatives to account for faster convergence or better adaptation to the optimum decision boundary. Only comparably little is known about theoretical guarantees for LVQ such as the convergence of the algorithm. Recently, several prototypical scenarios have been investigated within the framework of so-called online learning [74] which demonstrate the divergence of some algorithms (e.g., LVQ2.1) also in quite simple situations. We would like to introduce one alternative which can be derived as stochastic gradient descent method of a cost function and, therefore, shows quite good numerical stability, *generalized LVQ* (GLVQ)[75].

**FIGURE 15.8**

An LVQ network: the classification is based on the winner-takes-all rule, this way decomposing the input space into receptive fields.

**FIGURE 15.9**

Training of an LVQ network by Hebbian learning; given a data point, its closest prototype is moved towards/away from the data point depending on whether the classification is correct.

Assume patterns  $(\mathbf{x}^p, y^p)$  are available. Denote by  $\mathbf{w}^+(\mathbf{x}^p)$  the prototype closest to  $\mathbf{x}^p$  with the same label and by  $\mathbf{w}^-(\mathbf{x}^p)$  the closest prototype with a different label. Then, GRLVQ minimizes the cost function

$$\frac{1}{2} \sum_p \frac{\|\mathbf{x}^p - \mathbf{w}^+(\mathbf{x}^p)\|^2 - \|\mathbf{x}^p - \mathbf{w}^-(\mathbf{x}^p)\|^2}{\|\mathbf{x}^p - \mathbf{w}^+(\mathbf{x}^p)\|^2 + \|\mathbf{x}^p - \mathbf{w}^-(\mathbf{x}^p)\|^2}$$

by means of a stochastic gradient decent. In some cases, the summands are nonlinearly transformed by a sigmoidal function. Note that the term  $\|\mathbf{x}^p - \mathbf{w}^+(\mathbf{x}^p)\|^2 - \|\mathbf{x}^p - \mathbf{w}^-(\mathbf{x}^p)\|^2$  is negative if and only if  $\mathbf{w}^p$  is correctly classified, thus, the nominator accounts for a correct classification of the data points. The denominator scales this term such that it lies in the interval  $(-1, 1)$ .

We use the shorthand notation  $d^+$  for  $\|\mathbf{x}^p - \mathbf{w}^+(\mathbf{x}^p)\|^2$  and  $d^-$  for  $\|\mathbf{x}^p - \mathbf{w}^-(\mathbf{x}^p)\|^2$ . Taking the derivatives yields the following update rules which are also valid for the classification boundaries, as shown in [76]:

$$\mathbf{w}^+(\mathbf{x}^p) := \mathbf{w}^+(\mathbf{x}^p) + \eta \cdot \frac{2d^-}{(d^+ + d^-)^2} \cdot (\mathbf{x}^p - \mathbf{w}^+(\mathbf{x}^p))$$

for the closest correct prototype and

$$\mathbf{w}^-(\mathbf{x}^p) := \mathbf{w}^-(\mathbf{x}^p) - \eta \cdot \frac{2d^+}{(d^+ + d^-)^2} \cdot (\mathbf{x}^p - \mathbf{w}^-(\mathbf{x}^p))$$

for the closest incorrect prototype, i.e., this method differs from standard LVQ mainly by different scaling terms of the step size. The method is often more robust and yields a better classification accuracy compared to simple LVQ [76].

For LVQ type networks, strong generalization bounds hold in terms of the so-called hypothesis margin of the system. Consider the difference  $d^- - d^+$ , the *hypothesis margin* for pattern  $\mathbf{x}^p$ . Obviously, the larger the term, the more secure the classification of  $\mathbf{x}^p$ . For large margin, the hypothesis can be altered, i.e., the classification boundary can be shifted, without affecting the classification of  $\mathbf{x}^p$ . For small hypothesis margin, already a small shift of the hypothesis will alter the classification. It has been shown in [77] that, for a two-class classification problem, the generalization error of LVQ type networks can be limited by the following inequality:

$$E \leq \hat{E}_\rho + \mathcal{O}\left(\frac{P^2 B^3 \sqrt{\ln(1/\delta)}}{\rho \sqrt{m}} + \frac{\sqrt{\ln(1/\delta)}}{\sqrt{m}}\right),$$

where  $P$  is the number of prototypes,  $B$  the size of the support of the training set,  $\delta$  the confidence of the bound,  $m$  the number of examples, and  $\rho$  a priorly fixed constant.  $E$  denotes the error of the classifier for unknown training samples, i.e., the error we are interested in, and  $\hat{E}_\rho$  denotes the number of examples which are classified with margin smaller than  $\rho$  or misclassified, divided by  $m$ . Hence, the larger the margin achieved by an LVQ classifier, the better the generalization.

Note that GLVQ includes the margin in the cost function, hence it directly aims at margin maximization during training, achieving excellent generalization ability.

#### 1.15.4.4 General metrics

Note that, in principle, every differentiable metric  $d$  can be used in GRLVQ instead of the euclidian one. Substituting this metric into  $d^+$  and  $d^-$  and taking the derivative directly yields to adaptation formulas for training as shown in [78]. The metric can, for example, be derived from an arbitrary kernel (in which case the generalization bounds are still valid), or any other similarity measure (for which the generalization bounds do not necessarily hold.) The choice of the metric as the standard euclidian metric enhanced by *relevance terms* proved particularly suitable for practical applications:

$$d^\lambda(\mathbf{x}, \mathbf{w}) = \sum_i \lambda_i (x_i - w_i)^2,$$

where  $\lambda_i \geq 0$  and  $\sum \lambda_i = 1$ . The quantity  $\lambda_i$  scales input dimension  $i$  according to its relevance, therefore the algorithm is referred to as *generalized relevance LVQ* (GRLVQ) [76]. Thereby, the relevance of the dimensions need not be known a priori, rather, the relevance terms can automatically be determined during training by applying a stochastic gradient decent with respect to both, the prototypes and the relevance terms. The resulting updates for the relevance terms are

$$\lambda_l := \lambda_l + \eta \cdot \sum_p \left( \frac{4d^-}{(d^+ - d^-)^2} \cdot (x_l^p - w_l^+(\mathbf{x}^p))^2 - \frac{4d^+}{(d^+ - d^-)^2} \cdot (x_l^p - w_l^-(\mathbf{x}^p))^2 \right).$$

After every adaptation step, the relevance terms have to be normalized. Note that the generalization bounds are also valid for this more general approach as shown in [77].

A variety of further alternatives have been proposed and successfully applied in practice, whereby the metrics can be equipped with adaptive relevance parameters which are automatically adapted during training by means of a gradient descent. Examples for alternative metrics include specific choices for time series [78,79] or a full adaptive matrix [80].

## 1.15.5 Unsupervised learning

So far, we have considered classification or regression tasks, i.e., input-output pairs have been available. In unsupervised learning, only input vectors are given and the task is to extract useful information from these data. This problem is, of course, rather vague and a variety of different application domains exists: data preprocessing and dimensionality reduction, data visualization, data clustering, data compression, data mining, etc. We will consider a few popular approaches within this framework.

### 1.15.5.1 Unsupervised training of single neurons

Assume a stream of vectors  $\mathbf{x}^p$  is given. For simplicity, we assume that the expectation  $E(x_i^p)$  of each component is 0, otherwise, we can easily normalize every dimension. We assume that a single neuron is given with the identity as activation function and bias 0, i.e., it computes the function

$$\mathbf{x} \mapsto y = \mathbf{w}' \mathbf{x}.$$

Since we do not exactly know what we should do, we can formulate simple Hebbian learning in this scenario

$$\mathbf{w} := \mathbf{w} + \eta \cdot y \cdot \mathbf{x},$$

which, for positive output, reinforces those weights of the neuron which receive a positive input signal, for negative  $y$ , it is vice versa. However, unfortunately, this simple rule diverges, i.e., the size of  $\mathbf{w}$  becomes infinite.

#### 1.15.5.1.1 Oja's rule

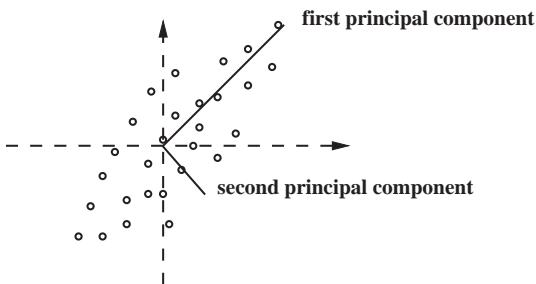
*Oja's rule* introduces an additional correction to Hebb learning [5]:

$$\mathbf{w} := \mathbf{w} + \eta \cdot (y \cdot \mathbf{x} - y^2 \cdot \mathbf{w}).$$

What is the behavior of this dynamics? We define the correlation matrix  $C$  by  $E(\mathbf{x}\mathbf{x}^t)$ . For a finite set of samples this is proportional to the sum  $\sum_p \mathbf{x}^p (\mathbf{x}^p)^t$ . Now consider the function  $-1/2 \cdot \mathbf{w}' C \mathbf{w}$ . A stochastic gradient descent of this function yields an update proportional to  $\mathbf{x}\mathbf{x}^t \mathbf{w} = y\mathbf{x}$ , i.e., it yields the Hebb term as introduced above. Obviously, the above function is smaller the larger the size of  $\mathbf{w}$ , hence the divergence. One can show that the additional correction term introduced by Oja's rule also optimizes this function, but it restricts the solution to  $\|\mathbf{w}\| = 1$  [5].

Thus, Oja's rule leads to a vector  $\mathbf{e}$  with  $\|\mathbf{e}\|^2 = 1$  such that  $\mathbf{e}' C \mathbf{e}$  is maximum. The matrix  $C$  is a symmetric positive (semi-)definite matrix, therefore it can be diagonalized, i.e., there exist  $n$  eigenvectors  $\mathbf{e}_i$  with  $\|\mathbf{e}_i\|^2 = 1$  and eigenvalues  $\lambda_i$  such that  $C\mathbf{e}_i = \lambda_i \mathbf{e}_i$ , and the matrix  $C$  is the diagonal matrix with entries  $\lambda_i$  when represented in the base  $\mathbf{e}_i$ . (Here we assume that  $C$  is positive definite and the eigenvalues are pairwise different.) The optimum  $\mathbf{e}$  is nothing else but the eigenvector corresponding to the largest eigenvalue  $\lambda_1$ .

The decomposition of  $C$  into its eigenvectors is also called *principal component analysis* (PCA) and the eigenvectors are the *principal components*. They have an intuitive meaning: the principal components are pairwise orthogonal, and the first principal component is the direction of the data with largest

**FIGURE 15.10**

Principal components of a data set: the directions with maximum variation are captured this way.

variance, the second one is the direction with second largest variance, and so on, see Figure 15.10. Therefore, PCA is often used for data preprocessing and data compression since they preserve as much information in terms of variances of the points as possible. Oja's rule does nothing else but extract the dimension with maximum information in this sense.

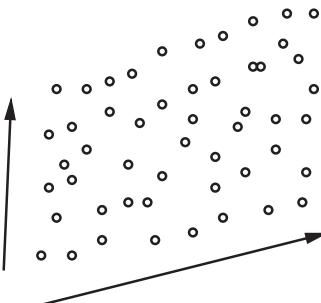
Extensions of Oja's rule to extract more than only the first principle component exists by repeated application of Oja's rule and decorrelation of the directions, e.g., Sanger's rule.

### 1.15.5.1.2 Independent component analysis

*Independent component analysis* (ICA) [81] is, in some sense, dual to PCA. As for PCA,  $n$ -dimensional patterns  $\mathbf{x}$  are observed, which are generated by unknown sources  $\mathbf{s}$ . Both, PCA and ICA assume that the observations are generated by a linear mapping  $\mathbf{x} = \mathbf{A} \cdot \mathbf{S}$  with  $\mathbf{A} = \mathbf{W}^{-1}$ ,  $\mathbf{W}$  denoting the weights of the trained neurons. The goal of PCA is to recover these sources under the assumption that they are pairwise orthogonal and they explain as much information as possible with respect to the variance, which is relevant, e.g., for compression. ICA tackles the problem that the sources are mixed as in real life, e.g., the mixing of different speakers in a cocktail party where the original speakers should be retrieved from a mixture of different speakers. These signals are usually not orthogonal. The assumption of ICA is, that the signals are (almost) independent (and non Gaussian since, otherwise, the mixture would just look the same).

The task of ICA is, thus, to retrieve the sources  $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$  from a mixed signal  $\mathbf{x}$  such that the signals  $s_1, s_2, \dots$  are as independent as possible. See Figure 15.11 for an example where two independent uniformly distributed sources are mixed, here, the task is to retrieve the two directions of the parallelogram. Now the question occurs how to measure independence? It is assumed that the expectation of each component of  $\mathbf{x}$  is 0 and the variance is 1, further, it is assumed that they are uncorrelated. If these assumptions are not fulfilled, data have to be preprocessed accordingly, so called *whitening*.

Independence of sources can be measured by the mutual information  $I(\mathbf{x}) = \sum_i H(x_i) - H(x_1, \dots, x_n)$  where  $H(\mathbf{x}) = -\sum P(x_i) \log P(x_i)$  denotes the entropy. For whitened data it holds  $I(\mathbf{x}) = \text{const} - \sum_i J(x_i)$  where  $J(x) = H(x_G) - H(x)$  is the negentropy which compares the variable  $x$  to a Gaussian variable  $x_G$ . Thus, minimizing the mutual information is the same as maximizing the negentropy, i.e., minimizing the Gaussianity of a source. Thus, we have to find sources which are

**FIGURE 15.11**

Linear mixture of two independent uniform distributions; unlike principal component analysis, independent component analysis aims at retrieving these dimensions.

minimum Gaussian to find independent sources. A classical measure for the Gaussianity of a variable is the kurtosis  $\text{kurt}(x) = E(x^4) - 3E(x^2)^2$ . This is maximized by one of the first ICA methods by means of a simple gradient ascent, obtaining the first independent component. Proceeding this way, all independent components can be obtained.

However, the optimization of kurtosis is not very robust. Therefore, alternatives have been proposed, among them fast-ICA, which approximates the negentropy and maximizes this approximation by a fast fixed point method resulting in the update:

$$\mathbf{w} := E(\mathbf{x}g(\mathbf{w}^t \mathbf{x})) - E(g'(\mathbf{w}^t \mathbf{x}))\mathbf{w}$$

followed by normalization where  $\mathbf{x}$  is the whitened signal and  $g$  a non quadratic function such as  $g(x) = -\exp(-x^2/2)$ .

#### 1.15.5.1.3 Nonlinear components

Both, PCA and ICA retrieve linear mixtures. This gives quite remarkable results in a variety of applications such as blind source separation of signals. However, in practice, there is no guarantee that sources are mixed in a linear way. Therefore, both ICA and PCA have been extended to nonlinear methods, e.g., introducing kernelization [61].

Here, we shortly mention another very intuitive method. Assume, as before, signals  $\mathbf{x}$  are given. PCA finds directions such that as much information as possible is preserved, i.e., one can think of PCA as a form of compression or dimensionality reduction. Consider a simple feedforward network with  $n$  inputs and outputs and one hidden layer with  $n' < n$  neurons,  $n$  being the dimensionality of the data. Assume this network is trained such that it approximates the identity on a given set of patterns  $\mathbf{x}^P$ . Since  $n' < n$ , the hidden layer serves as an encoder layer where the information of the data has to be compressed. Thus, the transformation of the hidden neurons likely resembles directions similar to the first principle components, since these directions allow a fairly effective encoding. Naturally, transformation within the vector space spanned by the first components do not alter this result. This idea can immediately be transferred to nonlinear directions: instead of just one hidden layer, more than one hidden layer is added. This way, the encoding function becomes nonlinear.

### 1.15.5.2 Slow feature analysis

Another recent approach to blind source separation and, more generally, to retrieve semantically meaningful information in a stream of observations is offered by slow feature analysis (SFA) [82]. The idea is to nonlinearly transform time dependent input signals such that the results have small variation over time, the function depends on one time step only, and the results are orthogonal to avoid trivial solutions. This way, the technique is forced to extract signals which do not change quickly over time; these signals correspond to potentially meaningful semantic entities in the given data. Under certain assumptions on the form of the nonlinear transformation, the problem can be solved algorithmically by a standard eigenvalue problem for preprocessed signals. Slow feature analysis has proven beneficial in diverse areas ranging from brain computer interfaces to image processing.

### 1.15.5.3 Training topographic maps

Unlike PCA and ICA, topographic maps combine several neurons which are trained in an unsupervised way, whereby the neurons are cooperating. This way, complex behavior such as topographic ordering according to local PCA directions can be achieved.

#### 1.15.5.3.1 Self organizing map

The self-organizing map (SOM) has been proposed by Kohonen [12] as a plausible learning method to achieve a topographic mapping of signals as can be observed in the human cortex. A SOM consists of a number of neurons which are equipped with a weight  $\mathbf{w}_i \in \mathbb{R}^n$  representing the typical sensor signal for this neuron, and a neighborhood structure of the neurons  $n(i, j)$ . Often, the neighborhood structure is induced by a regular lattice structure such as a square, hexagonal, or hyperbolic lattice. A SOM computes a *winner takes all* function just as LVQ networks  $\mathbf{x} \mapsto i$  where  $\|\mathbf{x} - \mathbf{w}_i\|$  is minimum. Thus, a new signal is mapped to a position in the map.

#### 1.15.5.3.2 Online training

Training takes place such that a topographic mapping is found which faithfully represents the input signals. Online SOM training differs from LVQ in the fact that the neighborhood structure is taken into account:

```

init
repeat
  choose  $\mathbf{x}$ 
  determine the winner neuron  $\mathbf{w}_{i_0}$ 
  adapt all neurons  $\mathbf{w}_i := \mathbf{w}_i + \eta \cdot \exp(-n(i, i_0)/\sigma^2) \cdot (\mathbf{x} - \mathbf{w}_i)$ 
  decrease  $\eta$  and  $\sigma$ 

```

Hence, all neurons are adapted whereby the strength of the adaptation is defined by a Gaussian function around the winning neuron.  $\sigma$  defines the neighborhood range of the adaptation.

Neighbored neurons in the map represent neighbored regions in the space of stimuli. If a two-dimensional map is used, a SOM can be used for data visualization: data are assigned to the place in the plane which corresponds to the respective winning neuron.

The degree of topology preservation of the resulting map can be measured using, e.g., the topographic product (if the curvature of the manifold is not too strong) or the topographic function [83]. Both measures relate the neighborhood structure in the lattice space to the neighborhood structure in the input space.

Despite the simple training model, the mathematics of SOM is rather difficult and, in several parts, still unexplored [84, 85]. Important topics concern the convergence of SOM to a topology preserving state, and the relation of the weight distribution to the data density. For both questions, only preliminary solutions (e.g., for low dimensional settings or constant learning rate) could be found so far. If the notion of the winner of  $x$  is changed to the neuron  $w(x) = i$  with average smallest distance, i.e., smallest  $\sum_j \exp(-n(i, j)/\sigma^2) \|w_j - x\|^2$ , a cost function which is also valid for continuous input distributions can be found [86]

$$\int \sum_i \delta_{w(x)}^i \sum_j \exp(-n(i, j)/\sigma^2) \|w_j - x\|^2 P(x) dx.$$

$\delta_i^j$  denoting the Kronecker symbol. In this case, a version of SOM with a slightly changed winner notation constitutes a stochastic gradient descent of this cost function.

#### 1.15.5.3.3 Batch training

SOM training can be accelerated by an alternative optimization scheme, *batch training*. This assumes that training data  $x^i$  are given a priori. Then batch optimization proceeds as follows:

```

init
repeat
    compute the winner  $w(x^p) = \operatorname{argmin}_i \sum_j \exp(-n(i, j)/\sigma^2) \|w_j - x^p\|^2$  for every  $x$ 
    compute the weights  $w_i = \frac{\sum_j \exp(-n(i, w(x^j))/\sigma^2) x^j}{\sum_j \exp(-n(i, w(x^j))/\sigma^2)}$ 
    decrease  $\sigma$ 

```

It has been shown in [87] that this procedure converges in a finite number of steps towards a (local) optimum of the cost function defined by Heskes [86]. However, this procedure is quite sensitive to initialization due to the rapid convergence. Therefore, the weights should be initialized properly, e.g., along the first two principal components of the data.

Interestingly, batch training can be transferred to more general metrics by the so-called generalized median. Assume only pairwise distances  $d(x^i, x^j)$  are available, but no embedding vector space of the data. The *median SOM* optimizes a lattice by restricting prototype locations to the location of example data points. The distance of prototypes and data points can be easily computed this way, and the winner of every data point can be determined as beforehand. Instead of an explicit formula for  $w_i$ , the prototype  $w_i$  is chosen as the location  $x_l$  for which  $\sum_j \exp(-n(i, w(x^j))d(x^l, x^j))$  is minimum. This algorithm also converges in a finite number of steps [87].

#### 1.15.5.3.4 Consciousness

For standard SOM, the final prototypes are arranged within the input space. Thereby, the number of input signals for which a neuron becomes winner is usually not the same for different prototypes.

Rather, regions of the data space with few input stimuli are emphasized, i.e., neurons which are located in regions of the data space with sparse coverage are less often winner compared to neurons in densely covered regions. This is suboptimal from an information theoretic point of view: prototypes are used in an optimum way if the winner frequency is the same for every neuron. There exists a variety of mechanisms which prevent this behavior and which introduce so-called *magnification control* in neural training [88].

A very popular and fast heuristics is offered by the *de Sieno-algorithm* [89]. During online training, each neuron adapts its mean winner frequency after the presentation of pattern  $\mathbf{x}$  by

$$p_i := p_i + B \cdot (\delta_{w(\mathbf{x})}^i - p_i),$$

where  $0 < B \ll 1$ . The winner is determined based on the distance which is enhanced by a conscious mechanism. Denote the number of neurons by  $N$ . The optimum winner frequency is  $1/N$ . The distance is altered by a term which relates this optimum frequency to the current one, emphasizing neurons with low winner frequency. Thus, the winner is

$$w(\mathbf{x}) = \operatorname{argmin}_i \|\mathbf{x} - \mathbf{w}_i\|^2 - C \cdot (1/N - p_i),$$

where  $C > 0$  is a constant. The size of  $C$  allows to determine the strength of consciousness learning in comparison to the standard Kohonen rule.

### 1.15.5.3.5 Neural gas

SOM is beneficial if data should be visualized. However, if only a clustering or topographic map should be obtained, the choice of a fixed prior topology limits the quality of the result in case this topology does not coincide with the data topology. *Neural gas* (NG) constitutes an alternative topographic mapping which determines the neighborhood cooperation in dependence of the data, resulting in an optimum lattice structure [90,91]. Unlike SOM, the resulting connections are not regular such that visualization requires additional work.

Neural gas can be derived from the cost function

$$\int \sum_{i,j} \exp(-r_i(\mathbf{x})/\sigma^2) (\|\mathbf{w}_j - \mathbf{x}\|^2 P(\mathbf{x}) d\mathbf{x}.$$

$r_i(\mathbf{x})$  denotes the rank of neuron  $i$  when the distances  $\|\mathbf{x} - \mathbf{w}_j\|^2$  are sorted according to their size, i.e., the rank is 0 for the winner, 1 for the second closest prototype, etc. Neural gas learning constitutes a stochastic gradient descent on this cost function, resulting in the update:

```

init
repeat
  choose  $\mathbf{x}$ 
  determine the ranks  $r_i(\mathbf{x})$ 
  adapt  $\mathbf{w}_i := \mathbf{w}_i + \eta \exp(-r_i(\mathbf{x})/\sigma^2) (\mathbf{x} - \mathbf{w}_i)$ 
  decrease  $\sigma$  and  $\eta$ 

```

Obviously, during adaptation, the neighborhood cooperation is given by the rank of the neurons with respect to an input pattern. One can introduce explicit connections which constitute an optimum lattice by connecting neurons if they are the first and second winner for at least one pattern [90].

Similar to SOM, an alternative *batch optimization scheme* exists given fixed patterns  $\mathbf{x}^j$ :

```

init
repeat
    determine the ranks  $r_i(\mathbf{x}^j)$ 
    adapt  $\mathbf{w}_i := \frac{\sum_j \exp(-r_i(\mathbf{x}^j)/\sigma^2) \mathbf{x}^j}{\exp(-r_i(\mathbf{x}^j)/\sigma^2)}$ 
    decrease  $\sigma$ 

```

Similar to SOM, this version can be adapted to proximity data not contained in a real-vector space.

Like SOM, neural gas does not yield to a uniform winner frequency of the prototypes. Rather, the prototype distribution and the data distribution follow a power law with exponent  $m/(m + 2)$  where  $m$  is the intrinsic dimension of the data manifold. This is different from the information theoretic optimum 1 in particular for low dimensional intrinsic dimensionality. A variety of magnification control schemes exist for NG and batch NG, see, e.g., [83, 92].

There exists a variety of alternatives to the standard SOM which rely on different fundamental mathematical principles such as a constraint mixture modeling proposed in the context of the generative topographic mapping [93], or information theoretic approaches [94].

#### 1.15.5.4 Extensions to structures

Albeit topographic mapping depends on the distance of data points only, the algorithms require an underlying Euclidean vector space for the updates of prototypes. If complex data are dealt with, the Euclidean distance is often not appropriate. There exist two principled approaches to extend unsupervised topographic maps to general data structures: one can equip the dynamics with recurrent connections, such that time series, tree structures, or general graph structures can be dealt with, resulting in architectures very similar to supervised recursive networks for structures. Examples for such approaches can be found in [95, 96].

Alternatively, a problem specific dissimilarity measure can be taken. Examples include alignment for biological sequence data, compression distance for text, graph distances for structures, and similar. In the last years, extensions of methods such as SOM and NG to kernels or general relational data described by pairwise dissimilarities have been proposed. Median SOM or NG, as mentioned above, constitute very simple (but also very limited) approaches.

As an alternative, kernel NG extends the standard NG cost function to a kernel mapping by means of the identity

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{w})\|^2 = k(\mathbf{x}, \mathbf{x}) - 2k(\mathbf{x}, \mathbf{w}) + k(\mathbf{w}, \mathbf{w})$$

assuming a kernel  $k$  with corrsponding feature map  $\Phi$  is given. For a linear combination  $\mathbf{w} = \sum \alpha_i \Phi(\mathbf{x}_i)$ , this reduces to a term which depends on the Gram matrix of the data only. Online adaptation of the coefficients  $\alpha_i$  or batch adaptation can be derived thereof, see [97, 98].

Instead of a kernel, a general dissimilarity measure can be chosen, because of the equality

$$\|\mathbf{x} - \sum \alpha_i \mathbf{x}_i\|^2 = [D^t \alpha]_i - 1/2 \cdot \alpha^t D \alpha,$$

where  $\alpha$  refers to the vector of coefficients which sums up to one, and  $D$  denotes a symmetric dissimilarity matrix storing the pairwise dissimilarities of  $\mathbf{x}_i$ . This formula allows a direct transfer of batch

learning to general dissimilarity data [99]. Unlike for kernels, however, these adaptations correspond to vector operations in the so-called pseudo-Euclidean embedding of the given dissimilarity matrix, where negative eigenvalues can occur in the dot product. In consequence, the NG algorithm is no longer guaranteed to converge to a local optimum of the cost function. Because of quadratic optimization being NP hard in such spaces, this is partially unavoidable.

## 1.15.6 Applications

Corresponding to the wide variety of neural architectures, these techniques populate nearly every possible application area one can think of. Applications can be found from commercial areas, e.g., in the context of the financial market, to standard tools in bioinformatics such as prediction of the secondary structure of proteins, up to hot research topics such as humanoid robotics. Thus, rather than accumulating a list of application scenarios, a list of typical tasks the different architectures are used for follows:

*Feed-forward network*: Classification, regression tasks.

*Recurrent network*: Associative memory, time series prediction and generation, dynamic system modeling; can be extended to deal with structures.

*Support vector machine*: Today, SVMs constitute the standard tool for classical regression and classification tasks.

*Learning vector quantization*: Multiclass classification, relevance learning, data compression and inspection.

*ICA/PCA/SFA*: Blind source separation, data preprocessing.

*Topographic mapping*: Data mining and visualization.

## 1.15.7 Open issues and problems

We have already discussed some open problems when considering the single architectures. In general, many open problems center around the following issues:

- How can the system be made more autonomous? Neural networks constitute very efficient methods provided a human has shaped the learning problem in a suitable way, e.g., as regression problem. Unlike human learners, however, neural networks are hardly capable of structuring a complex learning problem themselves, and of actively extracting important structures inherent in the scenario. Deep learning and autonomous learning are two topics which are currently discussed in this frame.
- How can neural systems learn from very few examples? Humans are capable of rapid learning and reasonable actions albeit they have experienced only few, probably only one example of how to react. In contrast, typical neural systems require a number of examples to perform valid generalization. This problem is tackled in topics such as instantaneous learning, or models to tackle the problem of compositionality, i.e., the capability of dealing with entirely novel combinations of data where only the parts are known, but they have never been experienced in this combination.
- How can neural systems deal with complex structures? At present, most neural systems still deal with standard Euclidean vectors. Great strides towards more general data structures such as graph

structures or dissimilarity data have been made in the context of recursive approaches or structure kernels. Still, however, problems such as the computational complexity, or structured outputs pose major challenges in this realm.

- How can neural systems be interpreted? Many neural networks offer high quality solutions, but, due to their distributed representation of information, a human can hardly understand why a neural network proposes a certain output. The question of how neural networks can be inspected by humans has been tackled in the literature at several places, but no universally accepted solution exists. Approaches deal with rule extraction, or visualization, for example, but usually still require knowledge of the systems to be interpretable. With problems becoming more and more complex, often a human-in-the-loop approach is taken to eventually arrive at the relevant information. For such approaches, human insight is vital for success. Current research can be found in the field of visual analytics, for example.
- How can larger and larger problems be addressed? Electronic data are getting more and more common, such that larger and larger data sets have to be dealt with. This opens the way to new research areas such as, e.g., the necessity for streaming algorithms which require at most one loop over the data sets, or parallelization schemes to make solutions feasible, thereby relying, e.g., on possibilities as offered by cloud computing or the map-reduce framework.

---

### 1.15.8 Implementation, code, and data sets

It becomes more and more common in Neuroinformatics that publications in the field are accompanied by links where to retrieve the corresponding programs and data sets. Further, the field of neural networks constitutes a standard topic taught at many universities, often accompanied by demo-source code or applets. In consequence, a huge number of code and benchmark data sets can be found all over the web.

Here, we list a few sites where program packages and data sets can be found, whereby the list is necessarily far from being complete, even far from being representative:

- The by far mostly used data repository with more than 200 data sets of different type is the UCI Machine learning repository at <http://archive.ics.uci.edu/ml/>.
- Large data sets are regularly published in the frame of the KDD contest pagination <http://www.kdd.org/kddcup/index.php>.
- A series of challenges with publicly available data is organized in the frame of the PASCAL network of excellence <http://www.pascal-network.org/?q=node/15>.
- Very large datasets of images with different preprocessing as well as a semantic annotation are prepared in the project ImageNet <http://www.image-net.org/>.
- A popular open source java package including machine learning tools and preprocessing is offered by Weka <http://www.cs.waikato.ac.nz/ml/weka/>.
- A (probably a bit old but still one of the largest open source software packages) simulator for neural networks is SNNS <http://www.ra.cs.uni-tuebingen.de/SNNS/>.
- Another relatively novel full-featured neural network simulator for all usual platforms is offered by Emergent [http://grey.colorado.edu/emergent/index.php/Main\\_Page](http://grey.colorado.edu/emergent/index.php/Main_Page).

---

## 1.15.9 Conclusions and future trends

Basically, a variety of different models and training algorithms has been introduced, including the popular standard algorithms such as perceptron training, back-propagation, recurrent networks, support vector machine, learning vector quantization, self organizing maps, and neural gas. Obviously, this is not a homogeneous set of models, rather different models with different motivations, biological relevance, heuristic motivation, or mathematical formulation exist. Thereby, the concrete algorithms are often surprisingly simple, formulated in just a few lines of code. However, both, mathematics behind these models as well as further algorithmic development are non trivial and subject of ongoing research.

---

## Glossary

Adatron	an online training algorithm which finds a linear separation of data with maximum margin
Back-propagation	efficient way to compute the gradients in a feed-forward neural network, standard training algorithm for such networks
Boltzmann machine	associative memory with probabilistic neurons, can involve hidden neurons
Cascade correlation	training algorithm which simultaneously determines the weights and the architecture of a feed-forward network
Chomsky hierarchy	hierarchy of formal languages based on the complexity of the rules of the underlying grammar, includes the standard model of what can be addressed by digital computation as the highest level
Convex optimization	optimization of a convex function in a convex set, under this condition any local optimum is a global one, such that efficient methods can be designed
Cross-validation	a technique for estimating the performance of a predictive model by means of a repeated split of the given data
Deep learning	learning multiple levels of representation autonomously, e.g., using deep convolutional networks or deep auto-encoders
Extreme learning machine	a feed-forward architecture where the first layer is random and only the output is trained
Feed-forward network	neural network with a dynamics determined by an acyclic graph for regression or classification tasks
Gradient descent	optimization method for a differentiable function by means of iterative adaptation steps into the direction of the steepest descent

Hebbian learning	learning paradigm which emphasizes connections in between neurons if they are simultaneously active
Hodgin-Huxley model	differential equations which describe the action potential of a biological neuron
Hopfield network	fully recurrent neural network used as associative memory
Independent component analysis	technique to decompose a linear mixture of signals in independent sources
Kernel	implicit nonlinear embedding function of data in a high dimensional feature space
Learning vector quantization	prototype-based classification algorithm trained by means of Hebbian learning
Long-short term memory	recurrent network where error back-propagation is restricted to linear units to avoid numerical difficulties
Magnification factor	quantity which characterizes the distribution of prototypes as compared to the underlying data in topographic maps
Mean squared error	common cost function to evaluate the quality of a neural network on a given data set
Neural gas	topographic mapping with data driven neighborhood cooperation
NP-hard problem	problem which is at least as hard as the famous satisfiability (SAT) problem, no efficient solution is known today
Oja learning rule	unsupervised learning rule for a single neuron to perform online principal component analysis
Perceptron	a single neuron used as a linear classifier
Principal component analysis	procedure to transform data corresponding to the directions of highest statistical variation
Real time recurrent learning	training algorithm to train recurrent networks in an online scenario
Recurrent neural network	neural network with cyclic connections to simulate associative processing or dynamical systems
Relational clustering	clustering data which are characterized by pairwise dissimilarities (relations) only
Relevance learning	automatic adaptation of the relevance of input dimensions according to a given task
Reservoir computing	recurrent neural networks based on a randomly connected rich recurrent reservoir and a trainable linear readout
Self-organizing map	unsupervised learning method for neurons arranged in lattice structure to arrive at a topographic mapping of given stimuli
Slow feature analysis	technique to extract slowly changing entities from dynamic stimuli
Support vector machine	learning principle which separates data linearly in a nonlinear high-dimensional feature space by optimizing the margin
Synapse	junctions between biological neurons

Topographic mapping	arrangement of neurons such that they represent stimuli according to their underlying topology
Vapnik-Chervonenkis dimension	combinatorial measure to judge the capacity of a function class

### Relevant Theory: Machine Learning

See this Volume, [Chapter 14](#) Learning Theory

See this Volume, [Chapter 16](#) Kernel Methods and SVM

See this volume, [Chapter 17](#) Online Learning in Reproducing Kernel Hilbert Spaces

See this Volume, [Chapter 25](#) A Tutorial on Model Selection

## References

- [1] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- [2] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, 2001.
- [3] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Series in Statistics, second ed., Springer, New York, NY, 2009.
- [4] S. Haykin, Neural Networks: A Comprehensive Foundation, Pearson, 1998.
- [5] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison Wesley, 1991.
- [6] S. Theodoridis, K. Koutroumbas, Pattern Recognition, third ed., Academic Press, Inc., Orlando, FL, USA, 2006.
- [7] W. McCulloch, W. Pitts, A logical calculus of ideas immanent in nervous activities, Bull. Math. Biophys. 5 (1943) 115–133.
- [8] D. Hebb, The Organization of Behavior, Wiley, 1949.
- [9] F. Rosenblatt, Principles of Neurodynamics, Spartan, 1962.
- [10] B. Widrow, M. Hoff, Adaptive switching circuits, in: 1960 IRE WESCON Convention Record, 1960, pp. 96–104.
- [11] M. Minsky, S. Papert, Perceptrons, MIT Press, 1969.
- [12] T. Kohonen, Self Organizing Maps, Springer, 1995.
- [13] C. von der Malsburg, Self-organization of orientation sensitive cells in the striate cortex, Kybernetik 14 (1973) 85–100.
- [14] D. Rumelhart, G. Hinton, R. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
- [15] P. Werbos, The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting, Wiley, 1994.
- [16] G.E. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
- [17] H. Jaeger, The echo state approach to analysing and training recurrent neural networks, Technical Report, GMD Report 148, German National Research Center for Information Technology, 2001.
- [18] W. Maass, Motivation, Theory, and Applications of Liquid State Machines, Imperial College Press, 2011.
- [19] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (1952) 500–544.
- [20] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

- [21] B. DasGupta, B. Hammer, On approximate learning by multi-layered feedforward circuits, *Theor. Comput. Sci.* 348 (1) (2005) 95–127.
- [22] J. Sima, Back propagation is not efficient, *Neural Networks* 9 (6) (1996) 1017–1023.
- [23] M. Riedmiller, H. Braun, Rprop—description and implementation details, Technical Report, Universitat Karlsruhe, 1994.
- [24] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1–3) (2006) 489–501.
- [25] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.
- [26] M. Karpinski, A. Macintyre, Polynomial bounds for VC dimension of sigmoidal and general pfaffian neural networks, *J. Comput. Syst. Sci.* 54 (1) (1997) 169–176.
- [27] D.J. MacKay, Bayesian interpolation, *Neural Comput.* 4 (1991) 415–447.
- [28] R.M. Neal, *Bayesian Learning for Neural Networks*, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 1996.
- [29] E.D. Sontag, Neural nets as systems models and controllers, in: 7th Yale Workshop on Adaptive and Learning Syst., 1992, pp. 73–79.
- [30] X. Wang, Period-doublings to chaos in a simple neural network: an analytic proof, *Complex Syst.* 5 (1991) 425–442.
- [31] B.A. Pearlmutter, Gradient calculations for dynamic recurrent neural networks: a survey, *IEEE Trans. Neural Networks* 6 (5) (1995) 1212–1228.
- [32] J.A.K. Suykens, B.D. Moor, J. Vandewalle, Robust local stability of multilayer recurrent neural networks, *IEEE TNN* 11 (1) (2000) 222–229.
- [33] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE TNN* 5 (2) (1994) 157–166.
- [34] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Comput.* 9 (8) (1997) 1735–1780.
- [35] T. Natschläger, W. Maass, H. Markram, The liquid computer: a novel strategy for real-time computing on time series, *TELEMATIK* 8 (1) (2002) 39–43.
- [36] P. Tino, G. Dorffner, Predicting the future of discrete sequences from fractal representations of the past, *Mach. Learn.* 45 (2) (2001) 187–217.
- [37] B. Hammer, P. Tiño, Recurrent neural networks with small weights implement definite memory machines, *Neural Comput.* 15 (8) (2003) 1897–1929.
- [38] B. Hammer, Generalization ability of folding networks, *IEEE Trans. Knowl. Data Eng.* 13 (2) (2001) 196–206.
- [39] J.L. Elman, Finding structure in time, *Cognitive Sci.* 14 (2) (1990) 179–211.
- [40] S. Frank, Learn more by training less: systematicity in sentence processing by recurrent networks, *Connect. Sci.* 18 (2006) 287–302.
- [41] M.H. Tong, A.D. Bickett, E.M. Christiansen, G.W. Cottrell, 2007 special issue: learning grammatical structure with echo state networks, *Neural Networks* 20 (3) (2007) 424–432.
- [42] F. Van der Velde, M. de Kamps, Neural blackboard architectures of combinatorial structures in cognition, *Behav. Brain Sci.* 29 (2006) 37–70.
- [43] R.C. Carrasco, M.L. Forcada, M.A. Valdes-Munoz, R.P. Neco, Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units, *Neural Comput.* 12 (2000) 2129–2174.
- [44] C. Omlin, C. Giles, Rule revision with recurrent neural network, *IEEE TKDE* 8 (1) (1996) 183–188.
- [45] P. Rodriguez, J. Wiles, J.L. Elman, A recurrent neural network that learns to count, *Connect. Sci.* 11 (1) (1999) 4–40.
- [46] H.T. Siegelmann, E.D. Sontag, Analog computation, neural networks, and circuits, *Theoretical Comput. Sci.* 131 (1994) 331–360.
- [47] J. Kilian, H.T. Siegelmann, The dynamic universality of sigmoidal neural networks, *Inf. Comp.* 128 (1996) 48–56.

- [48] B. Hammer, On the approximation capability of recurrent neural networks, *Neurocomputing* 31 (1–4) (2000) 107–123.
- [49] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, G. Soda, Exploiting the past and the future in protein secondary structure prediction, *Bioinformatics* 15 (11) (1999).
- [50] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, G. Soda, Bidirectional dynamics for protein secondary structure prediction, *Lect. Notes Comput. Sci.* 1828 (2001) 80.
- [51] P. Baldi, G. Pollastri, The principled design of large-scale recursive neural network architectures—DAG-RNNs and the protein structure prediction problem, *J. Mach. Learn. Res.* 4 (2003) 575–602.
- [52] A.M. Bianucci, A. Micheli, A. Sperduti, A. Starita, Application of cascade correlation networks for structures to chemistry, *J. Appl. Int.* 12 (2000) 117–146.
- [53] C.L. Giles, D. Chen, G.-Z. Sun, H.-H. Chen, Y.-C. Lee, M.W. Goudreau, Constructive learning of recurrent neural networks: limitations of recurrent cascade correlation and a simple solution, *IEEE Trans. Neural Networks* 6 (4) (1995) 829–836.
- [54] B. Hammer, A. Micheli, A. Sperduti, Universal approximation capability of cascade correlation for structures, *Neural Comput.* 17 (2005) 1109–1159.
- [55] P. Frasconi, M. Gori, A. Sperduti, A general framework for adaptive processing of data structures *IEEE TNN* 9 (5) (1997) 768–786.
- [56] B. Hammer, Learning with recurrent neural networks, *Lect. Notes Contr. Info. Sci.*, vol. 254, Springer, 2000.
- [57] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Trans. Neural Networks* 20 (1) (2009) 61–80.
- [58] T. Friess, N. Cristianini, C. Campbell, The kernel adatron algorithm: a fast and simple learning procedure for support vector machine, 1998.
- [59] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, MIT Press, Cambridge, MA, USA, 1999, pp. 185–208.
- [60] I.W. Tsang, J.T. Kwok, P.-M. Cheung, Core vector machines: fast svm training on very large data sets, *J. Mach. Learn. Res.* 6 (2005) 363–392.
- [61] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT, 2002.
- [62] B. Hammer, K. Gersmann, A note on the universal approximation capability of support vector machines, *Neural Process. Lett.* 17 (1) (2003) 43–53.
- [63] I. Steinwart, On the influence of the kernel on the consistency of support vector machines, *J. Mach. Learn. Res.* 2 (2002) 67–93.
- [64] B. Frénay, M. Verleysen, Parameter-insensitive kernel in extreme learning for non-linear support vector regression, *Neurocomputing* 74 (16) (2011) 2526–2531.
- [65] D. Haussler, Convolution kernels on discrete structures, Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California at Santa Cruz, 1999.
- [66] C. Watkins, Dynamic alignment kernels, in: A.J. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans (Eds.), *Advances in Large Margin Classifiers*, MIT Press, 1999 (Chapter 3).
- [67] C. Leslie, E. Eskin, W.S. Noble, The spectrum kernel: a string kernel for svm protein classification, in: *Pac. Symp. Biocomput.*, 2002, pp. 564–575.
- [68] H. Lodhi, J. Shawe-Taylor, N. Cristianini, C.J.C.H. Watkins, Text classification using string kernels, in: *NIPS*, 2000, pp. 563–569.
- [69] T. Kuboyama, H. Kashima, K.F. Aoki-Kinoshita, K. Hirata, H. Yasuda, A spectrum tree kernel, *J. Jpn. Soc. Artif. Int.* 22 (2) (2007).
- [70] S.V.N. Vishwanathan, A.J. Smola, Fast kernels on strings and trees, in: *NIPS* 15, 2002.
- [71] R. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in: *Proceedings of the 19th International Conference on Machine Learning ICML*, 2002.

- [72] T. Jaakkola, M. Diekhaus, D. Haussler, Using the Fisher kernel method to detect remote protein homologies, in: 7th Intell. Syst. Mol. Biol., 1999, 149–158.
- [73] M. Tipping, The relevance vector machine, in: Advances in Neural Information Processing Systems, Morgan Kaufmann, San Mateo, CA, 2000.
- [74] M. Biehl, A. Ghosh, B. Hammer, Dynamics and generalization ability of LVQ algorithms, *J. Mach. Learn. Res.* 8 (2007) 323–360.
- [75] S. Sato, K. Yamada, Generalized learning vector quantization, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), *Adv. in Neural Information Processing Systems*, vol. 7, 1995, pp. 423–429.
- [76] B. Hammer, T. Villmann, Generalized relevance learning vector quantization, *Neural Networks* 15 (8–9) (2002) 1059–1068.
- [77] B. Hammer, M. Strickert, T. Villmann, On the generalization ability of GRLVQ networks, *Neural Process. Lett.* 21 (2) (2005) 109–120.
- [78] B. Hammer, M. Strickert, T. Villmann, Supervised neural gas with general similarity measure, *Neural Process. Lett.* 21 (1) (2005) 21–44.
- [79] M. Strickert, U. Seiffert, N. Sreenivasulu, W. Weschke, T. Villmann, B. Hammer, Generalized relevance LVQ (GRLVQ) with correlation measures for gene expression analysis, *Neurocomputing* 69 (2006) 651–659, ISSN: 0925-2312.
- [80] P. Schneider, M. Biehl, B. Hammer, Adaptive relevance matrices in learning vector quantization, *Neural Comput.* 21 (12) (2009) 3532–3561.
- [81] A. Hyvärinen, J. Karhunen, E. Oja, *Independent Component Analysis*, Wiley, 2001.
- [82] L. Wiskott, T. Sejnowski, Slow feature analysis: unsupervised learning of invariances, *Neural Comput.* 14 (4) (2002) 715–770.
- [83] T. Villmann, R. Der, T.M. Martinetz, Topology preservation in self-organizing feature maps: exact definition and measurement, *IEEE Trans. Neural Networks* 8 (2) (1997) 256–266.
- [84] M. Cottrell, J.C. Fort, G. Pagès, Two or three things that we know about the Kohonen algorithm, in: M. Verleysen (Ed.), *Proc. ESANN'94*, European Symp. on Artificial Neural Networks, D Facto Conference Services, Brussels, Belgium, 1994, pp. 235–244.
- [85] J. Fort, SOM's mathematics, *Neural Networks* 19 (6) (2006) 812–816.
- [86] T. Heskes, Energy functions for self-organizing maps, in: E. Oja, S. Kaski (Eds.), *Kohonen Maps*, Elsevier, Amsterdam, 1999, pp. 303–315.
- [87] M. Cottrell, B. Hammer, A. Hasenfuss, T. Villmann, Batch and median neural gas, *Neural Networks* 19 (2006) 762–771.
- [88] T. Villmann, J.-C. Claussen, Magnification control in self-organizing maps and neural gas, *Neural Comput.* 18 (2005) 446–469.
- [89] D. de Sieno, Adding consciousness to competitive learning, in: Proc. 2nd IEEE Int. Conf. Neural Networks, vol. 1, 1988, pp. 117–124.
- [90] T. Martinetz, K. Schulten, Topology representing networks, *Neural Networks* 7 (3) (1994) 507–522.
- [91] T.M. Martinetz, K.J. Schulten, A neural-gas network learns topologies, in: T. Kohonen, K.M.O. Simula, J. Kangas (Eds.), *Artificial Neural Networks*, North-Holland, 1991, pp. 397–402.
- [92] B. Hammer, A. Hasenfuss, T. Villmann, Magnification control for batch neural gas, *Neurocomputing* 70 (7–9) (2007) 1225–1234.
- [93] C.M. Bishop, C.K.I. Williams, Gtm: the generative topographic mapping, *Neural Comput.* 10 (1998) 215–234.
- [94] M.M.V. Hulle, Entropy-based kernel mixture modeling for topographic map formation, *IEEE Trans. Neural Networks* 15 (4) (2004) 850–858.
- [95] C.M. Bishop, Gtm through time, in: IEE Fifth International Conference on Artificial Neural Networks, 1997, pp. 111–116.

- [96] B. Hammer, A. Micheli, A. Sperduti, M. Strickert, Recursive self-organizing network models, *Neural Networks* 17 (8–9) (2004) 1061–1085.
- [97] A.K. Qin, P.N. Suganthan, Kernel neural gas algorithms with application to cluster analysis, in: ICPR, vol. 4, 2004, pp. 617–620.
- [98] H. Yin, On the equivalence between kernel self-organising maps and self-organising mixture density networks, *Neural Networks* 19 (6–7) (2006) 780–784.
- [99] B. Hammer, A. Hasenfuss, Topographic mapping of large dissimilarity data sets, *Neural Comput.* 22 (9) (2010) 2229–2284.

# Kernel Methods and Support Vector Machines

# 16

John Shawe-Taylor<sup>\*</sup> and Shiliang Sun<sup>†</sup>

<sup>\*</sup>Department of Computer Science, University College London, Gower Street, London, United Kingdom

<sup>†</sup>Department of Computer Science and Technology, East China Normal University, 500 Dongchuan Road, Shanghai, China

---

## Nomenclature

$\mathbf{X}$	The data matrix with each row as an observation
$\kappa(\mathbf{x}, \mathbf{z})$	The kernel function with input vectors $\mathbf{x}$ and $\mathbf{z}$
$\langle \mathbf{x}, \mathbf{z} \rangle$	The inner product between two vectors $\mathbf{x}$ and $\mathbf{z}$
$\phi(\mathbf{x})$	The mapping of $\mathbf{x}$ to the feature space $F$
$\mathbf{I}_n$	The $n \times n$ identity matrix
$K$	The kernel matrix with entry $K_{ij}$ being the kernel function value for the $i$ th and $j$ th inputs
$\ \mathbf{w}\ $	The Euclidean norm of the vector $\mathbf{w}$
$\text{trace}(K)$	The trace of the matrix $K$
$\text{cov}(x, u)$	The covariance between two random scalar variable $x$ and $u$
$\text{var}(x)$	The variance of the random scalar variable $x$
$\mathbf{x} \succeq (\preceq) \mathbf{z}$	The vector $\mathbf{x}$ is larger (less) than the vector $\mathbf{z}$ elementwise
$K \succeq 0$	The matrix $K$ is positive semidefinite

---

### 1.16.1 Introduction

Data often possess some intrinsic regularities which, if revealed, can facilitate people to understand data themselves or make predictions about new data from the same source. These regularities are called patterns, and pattern analysis, which has been studied broadly such as in statistics, artificial intelligence and signal processing, deals with the automatic detection of patterns in data.

The development of pattern analysis algorithms can be summarized with three important stages [1]. In the 1950s and 1960s, efficient algorithms such as the perceptron [2] were used. They are well understood and effective for detecting linear patterns, though were shown to be limited in complexity. In the 1980s, with the introduction of both the backpropagation algorithm for multi-layer networks

[3] and decision trees [4,5], pattern analysis underwent a nonlinear revolution. These methods made a high impact to efficiently and reliably detect nonlinear patterns, though they are largely heuristical with limited statistical analysis and often get trapped with local minima. In the 1990s, the emerging of kernel methods [1,6] for which support vector machines (SVMs) [7,8] are the earliest and foremost influential finally enabled people to deal with nonlinear patterns in the input space via linear patterns in high dimensional spaces. This third generation of pattern analysis algorithms are well-founded just like their linear counterparts, but wipe off the drawbacks of local minima and limited statistical analysis which are typical for multi-layer neural networks and decision trees. Since the 1990s, the algorithms and application scopes of kernel methods have been extended rapidly, from classification to regression, to clustering and many other machine learning tasks.

The approach of kernel methods has four key aspects: (i) Data are embedded into a Euclidean feature space; (ii) Linear relations are sought in the feature space; (iii) Algorithms are implemented so that only inner products between vectors in the feature space are required; (iv) The products can be directly computed from the original data by an efficient “short-cut” known as a kernel function (or kernel for short). This is also known as the kernel trick. The idea of using kernel functions as inner products in a feature space is not new. It was introduced into machine learning in 1964 with the method of potential functions [9] and this work is mentioned in a footnote of Duda and Hart’s pattern classification book [10]. Through this route, the authors of [7] noticed this idea, combined it with large margin hyperplanes in the later SVMs and thus introduced the notion of kernels into the mainstream of the machine learning literature.

Although basic kernel methods are rather mature techniques, research combining them with other techniques is still going on, e.g., kernels have been successfully applied to multi-view learning, semi-supervised learning and multitask learning problems [11–16]. This forms a continual impetus along the line of research on kernel-based learning methods. More importantly, recent work on multiple kernel learning [17] has promoted the study of kernel methods to a new level. This article reviews both classical and some recent research developments on kernel methods, with emphases on the “plug-and-play” flavor of kernel methods.

The rest of this article is organized as follows. Section 1.16.2 introduces the kernel trick and properties and types of kernels, which constitute the foundations of kernel methods. In addition to the kernel ridge regression method presented in Section 1.16.2, Section 1.16.3 reviews some fundamental kernel methods including kernel principal component analysis, kernel canonical correlation analysis, kernel Fisher discriminant analysis, support vector machines, and Gaussian processes. Section 1.16.4 discusses the computational issues of kernel methods and algorithms towards their efficient implementations. Section 1.16.5 briefly surveys the recent developments on multiple kernel learning. Section 1.16.6 presents some practical applications of kernel methods and SVMs. Finally, open issues and problems are discussed in Section 1.16.7 after a brief concluding summary of the article.

## 1.16.2 Foundations of kernel methods

In this section, we first illustrate key concepts for kernel methods from kernel ridge regression, and then discuss properties of valid kernels. Finally, kernel design strategies are introduced.

### 1.16.2.1 The kernel trick: ridge regression as an example

Consider the problem of finding a homogeneous real-valued linear function

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{x}^\top \mathbf{w} = \sum_{i=1}^n w_i x_i, \quad (16.1)$$

that best interpolates a given training set  $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  of points  $\mathbf{x}_i \in \mathbb{R}^n$  with corresponding labels  $y_i \in \mathbb{R}$ . A commonly chosen measure of the discrepancy between a function output and the real observation is

$$f_g((\mathbf{x}, y)) = (g(\mathbf{x}) - y)^2. \quad (16.2)$$

Suppose the  $m$  inputs of  $S$  are stored in the matrix  $\mathbf{X}$  as row vectors, and the corresponding outputs constitute vector  $\mathbf{y}$  with  $\mathbf{y} = [y_1, \dots, y_m]^\top$ . Hence we can write  $\xi = \mathbf{y} - \mathbf{X}\mathbf{w}$  for the vector of differences between  $g(\mathbf{x}_i)$  and  $y_i$ . Ridge regression corresponds to solving the following optimization with a simple norm regularizer

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \|\xi\|^2, \quad (16.3)$$

where  $\lambda > 0$  defines the relative tradeoff between the norm and loss. Setting the derivative of  $\mathcal{L}_\lambda(\mathbf{w}, S)$  with respect to the parameter vector  $\mathbf{w}$  equal to 0 gives

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n)\mathbf{w} = \mathbf{X}^\top \mathbf{y}, \quad (16.4)$$

where  $\mathbf{I}_n$  is the  $n \times n$  identity matrix. Thus we get the primal solution (referring to the explicit representation) for the weight vector

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_n)^{-1} \mathbf{X}^\top \mathbf{y}, \quad (16.5)$$

from which the resulting prediction function  $g(\mathbf{x})$  can be readily given.

Alternatively, from (16.4) we get

$$\mathbf{w} = \mathbf{X}^\top \frac{1}{\lambda} (\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}^\top \boldsymbol{\alpha} = \sum_{i=1}^m \alpha_i \mathbf{x}_i, \quad (16.6)$$

where parameters  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_m]^\top \triangleq \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$  are known as the dual variables. Substituting  $\mathbf{w} = \mathbf{X}^\top \boldsymbol{\alpha}$  into  $\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$ , we obtain

$$\boldsymbol{\alpha} = (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}_m)^{-1} \mathbf{y}, \quad (16.7)$$

which is called the dual solution. The dual solution expresses the weight vector  $\mathbf{w}$  as a linear combination of the training examples. Denote the term  $\mathbf{X}\mathbf{X}^\top$  by  $\mathbf{K}$ . It follows that  $\mathbf{K}_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ . Now the resulting prediction function is formulated as

$$g(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} = \mathbf{x}^\top \mathbf{X}^\top \boldsymbol{\alpha} = \left\langle \mathbf{x}, \sum_{i=1}^m \alpha_i \mathbf{x}_i \right\rangle = \sum_{i=1}^m \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle. \quad (16.8)$$

There are two ingredients embedded in the dual form of ridge regression: computing vector  $\boldsymbol{\alpha}$  and evaluation of the prediction function. Both operations only involve inner products between data inputs.

Since the computation only involves inner products, we can substitute for all occurrences of  $\langle \cdot, \cdot \rangle$  a kernel function  $\kappa$  that computes  $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$  and we obtain an algorithm for ridge regression in the feature space  $F$  defined by the mapping  $\phi : \mathcal{X} \mapsto \phi(\mathbf{x}) \in F$ . This is an instantiation of the kernel trick for ridge regression and results in the kernel ridge regression algorithm. Through kernel ridge regression we can perform linear regression in very high-dimensional spaces efficiently, which is equivalent to performing non-linear regression in the original input space.

### 1.16.2.2 Properties of kernels

**Definition 1 (Kernel function).** A kernel is a function  $\kappa$  that for all  $\mathbf{x}, \mathbf{z}$  from a nonempty set  $\mathcal{X}$  (which need not be a vector space) satisfies

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \quad (16.9)$$

where  $\phi$  is a mapping from the set  $\mathcal{X}$  to a Hilbert space  $F$  that is usually called the feature space

$$\phi : \mathcal{X} \mapsto \phi(\mathbf{x}) \in F. \quad (16.10)$$

To verify whether a function is a valid kernel, one approach is to construct a feature space for which the function value for two inputs corresponds to first performing an explicit feature mapping and then computing the inner product between their images. An alternative approach, which is more widely used, is to investigate the finitely positive semidefinite property [1, 6, 18–20].

**Definition 2 (Finitely positive semidefinite function).** A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  satisfies the finitely positive semidefinite property if it is a *symmetric* function for which the kernel matrices  $K$  with  $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$  formed by restriction to any finite subset of  $\mathcal{X}$  are positive semidefinite.

The feasibility of the above property for characterizing kernels is justified by the following theorem [1, 6, 21].

**Theorem 3 (Characterization of kernels).** A function  $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  which is either continuous or has a countable domain, can be decomposed as an inner product in a Hilbert space  $F$  by a feature map  $\phi$  applied to both its arguments

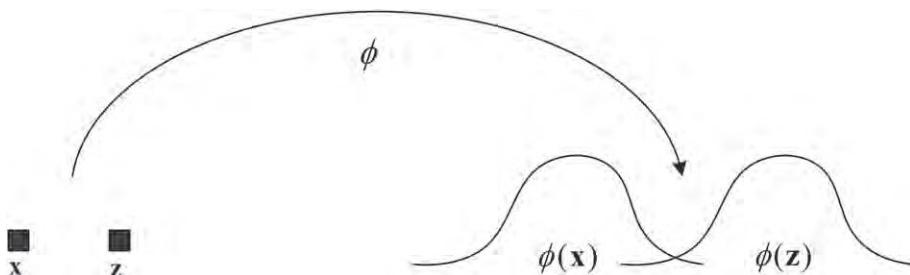
$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \quad (16.11)$$

if and only if it satisfies the finitely positive semidefinite property.

A Hilbert space  $F$  is defined as an inner product space that is complete where completeness means that every Cauchy sequence of elements of  $F$  converges to an element in this space. If the separability property is further added to the definition of a Hilbert space, where a space is separable if there is a countable set of elements from this space such that the distance between each element of this space and some element of this countable set is less than any predefined threshold, the existence of “kernels are continuous or the domain is countable” in Theorem 3 is then necessary.

The Hilbert space constructed in proving Theorem 3 is called the reproducing kernel Hilbert space (RKHS) because the following reproducing property of the kernel resulting from the defined inner product holds

$$\langle f_F(\cdot), \kappa(\mathbf{x}, \cdot) \rangle = f_F(\mathbf{x}), \quad (16.12)$$

**FIGURE 16.1**

One instantiation of the feature mapping using a Gaussian kernel.

where  $f_F$  is a function of the function space  $F$ , and function  $\kappa(\mathbf{x}, \cdot)$  is the mapping  $\phi(\mathbf{x})$  which actually represents the similarity of  $\mathbf{x}$  to all other points in  $\mathcal{X}$ , as shown in Figure 16.1 [6].

By construction,  $f_F(\cdot)$  takes the form of an arbitrarily-weighted linear combination of countable images of the original inputs. For any two such functions

$$f_{F1}(\cdot) = \sum_{i=1}^{\ell_1} \alpha_i \kappa(\mathbf{x}_i, \cdot), \quad f_{F2}(\cdot) = \sum_{j=1}^{\ell_2} \beta_j \kappa(\mathbf{x}'_j, \cdot) \quad (16.13)$$

where  $\ell_1, \ell_2 \in \mathbb{N}$ ,  $\alpha_i, \beta_j \in \mathbb{R}$  and  $\mathbf{x}_i, \mathbf{x}'_j \in \mathcal{X}$ , the dot product is defined as

$$\langle f_{F1}(\cdot), f_{F2}(\cdot) \rangle \triangleq \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \alpha_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}'_j). \quad (16.14)$$

The inner product space or pre-Hilbert space formed by  $f_F(\cdot)$  is then completed to form the Hilbert space  $F$  where the mathematical trick “completion” refers to adding all limit points of Cauchy sequences to the space [6].

It should be noted that there are different approaches to constructing feature spaces for any given kernel. Besides the above construction, the Mercer kernel map [22], though not mentioned much here, is also widely applicable, especially in the SVM literature. The feature spaces constructed in different ways can even have different dimensions. However, since we are only interested in dot products, these spaces can be regarded as identical.

For some kernels, the feature map and feature space can be explicitly built with a simple form. For instance, consider the homogeneous quadratic kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2, \quad (16.15)$$

which can be reformulated as

$$\kappa(\mathbf{x}, \mathbf{z}) = (\mathbf{x}' \mathbf{z})^2 = \mathbf{z}' (\mathbf{x} \mathbf{x}') \mathbf{z} = \langle \text{vec}(\mathbf{z} \mathbf{z}'), \text{vec}(\mathbf{x} \mathbf{x}') \rangle, \quad (16.16)$$

where  $\text{vec}(A)$  stacks the column of matrix  $A$  on top of each other in the manner that the first column situates at the top. The feature map corresponding to  $\kappa$  would be  $\phi(\mathbf{x}) = \text{vec}(\mathbf{x} \mathbf{x}')$ . The feature space can be the Euclidean space with dimensionality being the total number of entries of  $\text{vec}(\mathbf{x} \mathbf{x}')$ .

### 1.16.2.3 Types of kernels

The use of kernels provides a powerful and principled approach to modeling nonlinear patterns through linear patterns in a feature space. Another benefit is that the design of kernels and linear methods can be decoupled, which greatly facilitates the modularity of machine learning methods.

Representative kernels include the linear kernel  $\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$ , inhomogeneous polynomial kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + R)^d \quad (16.17)$$

where  $d$  is the degree of the polynomial and parameter  $R \in \mathbb{R}$ , and the Gaussian radial basis function (RBF) kernel (Gaussian kernel for short) with parameter  $\sigma > 0$

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right). \quad (16.18)$$

The polynomial kernel (16.17) can be expanded by the binomial theorem as

$$(\langle \mathbf{x}, \mathbf{z} \rangle + R)^d = \sum_{s=0}^d \binom{d}{s} R^{d-s} \langle \mathbf{x}, \mathbf{z} \rangle^s. \quad (16.19)$$

Hence, the features for each component in the sum together form the features of the kernel. In other words, we have a reweighting of the features of the homogeneous polynomial kernel

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^s, \quad s = 0, \dots, d, \quad (16.20)$$

where one construction of the feature map corresponding to kernel (16.20) is using a vector with entries being all ordered monomials (e.g.,  $x_1x_2$  and  $x_2x_1$  are treated as separate features) of degree  $s$ , that is, each entry is an instantiation of product  $x_{j_1} \dots x_{j_s}$  with  $j_1, \dots, j_s \in \{1, \dots, n\}$  [6]. The parameter  $R$  allows the control of the relative weightings of the monomials with different degrees. The weight formulation  $\binom{d}{s} R^{d-s}$  indicates that increasing  $R$  decreases the relative weighting of higher order monomials [1].

For the Gaussian kernel (16.18) the images of all points have norm 1 in the feature space as a result of  $\kappa(\mathbf{x}, \mathbf{x}) = 1$ . It can be obtained by normalizing  $\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)$

$$\begin{aligned} \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) &= \exp\left(\frac{\langle \mathbf{x}, \mathbf{z} \rangle}{\sigma^2} - \frac{\langle \mathbf{x}, \mathbf{x} \rangle}{2\sigma^2} - \frac{\langle \mathbf{z}, \mathbf{z} \rangle}{2\sigma^2}\right) \\ &= \frac{\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)}{\sqrt{\exp(\|\mathbf{x}\|^2 / \sigma^2) \exp(\|\mathbf{z}\|^2 / \sigma^2)}}. \end{aligned} \quad (16.21)$$

Because an exponential function can be arbitrarily closely approximated by polynomials with positive coefficients

$$\exp(x) = \sum_{i=0}^{\infty} \frac{1}{i!} x^i, \quad (16.22)$$

the function  $\exp(\langle \mathbf{x}, \mathbf{z} \rangle / \sigma^2)$  is arguably a kernel. Therefore, the Gaussian kernel (16.18) is a polynomial kernel of infinite degree, and its features can be all ordered monomials of input features with no restriction placed on the degrees. However, with increasing degree the weighting of individual monomials falls off as  $i!$  [1].

One appeal of using kernel methods is that kernels are not restricted to vectorial data, making it possible to apply the techniques to diverse types of objects. Not surprisingly, kernels can be designed

for sets, strings, text documents, graphs and graph-nodes [1]. For these kernels, we would not elaborate here. However, an effective design of kernels has to be embedded with some prior knowledge on how to characterize similarity between data.

We now focus on two types of kernels induced by probabilistic models, marginalization kernels and Fisher kernels. These techniques are useful for combining generative and discriminative methods for machine learning. The marginalization kernels are defined as follows.

**Definition 4 (Marginalization kernels).** Given a set of data models  $M$  and a prior distribution  $P_M$  on  $M$ , the probability that an example pair  $\mathbf{x}$  and  $\mathbf{z}$  is generated together can be computed as

$$P_M(\mathbf{x}, \mathbf{z}) = \sum_{m \in M} P(\mathbf{x}|m)P(\mathbf{z}|m)P_M(m). \quad (16.23)$$

If we consider the mapping function

$$\phi : \mathbf{x} \mapsto (P(\mathbf{x}|m))_{m \in M} \in F \quad (16.24)$$

in a feature space  $F$  indexed by  $M$ ,  $P_M(\mathbf{x}, \mathbf{z})$  corresponds to the inner product

$$\langle f, g \rangle = \sum_{m \in M} f_m g_m P_M(m) \quad (16.25)$$

between  $\phi(\mathbf{x})$  and  $\phi(\mathbf{z})$ .  $P_M(\mathbf{x}, \mathbf{z})$  is referred to as the marginalization kernel for the model class  $M$ .

The above computation can be viewed as a marginalization operation for the probability distribution of triples  $(\mathbf{x}, \mathbf{z}, m)$  over  $m$  (with conditional independence of  $\mathbf{x}$  and  $\mathbf{z}$  given a specific model  $m$ ), and therefore comes the name marginalization kernels. The assumption of conditional independence is a sufficient condition for positive semi-definiteness. For an input, marginalization kernels treat the output probability given one model as a feature. Since the information from a single model is quite limited, they usually adopt multiple different models to reach a representation of the input.

Fisher kernels, defined by [23,24], are an alternative way of extracting information, usually from a single generative model, however. The single model is required to be smoothly parameterized so that derivatives of the model with respect to the parameters is computable. An intuitive interpretation of Fisher kernels is that it describes data points by the variation of some quantity (say the log of the likelihood function) caused by slight parameter perturbations.

**Definition 5 (Fisher score and Fisher information matrix).** For a given setting of the parameters  $\boldsymbol{\theta}^0$  (e.g., obtained by the maximum likelihood rule) the log-likelihood of a data point  $\mathbf{x}$  with respect to the model  $m(\boldsymbol{\theta}^0)$  is defined to be  $\log P(\mathbf{x}|\boldsymbol{\theta}^0)$ . Consider the gradient vector of the log-likelihood

$$\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}) = \left( \frac{\partial \log P(\mathbf{x}|\boldsymbol{\theta})}{\partial \theta_i} \right)_{i=1}^N, \quad (16.26)$$

where  $\boldsymbol{\theta} \in \mathbb{R}^N$ . The Fisher score of a data point  $\mathbf{x}$  with respect to the model  $m(\boldsymbol{\theta}^0)$  is  $\mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})$ . The Fisher information matrix with respect to the model  $m(\boldsymbol{\theta}^0)$  is given by

$$\mathbf{I}_{\text{Fisher}} = \mathbb{E} \left[ \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x}) \mathbf{g}(\boldsymbol{\theta}^0, \mathbf{x})^\top \right], \quad (16.27)$$

where the expectation is over the distribution of the data point  $\mathbf{x}$ .

The Fisher score embeds a data point into the feature space  $\mathbb{R}^N$ , and provides direct constructions of kernels.

**Definition 6 (Fisher kernel).** The invariant Fisher kernel with respect to the model  $m(\theta^0)$  for a given setting of the parameters  $\theta^0$  is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{g}(\theta^0, \mathbf{x})^\top \mathbf{I}_{Fisher}^{-1} \mathbf{g}(\theta^0, \mathbf{z}). \quad (16.28)$$

The practical Fisher kernel is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{g}(\theta^0, \mathbf{x})^\top \mathbf{g}(\theta^0, \mathbf{z}). \quad (16.29)$$

The invariant Fisher kernel is computationally more demanding as it requires the computation and inversion of the Fisher information matrix. It is named “invariant” because the resulting kernel would not change if we reparameterize the model with an invertible differentiable transformation  $\psi = \psi(\theta)$ . Suppose  $\tilde{\kappa}$  is the transformed kernel. It follows that

$$\mathbf{g}(\theta^0, \mathbf{x})^\top = \left( \left( \frac{\partial \log P(x|\psi)}{\partial \psi_i} \right)_{i=1}^N \right)^\top \mathbf{J}(\psi) = \mathbf{g}(\psi^0, \mathbf{x})^\top \mathbf{J}(\psi^0), \quad (16.30)$$

where matrix  $\mathbf{J}(\psi^0)$  is the Jacobian of the transformation  $\psi$  evaluated at  $\psi^0$  [1]. Now we have

$$\begin{aligned} \tilde{\kappa}(\mathbf{z}_1, \mathbf{z}_2) &= \mathbf{g}(\psi^0, \mathbf{z}_1)^\top \mathbb{E} \left[ (\mathbf{J}(\psi^0)^{-1})^\top \mathbf{g}(\theta^0, \mathbf{x}) \mathbf{g}(\theta^0, \mathbf{x})^\top \mathbf{J}(\psi^0)^{-1} \right]^{-1} \mathbf{g}(\psi^0, \mathbf{z}_2) \\ &= \mathbf{g}(\theta^0, \mathbf{z}_1)^\top \mathbb{E} \left[ \mathbf{g}(\theta^0, \mathbf{x}) \mathbf{g}(\theta^0, \mathbf{x})^\top \right]^{-1} \mathbf{g}(\theta^0, \mathbf{z}_2) \\ &= \kappa(\mathbf{z}_1, \mathbf{z}_2). \end{aligned} \quad (16.31)$$

Hence, the invariant Fisher kernel is desirable if the choice of parameterizations is somewhat arbitrary. But for this kernel there is a caveat when the natural approximation of the Fisher information matrix by its empirical estimate is used

$$\hat{\mathbf{I}}_{Fisher} = \hat{\mathbb{E}} \left[ \mathbf{g}(\theta^0, \mathbf{x}) \mathbf{g}(\theta^0, \mathbf{x})^\top \right] = \frac{1}{m} \sum_{i=1}^m \mathbf{g}(\theta^0, \mathbf{x}_i) \mathbf{g}(\theta^0, \mathbf{x}_i)^\top, \quad (16.32)$$

in which case  $\hat{\mathbf{I}}_{Fisher}$  is the empirical covariance matrix of the Fisher scores. The invariant Fisher kernel is thus equivalent to whitening the scores. The negative effect is that we may amplify noise if some parameters are not relevant for the information, and therefore the signal to noise ratio is possibly reduced. This can be regarded as the cost of the invariance.

Apart from the kernels introduced so far, more complicated kernels can be constructed with them as building blocks. The following theorem [1] lists some strategies for kernel constructions.

**Theorem 7 (Kernel constructions).** Let  $\kappa_1, \kappa_2$ , and  $\kappa_3$  be valid kernels,  $\phi$  any feature map to the domain of  $\kappa_3$ ,  $a \geq 0$ ,  $f(\cdot)$  any real-valued function, and  $\mathbf{B}$  a positive semi-definite matrix. Then the following functions are valid kernels:

- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) + \kappa_2(\mathbf{x}, \mathbf{z})$ ,
- $\kappa(\mathbf{x}, \mathbf{z}) = a\kappa_1(\mathbf{x}, \mathbf{z})$ ,

- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$ ,
- $\kappa(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$ ,
- $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$ ,
- $\kappa(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{Bz}$  (for now  $\mathbf{x}$  and  $\mathbf{z}$  are vectorial data).

### 1.16.3 Fundamental kernel methods

In this section, we introduce some fundamental kernel methods ranging from unsupervised learning to supervised learning. These methods have a large popularity either because they are among the first uses of kernels or because they address very fundamental learning problems.

#### 1.16.3.1 Kernel principal component analysis

Principal component analysis (PCA) finds a set of orthogonal directions which forms a subspace to maximize variances. In this way, data can be reconstructed with minimal quadratic error. Suppose the inputs of the data set  $S$  given in Section 1.16.2.1 is centered with mean  $\mathbf{0}$ . The direction that maximizes the variance can be found by solving the following problem

$$\begin{aligned} & \max_{\mathbf{w}} \mathbf{w}^\top \mathbf{Cw} \\ & \text{s.t. } \|\mathbf{w}\| = 1, \end{aligned} \quad (16.33)$$

where  $\mathbf{C} = \frac{1}{m} \mathbf{X}^\top \mathbf{X}$  is the covariance matrix (strictly speaking, an empirical estimate of the covariance) of the input data. The solution is given by the eigenvector of  $\mathbf{C}$  corresponding to the largest eigenvalue with the objective value being the eigenvalue. The direction of the second largest variance can be searched for in the subspace orthogonal to the direction already found. This results in the eigenvector corresponding to the second largest eigenvalue. It is readily provable that PCA projects data into the space spanned by the  $k$  largest eigenvectors of  $\mathbf{C}$  if we would like to find a  $k$ -dimensional subspace. The new coordinates by which we represent the data are known as principal components. Although centering data before performing PCA is not a must, it has the advantage of reducing the overall sum of the eigenvalues and thus removing irrelevant variance arising from data shift [1].

The kernel PCA [25] extends the linear PCA algorithm to extracting nonlinear structures in terms of kernels. Now we provide a simple derivation of the kernel PCA by exploiting the relationship between  $\mathbf{X}^\top \mathbf{X}$  and  $\mathbf{XX}^\top$  [1]. It is easy to show that these two matrices have the same rank. More interestingly, their eigen-decompositions correspond to each other. Suppose that  $\mathbf{w}, \lambda$  is an eigenvector-eigenvalue pair for  $\mathbf{X}^\top \mathbf{X}$ , then  $\mathbf{Xw}, \lambda$  is for  $\mathbf{XX}^\top$

$$(\mathbf{XX}^\top)\mathbf{Xw} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})\mathbf{w} = \lambda \mathbf{Xw}, \quad (16.34)$$

and conversely, if  $\alpha, \lambda$  is an eigenvector-eigenvalue pair for the matrix  $\mathbf{XX}^\top$ , then  $\mathbf{X}^\top \alpha, \lambda$  is for  $\mathbf{X}^\top \mathbf{X}$

$$(\mathbf{X}^\top \mathbf{X})\mathbf{X}^\top \alpha = \mathbf{X}^\top (\mathbf{XX}^\top)\alpha = \lambda \mathbf{X}^\top \alpha. \quad (16.35)$$

This gives a dual representation for the eigenvector of  $\mathbf{X}^\top \mathbf{X}$  from the eigen-decomposition of  $\mathbf{XX}^\top$ .  $\mathbf{XX}^\top$  is actually a kernel matrix if we replace each row  $\mathbf{x}_i^\top$  of  $\mathbf{X}$  by its image  $\phi(\mathbf{x}_i)^\top$  in a feature space, and  $\mathbf{X}^\top \mathbf{X}$  would be the scaled covariance matrix without centering.

Centering data in a feature space is not so simple as in the original space. Suppose that a kernel  $\kappa$  is adopted with the kernel matrix  $\mathbf{K}$  computed from the original data. Centering data in the feature space corresponds to defining a new feature map  $\hat{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \frac{1}{m} \sum_{i=1}^m \phi(\mathbf{x}_i)$ . The new kernel matrix for the centered data would be

$$\hat{\mathbf{K}} = \mathbf{K} - \frac{1}{m} \mathbf{j} \mathbf{j}^\top \mathbf{K} - \frac{1}{m} \mathbf{K} \mathbf{j} \mathbf{j}^\top + \frac{1}{m^2} (\mathbf{j}^\top \mathbf{K} \mathbf{j}) \mathbf{j} \mathbf{j}^\top, \quad (16.36)$$

where  $\mathbf{j}$  is the all 1s vector [1]. Suppose that  $\hat{\alpha}, \hat{\lambda}$  is an eigenvector-eigenvalue pair for the kernel matrix  $\hat{\mathbf{K}} = \hat{\mathbf{X}} \hat{\mathbf{X}}^\top$  where  $\|\hat{\alpha}\| = 1$  and the  $i$ th row of  $\hat{\mathbf{X}}$  is  $\hat{\phi}(\mathbf{x}_i)^\top$ . Then  $\hat{\mathbf{X}}^\top \hat{\alpha}$  is the eigenvector of the covariance matrix  $\frac{1}{m} \hat{\mathbf{X}}^\top \hat{\mathbf{X}}$  which has the same eigenvectors with  $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}$ . Usually we require that the final projection vector is normalized, that is,  $\|\hat{\mathbf{X}}^\top \hat{\alpha}\| = 1$ . Because for  $\|\hat{\alpha}\| = 1$  we have

$$\|\hat{\mathbf{X}}^\top \hat{\alpha}\|^2 = \hat{\alpha}^\top \hat{\mathbf{X}} \hat{\mathbf{X}}^\top \hat{\alpha} = \hat{\alpha}^\top \hat{\mathbf{K}} \hat{\alpha} = \hat{\lambda}, \quad (16.37)$$

to meet  $\|\hat{\mathbf{X}}^\top \hat{\alpha}\| = 1$ ,  $\hat{\alpha}$  should be further divided by  $\sqrt{\hat{\lambda}}$ . Hence, the  $k$  projection directions derived from kernel PCA should be

$$\left\{ \frac{1}{\sqrt{\hat{\lambda}_i}} \hat{\mathbf{X}}^\top \hat{\alpha}_i \right\}_{i=1}^k, \quad (16.38)$$

where  $\{\hat{\alpha}_i, \hat{\lambda}_i\}_{i=1}^k$  are the  $k$  leading eigenvector-eigenvalue pairs for the kernel matrix  $\hat{\mathbf{K}}$  and the norms of  $\{\hat{\alpha}_i\}_{i=1}^k$  are all 1. The projections of a new input  $\mathbf{x}$  would be the inner products between the above directions and  $\phi(\mathbf{x}) - \frac{1}{m} \sum_{i=1}^m \phi(\mathbf{x}_i)$ .

### 1.16.3.2 Kernel canonical correlation analysis

Canonical correlation analysis (CCA), proposed by [26], works on a paired dataset (i.e., data with two representations) to find two linear transformations each for one of the two representations such that the correlations between the transformed variables are maximized. It was later generalized to more than two sets of variables in several ways [27, 28]. Here we only focus on the situation of two sets of variables.

Suppose we have a paired dataset  $S_{\mathbf{x}, \mathbf{u}} = \{(\mathbf{x}_1, \mathbf{u}_1), \dots, (\mathbf{x}_m, \mathbf{u}_m)\}$ . For example,  $\mathbf{x}_i$  and the corresponding  $\mathbf{u}_i$  can be the representations of a same semantic content in two different languages. CCA attempts to seek the projection directions  $\mathbf{w}_x$  and  $\mathbf{w}_u$  to maximize the following empirical correlation

$$\frac{\text{cov}(\mathbf{w}_x^\top \mathbf{x}, \mathbf{w}_u^\top \mathbf{u})}{\sqrt{\text{var}(\mathbf{w}_x^\top \mathbf{x}) \text{var}(\mathbf{w}_u^\top \mathbf{u})}} = \frac{\mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u}{\sqrt{(\mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x)(\mathbf{w}_u^\top \mathbf{C}_{uu} \mathbf{w}_u)}}, \quad (16.39)$$

where covariance matrix  $\mathbf{C}_{xu}$  is defined as (definitions for  $\mathbf{C}_{xx}$  and  $\mathbf{C}_{uu}$  can be obtained analogously)

$$\mathbf{C}_{xu} = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i - \mathbf{m}_x)(\mathbf{u}_i - \mathbf{m}_u)^\top \quad (16.40)$$

with  $\mathbf{m}_x$  and  $\mathbf{m}_u$  being the means of the two representations, respectively

$$\mathbf{m}_x = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i, \quad \mathbf{m}_u = \frac{1}{m} \sum_{i=1}^m \mathbf{u}_i. \quad (16.41)$$

Because the scales of  $\mathbf{w}_x$  and  $\mathbf{w}_u$  have no effects on the value of (16.39), we can constrain each of the two terms in the denominator to take value 1. Thus we reach another widely used objective for CCA

$$\begin{aligned} \max_{\mathbf{w}_x, \mathbf{w}_u} \quad & \rho = \mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u \\ \text{s.t.} \quad & \mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x = 1, \quad \mathbf{w}_u^\top \mathbf{C}_{uu} \mathbf{w}_u = 1. \end{aligned} \quad (16.42)$$

The solution is given by first solving the generalized eigenvalue problem [1]

$$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xu} \\ \mathbf{C}_{ux} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{uu} \end{pmatrix} \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix}, \quad (16.43)$$

and then normalizing the resulting directions to comply with the constraints of (16.42). Note that the eigenvalue  $\lambda$  for a particular eigenvector  $\begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix}$  gives the corresponding correlation value

$$\rho = \mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u = \mathbf{w}_x^\top (\lambda \mathbf{C}_{xx} \mathbf{w}_x) = \lambda. \quad (16.44)$$

Consequently, all eigenvalues lie in the interval  $[-1, +1]$ . Interestingly, if  $\begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_u \end{pmatrix}$ ,  $\lambda$  is an eigenvector-eigenvalue pair, so is  $\begin{pmatrix} \mathbf{w}_x \\ -\mathbf{w}_u \end{pmatrix}$ ,  $-\lambda$ . Therefore, only half the spectrum, e.g., the positive eigenvalues, are necessary to be considered, and the corresponding eigenvectors constitute desirable projection directions (as with PCA, we often need more than one projection directions). The eigenvectors with largest eigenvalues identify the strongest correlations.

Now we give the dual form of CCA to facilitate the derivation of kernel CCA [29–31]. Assume that the dataset  $S_{x,u}$  is centered, that is, the mean value of each of the two representations is zero. We consider expressing  $\mathbf{w}_x$  and  $\mathbf{w}_u$  as linear combinations of training examples

$$\mathbf{w}_x = \mathbf{X}^\top \boldsymbol{\alpha}_x, \quad \mathbf{w}_u = \mathbf{U}^\top \boldsymbol{\alpha}_u, \quad (16.45)$$

where the rows of  $\mathbf{X}$  and  $\mathbf{U}$  are vectors  $\mathbf{x}_i^\top$  and  $\mathbf{u}_i^\top$  ( $i = 1, \dots, m$ ), respectively. Substituting (16.45) into (16.42) results in

$$\begin{aligned} \max_{\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_u} \quad & \boldsymbol{\alpha}_x^\top \mathbf{X} \mathbf{X}^\top \mathbf{U} \mathbf{U}^\top \boldsymbol{\alpha}_u \\ \text{s.t.} \quad & \boldsymbol{\alpha}_x^\top \mathbf{X} \mathbf{X}^\top \mathbf{X} \mathbf{X}^\top \boldsymbol{\alpha}_x = 1, \quad \boldsymbol{\alpha}_u^\top \mathbf{U} \mathbf{U}^\top \mathbf{U} \mathbf{U}^\top \boldsymbol{\alpha}_u = 1. \end{aligned} \quad (16.46)$$

Since the above formulation only involves inner products among training examples, we can write down the objective for kernel CCA simply as

$$\begin{aligned} \max_{\boldsymbol{\alpha}_x, \boldsymbol{\alpha}_u} \quad & \boldsymbol{\alpha}_x^\top \mathbf{K}_x \mathbf{K}_u \boldsymbol{\alpha}_u \\ \text{s.t.} \quad & \boldsymbol{\alpha}_x^\top \mathbf{K}_x^2 \boldsymbol{\alpha}_x = 1, \quad \boldsymbol{\alpha}_u^\top \mathbf{K}_u^2 \boldsymbol{\alpha}_u = 1, \end{aligned} \quad (16.47)$$

where  $\mathbf{K}_x$  and  $\mathbf{K}_u$  are the kernel matrices for the two representations, respectively (if data are not centered in feature spaces, techniques similar to centering for kernel PCA can be adopted).

It was shown that overfitting with perfect correlations which fail to distinguish spurious features from those revealing the underlying semantics can appear using the above versions of CCA and kernel CCA

[1,27]. In other words, some kind of regularization is needed to detect meaningful patterns. Statistical stability analysis shows that controlling the norms of the two projection directions is a good way for regularization [1]. Hence, we have the regularized CCA whose objective is to maximize

$$\frac{\mathbf{w}_x^\top \mathbf{C}_{xu} \mathbf{w}_u}{\sqrt{((1 - \tau_x) \mathbf{w}_x^\top \mathbf{C}_{xx} \mathbf{w}_x + \tau_x \|\mathbf{w}_x\|^2) ((1 - \tau_u) \mathbf{w}_u^\top \mathbf{C}_{uu} \mathbf{w}_u + \tau_u \|\mathbf{w}_u\|^2)}}, \quad (16.48)$$

where regularization parameters  $\tau_x$  and  $\tau_u$  vary in the interval  $[0, 1]$ . The kernel regularized CCA corresponding to (16.47) is given by optimizing

$$\begin{aligned} & \max_{\alpha_x, \alpha_u} \alpha_x^\top \mathbf{K}_x \mathbf{K}_u \alpha_u \\ & \text{s.t. } (1 - \tau_x) \alpha_x^\top \mathbf{K}_x^2 \alpha_x + \tau_x \alpha_x^\top \mathbf{K}_x \alpha_x = 1, \\ & \quad (1 - \tau_u) \alpha_u^\top \mathbf{K}_u^2 \alpha_u + \tau_u \alpha_u^\top \mathbf{K}_u \alpha_u = 1. \end{aligned} \quad (16.49)$$

### 1.16.3.3 Kernel Fisher discriminant analysis

The Fisher discriminant is a classification function

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b), \quad (16.50)$$

where the weight vector  $\mathbf{w}$  is found through a specific optimization to well separate different classes. In particular, a direction is found which maximizes the distance between projected class means and simultaneously minimizes the projected class variances. In this article, the binary case is considered. The parameter  $b$  in the Fisher discriminant is usually determined by projecting training data to  $\mathbf{w}$  and then identifying the middle point of two class means.

Suppose examples from two different classes are given by  $S_1 = \{\mathbf{x}_1^1, \dots, \mathbf{x}_{m_1}^1\}$  and  $S_2 = \{\mathbf{x}_1^2, \dots, \mathbf{x}_{m_2}^2\}$ . Fisher discriminant analysis [32,33] finds  $\mathbf{w}$  which maximizes

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}, \quad (16.51)$$

where

$$\begin{aligned} \mathbf{S}_B &= (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top, \\ \mathbf{S}_W &= \sum_{i=1,2} \sum_{\mathbf{x} \in S_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^\top \end{aligned} \quad (16.52)$$

are respectively the between and within class scatter matrices and  $\mathbf{m}_i$  is defined by  $\mathbf{m}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathbf{x}_j^i$ . The solution is the eigenvector corresponding to the largest eigenvalue of the generalized eigen-decomposition

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}. \quad (16.53)$$

Since the matrix  $\mathbf{S}_B$  has rank 1, only the leading eigenvector contains meaningful information.

Let  $\phi$  be a nonlinear map to some feature space  $F$ . Kernel Fisher discriminant analysis attempts to find a direction  $\mathbf{w} \in F$  to maximize

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W^\phi \mathbf{w}}, \quad (16.54)$$

where

$$\begin{aligned} \mathbf{S}_B^\phi &= (\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)(\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)^\top, \\ \mathbf{S}_W^\phi &= \sum_{i=1,2} \sum_{\mathbf{x} \in S_i} (\phi(\mathbf{x}) - \mathbf{m}_i^\phi) (\phi(\mathbf{x}) - \mathbf{m}_i^\phi)^\top \end{aligned} \quad (16.55)$$

with  $\mathbf{m}_i^\phi = \frac{1}{m_i} \sum_{j=1}^{m_i} \phi(\mathbf{x}_j^i)$ .

Define  $S = S_1 \cup S_2$  and denote its elements by  $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$  with  $m = m_1 + m_2$ . We would like to find an expansion for  $\mathbf{w}$  in the form  $\mathbf{w} = \sum_{i=1}^m \alpha_i \phi(\mathbf{x}_i)$ . It follows that

$$\mathbf{w}^\top \mathbf{m}_i^\phi = \frac{1}{m_i} \sum_{j=1}^{m_i} \sum_{k=1}^{m_i} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_k^i) = \boldsymbol{\alpha}^\top \mathbf{M}_i, \quad (16.56)$$

where vector  $\mathbf{M}_i$  is defined as  $(\mathbf{M}_i)_j = \frac{1}{m_i} \sum_{k=1}^{m_i} \kappa(\mathbf{x}_j, \mathbf{x}_k^i)$  and the dot products are replaced with kernels [33]. Based on (16.56), the numerator of (16.54) can be rewritten as

$$\mathbf{w}^\top \mathbf{S}_B^\phi \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}, \quad (16.57)$$

where  $\mathbf{M} = (\mathbf{M}_1 - \mathbf{M}_2)(\mathbf{M}_1 - \mathbf{M}_2)^\top$ . And the denominator is rewritten as

$$\mathbf{w}^\top \mathbf{S}_W^\phi \mathbf{w} = \boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}, \quad (16.58)$$

where  $\mathbf{N} = \sum_{j=1,2} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{m_j}) \mathbf{K}_j^\top$ ,  $\mathbf{K}_j$  is an  $m \times m_j$  matrix with  $(\mathbf{K}_j)_{ik} = \kappa(\mathbf{x}_i, \mathbf{x}_k^j)$ ,  $\mathbf{I}$  is the identity matrix and  $\mathbf{1}_{m_j}$  is the matrix with all entries  $\frac{1}{m_j}$  [33].

Hence, (16.54) is reformulated as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^\top \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \mathbf{N} \boldsymbol{\alpha}}. \quad (16.59)$$

The problem can be solved similarly to (16.51). To enhance numerical stability and perform capacity control in the feature space,  $\mathbf{N}$  in the above formulation is usually replace by  $\mathbf{N} + \mu \mathbf{I}$  with positive  $\mu$ . An alternative regularization is penalizing  $\|\mathbf{w}\|^2$  as in kernel CCA instead of the current  $\|\boldsymbol{\alpha}\|^2$  which corresponds to the term  $\mu \mathbf{I}$ .

#### 1.16.3.4 SVMs for classification and regression

Given the training set  $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  of points  $\mathbf{x}_i \in \mathbb{R}^n$  with corresponding labels  $y_i \in \{1, -1\}$ , SVM classifiers attempt to find a classification hyperplane induced from the maximum

margin principle [7,8]. In real applications data are usually not linearly separable. Thus a loss on the violation of the linearly separable constraints has to be introduced. A common choice is the hinge loss

$$\max \left( 0, 1 - y_i (\mathbf{w}^\top \mathbf{x}_i + b) \right), \quad (16.60)$$

which can be represented by a slack variable  $\xi_i$ .

The optimization problem for SVM classification is formulated as

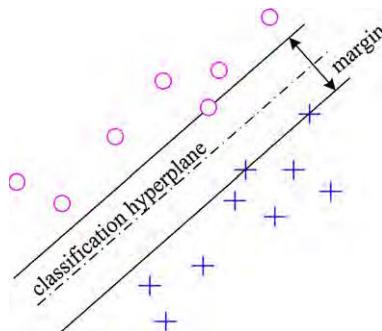
$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, m, \\ & \xi_i \geq 0, \quad i = 1, \dots, m, \end{aligned} \quad (16.61)$$

where the scalar  $C$  controls the balance between the margin and empirical loss, and  $\xi = [\xi_1, \dots, \xi_m]^\top$ . The large margin principle is reflected by minimizing  $\frac{1}{2} \|\mathbf{w}\|^2$  with  $2/\|\mathbf{w}\|$  being the margin between two hyperplanes  $\mathbf{w}^\top \mathbf{x} + b = 1$  and  $\mathbf{w}^\top \mathbf{x} + b = -1$  (For the linearly separable case, the concepts of the margin and classification hyperplane are illustrated in Figure 16.2). The SVM classifier would be

$$c_{\mathbf{w}, b}(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b). \quad (16.62)$$

The Lagrangian of problem (16.61) is

$$\begin{aligned} L(\mathbf{w}, b, \xi, \lambda, \gamma) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \lambda_i \left[ y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i \right] \\ & - \sum_{i=1}^m \gamma_i \xi_i, \quad \lambda_i \geq 0, \quad \gamma_i \geq 0, \end{aligned} \quad (16.63)$$



**FIGURE 16.2**

An illustration of the margin and classification hyperplane for the linearly separable binary case.

where  $\lambda = [\lambda_1, \dots, \lambda_m]^\top$  and  $\gamma = [\gamma_1, \dots, \gamma_m]^\top$  are the associated Lagrange multipliers. Using the superscript star to denote the solutions of the optimization problem, according to the KKT (Karush-Kuhn-Tucker) conditions [34, 35], we obtain

$$\partial_w L(\mathbf{w}^*, b^*, \xi^*, \lambda^*, \gamma^*) = \mathbf{w}^* - \sum_{i=1}^m \lambda_i^* y_i \mathbf{x}_i = 0, \quad (16.64)$$

$$\partial_b L(\mathbf{w}^*, b^*, \xi^*, \lambda^*, \gamma^*) = - \sum_{i=1}^m \lambda_i^* y_i = 0, \quad (16.65)$$

$$\partial_{\xi_i} L(\mathbf{w}^*, b^*, \xi^*, \lambda^*, \gamma^*) = C - \lambda_i^* - \gamma_i^* = 0, \quad i = 1, \dots, m. \quad (16.66)$$

From (16.64), the solution  $\mathbf{w}^*$  has the form

$$\mathbf{w}^* = \sum_{i=1}^m \lambda_i^* y_i \mathbf{x}_i. \quad (16.67)$$

Since examples with  $\lambda_i^* = 0$  can be omitted from the expression, the training examples for which  $\lambda_i^* > 0$  are called support vectors.

By substituting (16.64)–(16.66) into the Lagrangian, we can finally get the dual optimization problem [35]

$$\begin{aligned} \max_{\lambda} \quad & \lambda^\top \mathbf{j} - \frac{1}{2} \lambda^\top D \lambda \\ \text{s.t.} \quad & \lambda^\top \mathbf{y} = 0, \\ & \lambda \succeq \mathbf{0}, \\ & \lambda \leq C \mathbf{j}, \end{aligned} \quad (16.68)$$

where  $\mathbf{j}$  is the vector with all entries being 1,  $\mathbf{y} = [y_1, \dots, y_m]^\top$  and  $D$  is a symmetric  $m \times m$  matrix with entries  $D_{ij} = y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$  [1, 36].

The complementary slackness condition (also called the zero KKT-gap requirement) [6] implies

$$\begin{aligned} \lambda_i^* \left[ y_i (\mathbf{x}_i^\top \mathbf{w}^* + b^*) - 1 + \xi_i^* \right] &= 0, \quad i = 1, \dots, m, \\ \gamma_i^* \xi_i^* &= 0, \quad i = 1, \dots, m. \end{aligned} \quad (16.69)$$

Combining (16.66) and (16.69), we can solve  $b^* = y_i - \mathbf{x}_i^\top \mathbf{w}^*$  for any support vector  $\mathbf{x}_i$  with  $0 < \lambda_i^* < C$ . The existence of  $0 < \lambda_i^* < C$  is a reasonable assumption, though there lacks a rigorous justification [36]. Once  $\lambda^*$  and  $b^*$  are solved, the SVM classifier is given by

$$c^*(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^m y_i \lambda_i^* \mathbf{x}_i^\top \mathbf{x} + b^* \right). \quad (16.70)$$

Using the kernel trick, the optimization problem (16.68) for SVMs becomes

$$\max_{\lambda} \quad \lambda^\top \mathbf{j} - \frac{1}{2} \lambda^\top D \lambda$$

$$\begin{aligned} \text{s.t. } & \lambda^\top \mathbf{y} = 0, \\ & \lambda \succeq \mathbf{0}, \\ & \lambda \leq C\mathbf{j}, \end{aligned} \quad (16.71)$$

where the entries of  $D$  are  $D_{ij} = y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$ . The solution for the corresponding SVM classifier is formulated as

$$c^*(\mathbf{x}) = \text{sign} \left( \sum_{i=1}^m y_i \lambda_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + b^* \right). \quad (16.72)$$

For regression problems, the labels in the training set  $S$  are real numbers, that is  $y_i \in \mathbb{R}$  ( $i = 1, \dots, m$ ). In order to induce a sparse representation for the decision function (i.e., some training examples can be ignored), [8] devised the following  $\epsilon$ -insensitive function and applied it to support vector regression

$$|y - f(\mathbf{x})|_\epsilon = \max\{0, |y - f(\mathbf{x})| - \epsilon\}, \quad \epsilon \geq 0. \quad (16.73)$$

The standard form of support vector regression is to minimize

$$\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m |y_i - f(\mathbf{x}_i)|_\epsilon, \quad (16.74)$$

where the positive scalar  $C$  reflects the trade-off between the margin and the empirical loss. An equivalent optimization that is commonly used is

$$\begin{aligned} \min_{\mathbf{w}, b, \xi, \xi^*} & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i + C \sum_{i=1}^m \xi_i^* \\ \text{s.t. } & \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b - y_i \leq \epsilon + \xi_i, \\ & y_i - \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle - b \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, m, \end{aligned} \quad (16.75)$$

where  $\phi(\mathbf{x}_i)$  is the image of  $\mathbf{x}_i$  in the feature space, and  $\xi, \xi^*$  are defined similarly as before. The prediction output of support vector regression is

$$c^*(\mathbf{x}) = \langle \mathbf{w}^*, \phi(\mathbf{x}) \rangle + b^*, \quad (16.76)$$

where  $\mathbf{w}^*$  and  $b^*$  are the solution of (16.75). For support vector regression, the derivation for the dual representation of solutions and the dual optimization problem can consult the counterpart for classification, and thus is omitted here.

### 1.16.3.5 Bayesian kernel methods: Gaussian processes

All the previous methods introduced in this section can be summarized into the framework of risk minimization. The Bayesian learning approach differs from them in several aspects. The key distinction is that the Bayesian approach intuitively incorporates prior knowledge into the process of estimation [6]. Another benefit of the Bayesian framework is the possibility of measuring the confidence of the estimation in a straightforward manner. However, algorithms designed by the Bayesian approach (e.g., with maximum a posterior estimation) can have similar counterparts originating from the risk minimization

framework. Below we focus on the Gaussian process approach for regression, which is a classical Bayesian kernel method.

The Gaussian process models have two kinds of equivalent representations, namely the function-space view and the weight-space view [37]. We will start with the weight-space view to illustrate the explicit roles of kernels using the Bayesian treatment of linear regression, followed by a very brief introduction of the function-space view.

Suppose the training set  $S$  is  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  as defined in Section 1.16.2.1. The standard linear regression model with Gaussian noise is

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \quad y = f(\mathbf{x}) + \varepsilon, \quad (16.77)$$

where  $f$  is the function value,  $y$  is the noisy observed value, and the noise obeys an independent, identically distributed Gaussian distribution with mean zero and variance  $\sigma_n^2$

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2). \quad (16.78)$$

This gives rise to the likelihood of the independent observations in the training set

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= \prod_{i=1}^m p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{m/2}} \exp\left(-\frac{\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2}{2\sigma_n^2}\right) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma_n^2 \mathbf{I}), \end{aligned} \quad (16.79)$$

where  $\mathbf{y} = [y_1, \dots, y_m]^\top$  and  $\mathbf{X}^\top = [\mathbf{x}_1, \dots, \mathbf{x}_m]$ . Suppose we specify a Gaussian prior on the parameters with mean zero and covariance matrix  $\Sigma_p$  [37]

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p). \quad (16.80)$$

According to Bayes' rule, the posterior of the parameters is proportional to the product of the prior and likelihood

$$\begin{aligned} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto \exp\left(-\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1} \mathbf{w}\right) \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w})\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^\top A(\mathbf{w} - \bar{\mathbf{w}})\right), \end{aligned} \quad (16.81)$$

where  $A = \frac{1}{\sigma_n^2} \mathbf{X}^\top \mathbf{X} + \Sigma_p^{-1}$ , and  $\bar{\mathbf{w}} = \frac{1}{\sigma_n^2} A^{-1} \mathbf{X}^\top \mathbf{y}$ . It turns out that the posterior is a Gaussian distribution with mean  $\bar{\mathbf{w}}$  and covariance  $A^{-1}$ .

The predictive distribution for a test example  $\mathbf{x}$  is given by averaging the outputs of all possible linear models from the above Gaussian posterior

$$\begin{aligned} p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) &= \int p(f(\mathbf{x})|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{x}^\top A^{-1} \mathbf{X}^\top \mathbf{y}, \mathbf{x}^\top A^{-1} \mathbf{x}\right). \end{aligned} \quad (16.82)$$

Now suppose we use a function  $\phi(\cdot)$  to map the inputs in the original space to a feature space, and perform linear regression there. The predictive distribution would be

$$p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2} \phi(\mathbf{x})^\top A^{-1} \Phi^\top \mathbf{y}, \phi(\mathbf{x})^\top A^{-1} \phi(\mathbf{x})\right), \quad (16.83)$$

where  $\Phi^\top = [\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_m)]$ , and  $A = \frac{1}{\sigma_n^2} \Phi^\top \Phi + \Sigma_p^{-1}$ . Using matrix transformations such as the matrix inversion lemma, we can rewrite (16.83) as

$$\begin{aligned} p(f(\mathbf{x})|\mathbf{x}, \mathbf{X}, \mathbf{y}) &= \mathcal{N}\left(\phi(\mathbf{x})^\top \Sigma_p \Phi^\top (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \right. \\ &\quad \left. \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}) - \phi(\mathbf{x})^\top \Sigma_p \Phi^\top (K + \sigma_n^2 \mathbf{I})^{-1} \Phi \Sigma_p \phi(\mathbf{x})\right), \end{aligned} \quad (16.84)$$

where  $K = \Phi \Sigma_p \Phi^\top$  [37]. Notice that in the above formulation the terms related to the images in the feature space can be represented in the form of  $\phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$  with  $\mathbf{x}$  and  $\mathbf{x}'$  in either the training or test sets [37]. Define  $\kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$ . By Theorem 7, we know that  $\kappa(\mathbf{x}, \mathbf{x}')$  is a valid kernel function. In the Gaussian process literature, it is often called the covariance function.

The function-space view of the Gaussian processes is given by the following definition which describes a distribution over functions [37].

**Definition 8 (Gaussian processes).** A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is specified by its mean function and covariance function. If we define the mean function  $m(\mathbf{x})$  and the covariance function  $k(\mathbf{x}, \mathbf{x}')$  of a real process  $f(\mathbf{x})$  as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x})) (f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \quad (16.85)$$

the Gaussian process can be written as

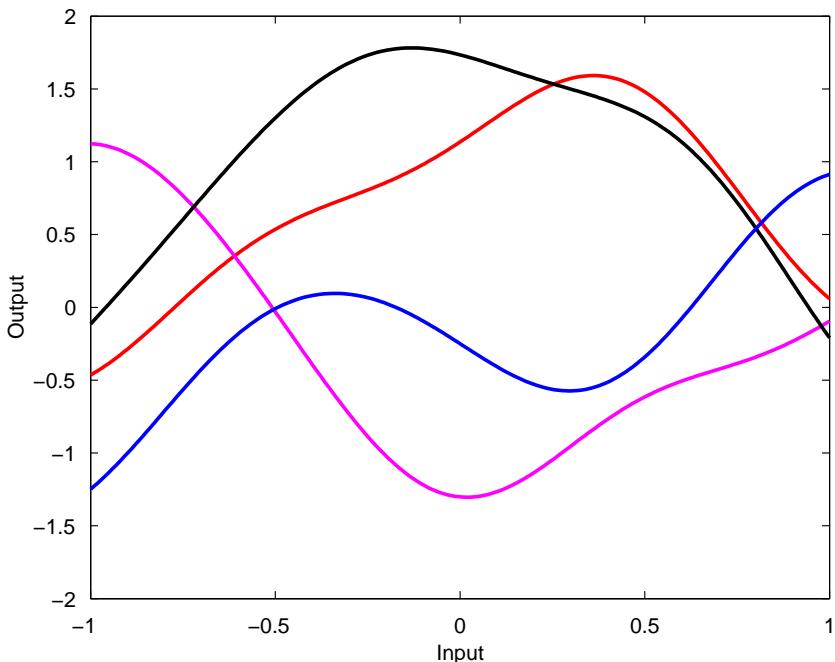
$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \quad (16.86)$$

Figure 16.3 shows samples of functions drawn from a specific Gaussian process.

The Bayesian linear regression model  $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$  with parameter prior  $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$  can be cast into the above function-space view. It is simple to see that the function values  $f(\mathbf{x}_1), \dots, f(\mathbf{x}_q)$  corresponding to any number of inputs  $q$  are jointly Gaussian, and the mean and covariance are given by

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \mathbb{E}[\mathbf{w}^\top] \phi(\mathbf{x}) = 0, \\ \mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] &= \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w} \mathbf{w}^\top] \phi(\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}'), \end{aligned} \quad (16.87)$$

where the second equation recovers our definition of the kernel function for the weight-space view. In other words, now  $m(\mathbf{x}) = 0$  and  $k(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$ .

**FIGURE 16.3**

Samples from a Gaussian process with zero mean and a Gaussian kernel as the covariance function.

## 1.16.4 Computational issues of kernel methods

Implementation of kernel methods often involve eigen-decomposition of kernel matrices or inversion of the sum of a kernel matrix and a scaled identity matrix. The computational complexity of this operation is typically  $O(m^3)$  with  $m$  being the number of training examples. This can be very demanding for large training sets, and therefore approximation algorithms are desirable.

Good low-rank approximations of kernel matrices are usually enough to deal with the above problems. For example, the matrix inversion lemma can use the low-rank decomposition to invert the sum of the kernel matrix and a scaled identity matrix efficiently. Eigen-decomposition of kernel matrices can also be converted to do the same operation on much smaller matrices. Here we just give a pointer to some of the approximation methods. Interested readers can refer to the corresponding literature for detailed implementation techniques.

Partial Gram-Schmidt orthogonalization [38] and incomplete Cholesky decomposition [27] are good approaches for finding low-rank approximations of kernel matrices. These two approaches are essentially equivalent, since performing a Cholesky decomposition of the kernel matrix is equivalent to performing

Gram-Schmidt orthogonalization in the feature space [1]. In other words, incomplete Cholesky decomposition can be viewed as the dual implementation of the partial Gram-Schmidt orthogonalization.

The sparse greedy matrix approximation [39] and the Nyström approximation [40] are two alternative approaches. The idea of the former is to select a collection of basis functions to obtain an approximate kernel matrix  $\tilde{K}$  whose distance to the original kernel matrix  $K$  is small. The Nyström approximation is much simpler, which randomly chooses  $r$  rows/columns of  $K$  without replacement, and then sets  $\tilde{K} = K_{m,r} K_{r,r}^{-1} K_{r,m}$ , where  $K_{m,r}$  is the  $m \times r$  block of  $K$  and similar definitions apply to the other blocks. For a given  $r$ , the sparse greedy matrix approximation produces a better approximation to  $K$ , but computationally more demanding [40].

### 1.16.5 Multiple kernel learning

In practical problems, for a given decision-making task, there can be multiple different data sources. These data can be heterogeneous, which means that they represent different properties (e.g., visual features or lingual features) or have different forms (e.g., continuous value or discrete value). Consequently, using a different kernel to account for each of the data sources and then combining them is sensible. In other cases, even if the data are homogeneous, we may still want to adopt multiple kernels to capture more information. This problem of learning a combination of multiple kernels is termed multiple kernel learning [17] and now is an active research topic [41–47]. Here we review some multiple kernel learning methods, several of which have sparsity regularizations [48, 49], to provide a brief outline of the research progress.

The kernel learning approach proposed by [17] is to add to the original optimization problems some extra constraints on the symmetric kernel matrix  $K$ , e.g., by

$$\begin{aligned} K &= \sum_{i=1}^t \mu_i K_i, \\ \mu_i &\in \mathbb{R}, \quad i = 1, \dots, t, \\ K &\succeq 0, \\ \text{trace}(K) &\leq c, \end{aligned} \tag{16.88}$$

or

$$\begin{aligned} K &= \sum_{i=1}^t \mu_i K_i, \\ \mu_i &\geq 0, \quad i = 1, \dots, t, \\ K &\succeq 0, \\ \text{trace}(K) &\leq c, \end{aligned} \tag{16.89}$$

where  $t$  is the number of individual kernels, and then formulate the problem in terms of semidefinite programming (SDP) [34, 35]. The advantages of the second set of constraints over the first include reducing computational burden and facilitating the study of some statistical properties of kernel matrices [17].

However, the learning problem would become intractable when the number of training examples or kernels grow large.

[48] reformulated the problem and proposed a sequential minimal optimization (SMO) algorithm to improve the efficiency. They used second-order cone programming (SOCP) and Moreau-Yosida regularization to derive the SMO algorithm and made multiple kernel learning applicable for medium-scale problems. The corresponding KKT conditions not only lead to support vectors, but also to “support kernels” which means a sparse combination of candidate kernels can be expected. [50] adopted semi-infinite liner programming (SILP) to formulate the multiple kernel learning problem, based on which they iteratively solved a standard SVM problem with a single kernel and a linear program whose constraints increase with iterations. This approach makes multiple kernel learning applicable to large-scale problems. Later, [49] further improved the efficiency by using a formulation of a weighted 2-norm regularization with sparsity considerations imposed on the weights. Evidence show that it is globally faster than the mentioned SILP approach but with more kernels selected.

[41] considered multiple kernel learning with infinite number of basic kernels. In particular, kernels are selected from the convex hull of continuously parameterized kernels. Making use of the conjugate function and von Neumann minimax theorem [44], they adopted a greedy algorithm to solve the optimization problem, where the DC (difference of convex functions) programming techniques that attempt to optimize a non-convex function by the difference of two convex functions [51] were used to optimize a subroutine of the algorithm. Experimental results indicated the advantage of working with a continuous parameterization over a predesignated finite number of basic kernels.

For Bayesian multiple kernel learning, recently, [52] proposed a new variational approximation for infinite mixtures of Gaussian processes. The mixtures of Gaussian processes have the advantages of characterizing varying covariances or multimodal data and reducing the cubic computational complexity of the single Gaussian process model [53–55]. They used mean field variational inference and a truncated stick-breaking representation of the Dirichlet process to approximate the posterior of latent variables, and applied the approach to traffic prediction problems.

## 1.16.6 Applications

The applications of kernel methods and SVMs are rather broad. Here we just list some of its typical applications.

Biometrics refers to the identification of humans based on their physical or behavioral traits, which can be used for access control. Typical methods for biometrics include face recognition, signature recognition and EEG-based biometrics [56]. Over the past years, kernel methods and SVMs have been successfully applied to this field [57,58].

Intelligent transportation systems are an important application platform for machine learning techniques. Representative applications include pedestrian recognition, traffic flow forecasting, and traffic bottleneck identification. Kernel methods including Gaussian processes have achieved very good performance in this area [52,59].

Research on brain-computer interfaces which aim to enable severely disabled people to drive communication or control devices, arouses many interests recently. The discrimination of different brain

signals is essentially a pattern classification problem, where SVMs have been shown to be a very useful tool [60,61].

Natural language processing, e.g., text classification and retrieval, is an active research field which has used a lot of machine learning methods. Kernel techniques applied to this task include kernel design, supervised classification and semi-supervised classification [14,62,63].

### 1.16.7 Open issues and problems

In this article, we have presented some key techniques for using kernel methods, such as how to derive the dual formulation of an original method, what are essential conditions for valid kernels, typical kernel functions, and how to construct new kernels. This constitutes the foundations of kernel methods. Then, we introduced some fundamental kernel methods which are well-known and now used widely for unsupervised or supervised learning. In particular, as a representative of Bayesian kernel methods, Gaussian processes were introduced.

The computational complexity of kernel methods is usually cubic with respect to the number of training examples. Therefore, reducing the computational costs has been an important research topic. For this problem, we briefly pointed out four methods—partial Gram-Schmidt orthogonalization, incomplete Cholesky decomposition, sparse greedy matrix approximation and the Nyström approximation, and explained the idea on why they can be used to alleviate the computational burden.

In addition, we have introduced the recent developments on multiple kernel learning which has shown its merit over single kernel learning in the past few years. However, for multiple kernel learning, we have to learn both the combination coefficients for candidate kernels and other parameters inherited from traditional single kernel learning. Therefore, there are various efforts to reformulate the optimization problem to accelerate learning, and indeed people have achieved some encouraging results.

Studies on kernel methods can be further deepened in different aspects, e.g., the above mentioned multiple kernel learning, and combining kernel techniques with other machine learning mechanisms. Another line of important open problems would be performing theoretical analysis on the generalization errors of newly emerging kernel methods, such as the multitask SVMs and multitask multiclass SVMs. We hope this article is helpful to promote the applications and theoretical developments of kernel methods in the future.

---

## Glossary

Canonical correlation analysis	A method to find two linear transformations respectively for two representations such that the correlations between the transformed variables are maximized
Fisher discriminant analysis	A method for classification which seeks a direction to maximize the distance between projected class means and simultaneously minimize the projected class variances
Gaussian process	A collection of random variables, any finite number of which have a joint Gaussian distribution

Kernel trick	A method to extend any algorithm only involving computations of inner products from the original space to a feature space with kernel functions
Multiple kernel learning	A learning mechanism which aims to learn a combination of multiple kernels to capture more information or reduce the computational complexity
Principal component analysis	A method to find a set of orthogonal directions which forms a subspace to maximize data variances along the directions in this subspace
Reproducing kernel Hilbert space	A function space which is a Hilbert space possessing a reproducing kernel
Support vector machine	A method to learn a hyperplane induced from the maximum margin principle, which has wide applications including classification and regression

---

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Project 61075005, the Fundamental Research Funds for the Central Universities, and the PASCAL2 Network of Excellence. This publication only reflects the authors' views.

---

## References

- [1] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004.
- [2] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, *Psychol. Reviews* 65 (1958) 386–408.
- [3] J. Hertz, A. Krogh, R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, MA, 1991.
- [4] L. Breiman, J. Friedman, R. Olshen, C. Stone, *Classification and Regression Trees*, Wadsworth International, Belmont, CA, 1984.
- [5] J. Quinlan, *C4.5: programs for Machine Learning*, San Mateo, California: Morgan Kauffmann, 1993.
- [6] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.
- [7] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifier, in: *Proceedings of the 5th ACM Worksop on Computational Learning Theory*, 1992, pp. 144–152.
- [8] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
- [9] M. Aizerman, E. Braverman, L. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, *Autom. Remote Control* 25 (1964) 821–837.
- [10] R. Duda, P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.
- [11] T. Evgeniou, M. Pontil, Regularized multi-task learning, in: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 109–117.
- [12] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, S. Szegdak, Two view learning: SVM-2K, theory and practice, *Adv. Neural Inform. Process. Syst.* 18 (2006) 355–362.

- [13] D. Rosenberg, V. Sindhwani, P. Bartlett, P. Niyogi, Multiview point cloud kernels for semisupervised learning, *IEEE Signal Process. Mag.* 26 (2009) 145–150.
- [14] S. Sun, J. Shawe-Taylor, Sparse semi-supervised learning using conjugate functions, *J. Mach. Learn. Res.* 11 (2010) 2423–2455.
- [15] Y. Ji, S. Sun, Multitask multiclass support vector machines, in: Proceedings of the IEEE International Conference on Data Mining Workshops, 2011, pp. 512–518.
- [16] S. Sun, Multi-view Laplacian support vector machines, *Lect. Notes Comput. Sci.* 7121 (2011) 209–222.
- [17] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. Jordan, Learning the kernel matrix with semidefinite programming, *J. Mach. Learn. Res.* 5 (2004) 27–72.
- [18] C. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, New York, 2006.
- [19] R. Duda, P. Hart, D. Stork, *Pattern Classification*, John Wiley & Sons, New York, 2001.
- [20] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press 2008.
- [21] N. Aronszajn, Theory of reproducing kernels, *Trans. Am. Math. Soc.* 68 (1950) 337–404.
- [22] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, *Philos. Trans. Roy. Soc., London, A* 209 (1909) 415–446.
- [23] T. Jaakkola, D. Haussler, Exploiting generative models in discriminative classifiers, *Adv. Neural Inform. Process. Syst.* 11 (1999a) 487–493.
- [24] T. Jaakkola, D. Haussler, Probabilistic kernel regression models, in: Proceedings of the International Conference on AI and, Statistics, 1999b, pp. 1–9.
- [25] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, *Neural Comput.* 10 (1998) 1299–1319.
- [26] H. Hotelling, Relations between two sets of variates, *Biometrika* 28 (1936) 321–377.
- [27] F. Bach, M. Jordan, Kernel independent component analysis, *J. Mach. Learn. Res.* 3 (2002) 1–48.
- [28] J. Kettenring, Canonical analysis of several sets of variables, *Biometrika* 58 (1971) 433–451.
- [29] S. Akaho, A kernel method for canonical correlation analysis, in: Proceedings of the International Meeting of the Psychometric Society 2001.
- [30] C. Fyfe, P. Lai, ICA using kernel canonical correlation analysis, in: Proceedings of the International Workshop on Independent Component Analysis and Blind Singal Separation, 2000, pp. 279–284.
- [31] T. Melzer, M. Reiter, H. Bischof, Nonlinear feature extraction using generalized canonical correlation analysis, in: Proceedings of the International Conference on Artificial, Neural Networks, 2001, pp. 353–360.
- [32] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, CA, 1990.
- [33] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K. Müller, Fisher discriminant analysis with kernels, *Neural Netw. Signal Process.* IX (1999) 41–48.
- [34] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, England, 2004.
- [35] J. Shawe-Taylor, S. Sun, A review of optimization methodologies in support vector machines, *Neurocomputing* 74 (2011) 3609–3618.
- [36] E. Osuna, R. Freund, F. Girosi, Support vector machines: training and applications, Technical Report AIM-1602, Massachusetts Institute of Technology, MA, 1997.
- [37] C. Rasmussen, C. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.
- [38] D. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: an overview with application to learning methods, *Neural Comput.* 16 (2004) 2639–2664.
- [39] A. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: Proceedings of the 17th International Conference on, Machine learning, 2000, pp. 911–918.
- [40] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, *Adv. Neural Inform. Process. Syst.* 13 (2001) 682–688.
- [41] A. Argyriou, R. Hauser, C. Micchelli, M. Pontil, A DC-programming algorithm for kernel selection, in: Proceedings of the 23rd International Conference on, Machine Learning, 2006, pp. 41–48.

- [42] M. Girolami, S. Rogers, Hierachic Bayesian models for kernel learning, in: Proceedings of the 22nd International Conference on, Machine Learning, 2005, pp. 241–248.
- [43] J. Li, S. Sun, Nonlinear combination of multiple kernels for support vector machines, in: Proceedings of the 20th International Conference on, Pattern Recognition, 2010, pp. 1–4.
- [44] A. Micchelli, M. Pontil, Learning the kernel function via regularization, *J. Mach. Learn. Res.* 6 (2005) 1099–1125.
- [45] C. Ong, A. Smola, R. Williamson, Learning the kernel with hyperkernels, *J. Mach. Learn. Res.* 6 (2005) 1043–1071.
- [46] Y. Ying, D. Zhou, Learnability of Gaussians with flexible variances, *J. Mach. Learn. Res.* 8 (2007) 249–276.
- [47] A. Zien, C. Ong, Multiclass multiple kernel learning, in: Proceedings of the 24th International Conference on, Machine Learning, 2007, pp. 1191–1198.
- [48] F. Bach, G. Lanckriet, M. Jordan, Multiple kernel learning, conic duality and the SMO algorithm, in: Proceedings of the 21st International Conference on, Machine Learning, 2004, pp. 6–13.
- [49] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, More efficiency in multiple kernel learning, in: Proceedings of the 24th International Conference on, Machine Learning, 2007, pp. 775–782.
- [50] S. Sonnenburg, G. Raetsch, C. Schaefer, B. Schölkopf, Large scale multiple kernel learning, *J. Mach. Learn. Res.* 7 (2006) 1531–1565.
- [51] R. Horst, V. Thoai, DC programming: Overview, *J. Optimiz. Theory App.* 103 (1999) 1–41.
- [52] S. Sun, X. Xu, Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction, *IEEE Trans. Intel. Transp. Syst.* 12 (2011) 466–475.
- [53] E. Meeds, S. Osindero, An alternative infinite mixture of Gaussian process experts, *Adv. Neural Inform. Process. Syst.* 18 (2006) 883–890.
- [54] C. Rasmussen, Z. Ghahramani, Infinite mixtures of Gaussian process experts, *Adv. Neural Inform. Process. Syst.* 14 (2002) 881–888.
- [55] V. Tresp, Mixtures of Gaussian processes, *Adv. Neural Inform. Process. Syst.* 13 (2001) 654–660.
- [56] S. Sun, Multitask learning for EEG-based biometrics. in: Proceedings of the 19th International Conference on, Pattern Recognition, 2008, pp. 1–4.
- [57] A. Tefas, C. Kotropoulos, I. Pitas, Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication, *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (2001) 735–746.
- [58] E. Justino, F. Bortolozzi, R. Sabourin, A comparison of SVM and HMM classifiers in the off-line signature verification. *Pattern Recogn. Lett.* 26, 1377–1385.
- [59] S. Munder, D. Gavrila, An experimental study on pedestrian classification, *IEEE Trans. Pattern Anal. Mach. Intell.* 28 (2006) 1863–1868.
- [60] D. Garrett, D. Peterson, C. Anderson, M. Thaut, Comparison of linear, nonlinear, and feature selection methods for EEG signal classification, *IEEE Trans. Neural Syst. Rehabil. Eng.* 11 (2003) 141–144.
- [61] S. Sun, C. Zhang, D. Zhang, An experimental evaluation of ensemble methods for EEG signal classification, *Pattern Recogn. Lett.* 28 (2007) 2157–2163.
- [62] M. Collins, N. Duffy, Convolution kernels for natural language, *Advances in Neural Information Processing Systems* 14 (2001) 625–632.
- [63] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, *Lect. Notes Comput. Sci.* 1398 (1998) 137–142.

# Online Learning in Reproducing Kernel Hilbert Spaces

# 17

**Konstantinos Slavakis\***, **Pantelis Bouboulis†**, and **Sergios Theodoridis†**

\**University of Peloponnese, Department of Telecommunications Science and Technology,  
Karaiskaki St., Tripolis 22100, Greece*

†*University of Athens, Department of Informatics and Telecommunications, Ilissia, Athens 15784, Greece*

---

## Nomenclature

Training Data	A set of input-output measurements that are used to train a specific learning machine
Online Learning	A learning task, where the data points arrive sequentially and used only once
Batch Learning	A learning task, where the training data are known beforehand
Regression Task	A learning task that estimates a parametric input-output relationship that best fits the available data
Classification Task	A learning task that classifies the data into one or more classes
Overfitting	A common notion in the Machine Learning literature. If the adopted model is too complex then the learning algorithm tends to fit to the training data as much as possible (e.g., it fits even the noisy data) and cannot perform well when new data arrive
Regularization	A technique that biases the solution towards a smoother result. This is usually achieved by adding the norm of the solution to the cost function of the minimization task
Sparsification	A method that biases the solution towards a result, that the least significant components are pushed to zero
Reproducing Kernel Hilbert Spaces (RKHS)	An inner product space of functions that is complete and has a special structure
Kernel	A positive definite function of two variables that is associated to a specific RKHS

Fréchet Differentiation	A mathematical framework that generalizes the notion of the standard derivative, to more general function spaces
Least Mean Squares	This is a class of stochastic online learning tasks, which aim to find the parameters of an input-output relation by minimizing the least mean squares of the error at each iteration (difference between the desired and the actual output), using a gradient descent rationale
Recursive Least Squares	A class of deterministic online learning tasks, which aim to find the parameters of an input-output relation by minimizing the least mean squares of the total error (the sum of differences between the desired and the actual output up to the current iteration) using a recursive Newton-like method
Adaptive Projected Subgradient Method (APSM)	A convex analytic framework for online learning. Based on set theoretic estimation arguments, and fixed point theory, the APSM is tailored to attack online learning problems related to general, convexly constrained, non-smooth optimization tasks
Wirtinger's Calculus	A methodology for computing gradients of real valued functions that are defined on complex domains, using the standard rules of differentiation

### 1.17.1 Introduction

A large class of Machine Learning tasks becomes equivalent with estimating an unknown functional dependence that relates the so called input (cause) to the output (effect) variables. Most often, this function is chosen to be of a specific type, e.g., linear, quadratic, etc., and it is described by a set of unknown parameters. There are two major paths in dealing with this set of parameters.

The first one, known as *Bayesian Inference*, treats the parameters as random variables, [1]. According to this approach, the dependence between the input and output variables is expressed via a set of pdfs. The task of learning is to obtain estimates of the pdf that relates the input-output variables as well as the pdf that describes the random nature of the parameters. However, the main concern of the Bayesian Inference approach is not in obtaining specific values for the parameters. Bayesian Inference can live without this information. In contrast, the alternative path, which will be our focus in this paper, considers the unknown parameters to be deterministic and its major concern is to obtain specific estimates for their values.

Both trends share a common trait; they dig out the required information, in order to build up their estimates, from an available set of input-output measurements known as the *training set*. The stage

of estimating the pdfs/parameters is known as training and this philosophy of learning is known as *supervised learning*. This is not the only approach which Machine Learning embraces in order to “learn” from data. *Unsupervised* and *semi-supervised* learning are also major learning directions, [2]. In the former, no measurements of the output variable(s) are available in the training data set. In the latter, there are only a few. In this paper, we will focus on the supervised learning of a set of deterministically treated parameters. We will refer to this task as *parameter estimation*.

In the parameter estimation task, depending on the adopted functional dependence, different models result. At this point, it has to be emphasized that the only “truth” that the designer has at his/her disposal is the set of training data. The model that the designer attempts to “fit” in the data is just a concept in an effort to be able to exploit and use the available measurements for his/her own benefit; that is, to be able to make useful predictions of the output values in the future, when new measurements of the inputs, outside the training set, become available. The ultimate goal of the obtained predictions is to assist the designer to make decisions, once the input measurements are disclosed to him/her. In nature, one can never know if there is a “true” model associated with the data. Our confidence in “true” models only concerns simulated data. In real life, one is obliged to adopt a model and a training method to learn its parameters so that the resulting predictions can be useful in practice. The designed Machine Learning system can be used till it is substituted by another, which will be based on a new model and/or a new training method, that leads to improved predictions and which can be computed with affordable computational resources.

In this paper, we are concerned with the more general case of nonlinear models. Obviously, different nonlinearities provide different models. Our approach will be in treating a large class of nonlinearities, which are usually met in practice, in a unifying way by mobilizing the powerful tool of Reproducing Kernel Hilbert Spaces (RKHS). This is an old concept, that was gradually developed within the functional analysis discipline [3] and it was, more recently, popularized in Machine Learning in the context of *Support Vector Machines*. This methodology allows one to treat a wide class of different nonlinearities in a unifying way, by solving an equivalent linear task in a space different than the one where the original measurements lie. This is done by adopting an implicit mapping from the input space to the so called *feature* space. In the realm of RKHSs, one needs not to bother about the specific nature of this space, not even about its dimensionality. Once the parametric training has been completed, the designer can “try” different implicit mappings, which correspond to different nonlinearities and keep the one that fits best his/her needs. This is achieved during the so-called *validation* stage. Ideally, this can be done by testing the performance of the resulting estimator using a data set that is independent of the one used for training. In case this is not possible, then various techniques are available in order to exploit a single data set, both for training and validation, [2].

Concerning the estimation process, once a parametric modeling has been adopted, the parameter estimation task relies on adopting, also, a criterion that measures the quality of fit between the predicted values and the measured ones, over the training set. Different criteria result in different estimates and it is up to the designer to finally adopt the one that better fits his/her needs. In the final step of parameter estimation, an algorithm has to be chosen to optimize the criterion. There are two major philosophies that one has to comply with. The more classical one is the so-called *batch processing* approach. The training data are available in “one go.” The designer has access to all of them simultaneously, which are then used for the optimization. In some scenarios, algorithms are developed so that to consider one

training point at a time and consider all of them sequentially, and this process goes on till the algorithm converges. Sometimes, such schemes are referred to as online; however, these schemes retain their batch processing flavor, in the sense that the training data set is *fixed* and the number of the points in it is known before the training starts. Hence, we will refer to these as batch techniques and keep the term online for something different.

All batch schemes suffer from a major disadvantage. Once a new measurement becomes available, after the training has been completed, the whole training process has to start from scratch, with a new training set that has been enriched by the extra measurement. No doubt, this is an inefficient way to attack the task. The most efficient way is to have methods that perform the optimization recursively, in the sense of updating the current estimate every time a new measurement is received, by considering *only* this newly received information and the currently available estimate of the parameters. The previously used training data will *never be needed again*. They have become part of the history; all information that they had to contribute to the training process now resides in the current estimate. We will refer to such schemes as *online* or *adaptive*, although many authors call them *time-recursive* or *sequential*. The term Adaptive Learning is mainly used in Signal Processing and such schemes have also been used extensively in Communications, System Theory and Automatic Control from the early sixties, in the context of the LMS, RLS and Kalman Filtering, e.g., [4–6]. The driving force behind the need for such schemes was the need for the training mechanism to be able to accommodate slow time variations of the learning environment and slowly forget the past. As a matter of fact, such a philosophy imitates better the way that human brain learns and tries to adapt to new situations. Online learning mechanisms have recently become very popular in applications such as Data Mining and Bioinformatics and more general in cases where data reside in large data bases, with massive number of training points, which cannot be stored in the memory and have to be considered one at a time. It is this path of online parameter estimation learning that we will pursue in this paper. As we will see, it turns out that very efficient schemes, of even linear complexity per iteration with respect to the number of the unknown parameters, are now available.

Our goal in this overview is to make an effort to collect a number of online/adaptive algorithms, which have been proposed over a number of decades, in a systematic way under a common umbrella; they are schemes that basically stem from only a few generic recurrences. It is interesting to note that some of these algorithms have been discovered and used under different names in different research communities, without one being referring to the other.

In an overview paper, there is always the dilemma how deep one can go in proofs. Sometimes proofs are completely omitted. We decided to keep some proofs, associated with the generic schemes; those which do not become too technical. Proofs give the newcomer to the field the flavor of the required mathematical tools and also give him/her the feeling of what is behind the performance of an algorithm; in simple words, why the algorithm works. For those who are not interested in proofs, simply, they can bypass them. It was for the less mathematically inclined readers, that we decided to describe first two structurally simple algorithms, the LMS and the RLS. We left the rest, which are derived via convex analytic arguments, for the later stage. As we move on, the treatment may look more mathematically demanding, although we made an effort to simplify things as much as we could. After all, talking about algorithms one cannot just use “words” and a “picture” of an algorithm. Although sometimes this is a line that is followed, the authors of this paper belong to a different school. Thus, this paper may not address the needs of a black-box user of the algorithms.

## 1.17.2 Parameter estimation: The regression and classification tasks

In the sequel,  $\mathcal{R}$ ,  $\mathcal{N}$ , and  $\mathcal{N}_*$  will stand for the set of all real numbers, non-negative and positive integers, respectively.

Two of the major pillars in parametric modeling in Machine Learning are the *regression* and *classification* tasks. These are two closely related, yet different tasks.

In regression, given a set of training points,  $(y_n, \mathbf{x}_n), n = 1, 2, \dots, y_n \in \mathcal{R}, \mathbf{x}_n \in \mathcal{R}^l$ <sup>1</sup>, the goal is to establish the input-output relationship via a model of the form

$$y_n = f(\mathbf{x}_n) + \eta_n, \quad n = 1, 2, \dots, \quad (17.1)$$

where  $\eta_n$  is a noise, unobservable, sequence and the nonlinear function is modeled in the form

$$f(\mathbf{x}) = \theta_0 + \sum_{k=1}^{K-1} \theta_k \phi_k(\mathbf{x}), \quad (17.2)$$

where  $\theta_k, k = 0, 1, \dots, K - 1$ , comprise the set of the unknown parameters, and  $\phi_k(\mathbf{x})$  are preselected nonlinear functions

$$\phi_k(\cdot) : \mathcal{R}^l \rightarrow \mathcal{R}, \quad k = 1, 2, \dots, K - 1.$$

The input vectors are also known as *regressors*. Combining (17.1) and (17.2) we obtain

$$y_n = \theta_0 + \sum_{k=1}^{K-1} \theta_k \phi_k(\mathbf{x}_n) + \eta_n := \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}_n) + \eta_n, \quad (17.3)$$

where

$$\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot), \phi_2(\cdot), \dots, \phi_{K-1}(\cdot), 1]^T, \text{ and } \boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_{K-1}, \theta_0]^T.$$

$\theta_0$  is known as the *bias* or the *intercept* and it has been absorbed in  $\boldsymbol{\theta}$  by simultaneously adding 1 as the last element in  $\boldsymbol{\phi}$ . The goal of the parameter estimation task is to obtain estimates  $\hat{\boldsymbol{\theta}}$  of  $\boldsymbol{\theta}$  using the available training set. Once  $\hat{\boldsymbol{\theta}}$  has been obtained and given a value of  $\mathbf{x}$ , the prediction of the corresponding output value is computed according to the model

$$\hat{y} = \hat{\boldsymbol{\theta}}^T \boldsymbol{\phi}(\mathbf{x}). \quad (17.4)$$

Figure 17.1 shows the graphs of (17.4) for two different cases of training data,  $(y_n, \mathbf{x}_n)$ . In Figure 17.1a, the fitted model is

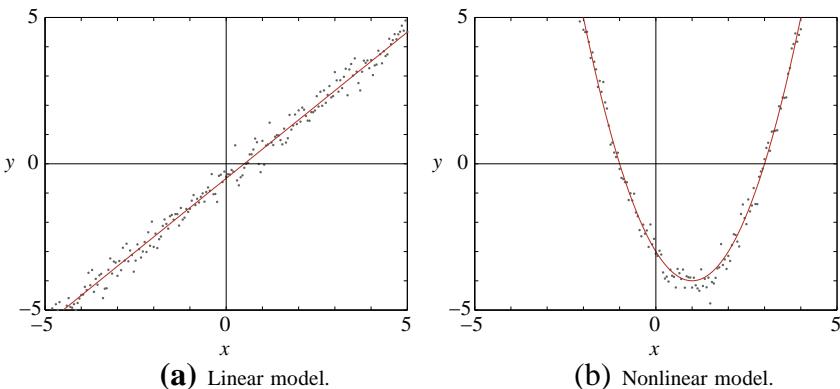
$$\hat{y} = -0.5 + x,$$

and for the case of Figure 17.1b, the model is

$$\hat{y} = -3 - 2\phi_1(x) + \phi_2(x),$$

---

<sup>1</sup>In general,  $y \in \mathcal{R}^m$ . Unless otherwise stated, we will consider  $y$  to be a real value for simplicity. Generalizations to Euclidean spaces are also possible.



## FIGURE 17.1

Fitting sets of training data  $(y_n, \mathbf{x}_n)_{n=1}^N$  by a linear and a quadratic function.

where

$$\phi_1(x) = x, \quad \phi_2(x) = x^2.$$

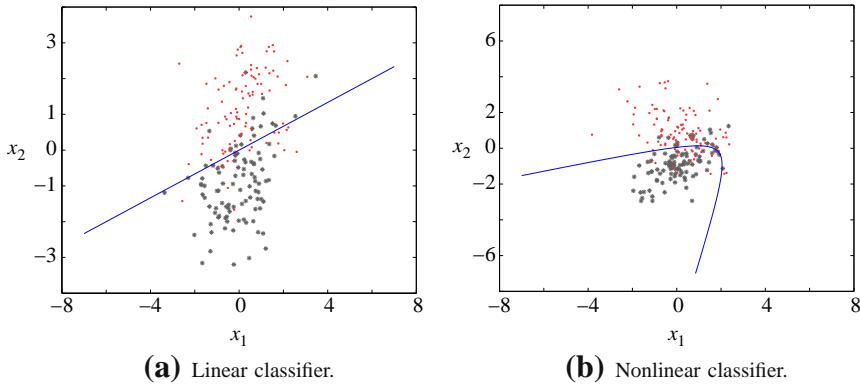
The figures also reveal the main goal of regression, which is to “explain” the generation mechanism of the data, and the graph of the curve defined in (17.4) should be designed so that to follow the spread of the data in the  $(y, x)$  space as close as possible.

In classification, the task is slightly different. The output variables are of a discrete/qualitative nature. That is,  $y_n \in D$ , where  $D$  is a finite set of discrete values. For example, for the simple case of a two-class classification problem, one may select  $D = \{1, -1\}$  or  $D = \{1, 0\}$ . The output values are also known as *class labels*. The input vectors,  $\mathbf{x}$ , are known as *feature vectors*, and they are chosen so that the respective components to encapsulate as much class-discriminatory information as possible. The goal of a classification task is to design a function (or in the more general case a set of functions), known as the classifier, so that the corresponding (hyper)surface  $f(\mathbf{x}) = 0$ , in the  $\mathbf{x}$  space, to separate the points that belong to different classes as much as possible. That is, the purpose of a classifier is to divide the input space into regions, where each one of the regions is associated with one and only one of the classes. The surface (and let us stay here in the simple case of the two-class problem with a single function) that is realized in the input space is known as decision surface. In this paper, we will consider functions of the same form as in (17.2). Hence, once more, designing a classifier is cast as a parameter estimation task. Once estimates of the unknown parameters are computed, say,  $\hat{\theta}$ , given a feature vector  $\mathbf{x}$ , which results from a pattern whose class label,  $y$ , is unknown, the predicted label is obtained as

$$\hat{y} = g(f(\mathbf{x})), \quad (17.5)$$

where  $g(\cdot)$  is a nonlinear indicator function, which indicates on which side of  $f(x) = 0$  the pattern  $x$  lies; typically  $g(\cdot) = \text{sign}(\cdot)$ , that is, the sign function. Figure 17.2a shows examples of two classifiers. In Figure 17.2 the classifier is linear, i.e.,

$$f(\mathbf{x}) = x_1 - 3x_2 = 0,$$

**FIGURE 17.2**

Two examples of classifiers.

and in Figure 17.2b is a nonlinear one

$$f(\mathbf{x}) = -0.23 + \phi_1(\mathbf{x}) + \phi_2(\mathbf{x}) = 0,$$

where

$$\phi_1(\mathbf{x}) = \mathbf{x}^T \begin{bmatrix} 0.61 \\ -4.64 \end{bmatrix}, \quad \phi_2(\mathbf{x}) = \mathbf{x}^T \begin{bmatrix} -0.45 & 0.93 \\ 0.93 & -0.45 \end{bmatrix} \mathbf{x}.$$

Note that, as it is apparent from both examples, the classes cannot, in general, be completely separated by the classifiers and errors in predicting the class labels are unavoidable. Tasks with separable classes are possible to occur, yet this is rather the exception than the rule.

Regression and Classification are two typical Machine Learning tasks, and a wide range of learning applications can be formulated as either one of the two. Moreover, we discussed that both tasks can be cast as parameter estimations problems, which comprise the focus of the current paper. Needless to say that both problems can also be attacked via alternative paths. For example, the  $k$ -nearest neighbor method is another possibility, which does not involve any parameter estimation phase; the only involved parameter,  $k$ , is determined via cross validation, e.g., [2].

Let us now put our parameter estimation task in a more formal setting. We are given a set of training points  $(y_n, \mathbf{x}_n) \in \mathcal{R} \times \mathcal{R}^l$ . For classification tasks, we replace  $\mathcal{R}$  with  $D$ . We adopt a parametric class of functions,

$$\mathcal{F} = \left\{ f_{\boldsymbol{\theta}}(\cdot) : \boldsymbol{\theta} \in \mathcal{A} \subseteq \mathcal{R}^K \right\}. \quad (17.6)$$

Our goal is to find a function in  $\mathcal{F}$ , denoted as  $f_{\boldsymbol{\theta}_*}(\cdot)$ , such that, given a measured value of  $\mathbf{x}$  and an adopted prediction model (e.g., (17.4) for regression or (17.5) for classification), it approximates the respective output value,  $\hat{y}$ , in an optimal way. Obviously, the word optimal paves the way to look for an optimizing criterion. To this end, a *loss function* is selected, from an available palette of non-negative functions,

$$\mathcal{L} : \mathcal{R} \times \mathcal{R} \rightarrow [0, \infty), \quad (17.7)$$

and compute  $\boldsymbol{\theta}_*$  so that to minimize the total loss over all the training data points, i.e.,

$$f_{\boldsymbol{\theta}_*}(\cdot) : \boldsymbol{\theta}_* \in \underset{\boldsymbol{\theta} \in \mathcal{A}}{\operatorname{argmin}} J(\boldsymbol{\theta}), \quad (17.8)$$

where

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n)), \quad (17.9)$$

assuming that a minimum exists. In an adaptive setting, minimizing the cost function takes place iteratively in the form

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \text{error correction term}, \quad (17.10)$$

as  $n$  increases. Different class of functions  $\mathcal{F}$  and different loss functions result in different estimators. In practice, for fixed number of training points, one uses cross validation to determine the “best” choice. Of course, in practice, when adaptive techniques are used and  $n$  is left to vary, cross validation techniques loose their meaning, especially when the statistical properties of the environment are slowly time varying. In such cases, the choice of the class of functions as well as the loss functions, which now has to be minimized adaptively, are empirically chosen. The critical factors, as we will see, for such choices are (a) *computational complexity*, (b) *convergence speed*, (c) *tracking agility*, (d) *robustness to noise and to numerical error accumulation*. In this paper, a number of different class functions as well as loss functions will be considered and discussed.

### 1.17.3 Overfitting and regularization

A problem of major importance in any Machine Learning task is that of *overfitting*. This is related to the complexity of the adopted model. If the model is too complex, with respect to the number of training points,  $N$ , then it tends to learn too much from the specific training set on which it is trained. For example, in regression, it tries to learn not only the useful information associated with the input data but also the contribution of the noise. So, if the model is too complex, it tends to fit very well the data points of the training set, but it does not cope well with data outside the training set; as we say, it cannot *generalize* well. In classification, the source of overfitting lies in the class overlap. A complex classifier will attempt to classify correctly all the points of the training set, and this is quite common in practice when very complex classifiers are employed. However, most of the class discriminatory information lies in the points lying in the “non-overlapping” regions. This is the information that a classifier has to learn in order to be able to generalize well, when faced with data outside the training set. A classifier which is designed to give very low error rates on the training set, usually results in large errors when faced with unseen data.

On the other hand, if our model is too simple it does not have the ability to learn even the useful information, which is contributed by the input data themselves, hence leading to very poor predictions. In practice, one seeks for a tradeoff, see, e.g., [2]. Figure 17.3 shows three cases of fitting by a very simple, a very complex and the correct model (that corresponds to the model which generated the data points). The first one is a linear model, the second one corresponds to a polynomial of high degree, i.e., 80, and the third one to a quadratic polynomial. Note that for the case of Figure 17.3b, the resulting

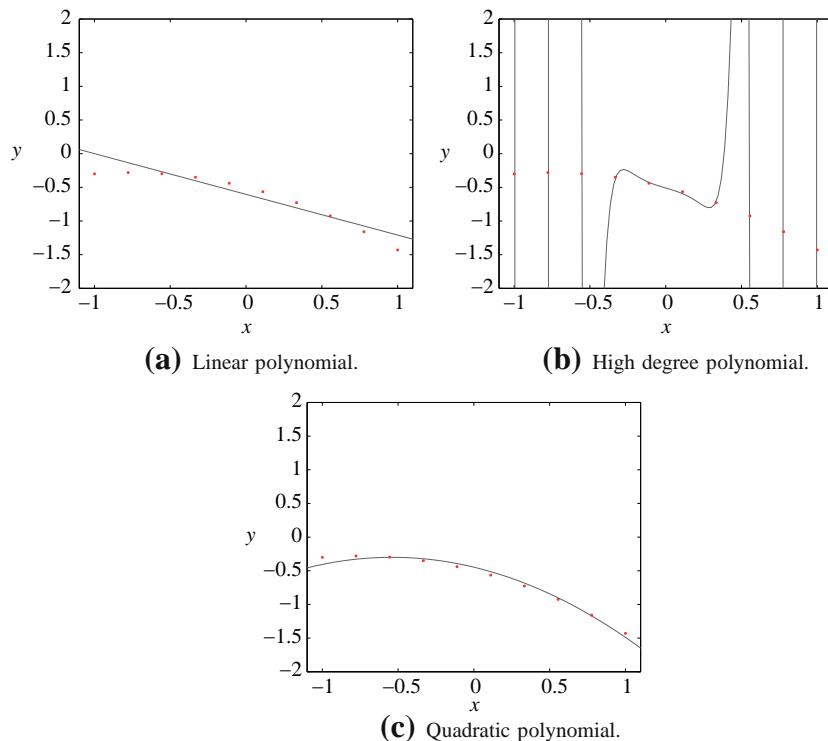


FIGURE 17.3

## Fitting noisy data by polynomials of various degrees.

curve passes through all the points. However, this is not what we want, since the points we have are the result of a noisy process, and the curve “follows” a path which is the combined result of the useful as well as the noise-contributed information. In practice, the performance of the resulting estimator is measured as the misfit, using a loss criterion (e.g., mean squared error, probability of error) between the predicted values and the true ones, over an independent set of data or using cross-validation techniques; for example, the resulting misfit is expected to be larger for the case of fitting the high order polynomial compared to the one corresponding to the quadratic polynomial of Figure 17.3c.

The reader may have noticed that we have avoided to define the term “complex” and we used it in a rather vague context. Complexity is often directly related to the number of the unknown parameters. However, this is not always the case. Complexity is a more general property and sometimes we may have classifiers in, even, infinite dimensional spaces, trained with a finite number of points, which generalize very well, [7].

A way to cope with the overfitting problem is to employ the *regularization* method. This is an elegant mathematical tool, that simultaneously cares to serve both desires; to keep the contribution of the loss function term as small as possible, since this accounts for the accuracy of the estimator, but at the same time to retain the fitted model as less complex as possible. This is achieved by adding a second term in

the cost function in (17.9), whose goal is to keep the norm of the parameters as small as possible, i.e.,

$$J(\boldsymbol{\theta}) := \sum_{n=1}^N \mathcal{L}(y_n, f_{\boldsymbol{\theta}}(\mathbf{x}_n)) + \lambda \Omega(\|\boldsymbol{\theta}\|), \quad (17.11)$$

where  $\Omega(\cdot)$  is a strictly increasing monotonic non-negative function and  $\|\cdot\|$  is a norm, e.g., the Euclidean ( $l_2$ ) or the  $l_1$  norms. The parameter  $\lambda$  is known as the regularization parameter and it controls the relative significance of the two terms in the cost function. For fixed  $N$ ,  $\lambda$  is usually determined via cross-validation. In a time varying adaptive mode of operation, when cross-validation is not possible, then empirical experience comes to assist or some adhoc techniques can be mobilized, to make  $\lambda = \lambda_N$ , i.e., time varying. Take as an example the case where the environment does not change and the only reason that we use an adaptive algorithm is due to complexity reasons. Then, as  $N \rightarrow \infty$ , the value of  $\lambda$  should go to zero. For very large values of  $N$ , overfitting is no problem, since the complexity remains a small portion of the data and only the loss-related term in the cost function is sufficient. As a matter of fact, the regularization term, in this case, will only do harm by biasing the solution away from the optimum, this time without reason.

Very often,  $\Omega$  is chosen so that

$$\Omega(\|\boldsymbol{\theta}\|) = \sum_{k=0}^{K-1} |\theta_k|^2, \text{ or } \Omega(\|\boldsymbol{\theta}\|) = \sum_{k=0}^{K-1} |\theta_k|,$$

the latter being the  $l_1$  norm, which has been popularized in the context of sparsity-aware learning/compressed sensing. Note that, usually, the bias term is not included in the norm, since it can make the solution sensitive to the shifts of the origin. This can easily be checked out for the case of the Least Squares loss function

$$\mathcal{L}(y, f_{\boldsymbol{\theta}}(\mathbf{x})) = (y - \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}))^2.$$

In practice, very often, if the mean values are not zero, we use centered data by subtracting from  $\mathbf{x}$  its sample mean, and then we can neglect  $\theta_0$ . Besides the norm, other functions have also been used as regularizers, e.g., [8].

Using (17.2) and plugging it in the loss function-related term (17.11), then after minimization and due to the presence of the norm-related term, some of the components of  $\boldsymbol{\theta}$  are pushed towards zero. This pushing towards zero is more aggressive when the  $l_1$  norm is used, see, e.g., [9]. Obviously, the terms which are pushed to zero are those whose elimination affects the least the loss function term, which implicitly results in less complex models.

*Sparsification* is another term used to indicate that the adopted estimation method has the ability to push the least significant, from the accuracy point of view (as this is measured my the loss function-related term in the regularized cost function) to zero. This is a way to guard against overfitting. Regularization techniques gain in importance and become unavoidable within a certain class of nonlinear modeling.

Consider the following modeling,

$$\phi_k(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}_k), \quad k = 1, 2, \dots, N, \quad (17.12)$$

where

$$\kappa(\cdot, \cdot) : \mathcal{R}^l \times \mathcal{R}^l \rightarrow \mathcal{R}, \quad (17.13)$$

which, dressed with some properties, as we will soon see, is known as kernel. In other words, the expansion of our nonlinear function in (17.2) involves as many terms as our training data points, and each term in the expansion is related with one point from the training set, that is,

$$f(\cdot) = \sum_{k=1}^N \theta_k \kappa(\cdot, \mathbf{x}_k) + \theta_0. \quad (17.14)$$

Such models are met in the context of Support Vector Machines as well as in Relevance Vector Machines, [1, 2, 7]. As we will soon see, this class of models has a strong theoretical justification. In any case, such models demand drastic sparsification, otherwise they will suffer for strong overfitting. Moreover, besides overfitting, in the adaptive case, where  $N$  keeps increasing, such a model would soon become unmanageable. This paper will discuss a number of related sparsification techniques.

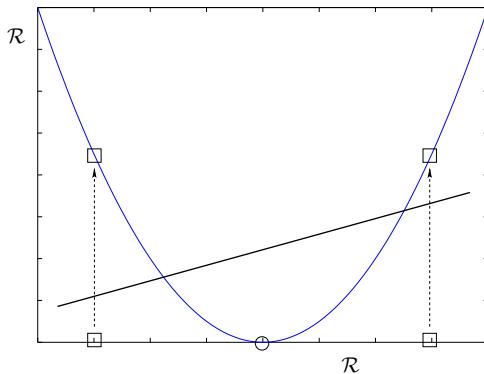
#### 1.17.4 Mapping a nonlinear to a linear task

Let us now look at the expansion in (17.2) in conjunction with our prediction models given in (17.4) for the regression and (17.5) for the classification tasks. Basically, all that such a modeling says is that if we map our original input  $l$ -dimensional space into a new  $K$ -dimensional space,

$$\mathcal{R}^l \ni \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathcal{R}^K, \quad (17.15)$$

we expect our task to be sufficiently well modeled by adopting a linear model in this new space. It is only up to us to choose the proper functions,  $\phi_k(\cdot)$ , as well as the appropriate dimensionality of the new space,  $K$ . For the classification task, such a procedure has a strong theoretical support from *Cover's theorem*, which states that: Consider  $N$  randomly located points in an  $l$ -dimensional space and use a mapping to map these points into a higher dimensional space. Then the probability of locating them, in the new space, in linearly separable groupings tends to one as the dimensionality,  $K$ , of the new space tends to infinity. Figure 17.4 illustrates how an appropriate mapping can transform a nonlinearly separable task to a linearly separable one. Notice that nature cannot be fooled. After the mapping in the higher 2-dimensional space, the points continue to live in an  $l$ -dimensional manifold [10] (paraboloid in this case). The mapping gives the opportunity to our  $l$ -dimensional points to take advantage of the freedom, which a higher dimensional space provides, so that to move around and be placed in linearly separable positions, while retaining their original  $l$ -dimensional (true degrees of freedom) nature; a paraboloid in the 2-dimensional space can be fully described in terms of one free parameter.

Although such a method sounds great, it has its practical drawbacks. To transform a problem into a linear one, we may have to map the original space to a very high dimensional one, and this can lead to a huge number of unknown parameters to be estimated. However, the mathematicians have, once more, offered to us an escape route that has been developed many years ago. The magic name is *Reproducing Kernel Hilbert Spaces*. Choosing to map to such spaces, and some of them can even be of

**FIGURE 17.4**

Transforming linearly non-separable learning tasks to separable ones by using nonlinear mappings to higher dimensional spaces. The original data, i.e., two “squares” and a “circle,” are placed on the 1-dimensional space,  $\mathcal{R}$ , rendering thus the linear classification task intractable. However, by using the polynomial function  $\Phi(x) := x^2, \forall x \in \mathcal{R}$ , we are able to map the original data to a manifold which lies in the 2-dimensional space  $\mathcal{R}^2$ . The mapped data can be now easily linearly separated, as this can be readily verified by the linear classifier which is illustrated by the straight, black, solid line of the figure.

infinite dimension, makes computations much easier; moreover, there is an associated strong theorem that justifies the expansion in (17.14). Assume that we have selected a mapping

$$\mathcal{R}^l \ni \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathcal{H}, \quad (17.16)$$

where  $\mathcal{H}$  is RKHS. Then, the following statements are valid:

- *The Kernel Trick:* Let  $\mathbf{x}_1, \mathbf{x}_2$  be two points in our original  $l$ -dimensional space and  $\phi(\mathbf{x}_1), \phi(\mathbf{x}_2)$  their respective images in an RKHS,  $\mathcal{H}$ . Since  $\mathcal{H}$  is a Hilbert space, an inner product,  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ , is associated with it, by the definition of Hilbert spaces. As we will more formally see in the next section, the inner product of any RKHS is uniquely defined in terms of the so called kernel function,  $\kappa(\cdot, \cdot)$  and the following holds true

$$\langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}_1, \mathbf{x}_2). \quad (17.17)$$

This is a remarkable property, since it allows to compute inner products in a (high dimensional) RKHS as a function operating in the lower dimensional input space. Also, this property allows the following “black box” approach, that has extensively been exploited in Machine Learning:

- Design an algorithm, which operates in the input space and computes the estimates of the parameters of a linear modeling.
- Cast this algorithm, if of course this is possible, in terms of inner product operations *only*.
- Replace each inner product with a kernel operation.

Then the resulting algorithm is equivalent with solving the (linear modeling) estimation task in an RKHS, which is defined by the selected kernel. Different kernel functions correspond to different RKHSs and, consequently, to different nonlinearities; each inner product in the RKHS is a nonlinear function operation in the input space. The kernel trick was first used in [11, 12].

- *Representer Theorem:* Consider the cost function in (17.11), and constrain the solutions to lie within an RKHS space, defined by a kernel function  $\kappa(\cdot, \cdot)$ . Then the minimizer of (17.11) is of the form

$$f(\cdot) = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n) + \theta_0. \quad (17.18)$$

This makes crystal clear the reason for the specific modeling in (17.14). In other words, if we choose to attack a nonlinear modeling by “transforming” it to a linear one, after performing a mapping of our data into a RKHS in order to exploit the advantage of efficient inner product operations, then the (linear with respect to the parameters) modeling in (17.14) results naturally from the theory. Our only concern now is to use an appropriate regularization, in order to cope with overfitting problems.

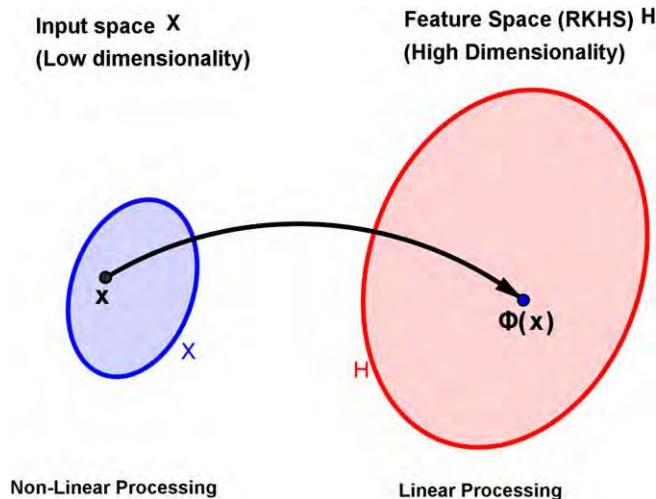
All the previously presented issues will be discussed in more detail in the sections to come. Our goal, so far, was to try to give the big picture of the paper in simple terms and focus more on the notions than on mathematics. We will become more formal in the sections to come. The reader, however, may bypass some of the sections, if his/her current focus is not on the mathematical details but on the algorithms. As said already before, we have made an effort to write the paper in such a way, so that to serve both needs: mathematical rigor and practical flavor, as much as we could.

## 1.17.5 Reproducing kernel Hilbert spaces

In kernel-based methods, the notion of the Reproducing Kernel Hilbert Space (RKHS) plays a crucial role. A RKHS is a rich construct (roughly, a space of functions with an inner product), which has been proven to be a very powerful tool. Kernel-based methods are utilized in an increasingly large number of scientific areas, especially where nonlinear models are required. For example, in pattern analysis, a classification task of a set  $X \subset \mathcal{R}^l$  is usually reformed by mapping the data into a higher dimensional space (possibly of infinite dimension)  $\mathcal{H}$ , which is a Reproducing Kernel Hilbert Space (RKHS). The advantage of such a mapping is to make the task more tractable, by employing a linear classifier in the feature space  $\mathcal{H}$ , exploiting Cover’s theorem (see [2, 13]). This is equivalent with solving a nonlinear problem in the original space.

Similar approaches have been used in principal components analysis, in Fisher’s linear discriminant analysis, in clustering, regression, image processing and in many other subdisciplines. Recently, processing in RKHS is gaining in popularity within the Signal Processing community in the context of adaptive learning.

Although the kernel trick works well for most applications, it may conceal the basic mathematical steps that underlie the procedure, which are essential if one seeks a deeper understanding of the problem. These steps are: 1) Map the finite dimensionality input data from the input space  $X$  (usually  $X \subset \mathcal{R}^l$ ) into a higher dimensionality (possibly infinite) RKHS  $\mathcal{H}$  (feature space) and 2) Perform a linear processing (e.g., adaptive filtering) on the mapped data in  $\mathcal{H}$ . The procedure is equivalent with a nonlinear processing

**FIGURE 17.5**

Mapping from input space  $X$  to feature space  $H$ .

(nonlinear filtering) in  $X$  (see Figure 17.5). The specific choice of the kernel  $\kappa$  defines, implicitly, an RKHS with an appropriate inner product. Moreover, the specific choice of the kernel defines the type of nonlinearity that underlies the model to be used.

### 1.17.5.1 A historical overview

In the past, there have been two trends in the study of these spaces by the mathematicians. The first one originated in the theory of integral equations by Mercer [14, 15]. He used the term “positive definite kernel” to characterize a function of two points  $\kappa(\mathbf{x}, \mathbf{y})$  defined on  $X^2$ , which satisfies Mercer’s theorem:

$$\sum_{n,m=1}^N a_n a_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \geq 0, \quad (17.19)$$

for any numbers  $a_n, a_m$ , any points  $\mathbf{x}_n, \mathbf{x}_m, n = 1, \dots, N$ , and for all  $N \in \mathcal{N}_*$ . Later on, Moore [16–18] found that to such a kernel there corresponds a well determined class of functions,  $\mathcal{H}$ , equipped with a specific inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ , in respect to which the kernel  $\kappa$  possesses the so called reproducing property:

$$f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}, \quad (17.20)$$

for all functions  $f \in \mathcal{H}$  and  $\mathbf{x} \in X$ . Those that followed this trend used to consider a specific given positive definite kernel  $\kappa$  and studied it in itself, or eventually applied it in various domains (such as integral equations, theory of groups, general metric theory, interpolation, etc.). The class  $\mathcal{H}$  corresponding to  $\kappa$  was mainly used as a tool of research and it was usually introduced a posteriori. The work of Bochner

[19, 20], which introduced the notion of the “positive definite function” in order to apply it in the theory of Fourier transforms, also belongs to the same path as the one followed by Mercer and Moore.

On the other hand, those who followed the second trend were primarily interested in the class of functions  $\mathcal{H}$ , while the associated kernel was employed essentially as a tool in the study of the functions of this class. This trend is traced back to the works of Zaremba [21, 22] during the first decade of the 20th century. He was the first to introduce the notion of a kernel, which corresponds to a specific class of functions and to state its reproducing property. However, he did not develop any general theory, nor did he give any particular name to the kernels he introduced. In this, second trend, the mathematicians were primarily interested in the study of the class of functions  $\mathcal{H}$  and the corresponding kernel  $\kappa$ , which satisfies the reproducing property, was used as a tool in this study. To the same trend belong also the works of Bergman [23] and Aronszajn [3]. Those two trends evolved separately during the first decades of the 20th century, but soon the links between them were noticed. After the second world war, it was known that the two concepts of defining a kernel, either as a positive definite kernel, or as a reproducing kernel, are equivalent. Furthermore, it was proved that there is a one to one correspondence between the space of positive definite kernels and the space of reproducing kernel Hilbert spaces.

It has to be emphasized that examples of such kernels have been known for a long time prior to the works of Mercer and Zaremba; for example, all the Green’s functions of self-adjoint ordinary differential equations belong to this type of kernels. However, some of the important properties that these kernels possess have only been realized and used in the beginning of the 20th century and since then have been the focus of research. In the following, we will give a more detailed description of these spaces and establish their main properties, focusing on the essentials that elevate them to such a powerful tool in the context of machine learning. Most of the material presented here can also be found in more detail in several other papers and textbooks, such as the celebrated paper of Aronszajn [3], the excellent introductory text of Paulsen [24] and the popular books of Schölkopf and Smola [13] and Shawe-Taylor and Cristianini [25]. Here, we attempt to portray both trends and to highlight the important links between them. Although the general theory applies to complex spaces, to keep the presentation as simple as possible, we will mainly focus on real spaces.

### 1.17.5.2 Definition

We begin our study with the classic definitions on positive definite matrices and kernels as they were introduced by Mercer. Given a function  $\kappa : X \times X \rightarrow \mathcal{R}$  and  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$  (typically  $X$  is a compact subset of  $\mathcal{R}^l$ ,  $l > 0$ ), the square matrix  $\mathbf{K} = (K_{n,m})$  with elements  $K_{n,m} = \kappa(\mathbf{x}_n, \mathbf{x}_m)$ , for  $n, m = 1, \dots, N$ , is called the *Gram matrix* (or *kernel matrix*) of  $\kappa$  with respect to  $\mathbf{x}_1, \dots, \mathbf{x}_N$ . A symmetric matrix  $\mathbf{K} = (K_{n,m})$  satisfying

$$\mathbf{c}^T \cdot \mathbf{K} \cdot \mathbf{c} = \sum_{n=1, m=1}^N c_n c_m K_{n,m} \geq 0,$$

for all  $\mathbf{c} \in \mathcal{R}^N$ ,  $n = 1, \dots, N$ , where the notation  $(\cdot)^T$  denotes transposition, is called *positive definite*. In matrix analysis literature, this is the definition of a positive semidefinite matrix. However, as positive definite matrices were originally introduced by Mercer and others in this context, we employ the term positive definite, as it was already defined. If the inequality is strict, for all non-zero vectors  $\mathbf{c} \in \mathcal{R}^N$ , the

matrix will be called *strictly positive definite*. A function  $\kappa : X \times X \rightarrow \mathcal{R}$ , which for all  $N \in \mathcal{N}_*$  and all  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$  gives rise to a positive definite Gram matrix  $K$ , is called a *positive definite kernel*. In the following, we will frequently refer to a positive definite kernel simply as *kernel*. We conclude that a positive definite kernel is symmetric and satisfies

$$\sum_{n=1, m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \geq 0,$$

for all  $c \in \mathcal{R}^N$ ,  $n = 1, \dots, N$ ,  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$ , and for all  $N \in \mathcal{N}_*$ . Formally, a Reproducing Kernel Hilbert space is defined as follows:

**Definition 1 (Reproducing Kernel Hilbert Space (RKHS)).** Consider a linear space  $\mathcal{H}$  of real valued functions,  $f$ , defined on a set  $X$ . Suppose, further, that in  $\mathcal{H}$  we can define an inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  with corresponding norm  $\|\cdot\|_{\mathcal{H}}$  and that  $\mathcal{H}$  is complete with respect to that norm, i.e.,  $\mathcal{H}$  is a Hilbert space. We call  $\mathcal{H}$  a *Reproducing Kernel Hilbert Space (RKHS)*, if there exists a function  $\kappa : X \times X \rightarrow \mathcal{R}$  with the following two important properties:

1. For every  $\mathbf{x} \in X$ ,  $\kappa(\cdot, \mathbf{x})$  belongs to  $\mathcal{H}$  (or equivalently  $\kappa$  spans  $\mathcal{H}$ , i.e.,  $\mathcal{H} = \overline{\text{span}\{\kappa(\cdot, \mathbf{x}), \mathbf{x} \in X\}}$ , where the overline stands for the closure of a set).
2.  $\kappa$  has the so called *reproducing property*, i.e.,

$$f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}, \text{ for all } f \in \mathcal{H}, \mathbf{x} \in X, \quad (17.21)$$

in particular  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \kappa(\cdot, \mathbf{y}), \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}$ . Furthermore,  $\kappa$  is a positive definite kernel and the mapping  $\Phi : X \rightarrow \mathcal{H}$ , with  $\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$ , for all  $\mathbf{x} \in X$  is called the feature map of  $\mathcal{H}$ .  $\diamond$

To denote the RKHS associated with a specific kernel  $\kappa$ , we will also use the notation  $\mathcal{H}(\kappa)$ . Furthermore, under the aforementioned notations  $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{y}), \Phi(\mathbf{x}) \rangle_{\mathcal{H}}$ , i.e.,  $\kappa(\mathbf{x}, \mathbf{y})$  is the inner product of  $\Phi(\mathbf{y})$  and  $\Phi(\mathbf{x})$  in the feature space. This is the essence of the kernel trick mentioned in Section 1.17.4. The feature map  $\Phi$  transforms the data from the low dimensionality space  $X$  to the higher dimensionality feature space  $\mathcal{H}$ . Linear processing in  $\mathcal{H}$  involves inner products in  $\mathcal{H}$ , which can be calculated via the kernel  $\kappa$  disregarding the actual structure of  $\mathcal{H}$ .

### 1.17.5.3 Some basic theorems

In the following, we consider the definition of a RKHS as a class of functions with specific properties (following the second trend) and show the key ideas that underlie Definition 1. To that end, consider a linear class  $\mathcal{H}$  of real valued functions,  $f$ , defined on a set  $X$ . Suppose, further, that in  $\mathcal{H}$  we can define an inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  with corresponding norm  $\|\cdot\|_{\mathcal{H}}$  and that  $\mathcal{H}$  is complete with respect to that norm, i.e.,  $\mathcal{H}$  is a Hilbert space. Consider, also, a *linear* functional  $T$ , from  $\mathcal{H}$  into the field  $\mathcal{R}$ . An important theorem of functional analysis states that such a functional is *continuous*, if and only if it is *bounded*. The space consisting of all continuous linear functionals from  $\mathcal{H}$  into the field  $\mathcal{R}$  is called the *dual space* of  $\mathcal{H}$ . In the following, we will frequently refer to the so called *linear evaluation functional*  $T_y$ . This is a special case of a linear functional that satisfies  $T_y(f) = f(y)$ , for all  $f \in \mathcal{H}$ .

We call  $\mathcal{H}$  a *Reproducing Kernel Hilbert Space (RKHS)* on  $X$  over  $\mathcal{R}$ , if for every  $y \in X$ , the linear evaluation functional,  $T_y$ , is continuous. We will prove that such a space is related to a positive definite kernel, thus providing the first link between the two trends. Subsequently, we will prove that any positive definite kernel defines implicitly a RKHS, providing the second link and concluding the equivalent definition of RKHS (Definition 1), which is usually used in the machine learning literature. The following theorem establishes an important connection between a Hilbert space  $H$  and its dual space.

**Theorem 2 (Riesz Representation).** *Let  $\mathcal{H}$  be a general Hilbert space and let  $\mathcal{H}^*$  denote its dual space. Every element  $\Phi$  of  $\mathcal{H}^*$  can be uniquely expressed in the form:*

$$\Phi(f) = \langle f, \phi \rangle_{\mathcal{H}},$$

for some  $\phi \in \mathcal{H}$ . Moreover,  $\|\Phi\|_{\mathcal{H}^*} = \|\phi\|_{\mathcal{H}}$ .  $\diamond$

Following the Riesz representation theorem, we have that for every  $y \in X$ , there exists a *unique* element  $\kappa_y \in \mathcal{H}$ , such that for every  $f \in \mathcal{H}$ ,  $f(y) = T_y(f) = \langle f, \kappa_y \rangle_{\mathcal{H}}$ . The function  $\kappa_y$  is called the reproducing kernel for the point  $y$  and the function  $\kappa(x, y) = \kappa_y(x)$  is called the *reproducing kernel* of  $\mathcal{H}$ . In addition, note that  $\langle \kappa_y, \kappa_x \rangle_{\mathcal{H}} = \kappa_y(x) = \kappa(x, y)$  and  $\|T_y\|_{\mathcal{H}^*}^2 = \|\kappa_y\|_{\mathcal{H}}^2 = \langle \kappa_y, \kappa_y \rangle_{\mathcal{H}} = \kappa(y, y)$ .

**Proposition 3.** *The reproducing kernel of  $\mathcal{H}$  is symmetric, i.e.,  $\kappa(x, y) = \kappa(y, x)$ .*  $\diamond$

**Proof.** Observe that  $\langle \kappa_y, \kappa_x \rangle_{\mathcal{H}} = \kappa_y(x) = \kappa(x, y)$  and  $\langle \kappa_x, \kappa_y \rangle_{\mathcal{H}} = \kappa_x(y) = \kappa(y, x)$ . As the inner product of  $\mathcal{H}$  is symmetric (i.e.,  $\langle \kappa_y, \kappa_x \rangle_{\mathcal{H}} = \langle \kappa_x, \kappa_y \rangle_{\mathcal{H}}$ ) the result follows.  $\square$

In the following, we will frequently identify the function  $\kappa_y$  with the notation  $\kappa(\cdot, y)$ . Thus, we write the reproducing property of  $\mathcal{H}$  as:

$$f(y) = \langle f, \kappa(\cdot, y) \rangle_{\mathcal{H}}, \quad (17.22)$$

for any  $f \in \mathcal{H}$ ,  $y \in X$ . Note that due to the uniqueness provided by the Riesz representation theorem,  $\kappa$  is the unique function that satisfies the reproducing property. The following proposition establishes the first link between the positive definite kernels and the reproducing kernels.

**Proposition 4.** *The reproducing kernel of  $\mathcal{H}$  is a positive definite kernel.*  $\diamond$

**Proof.** Consider  $N > 0$ , the real numbers  $a_1, a_2, \dots, a_N$  and the elements,  $x_1, x_2, \dots, x_N \in X$ . Then

$$\begin{aligned} \sum_{n=1}^N \sum_{m=1}^N a_n a_m \kappa(x_n, x_m) &= \sum_{n=1}^N \sum_{m=1}^N a_n a_m \langle \kappa(\cdot, x_m), \kappa(\cdot, x_n) \rangle_{\mathcal{H}} = \sum_{n=1}^N a_n \left\langle \sum_{m=1}^N a_m \kappa(\cdot, x_m), \kappa(\cdot, x_n) \right\rangle_{\mathcal{H}} \\ &= \left\langle \sum_{m=1}^N a_m \kappa(\cdot, x_m), \sum_{n=1}^N a_n \kappa(\cdot, x_n) \right\rangle_{\mathcal{H}} = \left\| \sum_{n=1}^N a_n \kappa(\cdot, x_n) \right\|_{\mathcal{H}}^2 \geq 0. \end{aligned}$$

Combining Proposition 3 and the previous result, we complete the proof.  $\square$

**Remark 5.** In general, the respective Gram matrix associated with a reproducing kernel is strictly positive definite. For if not, then there must exist at least one non zero vector  $a$  such that

$\left\| \sum_{n=1}^N a_n \kappa(\cdot, \mathbf{x}_n) \right\|_{\mathcal{H}}^2 = 0$ , for some finite set of points  $x_1, x_2, \dots, x_N$ . Hence, for every  $f \in \mathcal{H}$ , we have that  $\sum_n a_n f(\mathbf{x}_n) = \langle f, \sum_n a_n \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = 0$ . Thus, in this case there exists a linear dependence between the values of *every* function in  $\mathcal{H}$  at some finite set of points. Such examples do exist (e.g., Sobolev spaces), but in most cases the reproducing kernels define Gram matrices that are always strictly positive and invertible.

The following proposition establishes a very important fact; any RKHS,  $\mathcal{H}$ , can be generated by the respective reproducing kernel  $\kappa$ . Note that the overline denotes the closure of a set (i.e., if  $A$  is a subset of  $\mathcal{H}$ ,  $\overline{A}$  is the closure of  $A$ ).

**Proposition 6.** *Let  $\mathcal{H}$  be a RKHS on the set  $X$  with reproducing kernel  $\kappa$ . Then the linear span of the functions  $\kappa(\cdot, \mathbf{x})$ ,  $\mathbf{x} \in X$ , is dense in  $\mathcal{H}$ , i.e.,  $\mathcal{H} = \overline{\text{span}\{\kappa(\cdot, \mathbf{x}), \mathbf{x} \in X\}}$ .*  $\diamond$

**Proof.** We will prove that the only function of  $\mathcal{H}$  orthogonal to  $A = \text{span}\{\kappa(\cdot, \mathbf{x}), \mathbf{x} \in X\}$  is the zero function. Let  $f$  be such a function. Then, as  $f$  is orthogonal to  $A$ , we have that  $f(\mathbf{x}) = \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = 0$ , for every  $\mathbf{x} \in X$ . This holds true if and only if  $f = 0$ . Thus  $A^\perp = \overline{A^\perp} = \{0\}$ . Suppose that there is  $f \in \mathcal{H}$  such that  $f \notin \overline{A}$ . As  $\overline{A}$  is a closed (convex) subspace of  $\mathcal{H}$ , there is a  $g \in \overline{A}$  which minimizes the distance between  $f$  and points in  $\overline{A}$  (theorem of best approximation). For the same  $g$  we have that  $f - g \perp \overline{A}$ . Thus, the non-zero function  $h = f - g$  is orthogonal to  $\overline{A}$ . However, we proved that there isn't any non-zero vector orthogonal to  $A$ . This leads us to conclude that  $\overline{A} = \mathcal{H}$ .  $\square$

In the following we give some important properties of the specific spaces.

**Proposition 7 (Norm convergence implies point-wise convergence).** *Let  $\mathcal{H}$  be a RKHS on  $X$  and let  $\{f_n\}_{n \in \mathcal{N}_*} \subseteq \mathcal{H}$ . If  $\lim_n \|f_n - f\|_{\mathcal{H}} = 0$ , then  $f(\mathbf{x}) = \lim_n f_n(\mathbf{x})$ , for every  $\mathbf{x} \in X$ . Conversely, if for any sequence  $\{f_n\}_{n \in \mathcal{N}_*}$  of a Hilbert space  $\mathcal{H}$ , such that  $\lim_n \|f_n - f\|_{\mathcal{H}} = 0$ , we have also that  $f(\mathbf{x}) = \lim_n f_n(\mathbf{x})$ , then  $\mathcal{H}$  is a RKHS.*  $\diamond$

**Proof.** For every  $\mathbf{x} \in X$  we have that

$$|f_n(\mathbf{x}) - f(\mathbf{x})|_{\mathcal{H}} = |\langle f_n, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} - \langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}| = |\langle f_n - f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}}| \leq \|f_n - f\|_{\mathcal{H}} \cdot \|\kappa(\cdot, \mathbf{x})\|_{\mathcal{H}}.$$

As  $\lim_n \|f_n - f\| = 0$ , we have that  $\lim_n |f_n(\mathbf{x}) - f(\mathbf{x})| = 0$ , for every  $\mathbf{x} \in X$ . Hence  $f(\mathbf{x}) = \lim_n f_n(\mathbf{x})$ , for every  $\mathbf{x} \in X$ .

For the converse, consider the evaluation functional  $T_y : \mathcal{H} \rightarrow \mathcal{R}$ ,  $T_y(f) = f(y)$  for some  $y \in \mathcal{H}$ . We will prove that  $T_y$  is continuous for all  $y \in \mathcal{H}$ . To this end, consider a sequence  $\{f_n\}_{n \in \mathcal{N}_*}$  of  $\mathcal{H}$ , with the property  $\lim_n \|f_n - f\|_{\mathcal{H}} = 0$ , i.e.,  $f_n$  converges to  $f$  in the norm. Then  $|T_y(f_n) - T_y(f)| = |f_n(y) - f(y)| \rightarrow 0$ , as  $f(\mathbf{x}) = \lim_n f(\mathbf{x}) = \lim_n f_n(\mathbf{x})$ . Thus  $T_y(f) = \lim_n T_y(f_n)$  for all  $y \in X$  and all converging sequences  $\{f_n\}_{n \in \mathcal{N}_*}$  of  $\mathcal{H}$ .  $\square$

**Proposition 8 (Different RKHS's cannot have the same reproducing kernel).** *Let  $\mathcal{H}_1, \mathcal{H}_2$  be RKHS's on  $X$  with reproducing kernels  $\kappa_1, \kappa_2$ . If  $\kappa_1(\mathbf{x}, \mathbf{y}) = \kappa_2(\mathbf{x}, \mathbf{y})$ , for all  $\mathbf{x}, \mathbf{y} \in X$ , then  $\mathcal{H}_1 = \mathcal{H}_2$  and  $\|f\|_{\mathcal{H}_1} = \|f\|_{\mathcal{H}_2}$  for every  $f$ .*  $\diamond$

**Proof.** Let  $\kappa(\mathbf{x}, \mathbf{y}) = \kappa_1(\mathbf{x}, \mathbf{y}) = \kappa_2(\mathbf{x}, \mathbf{y})$  and  $A_i = \text{span}\{\kappa_i(\cdot, \mathbf{x}), \mathbf{x} \in X\}$ ,  $i = 1, 2$ . As shown in Proposition 6,  $\mathcal{H}_i = \overline{A_i}$ ,  $i = 1, 2$ . Note that for any  $f \in A_i$ ,  $i = 1, 2$ , we have that  $f(\mathbf{x}) = \sum_n a_n \kappa_i(\cdot, \mathbf{x}_n)$ , for some real numbers  $a_n$  and thus the values of the function are independent of whether we regard it as in  $A_1$  or  $A_2$ . Furthermore, for any  $f \in A_i$ ,  $i = 1, 2$ , as the two kernels are identical, we have that  $\|f\|_{\mathcal{H}_1}^2 = \sum_{n,m} a_n a_m \kappa(\mathbf{x}_m, \mathbf{x}_n) = \|f\|_{\mathcal{H}_2}^2$ . Thus,  $\|f\|_{\mathcal{H}_1} = \|f\|_{\mathcal{H}_2}$ , for all  $f \in A_1 = A_2$ .

Finally, we turn our attention to the limit points of  $A_1$  and  $A_2$ . If  $f \in \mathcal{H}_1$ , then there exists a sequence of functions,  $\{f_n\}_{n \in \mathcal{N}} \subseteq A_1$  such that  $\lim_n \|f - f_n\|_{\mathcal{H}_1} = 0$ . Since  $\{f_n\}_{n \in \mathcal{N}}$  is a converging sequence, it is Cauchy in  $A_1$  and thus it is also Cauchy in  $A_2$ . Therefore, there exists  $g \in \mathcal{H}_2$  such that  $\lim_n \|g - f_n\|_{\mathcal{H}_2} = 0$ . Employing proposition 7, we take that  $f(\mathbf{x}) = \lim_n f_n(\mathbf{x}) = g(\mathbf{x})$ . Thus, every  $f$  in  $\mathcal{H}_1$  is also in  $\mathcal{H}_2$  and by analogous argument we can prove that every  $g \in \mathcal{H}_2$  is also in  $\mathcal{H}_1$ . Hence,  $\mathcal{H}_1 = \mathcal{H}_2$  and as  $\|f\|_{\mathcal{H}_1} = \|f\|_{\mathcal{H}_2}$  for all  $f$  in a dense subset (i.e.,  $A_1$ ), we have that the norms are equal for every  $f$ . To prove the latter, we use the relation  $\lim_n \|f_n\|_{\mathcal{H}_i} = \|f\|_{\mathcal{H}_i}, i = 1, 2$ .  $\square$

The following theorem is the converse of Proposition 4. It was proved by Moore and it gives us a characterization of reproducing kernel functions. Also, it provides the second link between the two trends that have been mentioned in Section 1.17.5.1. Moore's theorem, together with Proposition 4, Proposition 8 and the uniqueness property of the reproducing kernel of a RKHS, establishes a one-to-one correspondence between RKHS's on a set and positive definite functions on the set.

**Theorem 9 (Moore [16]).** *Let  $X$  be a set and let  $\kappa : X \times X \rightarrow \mathcal{R}$  be a positive definite kernel. Then there exists a RKHS of functions on  $X$ , such that  $\kappa$  is the reproducing kernel of  $\mathcal{H}$ .*  $\diamond$

**Proof.** We will give only a sketch of the proof. The interested reader is referred to [24]. The first step is to define  $A = \text{span}\{\kappa(\cdot, \mathbf{x}), \mathbf{x} \in X\}$  and the linear map  $P : A \times A \rightarrow \mathcal{R}$  such that

$$P \left( \sum_m a_m \kappa(\cdot, \mathbf{y}_m), \sum_n b_n \kappa(\cdot, \mathbf{y}_n) \right) = \sum_{n,m} a_m b_n \kappa(\mathbf{y}_n, \mathbf{y}_m).$$

We prove that  $P$  is well defined and that it satisfies the properties of the inner product. Then, given the vector space  $A$  and the inner product  $P$ , one may complete the space by taking equivalence classes of Cauchy sequences from  $A$  to obtain the Hilbert space  $\mathcal{A}$ . Finally, the reproducing property of the kernel  $\kappa$  with respect to the inner product  $P$  is proved.  $\square$

In view of the aforementioned theorems, the Definition 1 of the RKHS given in 1.17.5.2, which is usually used in the machine learning literature, follows naturally.

We conclude this section with a short description of the most important points of the theory developed by Mercer in the context of integral operators. Mercer considered integral operators  $T_\kappa$  generated by a kernel  $\kappa$ , i.e.,  $T_\kappa : L_2(X) \rightarrow L_2(X)$ , such that  $(T_\kappa f)(\mathbf{x}) := \int_X \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}$ . He concluded the following theorem [14]:

**Theorem 10 (Mercer kernels are positive definite [14]).** *Let  $X \subseteq \mathcal{R}^l$  be a nonempty set and let  $\kappa : X \times X \rightarrow \mathcal{R}$  be continuous. Then  $\kappa$  is a positive definite kernel, if and only if,*

$$\int_{X^2} f(\mathbf{x}) \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0,$$

for all functions  $f \in L_2(X)$ . Moreover, if  $\kappa$  is positive definite, the integral operator  $T_\kappa : L_2(X) \rightarrow L_2(X) : (T_\kappa f)(\mathbf{x}) := \int_X \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}$  is positive definite, that is,

$$\langle T_\kappa f, f \rangle = \int_X f(\mathbf{y}) (T_\kappa f)(\mathbf{x}) d\mathbf{y} = \int_{X^2} \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0,$$

for all  $f \in L_2(X)$ , and if  $\psi_i \in L_2(X)$  are the normalized orthogonal eigenfunctions of  $T_\kappa$  associated with the eigenvalues  $\sigma_i > 0$  then:

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_i \sigma_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y}).$$

◇

Note that the original form of above theorem is more general, involving  $\sigma$ -algebras and probability measures. However, as in the applications concerning this manuscript such general terms are of no importance, we decided to include this simpler form. The previous theorems established that Mercer's kernels, as they are positive definite kernels, are also reproducing kernels. Furthermore, the first part of Theorem 10 provides a useful tool of determining whether a specific function is actually a reproducing kernel.

#### 1.17.5.4 Examples of kernels

Before proceeding to some more advanced topics in the theory of RKHS, it is important to give some examples of kernels that appear more often in the literature and are used in various applications. Perhaps the most widely used reproducing kernel is the Gaussian radial basis function defined on  $X \times X$ , where  $X \subseteq \mathcal{R}^l$ , as:

$$\kappa_\sigma(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right), \quad (17.23)$$

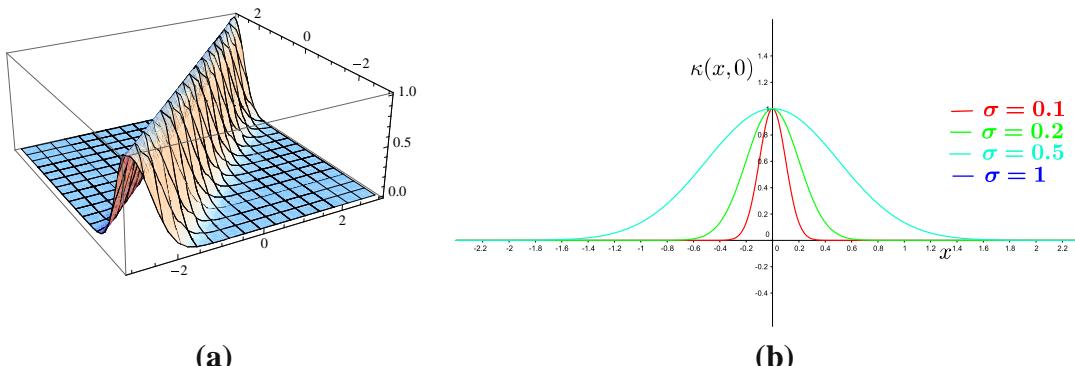
where  $\sigma > 0$ . Equivalently the Gaussian RBF function can be defined as:

$$\kappa_t(\mathbf{x}, \mathbf{y}) = \exp\left(-t\|\mathbf{x} - \mathbf{y}\|^2\right), \quad (17.24)$$

for  $t > 0$ .

Other well-known kernels defined in  $X \times X$ ,  $X \subseteq \mathcal{R}^l$  are:

- The homogeneous polynomial kernel:  $\kappa_d(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^d$ .
- The inhomogeneous polynomial kernel:  $\kappa_d(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^d$ , where  $c \geq 0$  a constant.



**FIGURE 17.6**

(a) The Gaussian kernel for the case  $X = \mathcal{R}$ ,  $\sigma = 0.5$ . (b) The element  $\Phi(0) = \kappa(\cdot, 0)$  of the feature space induced by the Gaussian kernel for various values of the parameter  $\sigma$ .

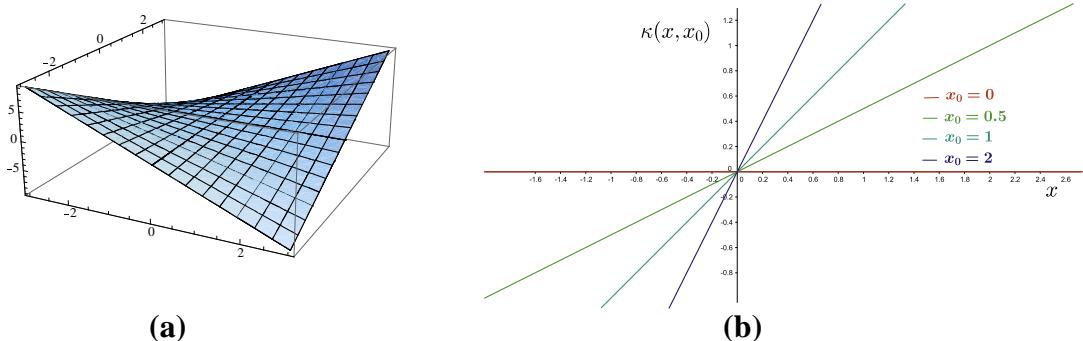


FIGURE 17.7

(a) The homogeneous polynomial kernel for the case  $X = \mathcal{R}$ ,  $d = 1$ . (b) The element  $\Phi(x_0) = \kappa(\cdot, x_0)$  of the feature space induced by the homogeneous polynomial kernel ( $d = 1$ ) for various values of  $x_0$ .

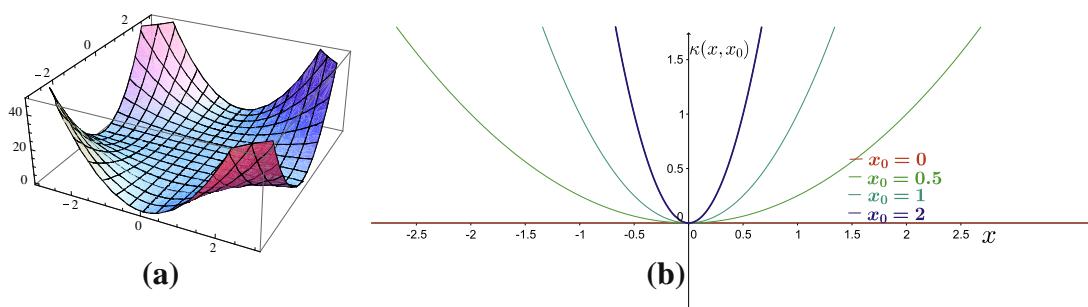


FIGURE 17.8

(a) The homogeneous polynomial kernel for the case  $X = \mathcal{R}$ ,  $d = 2$ . (b) The element  $\Phi(x_0) = \kappa(\cdot, x_0)$  of the feature space induced by the homogeneous polynomial kernel ( $d = 2$ ) for various values of  $x_0$ .

- The spline kernel:  $\kappa_p(\mathbf{x}, \mathbf{y}) = B_{2p+1}(\|\mathbf{x} - \mathbf{y}\|^2)$ , where  $B_n = \bigoplus_{i=1}^n I_{[-\frac{i}{2}, \frac{i}{2}]}$ .
- The cosine kernel:  $\kappa(\mathbf{x}, \mathbf{y}) = \cos(\angle(\mathbf{x}, \mathbf{y}))$ .
- The Laplacian kernel:  $\kappa_t(\mathbf{x}, \mathbf{y}) = \exp(-t\|\mathbf{x} - \mathbf{y}\|)$ .

Note that the RKHS associated to the Gaussian RBF kernel and the Laplacian kernel are infinite dimensional, while the RKHS associated to the polynomial kernels have finite dimension [13]. Figures 17.6–17.10, show some of the aforementioned kernels together with a sample of the elements  $\kappa(\cdot, \mathbf{x})$  that span the respective RKHS's for the case  $X = \mathcal{R}$ . Figures 17.11–17.13, show some of the elements  $\kappa(\cdot, \mathbf{x})$  that span the respective RKHS's for the case  $X = \mathcal{R}^2$ . Interactive figures regarding the aforementioned examples can be found in <http://bouboulis.mysch.gr/kernels.html>. Moreover, Appendix A highlights several properties of the popular Gaussian kernel.

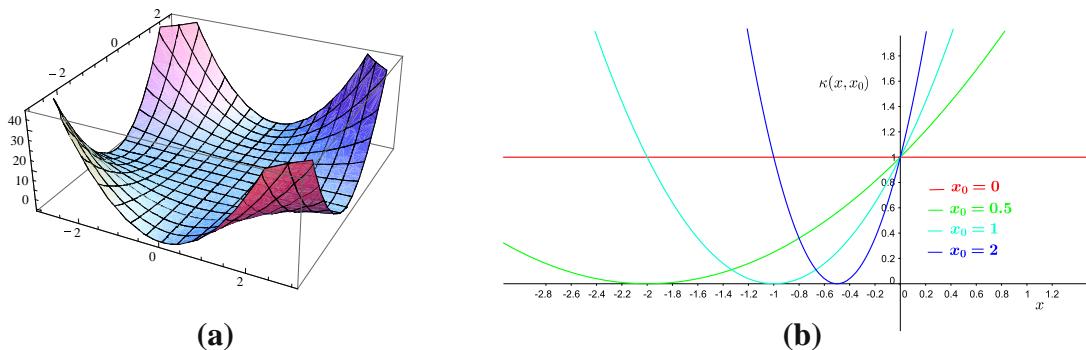


FIGURE 17.9

(a) The inhomogeneous polynomial kernel for the case  $X = \mathcal{R}$ ,  $d = 2$ . (b) The element  $\Phi(x_0) = \kappa(\cdot, x_0)$  of the feature space induced by the inhomogeneous polynomial kernel ( $d = 2$ ) for various values of  $x_0$ .

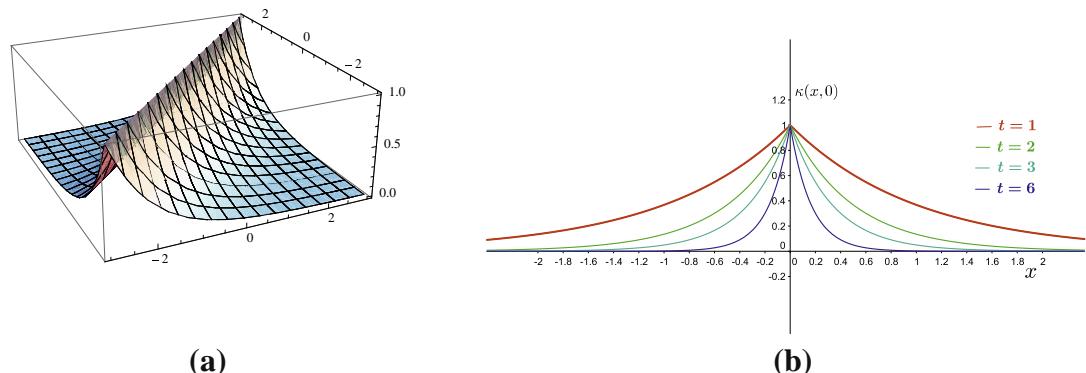
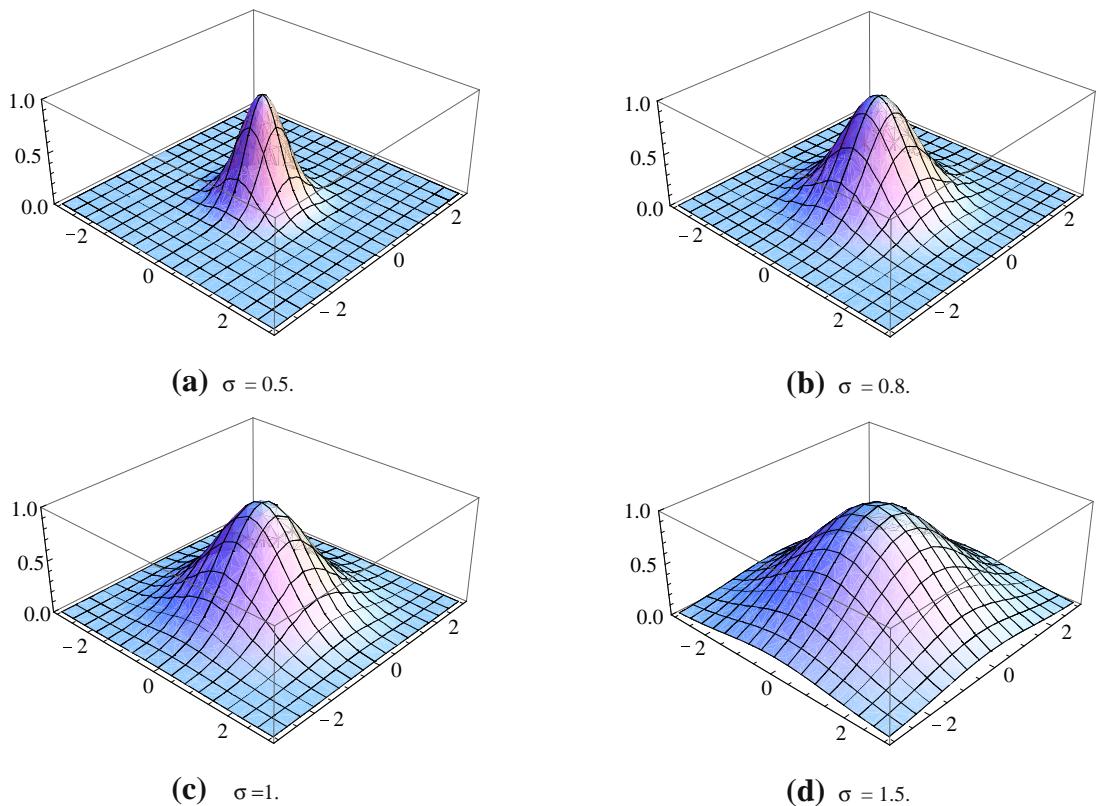


FIGURE 17.10

(a) The Laplacian kernel for the case  $X = \mathcal{R}$ ,  $t = 1$ . (b) The element  $\Phi(0) = \kappa(\cdot, 0)$  of the feature space induced by the Laplacian kernel for various values of the parameter  $t$ .

### 1.17.5.5 Some properties of the RKHS

In this section, we will refer to some more advanced topics on the theory of RKHS, which are useful for a deeper understanding of the underlying theory and show why RKHS's constitute such a powerful tool. We begin our study with some properties of RKHS's and conclude with the basic theorems that enable us to generate new kernels. As we work in Hilbert spaces, the two *Parseval's identities* are an extremely helpful tool. When  $\{e_s : s \in S\}$  (where  $S$  is an arbitrary set) is an orthonormal basis for a

**FIGURE 17.11**

The element  $\Phi(0) = \kappa(\cdot, 0)$  of the feature space induced by the Gaussian kernel ( $X = \mathcal{R}^2$ ) for various values of the parameter  $\sigma$ .

Hilbert space  $\mathcal{H}$ , then for any  $h \in \mathcal{H}$  we have that:

$$h = \sum_{s \in S} \langle h, e_s \rangle e_s, \quad (17.25)$$

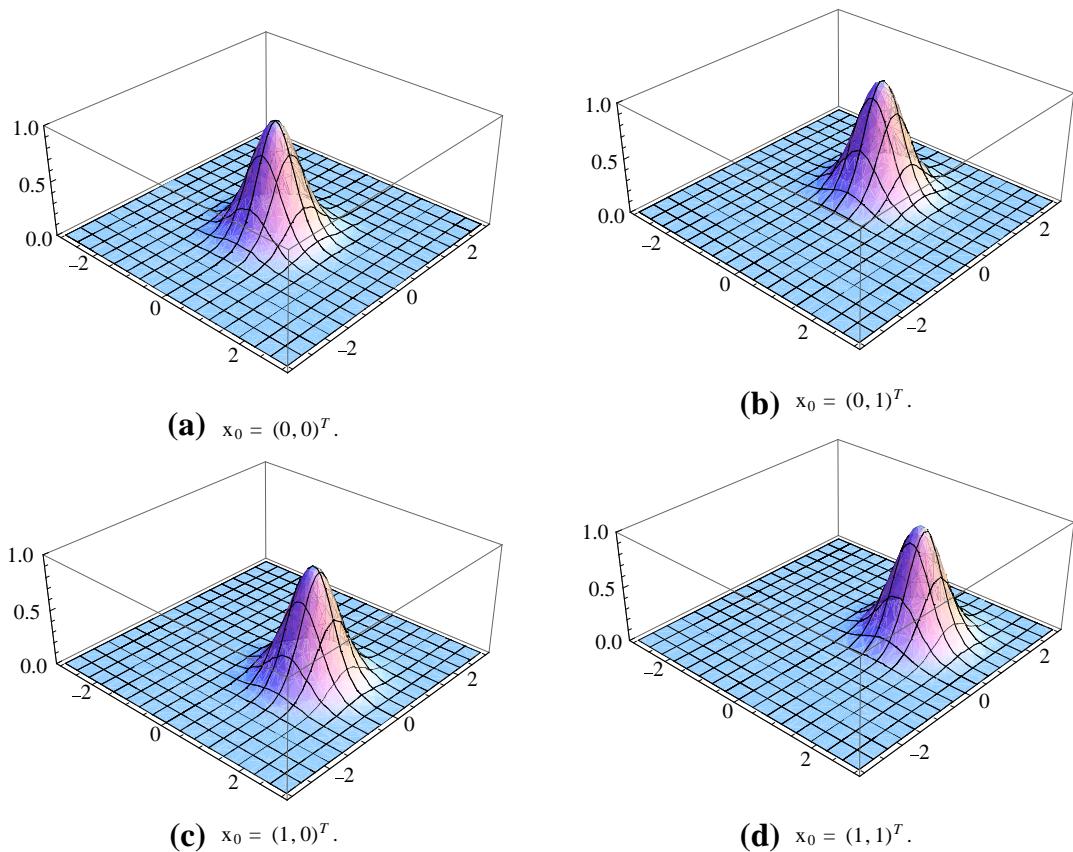
$$\|h\|^2 = \sum_{s \in S} |\langle h, e_s \rangle|^2. \quad (17.26)$$

Note that these two identities hold for a general arbitrary set  $S$  (not necessarily ordered). The convergence in this case is defined somewhat differently. We say that  $h = \sum_{s \in S} h_s$ , if for any  $\epsilon > 0$ , there exists a finite subset  $F_0 \subseteq S$ , such that for any finite set  $F : F_0 \subseteq F \subseteq S$ , we have that  $\|h - \sum_{s \in F} h_s\| < \epsilon$ .

**Proposition 11 (Cauchy-Schwarz Inequality).** *If  $\kappa$  is a reproducing kernel on  $X$  then*

$$\|\kappa(x, y)\|^2 \leq \kappa(x, x) \cdot \kappa(y, y).$$

◇

**FIGURE 17.12**

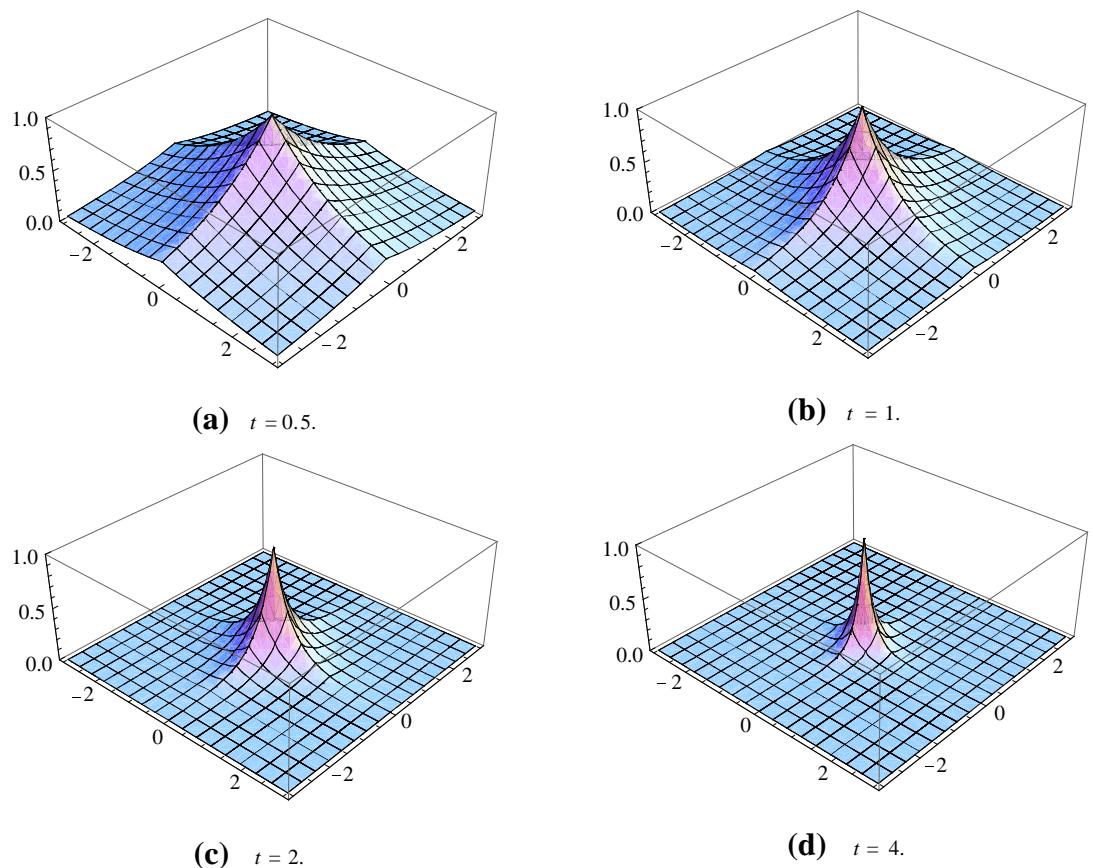
The element  $\Phi(x_0) = \kappa(\cdot, x_0)$  of the feature space induced by the Gaussian kernel ( $X = \mathcal{R}^2$ ) with  $\sigma = 0.5$ .

**Proof.** The proof is straightforward, as  $\kappa(x, y)$  is the inner product  $\langle \Phi(y), \Phi(x) \rangle_{\mathcal{H}}$  of the space  $\mathcal{H}(\kappa)$ .  $\square$

**Theorem 12.** Every finite dimensional class of functions defined on  $X$ , equipped with an inner product, is an RKHS. Let  $h_1, \dots, h_N$  constitute a basis of the space and the inner product is defined as follows

$$\langle f, g \rangle = \sum_{n,m=1}^N \alpha_{n,m} \gamma_n \zeta_m,$$

for  $f = \sum_{n=1}^N \gamma_n h_n$  and  $g = \sum_{n=1}^N \zeta_n h_n$ , where the  $N \times N$  real matrix  $A = (\alpha_{n,m})^N$  is strictly positive definite. Let  $B = (\beta_{n,m})^N$  be the inverse of  $A$ , then the kernel of the RKHS is given by

**FIGURE 17.13**

The element  $\Phi(0) = \kappa(\cdot, 0)$  of the feature space induced by the Laplacian kernel ( $X = \mathcal{R}^2$ ) for various values of the parameter  $t$ .

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{n,m=1}^N \beta_{n,m} h_n(\mathbf{x}) h_m(\mathbf{y}). \quad (17.27)$$

◇

**Proof.** The reproducing property is immediately verified by Eq. (17.27):

$$\langle f, \kappa(\cdot, \mathbf{x}) \rangle_{\mathcal{H}} = \left\langle \sum_{n=1}^N \gamma_n h_n, \sum_{n,m=1}^N \beta_{n,m} h_m(\mathbf{x}) \cdot h_m \right\rangle_{\mathcal{H}} = \sum_{n,m=1}^N \alpha_{n,m} \gamma_n \sum_{k=1}^N \beta_{m,k} h_k(\mathbf{x})$$

$$\begin{aligned}
&= \sum_{n,k=1}^N \left( \sum_{m=1}^N \alpha_{n,m} \beta_{m,k} \right) \gamma_n h_k(\mathbf{x}) = \sum_{n=1}^N \gamma_n h_n(\mathbf{x}) \\
&= f(\mathbf{x}). \quad \square
\end{aligned}$$

The following theorem gives the kernel of a RKHS (of finite or infinite dimension) in terms of the elements of an orthonormal basis.

**Theorem 13.** *Let  $\mathcal{H}$  be a RKHS on  $X$  with reproducing kernel  $\kappa$ . If  $\{e_s : s \in S \subset \mathcal{N}_*\}$  is an orthonormal basis for  $\mathcal{H}$ , then  $\kappa(\mathbf{x}, \mathbf{y}) = \sum_{s \in S} e_s(\mathbf{y}) e_s(\mathbf{x})$ , where this series converges pointwise.*  $\diamond$

**Proof.** For any  $\mathbf{y} \in X$  we have that  $\langle \kappa(\cdot, \mathbf{y}), e_s \rangle_{\mathcal{H}} = \langle e_s, \kappa(\cdot, \mathbf{y}) \rangle_{\mathcal{H}} = e_s(\mathbf{y})$ . Hence, employing Parseval's identity (17.25), we have that  $\kappa(\cdot, \mathbf{y}) = \sum_{s \in S} e_s(\mathbf{y}) e_s(\cdot)$ , where these sums converge in the norm on  $\mathcal{H}$ . Since the sums converge in the norm, they converge at every point. Hence,  $\kappa(\mathbf{x}, \mathbf{y}) = \sum_{s \in S} e_s(\mathbf{y}) e_s(\mathbf{x})$ .  $\square$

**Proposition 14.** *If  $\mathcal{H}$  is a RKHS on  $X$  with respective kernel  $\kappa$ , then every closed subspace  $\mathcal{F} \subseteq \mathcal{H}$  is also a RKHS. In addition, if  $\mathcal{F}_1(\kappa_1)$  and  $\mathcal{F}_2(\kappa_2)$  are complementary subspaces in  $\mathcal{H}$  then  $\kappa = \kappa_1 + \kappa_2$ .*  $\diamond$

**Proposition 15.** *Let  $\mathcal{H}$  be a RKHS on  $X$  with kernel  $\kappa$  and  $\{g_n\}$  is an orthonormal system in  $\mathcal{H}$ . Then for any sequence of numbers  $\{a_n\}$  such that  $\sum_n a_n^2 < \infty$  (i.e.,  $\{a_n\} \in \ell_2$ ) we have*

$$\sum_n |a_n| |g_n(\mathbf{x})| \leq \kappa(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} \left( \sum_n |a_n|^2 \right)^{\frac{1}{2}}. \quad \diamond$$

**Proof.** We have seen that  $g_n(\mathbf{y}) = \langle g_n, \kappa(\cdot, \mathbf{y}) \rangle$  and that  $\|\kappa(\cdot, \mathbf{y})\|_{\mathcal{H}}^2 = \kappa(\mathbf{y}, \mathbf{y})$ . Thus, considering that  $g_n$ 's are orthonormal and taking the Parseval's identity (17.26) for  $\kappa(\cdot, \mathbf{y})$  with respect to the orthonormal basis, we have:

$$\sum_n |g_n(\mathbf{y})|^2 = \sum_n |\langle g_n, \kappa(\cdot, \mathbf{y}) \rangle_{\mathcal{H}}|^2 \leq \|\kappa(\cdot, \mathbf{y})\|_{\mathcal{H}}^2 = \kappa(\mathbf{y}, \mathbf{y}).$$

Therefore, applying the Cauchy-Schwartz inequality we take

$$\sum_n |a_n| |g_n(\mathbf{x})| \leq \left( \sum_n |a_n|^2 \right)^{\frac{1}{2}} \left( \sum_n |g_n(\mathbf{x})|^2 \right)^{\frac{1}{2}} \leq \kappa(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} \left( \sum_n |a_n|^2 \right)^{\frac{1}{2}}. \quad \square$$

**Theorem 16 (Representer Theorem [26]).** *Denote by  $\Omega : [0, +\infty) \rightarrow \mathcal{R}$  a strictly monotonic increasing function, by  $X$  a nonempty set and by  $\mathcal{L} : X \times \mathcal{R}^2 \rightarrow \mathcal{R} \cup \{\infty\}$  an arbitrary loss function. Then each minimizer  $f \in \mathcal{H}$  of the regularized minimization problem:*

$$\min_f \mathcal{L}((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_N, y_N, f(\mathbf{x}_N)) + \Omega(\|f\|_{\mathcal{H}}^2),$$

admits a representation of the form

$$f = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n), \quad (17.28)$$

where  $\theta_n \in \mathbb{R}$ , for  $n = 1, 2, \dots, N$ .  $\diamond$

**Proof.** We may decompose each  $f \in \mathcal{H}$  into a part contained in the span of the kernels centered at the training points, i.e.,  $\kappa(\cdot, \mathbf{x}_1), \dots, \kappa(\cdot, \mathbf{x}_N)$ , (which is a closed linear subspace) and a part in the orthogonal complement of the previous span. Thus each  $f$  can be written as:

$$f = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n) + f_{\perp}.$$

Applying the reproducing property and considering that  $\langle f_{\perp}, \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = 0$ , for  $n = 1, \dots, N$ , we take:

$$f(\mathbf{x}_n) = \langle f, \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = \sum_{i=1}^N \theta_i \kappa(\mathbf{x}_n, \mathbf{x}_i) + \langle f_{\perp}, \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = \sum_{i=1}^N \theta_i \kappa(\mathbf{x}_n, \mathbf{x}_i).$$

Thus, the value of the loss function  $\mathcal{L}$  depends only on the part contained in the span of the kernels centered at the training points, i.e., on  $a_1, \dots, a_N$ . Furthermore, for all  $f_{\perp}$  we have:

$$\Omega(\|f\|^2) = \Omega\left(\left\|\sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n)\right\|^2 + \|f_{\perp}\|_{\mathcal{H}}^2\right) \geq \Omega\left(\left\|\sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n)\right\|^2\right).$$

Thus, for any fixed  $\theta_1, \dots, \theta_n$  the value of the cost function is minimized for  $f_{\perp} = 0$ . Hence, the solution of the minimization task will have to obey this property too.  $\square$

Examples of loss functions  $\mathcal{L}$  as the ones mentioned in Theorem 16 are for example the total squared error:

$$\mathcal{L}((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_N, y_N, f(\mathbf{x}_N))) = \sum_{n=1}^N (f(\mathbf{x}_n) - y_n)^2,$$

and the  $l_1$  norm measured error

$$\mathcal{L}((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_N, y_N, f(\mathbf{x}_N))) = \sum_{n=1}^N |f(\mathbf{x}_n) - y_n|.$$

The aforementioned theorem is of great importance to practical applications. Although one might be trying to solve an optimization task in an infinite dimensional RKHS  $\mathcal{H}$  (such as the one that generated by the Gaussian kernel), the Representer Theorem states that the solution of the problem lies in the span of  $N$  particular kernels, those centered on the training points.

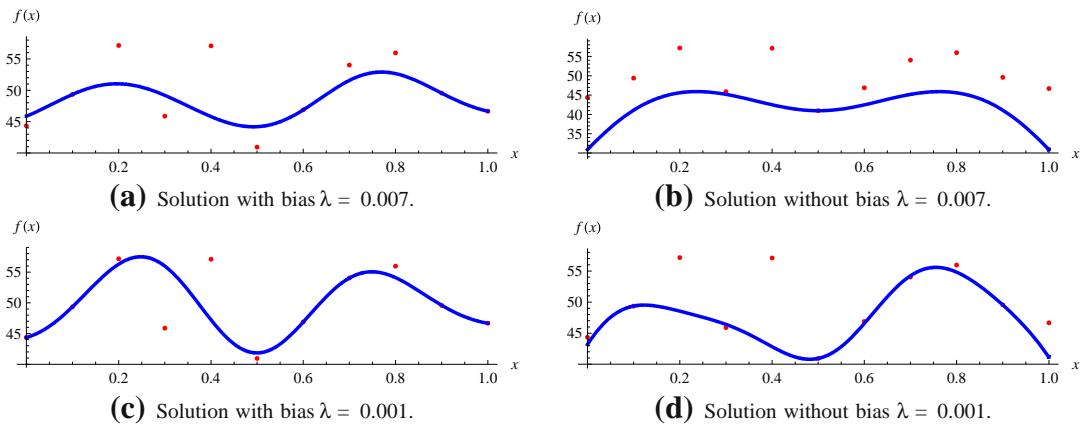


FIGURE 17.14

Solving the regression problem  $\min_f \frac{1}{N} \sum_{n=1}^N (y_n - f(x_n))^2 + \lambda \|f\|_{\mathcal{H}}^2$ , on a set of 11 points (a), (c) with a bias, i.e.,  $f$  admits the form of (17.29) and (b), (d) without a bias, i.e.,  $f$  admits the form of (17.28). In all the examples the Gaussian kernel function was employed. In (a) and (b) we set  $\sigma = 0.15$ ,  $\lambda = 0.007$ . In (c) and (d) we set  $\sigma = 0.15$ ,  $\lambda = 0.001$ . Observe that for  $\lambda = 0.007$ , the unbiased solution takes values significantly lower compared to the values of the training points. For the smaller  $\lambda = 0.001$ , the difference between (c) and (d) is reduced (compared to the case  $\lambda = 0.007$ ). Moreover, one may observe that for the larger value,  $\lambda = 0.007$ , the resulting curves are smoother compared to the curves that correspond to  $\lambda = 0.001$ .

In practice, we often include a bias factor to the solution of kernel-based regularized minimization tasks, that is, we assume that  $f$  admits a representation of the form

$$f = \sum_{n=1}^N \theta_n \kappa(\cdot, x_n) + b, \quad (17.29)$$

where  $b \in \mathcal{R}$ . This has been shown to improve the performance of the respective algorithms [2, 13, 27], for two main reasons. Firstly, the introduction of the bias,  $b$ , enlarges the family of functions in which we search for a solution, thus leading to potentially better estimations. Moreover, as the regularization factor  $\Omega(\|f\|_{\mathcal{H}}^2)$  penalizes the values of  $f$  at the training points, the resulting solution tends to take values as close to zero as possible, for large values of  $\lambda$  (compare Figures 17.14b and 17.14d). At this point, another comment is of interest also. Note that it turns out that the regularization factor  $\|f\|_{\mathcal{H}}^2$  for certain types of kernel functions (e.g., Gaussian) involves the derivatives of  $f$  of all orders [13, 28]. As a consequence the larger the  $\lambda$  is, the smoother the resulting curve becomes (compare Figures 17.14a with 17.14c and Figures 17.14b with 17.14d). The use of the bias factor is theoretically justified by the semi-parametric Representer theorem.

**Theorem 17 (Semi-parametric Representer Theorem [13]).** Suppose that in addition to the assumptions of theorem 16, we are given a set of  $M$  real valued functions  $\{\psi_m\}_{m=1}^M : X \rightarrow \mathcal{R}$ , with the property that the  $N \times M$  matrix  $(\psi_m(x_n))_{n,m}$  has rank  $M$ . Then any  $\tilde{f} := f + h$ , with  $f \in \mathcal{H}$  and

$h \in \text{span}\{\psi_m; m = 1, \dots, M\}$ , solving

$$\min_{\tilde{f}} \mathcal{L}((\mathbf{x}_1, y_1, \tilde{f}(\mathbf{x}_1)), \dots, (\mathbf{x}_N, y_N, \tilde{f}(\mathbf{x}_N)) + \Omega(\|f\|_{\mathcal{H}}^2),$$

admits a representation of the form

$$\tilde{f} = \sum_{n=1}^N \theta_n \kappa(\cdot, \mathbf{x}_n) + \sum_{m=1}^M b_m \psi_m(\cdot), \quad (17.30)$$

with  $\theta_n \in \mathcal{R}, b_m \in \mathcal{R}$ , for all  $n = 1, \dots, N, m = 1, \dots, M$ .  $\diamond$

The following results can be used for the construction of new kernels.

**Proposition 18 (Conformal Transformations).** *If  $f : X \rightarrow \mathcal{R}$  is any function, then  $\kappa_1(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})f(\mathbf{y})$  is a reproducing kernel. Moreover, if  $\kappa$  is any other reproducing kernel then  $\kappa_2(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})\kappa(\mathbf{x}, \mathbf{y})f(\mathbf{y})$  is also a reproducing kernel.*  $\diamond$

**Proof.** The first part is a direct consequence of theorem 12. For the second part, consider  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$  and  $a_1, \dots, a_N \in \mathcal{R}$ . Then

$$\begin{aligned} \sum_{n,m=1}^N a_n a_m f(\mathbf{x}_n) \kappa(\mathbf{x}_n, \mathbf{x}_m) f(\mathbf{x}_m) &= \sum_{n,m=1}^N a_n a_m f(\mathbf{x}_n) f(\mathbf{x}_m) \langle \Phi(\mathbf{x}_m), \Phi(\mathbf{x}_n) \rangle_{\mathcal{H}} \\ &= \left\langle \sum_m^N a_m f(\mathbf{x}_m) \Phi(\mathbf{x}_m), \sum_n^N a_n f(\mathbf{x}_n) \Phi(\mathbf{x}_n) \right\rangle_{\mathcal{H}} \\ &= \left\| \sum_n^N a_n f(\mathbf{x}_n) \Phi(\mathbf{x}_n) \right\|^2 \geq 0. \end{aligned}$$

Moreover, as

$$\begin{aligned} \cos(\angle(\Phi_2(\mathbf{x}), \Phi_2(\mathbf{y}))) &= \frac{f(\mathbf{x})\kappa(\mathbf{x}, \mathbf{y})f(\mathbf{y})}{\sqrt{f(\mathbf{x})\kappa(\mathbf{x}, \mathbf{x})f(\mathbf{x})}\sqrt{f(\mathbf{y})\kappa(\mathbf{y}, \mathbf{y})f(\mathbf{y})}} \\ &= \frac{\kappa(\mathbf{x}, \mathbf{y})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})}\sqrt{\kappa(\mathbf{y}, \mathbf{y})}} = \cos(\angle(\Phi(\mathbf{x}), \Phi(\mathbf{y}))), \end{aligned}$$

this transformation of the original kernel, preserves angles in the feature space.  $\square$

**Theorem 19 (Restriction of a kernel).** *Let  $\mathcal{H}$  be a RKHS on  $X$  with respective kernel  $\kappa$ . Then  $\kappa$  restricted to the set  $X_1 \subset X$  is the reproducing kernel of the class  $\mathcal{H}_1$  of all restrictions of functions of  $\mathcal{H}$  to the subset  $X_1$ . The respective norm of any such restricted function  $f_1 \in \mathcal{H}_1$  (originating from  $f \in \mathcal{H}$ ) has norm  $\|f_1\|_{\mathcal{H}_1} = \min\{\|f\|_{\mathcal{H}} : f \in \mathcal{H}, f|_{X_1} = f_1\}$ .*  $\diamond$

**Proposition 20 (Normalization of a kernel).** *Let  $\mathcal{H}$  be a RKHS on  $X$  with respective kernel  $\kappa$ . Then*

$$\hat{\kappa}(\mathbf{x}, \mathbf{y}) = \frac{\kappa(\mathbf{x}, \mathbf{y})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})\kappa(\mathbf{y}, \mathbf{y})}}, \quad (17.31)$$

*is also a positive definite kernel on  $X$ . Note that  $|\hat{\kappa}(\mathbf{x}, \mathbf{y})| \leq 1$ , for all  $\mathbf{x}, \mathbf{y} \in X$ .*  $\diamond$

**Proof.** Let  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in X$  and  $c_1, \dots, c_N$  be real numbers. Then

$$\begin{aligned} \sum_{n,m=1}^N c_n c_m \hat{\kappa}(\mathbf{x}_n, \mathbf{x}_m) &= \sum_{n,m=1}^N c_n c_m \frac{\kappa(\mathbf{x}_n, \mathbf{x}_m)}{\sqrt{\kappa(\mathbf{x}_n, \mathbf{x}_n)\kappa(\mathbf{x}_m, \mathbf{x}_m)}} \\ &= \sum_{n,m=1}^N \frac{c_n}{\sqrt{\kappa(\mathbf{x}_n, \mathbf{x}_n)}} \frac{c_m}{\sqrt{\kappa(\mathbf{x}_m, \mathbf{x}_m)}} \kappa(\mathbf{x}_n, \mathbf{x}_m) \geq 0, \end{aligned}$$

as  $\kappa$  is a positive definite kernel.  $\square$

**Theorem 21 (Sum of kernels [3]).** Let  $\mathcal{H}_1, \mathcal{H}_2$  be two RKHS's on  $X$  with respective kernels  $\kappa_1, \kappa_2$ . Then  $\kappa = \kappa_1 + \kappa_2$  is also a reproducing kernel. The corresponding RKHS,  $\mathcal{H}$ , contains the functions  $f = f_1 + f_2$ , where  $f_i \in \mathcal{H}_i$ ,  $i = 1, 2$ . The respective norm is defined by

$$\|f\|_{\mathcal{H}} = \min\{\|f_1\| + \|f_2\| : f = f_1 + f_2, f_i \in \mathcal{H}_i, i = 1, 2\}. \quad \diamond$$

**Proof.** It is trivial to show that  $\kappa_1 + \kappa_2$  is a positive definite kernel. The difficult part is to associate this kernel with the specific RKHS  $\mathcal{H}$ . Consider the Hilbert space  $F = \mathcal{H}_1 \times \mathcal{H}_2$ . The respective inner product and the corresponding norm are defined as

$$\begin{aligned} \langle (f_1, f_2), (g_1, g_2) \rangle_F &= \langle f_1, g_1 \rangle_{\mathcal{H}_1} + \langle f_2, g_2 \rangle_{\mathcal{H}_2}, \\ \|(f_1, f_2)\|_F^2 &= \|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2, \end{aligned}$$

for  $f_1, g_1 \in \mathcal{H}_1$  and  $f_2, g_2 \in \mathcal{H}_2$ . If  $\mathcal{H}_1$  and  $\mathcal{H}_2$  have only 0 in common, it is easy to show that there is a one-to-one correspondence between  $F$  and  $\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2$ , as each  $f \in \mathcal{H}$  can be decomposed into two parts (one belonging to  $\mathcal{H}_1$  and the other in  $\mathcal{H}_2$ ) uniquely. The difficult part is to discover such a relation, if  $\mathcal{H}_0 = \mathcal{H}_1 \cap \mathcal{H}_2$  is larger than  $\{0\}$ . To make this fact clear, consider this simple example: Let  $\mathcal{H}_1$  and  $\mathcal{H}_2$  be the linear classes of polynomials of orders up to 1 and up to 2 respectively. Obviously,  $\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2 = \mathcal{H}_2$ , as  $\mathcal{H}_1 \subset \mathcal{H}_2$ . Let  $f(x) = x^2 + 5x$ ,  $f \in \mathcal{H}$ . Then  $f$  can be decomposed into two parts (one belonging to  $\mathcal{H}_1$  and the other in  $\mathcal{H}_2$ ) in more than one ways. For example  $f(x) = (x^2) + (5x)$ , or  $f(x) = (x^2 + 4x) + (x)$ , or  $f(x) = (x^2 + 2x) + (3x)$ , e.t.c. Thus, the mapping between  $f = f_1 + f_2 \in \mathcal{H}$  and  $(f_1, f_2) \in F$  is not one-to-one. However, in such cases, we can still find a smaller subspace of  $F$ , which can be identified to  $\mathcal{H}$ .

To this end, define  $F_0 = \{(f, -f) : f \in \mathcal{H}_0\}$ . It is clear that  $F_0$  is a linear subspace of  $F$ . We will show that it is a closed one. Consider the converging sequence in  $F_0$ :  $(f_n, -f_n) \rightarrow (\tilde{f}_1, \tilde{f}_2)$ . Then  $f_n \rightarrow \tilde{f}_1$  and  $-f_n \rightarrow \tilde{f}_2$ . Thus  $\tilde{f}_1 = -\tilde{f}_2$  and  $(\tilde{f}_1, \tilde{f}_2)$  is in  $F_0$ . As  $F_0$  is a closed linear subspace of  $F$ , we may consider its complementary subspace  $F_0^\perp : F = F_0 \oplus F_0^\perp$ .

As a next step, consider the linear transformation  $T : F \rightarrow \mathcal{H} : T(f_1, f_2) = f_1 + f_2$ . The kernel of this transformation is the subspace  $F_0$ . Hence, there is a one-to-one correspondence between  $F_0^\perp$  and  $\mathcal{H}$ . Consider the inverse transformation  $(T|_{F_0^\perp})^{-1}$  and let  $(T|_{F_0^\perp})^{-1}(f) = (f', f'')$ , for  $f \in \mathcal{H}$ , where  $f' \in \mathcal{H}_1$  and  $f'' \in \mathcal{H}_2$ , i.e., through  $(T|_{F_0^\perp})^{-1}$  we decompose  $f$  uniquely into two components, one in  $\mathcal{H}_1$  and the other in  $\mathcal{H}_2$ . This decomposition enables us to define an inner product in  $\mathcal{H}$ , i.e.,

$$\langle f, g \rangle_{\mathcal{H}} = \langle f' + f'', g' + g'' \rangle_{\mathcal{H}} = \langle f', g' \rangle_{\mathcal{H}_1} + \langle f'', g'' \rangle_{\mathcal{H}_2} = \langle (f', f''), (g', g'') \rangle_F,$$

for  $f, g \in \mathcal{H}$ . To prove that to this  $\mathcal{H}$  there corresponds the kernel  $\kappa = \kappa_1 + \kappa_2$ , we make the following remarks:

1. For every  $y \in X$ ,  $\kappa(\cdot, y) = \kappa_1(\cdot, y) + \kappa_2(\cdot, y) \in \mathcal{H}$ .
2. For every  $y \in X$ , let  $T^{-1}(\kappa(\cdot, y)) = (\kappa'(\cdot, y), \kappa''(\cdot, y))$ . Thus

$$\kappa(x, y) = \kappa'(x, y) + \kappa''(x, y) = \kappa_1(x, y) + \kappa_2(x, y),$$

and consequently  $\kappa_1(x, y) - \kappa'(x, y) = -(\kappa_2(x, y) - \kappa''(x, y))$ . This means that  $(\kappa_1(x, y) - \kappa'(x, y), \kappa_2(x, y) - \kappa''(x, y)) \in F_0$ . Hence, for every  $y \in X$  we have

$$\begin{aligned} f(y) &= f'(y) + f''(y) = \langle f', \kappa_1(\cdot, y) \rangle_{\mathcal{H}_1} + \langle f'', \kappa_2(\cdot, y) \rangle_{\mathcal{H}_2} \\ &= \langle (f', f''), (\kappa_1(\cdot, y), \kappa_2(\cdot, y)) \rangle_F \\ &= \langle (f', f''), (\kappa'(\cdot, y), \kappa''(\cdot, y)) \rangle_F + \langle (f', f''), (\kappa_1(\cdot, y) - \kappa'(\cdot, y), \kappa_2(\cdot, y) - \kappa''(\cdot, y)) \rangle_F. \end{aligned}$$

As  $(\kappa_1(x, y) - \kappa'(x, y), \kappa_2(x, y) - \kappa''(x, y)) \in F_0$  and  $(f', f'') \in F_0^\perp$ , we conclude that

$$f(y) = \langle (f', f''), (\kappa'(\cdot, y), \kappa''(\cdot, y)) \rangle_F = \langle f' + f'', \kappa'(\cdot, y) + \kappa''(\cdot, y) \rangle_{\mathcal{H}} = \langle f, \kappa(\cdot, y) \rangle_{\mathcal{H}}.$$

This is the reproducing property.

Finally, to prove the last part of the theorem, consider again  $f \in \mathcal{H}$  and let  $f_i \in \mathcal{H}_i$ ,  $i = 1, 2$ , such that  $f = f_1 + f_2$  and let  $f' \in \mathcal{H}_1$  and  $f'' \in \mathcal{H}_2$  be the unique decomposition of  $f$  through  $T^{-1}$ . As  $f_1 + f_2 = f' + f''$  we obtain that  $f' - f_1 = -(f'' - f_2)$ , which implies that  $(f' - f_1, f'' - f_2) \in F_0$ . Thus, we take:

$$\begin{aligned} \|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2 &= \|(f_1, f_2)\|_F^2 = \|(f', f'')\|_F^2 + \|(f_1 - f', f_2 - f'')\|_F^2 \\ &= \|f'\|_{\mathcal{H}_1}^2 + \|f''\|_{\mathcal{H}_2}^2 + \|(f_1 - f', f_2 - f'')\|_F^2 \\ &= \|f\|_{\mathcal{H}}^2 + \|(f_1 - f', f_2 - f'')\|_F^2. \end{aligned}$$

From the last relation we conclude that  $\|f\|_{\mathcal{H}}^2 = \|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2$ , if and only if  $f_1 = f'$  and  $f_2 = f''$ . In this case we take the minimum value of  $\|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2$ , for all possible decompositions  $f = f_1 + f_2$ . This completes the proof.  $\square$

Despite the sum of kernels, other operations preserve reproducing kernels as well. Below, we give an extensive list of such operations. For a description of the induced RKHS and a formal proof (in the cases that are not considered here) the interested reader may refer to [3, 13].

1. If  $\kappa(x, y)$  is a positive definite kernel on  $X$ , then  $\lambda\kappa(x, y)$  is also a positive definite kernel for any  $\lambda \geq 0$ . It is obvious that in this case  $\mathcal{H}(\lambda\kappa) = \mathcal{H}(\kappa)$ , if  $\lambda > 0$ . If  $\lambda = 0$ , then  $\mathcal{H}(0) = \{0\}$ .
2. If  $\kappa_1(x, y)$  and  $\kappa_2(x, y)$  are positive definite kernels on  $X$ , then  $\kappa_1(x, y) + \kappa_2(x, y)$  is also a positive definite kernel, as Theorem 21 established.
3. If  $\kappa_1(x, y)$  and  $\kappa_2(x, y)$  are positive definite kernels on  $X$ , then  $\kappa_1(x, y) \cdot \kappa_2(x, y)$  is also a positive definite kernel.

4. If  $\kappa_n(\mathbf{x}, \mathbf{y})$  are positive definite kernels on  $X$ , such that  $\lim_n \kappa_n(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x}, \mathbf{y})$ , for all  $\mathbf{x}, \mathbf{y} \in X$ , then  $\kappa(\mathbf{x}, \mathbf{y})$  is also a positive definite kernel.
5. If  $\kappa(\mathbf{x}, \mathbf{y})$  is a positive definite kernel on  $X$  and  $p(z)$  is a polynomial with non-negative coefficients, then  $p(\kappa(\mathbf{x}, \mathbf{y}))$  is also a positive definite kernel.
6. If  $\kappa(\mathbf{x}, \mathbf{y})$  is a positive definite kernel on  $X$ , then  $e^{\kappa(\mathbf{x}, \mathbf{y})}$  is also a positive definite kernel. To prove this, consider the Taylor expansion formula of  $e^z$ , which may be considered as a limit of polynomials with non-negative coefficients.
7. If  $\kappa(\mathbf{x}, \mathbf{y})$  is a positive definite kernel on  $X$  and  $\Psi : X' \rightarrow X$  is a function, then  $\kappa(\Psi(\mathbf{x}), \Psi(\mathbf{y}))$  is a positive definite kernel on  $X'$ .
8. If  $\kappa_1(\mathbf{x}, \mathbf{y})$  and  $\kappa_2(\mathbf{x}', \mathbf{y}')$  are positive definite kernels on  $X$  and  $X'$  respectively, then their tensor product  $(\kappa_1 \otimes \kappa_2)(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') = \kappa_1(\mathbf{x}, \mathbf{y})\kappa_2(\mathbf{x}', \mathbf{y}')$ , is a kernel on  $X \times X'$ .
9. If  $\kappa_1(\mathbf{x}, \mathbf{y})$  and  $\kappa_2(\mathbf{x}', \mathbf{y}')$  are positive definite kernels on  $X$  and  $X'$  respectively, then their direct sum  $(\kappa_1 \oplus \kappa_2)(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') = \kappa_1(\mathbf{x}, \mathbf{y}) + \kappa_2(\mathbf{x}', \mathbf{y}')$ , is a kernel on  $X \times X'$ .

### 1.17.5.6 Dot product and translation invariant kernels

There are two important classes of kernels that follow certain rules and are widely used in practice. The first one includes the *dot product kernels*, which are functions defined as  $\kappa(\mathbf{x}, \mathbf{y}) = f(\langle \mathbf{x}, \mathbf{y} \rangle)$ , for some real function  $f$ . The second class are the *translation invariant kernels*, which are defined as  $\kappa(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} - \mathbf{y})$ , for some real function  $f$  defined on  $X$ . The following theorems establish necessary and sufficient conditions for such functions to be reproducing kernels.

**Theorem 22 (Power Series of dot product kernels [29]).** *Let  $f : \mathcal{R} \rightarrow \mathcal{R}$ . A function  $\kappa(\mathbf{x}, \mathbf{y}) = f(\langle \mathbf{x}, \mathbf{y} \rangle)$  defined on  $X$ , such that  $f$  has the power series expansion  $f(t) = \sum_n a_n t^n$ , is a positive definite kernel, if and only if we have  $a_n \geq 0$  for all  $n$ .* ◇

**Theorem 23 (Bochner's-Fourier Criterion for translation invariant kernels [20]).** *Let  $f : X \rightarrow \mathcal{R}$ . A function  $\kappa(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} - \mathbf{y})$  defined on  $X \subseteq \mathcal{R}^l$ , is a positive definite kernel, if the Fourier transform*

$$F[\kappa](\omega) = (2\pi)^{-\frac{N}{2}} \int_X e^{-i\langle \omega, \mathbf{x} \rangle} f(\mathbf{x}) d\mathbf{x},$$

*is non-negative.* ◇

**Remark 24.** Bochner's theorem is more general, involving Borel measures and topological spaces. For the sake of simplicity we give only this simple form.

Employing the tools provided in this section, one can readily prove the positivity of some of the kernels given in Section 1.17.5.4. For example:

- *Homogeneous polynomial kernel:* As  $\langle \mathbf{x}, \mathbf{y} \rangle$  is a positive definite kernel and  $p(z) = z^d$  is a polynomial with non-negative coefficients,  $p(\langle \mathbf{x}, \mathbf{y} \rangle) = (\langle \mathbf{x}, \mathbf{y} \rangle)^d$  is a positive definite kernel.
- *Inhomogeneous polynomial kernel:* As  $\langle \mathbf{x}, \mathbf{y} \rangle$  is a positive definite kernel, and  $p(z) = (z + c)^d$  is a polynomial with non-negative coefficients (for positive  $c$ ),  $p(\langle \mathbf{x}, \mathbf{y} \rangle) = (c + \langle \mathbf{x}, \mathbf{y} \rangle)^d$  is a positive definite kernel.
- *The cosine kernel:* Note that  $\cos(\angle(\mathbf{x}, \mathbf{y})) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}$ . Thus the cosine kernel is the normalization of the simple kernel  $\langle \mathbf{x}, \mathbf{y} \rangle$ .

To prove that the Gaussian and the Laplacian are positive kernels we need another set of tools. This is the topic of Appendix A.

### 1.17.5.7 Differentiation in Hilbert spaces

#### 1.17.5.7.1 Fréchet's differentiation

In the following sections, we will develop cost functions defined on RKHS, that are suitable for minimization tasks related with adaptive filtering problems. As most minimization procedures involve computation of gradients or subgradients, we devote this section to study differentiation on Hilbert spaces. The notion of Fréchet's Differentiability, which generalizes differentiability to general Hilbert spaces, lies at the core of this analysis.

**Definition 25 (Fréchet's Differential).** Let  $\mathcal{H}$  be a real Hilbert space,  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R}$  a function, and  $f \in \mathcal{H}$ . The function  $\mathcal{L}$  is said to be Fréchet differentiable at  $f$ , if there exists a  $g \in \mathcal{H}$  such that

$$\lim_{\|h\|_{\mathcal{H}} \rightarrow 0} \frac{\mathcal{L}(f + h) - \mathcal{L}(f) - \langle h, g \rangle_{\mathcal{H}}}{\|h\|_{\mathcal{H}}} = 0, \quad (17.32)$$

where  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  is the inner product of the Hilbert space  $\mathcal{H}$  and  $\|\cdot\|_{\mathcal{H}} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{H}}}$  is the induced norm.  $\diamond$

The element  $g \in \mathcal{H}$  is called the gradient of the operator at  $f$ , and is usually denoted as  $\nabla \mathcal{L}(f)$ . This relates to the standard gradient operator known by Calculus in Euclidean spaces. The Fréchet's Differential is also known as Strong Differential. There is also a weaker definition of Differentiability, named Gâteaux's Differential (or Weak Differential), which is a generalization of the directional derivative. The Gâteaux differential  $d\mathcal{L}(f, h)$  of  $\mathcal{L}$  at  $f \in \mathcal{H}$  in the direction  $h \in \mathcal{H}$  is defined as

$$d\mathcal{L}(f, h) = \lim_{\epsilon \rightarrow 0} \frac{\mathcal{L}(f + \epsilon h) - \mathcal{L}(f)}{\epsilon}. \quad (17.33)$$

In the following, whenever we are referring to a derivative or a gradient we will mean the one produced by Fréchet's notion of differentiability. The interested reader is addressed to [30–35], (amongst others) for a more detailed discussion on the subject. The well known properties of the derivative of a real valued function of one variable, which are known from elementary Calculus, like the product and chain rules, apply to the Fréchet's derivatives as well.

The following simple examples demonstrate the differentiation procedure in arbitrary spaces.

**Example 26.** Consider the real Hilbert space  $\mathcal{H}$ , with inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ , and  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R} : \mathcal{L}(f) = \langle f, g \rangle_{\mathcal{H}}$ , where  $g \in \mathcal{H}$  fixed. We can easily show (using Fréchet's definition) that  $\mathcal{L}$  is differentiable at any  $f \in \mathcal{H}$  and that  $\nabla \mathcal{L}(f) = g$ . In the case where  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R} : \mathcal{L}(f) = \|f\|_{\mathcal{H}}^2$ , we can easily show (using Fréchet's definition) that  $\mathcal{L}$  is differentiable at any  $f \in \mathcal{H}$  and that  $\nabla \mathcal{L}(f) = 2f$ .

---

## 1.17.6 Least squares learning algorithms

Having discussed the basic definitions and properties of RKHS we now turn to our main focus of the paper to discuss online learning schemes, which can operate in such spaces. We will start with two celebrated schemes, which are built around the Least Squares criterion, namely the LMS and the RLS.

Although we can look at these algorithms as special cases of the schemes to be considered later on, in the more general setting of convex analytic tools, we chose to treat them separately. The reason is that the LMS and RLS are widely known, and the less familiar reader could follow their extension to RKHSs easier.

### 1.17.6.1 Least mean square (LMS)

Consider the sequence of examples  $\{(\mathbf{x}_n, y_n)\}_{n=1,2,\dots}$ , where  $\mathbf{x}_n \in X \subset \mathcal{R}^l$ , and  $y_n \in \mathcal{R}$  for  $n = 1, \dots, N$ , and a parametric class of functions  $\mathcal{F} = \{f_\theta(\cdot) : \theta \in \mathcal{A} \subseteq \mathcal{R}^K\}$ . The goal of a typical online/adaptive learning task is to infer an input-output relationship  $f_\theta$  from the parametric class of functions  $\mathcal{F}$ , based on the given data, so that to minimize a certain loss function,  $\mathcal{L}(\theta)$ , that measures the error between the actual output,  $y_n$ , and the estimated output,  $\hat{y}_n = f_\theta(\mathbf{x}_n)$ , at iteration  $n$ .

Least mean squares (LMS) algorithms are a popular class of adaptive learning systems that adopt the least mean squares error (i.e., the mean squared difference between the desired and the actual signal) to estimate the unknown coefficients  $\theta$ . Its most typical form was invented in the 1960s by Bernard Widrow and his Ph.D. student Ted Hoff [36]. In the standard *least mean square* (LMS) setting, one adopts the class of linear functions, i.e.,  $\mathcal{F} = \{f_\theta(\mathbf{x}) = \theta^T \mathbf{x} : \theta \in \mathcal{R}^l\}$  and employs the mean square error,  $\mathcal{L}(\theta) = E[|y_n - \theta^T \mathbf{x}_n|^2]$ , as the loss function. To this end, the gradient descent rationale, e.g., [4–6], is employed and at each time instant,  $n = 1, 2, \dots, N$ , the gradient of the mean square error, i.e.,  $\nabla \mathcal{L}(\theta) = -2E[(y_n - \theta^T \mathbf{x}_n)\mathbf{x}_n]$ , is approximated via its current measurement, i.e.,  $\nabla \mathcal{L}(\theta) = -2E[(y_n - \theta^T \mathbf{x}_n)\mathbf{x}_n] \approx -2(y_n - \theta^T \mathbf{x}_n)\mathbf{x}_n = \nabla \mathcal{L}_n(\theta)$ , where

$$\mathcal{L}_n(\theta) := (y_n - \theta^T \mathbf{x}_n)^2, \quad n = 1, 2, \dots$$

If we define  $e_n := y_n - \theta_{n-1}^T \mathbf{x}_n$  as the a-priori error, where  $\theta_{n-1}$  is the current estimate, then the previous discussion leads naturally to the step update equation

$$\begin{aligned} \theta_n &= \theta_{n-1} - \mu \nabla \mathcal{L}_n(\theta_{n-1}) \\ &= \theta_{n-1} + \mu e_n \mathbf{x}_n, \quad n = 1, 2, \dots \end{aligned} \tag{17.34}$$

Another derivation of this classical recursion, through a convex analytic path, will be given in Section 1.17.7. Such a viewpoint will set the LMS as an instant of a larger family of algorithms.

Assuming that the initial  $\theta_0 = \mathbf{0}$ , a repeated application of (17.34) gives

$$\theta_n = \mu \sum_{i=1}^n e_i \mathbf{x}_i, \quad n = 1, 2, \dots, \tag{17.35}$$

while the predicted output of the learning system at iteration  $n$  becomes

$$\hat{y}_n = f_{\theta_{n-1}}(\mathbf{x}_n) = \mu \sum_{i=1}^{n-1} e_i \mathbf{x}_i^T \mathbf{x}_n, \quad n = 1, 2, \dots \tag{17.36}$$

There are a number of convergence criteria for the LMS algorithm [5]. Among the most popular ones are:

- The *convergence of the mean* property, i.e.,  $E[\epsilon_n] \rightarrow 0$ , as  $n \rightarrow \infty$ , where  $\epsilon_n = \theta_n - \theta_*$  and  $\theta_*$  is the Wiener solution. Unfortunately, this is of little practical use, as any sequence of zero mean arbitrary random numbers converges in this sense.
- The *convergence in the mean square* property: If the true input-output relation is given by  $y_n = \theta_*^T \mathbf{x}_n + \eta_n$ , and  $\mathbf{x}_n$  is a weakly stationary process, then  $\mathcal{L}(\theta_{n-1}) = E[|e_n|^2] \rightarrow \text{constant}$ , as  $n \rightarrow \infty$ , provided that  $\mu$  satisfies the condition  $0 < \mu < \frac{2}{\sigma_{\max}}$ , where  $\sigma_{\max}$  is the largest eigenvalue of the correlation matrix  $\mathbf{R} = E[\mathbf{x}_n \mathbf{x}_n^T]$ . We will see in Example 56 that the quantity  $\frac{2}{\sigma_{\max}}$  stems also from a convex analytic point of view of the LMS-related estimation task. In practice, one usually uses  $0 < \mu < \frac{2}{\text{tr}(\mathbf{R})}$ , where  $\text{tr}(\cdot)$  denotes the trace of a matrix.
- The *Misadjustment* property: The Misadjustment ratio, i.e.,  $\rho := \frac{\mathcal{L}(\infty) - \mathcal{L}_{\min}}{\mathcal{L}_{\min}}$ , where

$$\mathcal{L}(\infty) := \lim_{n \rightarrow \infty} \mathcal{L}(\theta_{n-1}) = \lim_{n \rightarrow \infty} E[|e_n|^2],$$

and  $\mathcal{L}_{\min}$  is the minimum mean squared error associated with the optimum Wiener filter, is equal to

$$\rho = \sum_{i=1}^l \frac{\mu \sigma_i}{2 - \mu \sigma_i},$$

provided that the step update parameter  $\mu$  satisfies  $\sum_{i=1}^l \frac{2\sigma_i}{2 - \mu \sigma_i} < 1$ , where  $\sigma_1, \dots, \sigma_l$  are the eigenvalues of the correlation matrix  $\mathbf{R}$ . This is a dimensionless quantity, providing a measure of closeness of the LMS algorithm from optimality in the mean square sense. The smallest  $\rho$  is (compared to 1), the more accurate the learning system is. Usually  $\rho$  is expressed as a percentage. A misadjustment of 10% is, ordinarily, considered to be satisfactory in practice [5].

LMS is sensitive to the scaling of the inputs  $\mathbf{x}_n$ . This makes it hard to choose a learning rate  $\mu$  that guarantees the stability of the algorithm for various inputs. The Normalized LMS (NLMS) is a variant of the LMS designed to cope with this problem. It solves the problem by normalizing with the power of the input. Hence, the step update equation becomes

$$\theta_n = \theta_{n-1} + \frac{\mu e_n}{\|\mathbf{x}_n\|^2} \mathbf{x}_n, \quad n = 1, 2, \dots, \quad (17.37)$$

where  $\mu \in (0, 2)$ . It has been proved that the optimal learning rate of the NLMS is  $\mu = 1$  [5, 6]. We will also see in Section 1.17.7 that the NLMS has a clear geometrical interpretation; it is the repetition of a relaxed (metric) projection mapping onto a series of hyperplanes in the Euclidean space  $\mathcal{R}^l$ . More on the LMS family of algorithms, and more specifically, on its impact in modern signal processing, can be found in [4–6].

### 1.17.6.2 The Kernel LMS

In kernel-based algorithms, we estimate the output of the learning system by a nonlinear function  $f$  in a specific RKHS,  $\mathcal{H}$ , which is associated to a positive definite kernel  $\kappa$ . The sequence of examples are transformed via the feature map  $\Phi : X \rightarrow \mathcal{H}$ ,  $\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$  and the LMS rationale is employed on

the transformed data

$$\{(\Phi(\mathbf{x}_1), y_1), (\Phi(\mathbf{x}_2), y_2), \dots, (\Phi(\mathbf{x}_n), y_n), \dots\},$$

while the estimated output at iteration  $n$  takes the form  $\hat{y}_n = \langle \Phi(\mathbf{x}_n), f \rangle_{\mathcal{H}}$ , for some  $f \in \mathcal{H}$ . Although this is a linear algorithm in the RKHS  $\mathcal{H}$ , it corresponds to a nonlinear processing in  $X$ . To calculate the gradient of the respective loss function,  $\mathcal{L}(f) = E[|y_n - \langle \Phi(\mathbf{x}_n), f \rangle_{\mathcal{H}}|^2]$ , the Fréchet's notion of differentiability is adopted, leading to:  $\nabla \mathcal{L}(f) = -2E[e_n \Phi(\mathbf{x}_n)]$ ; this, according to the LMS rationale, is approximated at each time instant by its current measurement, i.e.,  $-2e_n \Phi(\mathbf{x}_n)$ . Consequently, following a gradient descent rationale, the step update equation of the KLMS is given by

$$\begin{aligned} f_n &= f_{n-1} + \mu e_n \Phi(\mathbf{x}_n), \\ &= f_{n-1} + \mu e_n \kappa(\cdot, \mathbf{x}_n), \quad n = 1, 2, \dots \end{aligned} \quad (17.38)$$

As it was stated for (17.34), the recursion (17.38) will be re-derived through convex analytic arguments in Section 1.17.7.

Assuming that  $f_0 = 0$ , i.e., the zero function, a repeated application of (17.38) gives

$$f_n = \mu \sum_{i=1}^n e_i \Phi(\mathbf{x}_i), \quad n = 1, 2, \dots, \quad (17.39)$$

and the output of the filter at iteration  $n$  becomes

$$\hat{y}_n = \langle \Phi(\mathbf{x}_n), f_{n-1} \rangle_{\mathcal{H}} = \mu \sum_{i=1}^{n-1} e_i \langle \Phi(\mathbf{x}_n), \Phi(\mathbf{x}_i) \rangle_{\mathcal{H}} = \mu \sum_{i=1}^{n-1} e_i \kappa(\mathbf{x}_i, \mathbf{x}_n). \quad (17.40)$$

Equation (17.39) provides the KLMS estimate at time  $n$ . This is an expansion (in the RKHS  $\mathcal{H}$ ) in terms of the feature map function computed at each training point up to time  $n$ . Observe that, alternatively, one could derive Eq. (17.40), simply by applying the kernel trick to (17.36). It is important to emphasize that, in contrast to the standard LMS, where the solution is a vector of fixed dimension, which is updated at each time instant, in KLMS the solution is a function of  $\mathcal{H}$  and thus it cannot be represented by a machine. Nevertheless, as it is common in kernel-based methods, one needs not to compute the actual solution (17.39), but only the estimated output of the learning system,  $\hat{y}_n$ , which is given in terms of the kernel function centered at the points,  $\mathbf{x}_i$ ,  $i = 1, \dots, n-1$ , of the given sequence (Eq. (17.40)). Under this setting, the algorithm needs to store into memory two pieces of information: a) the centers (training input points),  $\mathbf{x}_i$ , of the expansion (17.39), which are stored into a dictionary  $\mathcal{D}$  and b) the coefficients of (17.39), i.e.,  $\mu e_i$ , which are represented by a growing vector  $\theta$ . Observe that (17.39) is inline with what we know by the Representer theorem given in Section 1.17.5 (Theorem 16). Algorithm 27 summarizes this procedure.

If in place of (17.38), we use its normalized version, i.e.,

$$\begin{aligned} f_n &= f_{n-1} + \frac{\mu}{\kappa(\mathbf{x}_n, \mathbf{x}_n)} e_n \Phi(\mathbf{x}_n) \\ &= f_{n-1} + \frac{\mu}{\kappa(\mathbf{x}_n, \mathbf{x}_n)} e_n \kappa(\cdot, \mathbf{x}_n), \end{aligned} \quad (17.41)$$

**Algorithm 27.** Kernel LMS (KLMS)

---

**Require:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n), \dots$

- 1: Set  $\boldsymbol{\theta} = 0$ ,  $\mathcal{D} = \emptyset$ . Select the step parameter  $\mu$  and the parameters of the kernel.
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3:   **if**  $n = 1$  **then**
- 4:      $\hat{y}_n = 0$ .
- 5:   **else**
- 6:     Compute the learning system's output:  $\hat{y}_n = \sum_{i=1}^{n-1} \theta_i \kappa(\mathbf{u}_i, \mathbf{x}_n)$ .
- 7:   **end if**
- 8:     Compute the error:  $e_n = y_n - \hat{y}_n$ .
- 9:      $\theta_n = \mu e_n$ .
- 10:   Add the new center  $\mathbf{u}_n = \mathbf{x}_n$  to the list of centers, i.e.,  $\mathcal{D} = \mathcal{D} \cup \{\mathbf{x}_n\}$ ,  $\boldsymbol{\theta} = (\boldsymbol{\theta}^T, \theta_n)^T$ .
- 11: **end for**

---

where  $\mu \in (0, 2)$ , the normalized KLMS counterpart (KNLMS) results. This comprises replacing in Algorithm 27 the step  $\theta_n = \mu e_n$  by  $\theta_n = \mu e_n / \gamma$ , where  $\gamma = \kappa(\mathbf{x}_n, \mathbf{x}_n)$ . The convergence and stability properties of KLMS is, still, a hot topic for research. One may consider that, as the KLMS is the LMS in a RKHS  $\mathcal{H}$ , the properties of the LMS are directly transferred to the KLMS [37]. However, we should note that the properties of the LMS have been proved for Euclidean spaces, while very often the RKHS used in practice are of infinite dimension.

### 1.17.6.2.1 Sparsifying the solution

The main drawback of the KLMS algorithm is that a growing number of training points,  $\mathbf{x}_n$ , is involved in the estimation of the learning system's output (17.40). The set of these points can be thought of as a “dictionary,”  $\mathcal{D}$ , which is stored into the memory, as in Algorithm 27. This is a typical situation that arises in any kernel-based online learning scheme. As a consequence, increasing memory and computational resources are needed, as time evolves (the complexity of KLMS, as presented in Section 1.17.6.2 is  $O(n)$  at each iteration). Furthermore, it is impossible to use the KLMS in real world's applications, since the dictionary grows unbounded. In such a case, after a significant amount of iterations the expansion (17.40) will become so large that it will fill up the memory of the machine. In addition, it will require a huge amount of time to be computed, rendering the application useless. Hence, it is important to find a way to limit the size of this expansion. Several strategies have been proposed to cope with this problem. According to these strategies, the dictionary of points is created at the beginning of the algorithm and new points are inserted into it, only if they satisfy a specific rule, as in Algorithm 28. Adopting such a strategy, the complexity of KLMS is reduced to  $O(M_n)$ , where  $M_n$  is the size of the dictionary at time instant  $n$ .

For example, in the popular Plat's *novelty criterion* sparsification scheme [38], whenever a new data pair  $(\mathbf{x}_n, y_n)$  is considered, a decision is immediately made of whether to add the new point,  $\mathbf{x}_n$ , to  $\mathcal{D}$ . The decision is reached following two simple rules:

**Algorithm 28.** Kernel LMS (KLMS) with sparsification

---

**Require:**  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n), \dots$

- 1: Set  $\boldsymbol{\theta} = 0$ ,  $\mathcal{D} = \emptyset$ ,  $M = 0$ . Select the step parameter  $\mu$  and the parameters of the kernel.
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3:   **if**  $n = 1$  **then**
- 4:      $\hat{y}_n = 0$ .
- 5:   **else**
- 6:     Compute the learning system's output:  $\hat{y}_n = \sum_{i=1}^M \theta_i \kappa(\mathbf{u}_i, \mathbf{x}_n)$ .
- 7:   **end if**
- 8:     Compute the error:  $e_n = y_n - \hat{y}_n$ .
- 9:      $\theta_n = \mu e_n$ .
- 10:   Check the Sparsification Rules.
- 11:   **if** Sparsification Rules are not satisfied, **then**
- 12:      $M = M + 1$ .
- 13:     Add the new center  $\mathbf{u}_M = \mathbf{x}_n$  to the list of centers, i.e.,  $\mathcal{D} = \mathcal{D} \cup \{\mathbf{x}_n\}$ ,  $\boldsymbol{\theta} = (\boldsymbol{\theta}^T, \theta_n)^T$ .
- 14:   **end if**
- 15: **end for**

**Output:** The  $M$ -dimensional vector  $\boldsymbol{\theta}$  and the dictionary  $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_M\}$ .  
The solution is then given as  $f_n = \sum_{i=1}^M \theta_i \kappa(\mathbf{u}_i, \cdot)$ .

---

- First, the distance of the new point,  $\mathbf{x}_n$ , from the current dictionary,  $\mathcal{D}_{n-1}$ , is evaluated:

$$d(\mathbf{x}_n, \mathcal{D}_{n-1}) = \inf_{\mathbf{u}_k \in \mathcal{D}_{n-1}} \{\|\mathbf{x}_n - \mathbf{u}_k\|\}.$$

If this distance is smaller than a given threshold  $\delta_1$  (i.e., the new input vector is close to a point, which is already in the existing dictionary), then the newly arrived point is not added to  $\mathcal{D}_{n-1}$ . Thus  $\mathcal{D}_n = \mathcal{D}_{n-1}$ .

- Otherwise, we compute the prediction error  $e_n = y_n - \hat{d}_n$ . If  $|e_n|$  is smaller than a predefined threshold,  $\delta_2$ , then the new point is discarded and we set  $\mathcal{D}_n = \mathcal{D}_{n-1}$ . Only if  $|e_n| \geq \delta_2$ , then  $\mathbf{x}_n$  is inserted into  $\mathcal{D}_{n-1}$ , forming the new dictionary  $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n\}$ .

Note that whenever we insert a new point into  $\mathcal{D}$ , we must also insert the respective coefficient,  $\theta_n = \mu e_n$ , to the growing vector  $\boldsymbol{\theta}$ .

Another popular scheme is the so called *coherence-based* sparsification strategy [39], where the point  $\mathbf{x}_n$  is inserted into the dictionary, if its coherence is above a given threshold  $\epsilon_0$ , i.e.

$$\max_{\mathbf{u}_i \in \mathcal{D}_n} \{|\kappa(\mathbf{x}_n, \mathbf{u}_i)|\} \leq \epsilon_0. \quad (17.42)$$

It has been proved that the dimension of the dictionary determined under rule (17.42) remains finite, as  $n$  goes to infinity [39].

A more sophisticated strategy is the so called *surprise criterion* [37], which employs ideas from information theoretic learning. The *surprise* of a new data pair  $(\mathbf{x}_n, y_n)$  with respect to a learning

system  $\mathcal{T}$  is defined as the negative log likelihood of  $(\mathbf{x}_n, y_n)$ , given the learning system's hypothesis on the data distribution, i.e.,

$$S_{\mathcal{T}}(\mathbf{x}_n, y_n) = -\ln p((\mathbf{x}_n, y_n)|\mathcal{T}).$$

According to this measure, one may classify the new data pair into the following three categories:

- *Abnormal*:  $S_{\mathcal{T}}(\mathbf{x}_n, y_n) > \delta_1$ ,
- *Learnable*:  $\delta_1 \geq S_{\mathcal{T}}(\mathbf{x}_n, y_n) \geq \delta_2$ ,
- *Redundant*:  $S_{\mathcal{T}}(\mathbf{x}_n, y_n) < \delta_2$ ,

where  $\delta_1, \delta_2$  are problem related parameters. In surprise-based sparsification, if the new data pair is classified as learnable, it is inserted into the dictionary. For the case of the KLMS with Gaussian inputs, the surprise measure is

$$S_n = \frac{1}{2} \ln(r_n) + \frac{e_n^2}{2r_n}, \quad (17.43)$$

where

$$r_n = \lambda + \kappa(\mathbf{x}_n, \mathbf{x}_n) - \max_{\mathbf{u}_i \in \mathcal{D}_n} \left\{ \frac{\kappa^2(\mathbf{x}_n, \mathbf{u}_i)}{\kappa(\mathbf{u}_i, \mathbf{u}_i)} \right\}$$

and  $\lambda$  is a user-defined regularization parameter.

A main drawback of the aforementioned sparsification techniques is that if one data point is inserted in the dictionary, it remains in it indefinitely. This may effect the tracking performance of the algorithm in time varying environments. Another technique, that one can adopt to impose sparsity on the solution of the KLMS, that, in addition, changes the coefficients of the solution's expansion, is the *quantization* of the training data in the input space, as in the Quantized KLMS [40]. While each data point  $\mathbf{x}_n$  arrives sequentially, the algorithm checks if it is a new point or a redundant point. If its distance from the current dictionary  $\mathcal{D}_n$  is greater than or equal to the quantization size  $\delta$  (i.e.,  $\mathbf{x}_n$  cannot be quantized to a point already contained in  $\mathcal{D}_n$ ) then  $\mathbf{x}_n$  is classified as a new point and it is inserted into the dictionary  $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{u}_n\}$ . Otherwise,  $\mathbf{x}_n$  is classified as a “redundant” point and the algorithm uses this information to update the coefficient of the closest center, say  $\mathbf{u}_{l_0} \in \mathcal{D}_n$ . Algorithm 29 summarizes this procedure.

An alternative method to impose sparsity in an adaptive setting has been considered in [41–43]. This is based on regularization and it has the additional benefit of introducing an exponential forgetting mechanism of past data. This mechanism is very natural to the projection-based philosophy of these algorithms, and sparsification only requires an extra projection onto a convex ball.

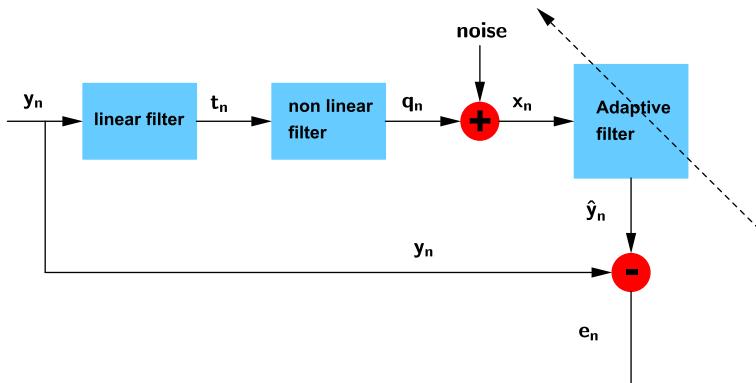
### 1.17.6.2.2 Simulation results

To demonstrate the performance of the KLMS, we consider a typical nonlinear channel equalization task (see Figure 17.15) similar to the one presented in [44,45]. The nonlinear channel consists of a linear filter:

$$t_n = -0.8 \cdot y_n + 0.7 \cdot y_{n-1},$$

and a memoryless nonlinearity

$$q_n = t_n + 0.25 \cdot t_n^2 + 0.11 \cdot t_n^3. \quad (17.44)$$

**FIGURE 17.15**

The equalization task.

**Algorithm 29.** Quantized Kernel LMS (QKLMS)

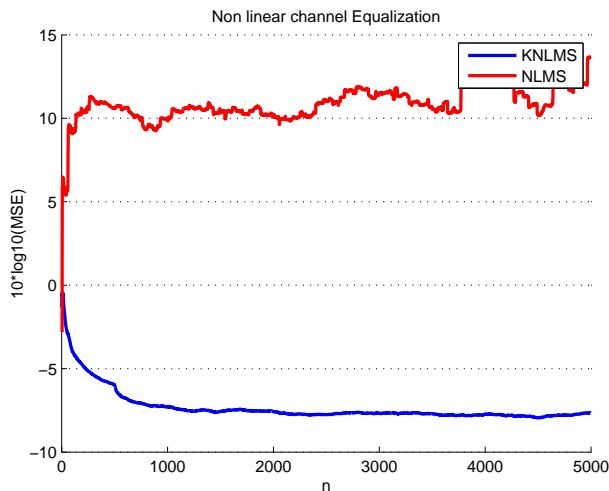
---

**Require:**  $(x_1, y_1), \dots, (x_n, y_n), \dots$

- 1: Set  $\theta = 0, \mathcal{D} = \emptyset, M = 0$ . Select the step parameter  $\mu$ , the parameters of the kernel and the quantization size  $\delta$ .
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3:   **if**  $n = 1$  **then**
- 4:      $\hat{y}_n = 0$ .
- 5:   **else**
- 6:     Compute the learning system's output:  $\hat{y}_n = \sum_{i=1}^M \theta_i \cdot \kappa(u_i, x_n)$ .
- 7:   **end if**
- 8:     Compute the error:  $e_n = y_n - \hat{y}_n$ .
- 9:      $\theta_n = \mu e_n$ .
- 10:   Compute the distance of  $x_n$  from  $\mathcal{D}$ :  $d(x_n, \mathcal{D}) = \inf_{u_i \in \mathcal{D}} \|x_n - u_i\| = \|x_n - u_{l_0}\|$ , for an  $l_0 \in \{1, \dots, M\}$ .
- 11:   **if**  $d(x_n, \mathcal{D}) > \delta$ , **then**
- 12:      $M = M + 1$ .
- 13:     Add the new center  $u_M = x_n$  to the list of centers, i.e.,  $\mathcal{D} = \mathcal{D} \cup \{u_n\}, \theta = (\theta^T, \theta_n)^T$ .
- 14:   **else**
- 15:     Keep the codebook unchanged and update the coefficient  $\theta_{l_0}$ , i.e.,  $\theta_{l_0} = \theta_{l_0} + \mu e_n$ .
- 16:   **end if**
- 17:     The  $M$ -dimensional vector  $\theta$  and the dictionary  $\mathcal{D} = \{u_1, \dots, u_M\}$ .  
The solution is then given as  $f_n = \sum_{i=1}^M \theta_i \kappa(\cdot, u_i)$ .
- 18: **end for**

---

The signal is then corrupted by additive white Gaussian noise and then it is observed as  $x_n$ . The level of the noise was set equal to 15 dB. The aim of a channel equalization task is to design an inverse filter, which acts on the output of the channel,  $x_n$ , and reproduces the original input signal,  $y_n$ , as close as

**FIGURE 17.16**

Learning curves of normalized LMS ( $\mu = 1$ ) and the normalized Gaussian Kernel LMS ( $\mu = 1/2, \sigma = 5$ ) for the equalization problem (filter length  $l = 5$ , delay  $D = 2$ ).

possible. To this end, we apply the normalized KLMS algorithm to the set of samples

$$(\mathbf{x}_n, y_{n-D}) := ((x_n, x_{n-1}, \dots, x_{n-l+1}), y_{n-D}), \quad (17.45)$$

where  $l > 0$  is the equalizer's length and  $D$  the equalization time delay, which is present to, almost, any equalization set up. In other words, the output of the equalizer at time  $n$ , provides the estimate of  $y_{n-D}$ . This is to account for the non minimum phase property of the linear filter that was used to model the channel [5].

Experiments were conducted on 50 sets of 5000 samples of the input signal (Gaussian random variable with zero mean and unit variance) comparing the standard LMS and the KLMS. Regarding the KLMS, we employed the Gaussian kernel (with  $\sigma = 5$ ) and we adopted the novelty criterion sparsification strategy ( $\delta_1 = 0.15, \delta_2 = 0.2$ ). The step update parameter,  $\mu$ , was tuned for the best possible results (in terms of the steady-state error rate). Time delay  $D$  was also set for optimality. Figure 17.16 shows the learning curves of KLMS versus the standard LMS algorithm. The superiority of the KLMS is evident and it was expected, since the standard LMS cannot cope with the nonlinearity.

Figures 17.17a and 17.17b compare the performance of the various sparsification strategies presented in Section 1.17.6.2.1, considering the same experimental set up. The user-defined parameters for the novelty detection approach were set to  $\delta_1 = 0.15, \delta_2 = 0.2$ , for the coherence-based scenario were set to  $\epsilon_0 = 0.99$  and for the surprise-based sparsification were set to  $\lambda = 0.01, \delta_1 = 2, \delta_2 = -2$ . In this experiment, one observes that although both the surprise and the coherence-based scenarios perform almost equally well in terms of sparsity, the latter gives a slightly lower steady state MSE, whereas the novelty detection approach outperforms the other two both in terms of sparsity and the steady state MSE. Moreover, we should note that the coherence-based and the surprise strategies are

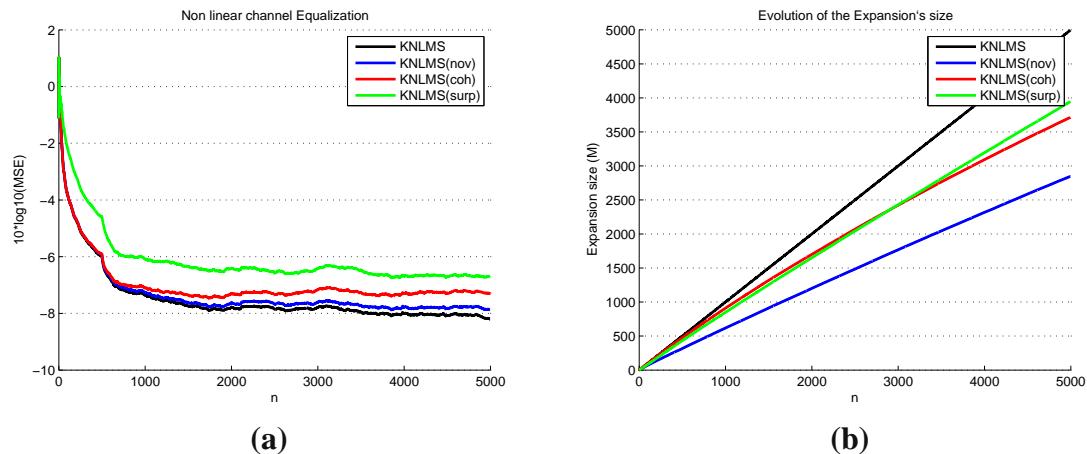


FIGURE 17.17

(a) Learning curves of the normalized Kernel LMS (KNLMS) with  $\mu = 1/2$ , using the Gaussian kernel ( $\sigma = 5$ ), under various sparsification strategies for the equalization problem (filter length  $l = 5$ , time delay  $D = 2$ ).  
(b) Evolution of the expansion's size at each time instant, for each sparsification strategy. The user-defined parameters for the novelty detection approach were set to  $\delta_1 = 0.15$ ,  $\delta_2 = 0.2$ , for the coherence-based scenario were set to  $\epsilon_0 = 0.99$  and for the surprise-based sparsification were set to  $\lambda = 0.01$ ,  $\delta_1 = 2$ ,  $\delta_2 = -2$ .

more computationally demanding compared to the novelty criterion, as they involve several evaluations of the kernel function. Figure 17.18 compares the learning curves and the expansion' size evolution of the QKLMS and the novelty detection KLMS at the same experimental set up. There it is shown that QKLMS, while offering a significant sparser solution, at the same time accomplishes a lower steady state MSE. In fact, the convergence behavior of QKLMS is almost identical to the KLMS without sparsification! Moreover, in terms of computational resources QKLMS is comparable to the novelty detection sparsification scenario.

Finally, Figures 17.19 and 17.20 compare the performances of the aforementioned sparsification strategies in a set up where the channel undergoes a significant change. In this particular experiment, the input signal was fed to a channel consisting of the linear part  $t_n = -0.8y_n + 0.7y_{n-1} - 0.6y_{n-2} + 0.1y_{n-3}$  and the nonlinear part  $q_n = t_n + 0.08t_n^2$ . The signal was then corrupted by additive white Gaussian noise and then observed as  $x_n$ . After the 2000th time instant the channel changes so that  $t_n = 0.6y_n - 0.9y_{n-1} + 0.5y_{n-2} - 0.4y_{n-3}$  and  $q_n = t_n + 0.1t_n^2$ . Experiments were conducted on 50 sets of 4000 samples of the input signal (Gaussian random variable with zero mean and unit variance). The QKLMS in this test gives excellent results, as it achieves the same steady state MSE as the unsparsified KLMS using only the 1/8th of the training centers. The three sparsification strategies (novelty, coherence, surprise) carry the information learned while the channel was in its original state throughout the learning phase. However, the two first (novelty and coherence) seem to be able to cope with the change of the channel, while the latter fails. The code for the experiments presented in this section can be found in <http://bouboulis.mysch.gr/kernels.html>. More simulation results for the KLMS

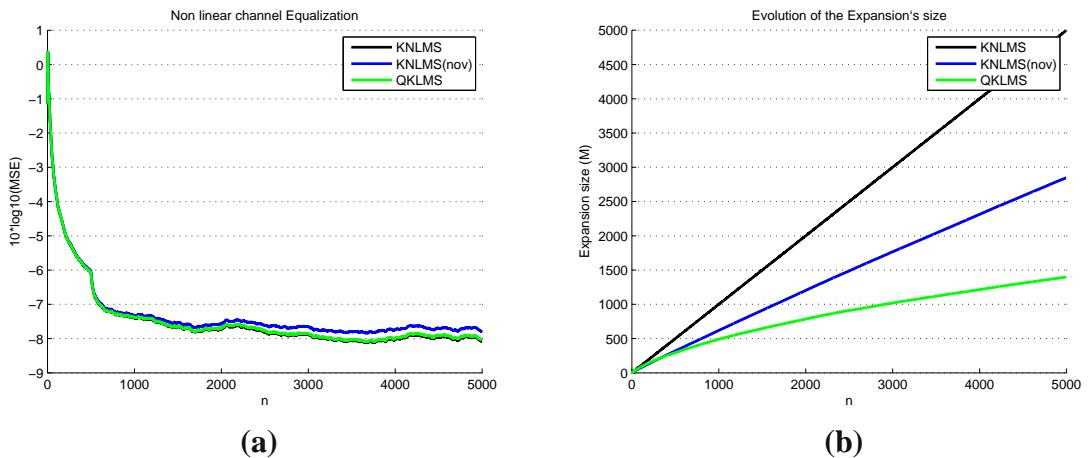


FIGURE 17.18

(a) Learning curves of the normalized Kernel LMS (KNLMS) with  $\mu = 1/2$ , using the Gaussian kernel ( $\sigma = 5$ ), the novelty-detection normalized KNLMS and the QKLMS for the equalization problem (filter length  $l = 5$ , time delay  $D = 2$ ). (b) Evolution of the expansion's size at each time instant, for the KNLMS, novelty KNLMS, QKNLMS. The user-defined parameters for the novelty detection approach were set to  $\delta_1 = 0.15$ ,  $\delta_2 = 0.2$ . The quantization size (for the QKNLMS) was set  $\delta = 1$ .

algorithm are given in Sections 1.17.6.6.1 and 1.17.7.6.2, where it is validated against other kernel-based online learning methodologies, such as the Kernel RLS, and the projection-based Adaptive Projected Subgradient Method (APSM).

### 1.17.6.3 The complex case

Complex-valued signals arise frequently in applications as diverse as communications, bioinformatics, radar, etc. In this section we present the LMS rationale suitably adjusted to treat data of this nature.

#### 1.17.6.3.1 Complex LMS

Let  $z_n \in \mathbb{C}^v$  and  $d_n \in \mathbb{C}$  be the input and the output of the original filter, respectively. In the typical Complex LMS (CLMS) rationale, we estimate the output of the filter using the so called C-linear function:  $\hat{d}_n = f_\theta(z_n) = \theta^H z_n$ . The CLMS aims to compute  $\theta \in \mathbb{C}^v$ , such that the error  $\mathcal{L}(\theta) = E[|d_n - \theta^H z_n|^2]$  is minimized. To compute the respective gradients, the notion of the Wirtinger's Calculus [46,47] is employed as it simplifies calculations. In a nutshell, Wirtinger's Calculus considers two types of derivatives, with respect to  $z$  and  $z^*$  respectively (following the standard differentiation rules). However, only the second one is important for optimization tasks. The alternative to using Wirtinger's calculus would be to consider the complex variables as pairs of two real ones and employ the common real partial derivatives. However, this approach, usually, is more time consuming and leads to more cumbersome expressions. Hence, applying the rules of Wirtinger's calculus to this case, we

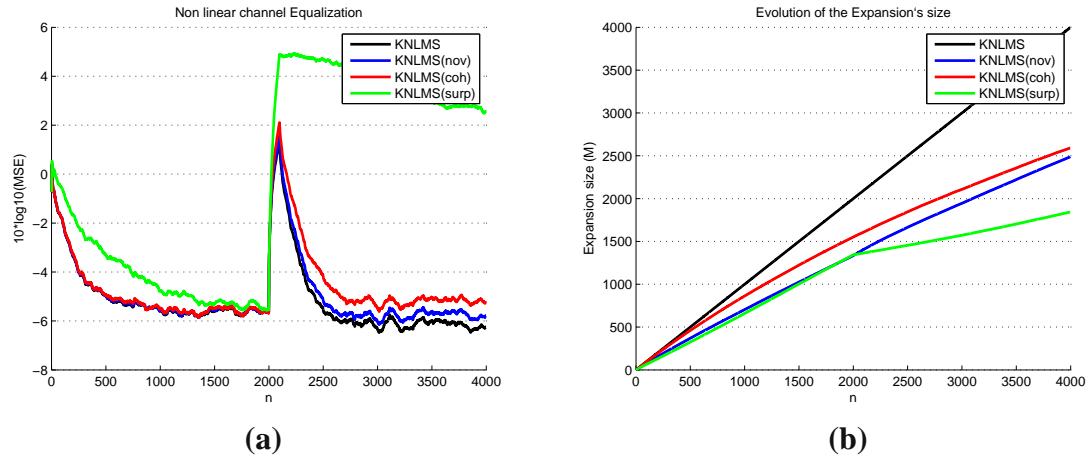


FIGURE 17.19

(a) Learning curves of the normalized Kernel LMS (KNLMS) with  $\mu = 1/2$ , using the Gaussian kernel ( $\sigma = 5$ ), under various sparsification strategies for the two states equalization problem (filter length  $l = 5$ , time delay  $D = 2$ ). (b) Evolution of the expansion's size at each time instant, for each sparsification strategy. The user-defined parameters for the novelty detection approach were set to  $\delta_1 = 0.15$ ,  $\delta_2 = 0.2$ , for the coherence-based scenario were set to  $\epsilon_0 = 0.99$  and for the surprise-based sparsification were set to  $\lambda = 0.01$ ,  $\delta_1 = 2$ ,  $\delta_2 = -2$ .

take that  $\frac{\partial \mathcal{L}(\theta)}{\partial \theta^*} = -e_n^* z_n$ , where  $e_n = d_n - \hat{d}_n$ . Thus, the step update for the CLMS is

$$\theta_n = \theta_{n-1} + \mu e_n^* z_n,$$

and the estimated output of the filter at iteration  $n$ , assuming that  $\theta_0 = \mathbf{0}$ , is

$$\hat{d}_n = \mu \sum_{i=1}^{n-1} e_i z_i^H z_n.$$

A different filtering structure has been brought to light more recently, mainly due to the works of Picinbono [48–50]. It is based on the notion of the *widely linear estimation functions*, where the output is estimated so that to be linear in terms of both  $z$  and  $z^*$ , i.e.,  $\tilde{d}_n = f_\theta, \eta(z_n) = \theta^H z_n + \eta^H z_n^*$ . It turns out that the traditional approach to linearity, as employed by the CLMS, is rather “unorthodox,” as it excludes a large class of linear functions from being considered in the estimation process. Moreover, it can be shown that the correct complex linear estimation is the widely linear one [51]. In the context of the widely linear, or Augmented, CLMS (ACLMS), the vectors  $\theta, \eta \in \mathbb{C}^v$  are computed, so that the error  $\mathcal{L}(\theta) = E[|d_n - f_{\theta,\eta}(z_n)|^2]$  is minimized. Applying Wirtinger’s Calculus we compute the respective gradients as  $\frac{\partial \mathcal{L}(\theta,\eta)}{\partial \theta^*} = -e_n^* z_n$  and  $\frac{\partial \mathcal{L}(\theta,\eta)}{\partial \eta^*} = -e_n^* z_n^*$ . Thus, the step updates of

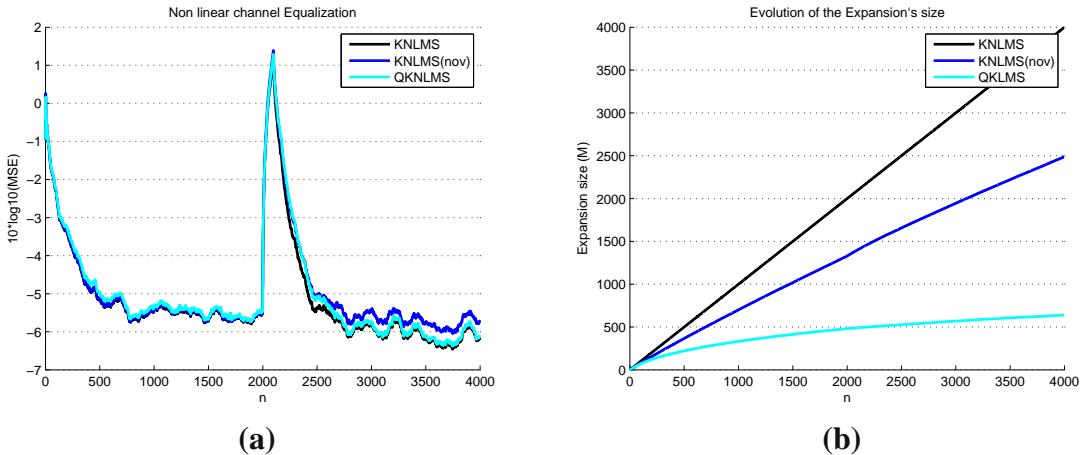


FIGURE 17.20

(a) Learning curves of the normalized Kernel LMS (KNLMS) with  $\mu = 1/2$ , using the Gaussian kernel ( $\sigma = 5$ ), the novelty-detection normalized KNLMS and the QKLMS for the two-states equalization problem (filter length  $l = 5$ , time delay  $D = 2$ ). (b) Evolution of the expansion's size at each time instant, for the KNLMS, novelty KNLMS, QKNLMS. The user-defined parameters for the novelty detection approach were set to  $\delta_1 = 0.15$ ,  $\delta_2 = 0.2$ . The quantization size (for the QKNLMS) was set  $\delta = 1$ .

the ACLMS become

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu e_n^* \mathbf{z}_n, \quad (17.46a)$$

$$\boldsymbol{\eta}_n = \boldsymbol{\eta}_{n-1} + \mu e_n^* \mathbf{z}_n^*, \quad (17.46b)$$

and the estimated output at iteration  $n$ , assuming that  $\boldsymbol{\theta}_0 = \boldsymbol{\eta}_0 = \mathbf{0}$ , is

$$\tilde{d}_n = \mu \sum_{i=1}^{n-1} e_i (\mathbf{z}_i + \mathbf{z}_i^*)^H \mathbf{z}_n. \quad (17.47)$$

An important notion that has been associated with the case of complex data is the so called *circularity*. Circularity is intimately related to the rotation in the geometric sense. A complex random variable  $Z$  is called circular, if for any angle  $\phi$  both  $Z$  and  $Ze^{i\phi}$  (i.e., the rotation of  $Z$  by angle  $\phi$ ) follow the same probability distribution [52,53]. It has been shown that, in the case of circular inputs, ACLMS and CLMS have identical performances [48,49,52,53]. However, ACLMS may show some performance gains when non-circular input sources are considered, although its convergence rate is somewhat slower than the CLMS. This is due to the fact that widely linear estimation processes are able to capture the full second-order statistics of a given complex signal, especially in the case where the signal is non-circular, by considering both the covariance and the pseudo-covariance matrices [49,50].

### 1.17.6.3.2 Complex kernel LMS

Following a similar procedure, as the one adopted in the derivation of the KLMS from the standard LMS, kernel-based analogues of the CLMS can be produced. In general, to generate kernel adaptive

filtering algorithms on complex domains, according to [54], one can adopt two methodologies. A first straightforward approach is to use directly a complex RKHS, using one of the complex kernels given in Section 1.17.5, and map the original data to the complex RKHS through the associated feature map  $\Phi_{\mathbb{C}}(z) = \kappa_{\mathbb{C}}(\cdot, z)$ . This is equivalent to applying the kernel trick to the standard complex LMS rationale in the usual black-box approach. Another alternative technique is to use real kernels through a rationale that is called *complexification* of real RKHSs. This method has the advantage of allowing modeling in complex RKHSs, using popular well-established and well understood, from a performance point of view, real kernels (e.g., Gaussian, polynomial, etc). While in the first case, we map the data directly to the complex RKHS through the feature map, in the complexification scenario we employ the map  $\Phi(z) = \Phi_{\mathbb{R}}(x, y) + i\Phi_{\mathbb{R}}(x, y)$ , where  $z = x + iy$ , and  $\Phi_{\mathbb{R}}$  is the feature map of the chosen real kernel  $\kappa_{\mathbb{R}}$ , i.e.,  $\Phi_{\mathbb{R}}(x, y) = \kappa_{\mathbb{R}}(\cdot, (x, y))$ . In the following, we will deal with the first scenario only. The complexification case is treated in [54].

Consider the complex RKHS  $\mathbb{H}$ , with the respective complex reproducing kernel  $\kappa_{\mathbb{C}}(\cdot, \cdot)$  (meaning that it takes complex arguments and returns a complex number) defined on  $X \times X$ , where  $X \subseteq \mathbb{C}^v$ . We apply the feature map  $\Phi_{\mathbb{C}}$  of the RKHS,  $\mathbb{H}$ , to the points  $z_n$  and employ the LMS rationale to the transformed data  $\{(\Phi_{\mathbb{C}}(z_n), d_n) : n = 1, 2, \dots\}$ , modeling the desired output either as  $\hat{d}_n = \langle \Phi_{\mathbb{C}}(z_n), f \rangle_{\mathbb{H}}$ , for some  $f \in \mathbb{H}$ , or as  $\tilde{d}_n = \langle \Phi_{\mathbb{C}}(z_n), f \rangle_{\mathbb{H}} + \langle \Phi_{\mathbb{C}}^*(z_n), g \rangle_{\mathbb{H}}$ , for some  $f, g \in \mathbb{H}$ , if the more general widely linear approach is considered. In the first case, considering the corresponding loss as  $\mathcal{L}(f) = E[|d_n - \hat{d}_n|^2]$ , the Pure Complex KLMS (PCKLMS) is generated. To this end, the gradient descent rationale is adopted and the gradient of the loss function is estimated via its current measurement. To this end, Wirtinger's calculus had to be generalized in the framework of Freshet derivation, to cope with infinite dimensional complex Hilbert spaces, [55]. It turns out that,  $\nabla_{f^*}\mathcal{L}(f) = -E[e_n^*\Phi_{\mathbb{C}}(z_n)] \approx -e_n^*\Phi_{\mathbb{C}}(z_n)$ . This leads to the following step update equation:

$$f_n = f_{n-1} + \mu e_n^* \Phi_{\mathbb{C}}(z_n). \quad (17.48)$$

Assuming that  $f_0 = 0$ , and by applying a repetition of (17.48), we take formulas similar to the KLMS case. Algorithm 30 summarizes the procedure of PCKLMS.

To sparsify the solution one may employ any of the methods described in Section 1.17.6.2.1. Adopting the widely linear estimation process, i.e.,  $\tilde{d}_n = \langle \Phi_{\mathbb{C}}(z_n), f \rangle_{\mathbb{H}} + \langle \Phi_{\mathbb{C}}^*(z_n), g \rangle_{\mathbb{H}}$ , and following a similar procedure, the *Augmented Pure Complex KLMS (APCKLMS)* is produced. In this case, as the corresponding loss is  $\mathcal{L}(\theta, \eta) = E[|d_n - \tilde{d}_n|^2]$ , the respective estimation of gradients via their current measurements becomes  $\nabla_{f^*}\mathcal{L}(f, g) \approx -e_n^*\Phi_{\mathbb{C}}(z_n)$  and  $\nabla_{g^*}\mathcal{L}(f, g) \approx -e_n^*\Phi_{\mathbb{C}}^*(z_n)$ . It turns out that the step update equations become:

$$\begin{aligned} f_n &= f_{n-1} + \mu e_n^* \Phi_{\mathbb{C}}(z_n), \\ g_n &= g_{n-1} + \mu e_n^* \Phi_{\mathbb{C}}^*(z_n). \end{aligned}$$

The APCKLMS is treated in details in [51], where it is shown that it performs significantly better than the PCKLMS. The problem of complex estimation using kernels in the special case of finite dimensional Euclidean spaces has also been considered in [56].

**Algorithm 30.** Pure Complex Kernel LMS (PCKLMS)

---

**Require:**  $(z_1, d_1), \dots, (z_n, d_n), \dots$

- 1: Set  $\theta = 0$ ,  $\mathcal{D} = \emptyset$ ,  $M = 0$ . Select the step parameter  $\mu$  and the parameters of the kernel.
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3:   **if**  $n = 1$  **then**
- 4:      $\hat{d}_n = 0$ .
- 5:   **else**
- 6:     Compute the filter output:  $\hat{d}_n = \sum_{i=1}^M \theta_i \kappa_{\mathbb{C}}(\mathbf{u}_i, z_n)$ .
- 7:   **end if**
- 8:     Compute the error:  $e_n = d_n - \hat{d}_n$ .
- 9:      $\theta_n = \mu e_n$ .
- 10:   Check Sparsification Rules.
- 11:   **if** Sparsification Rules are satisfied,
- 12:      $M = M + 1$ .
- 13:     Add the new center  $\mathbf{u}_M = z_n$  to the list of centers, i.e.,  $\mathcal{D} = \mathcal{D} \cup \{z_n\}$ ,  $\theta = (\theta^T, \theta_n)^T$ .
- 14:   **end if**
- Output:** The  $M$ -dimensional vector  $\theta$  and the dictionary  $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_M\}$ .

The solution is then given as  $f_n = \sum_{i=1}^M \theta_i \kappa(\mathbf{u}_i, \cdot)$ .

15: **end for**

---

**1.17.6.4 Recursive Least Squares (RLS)**

Recall that we assumed the sequence of examples  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n), \dots\}$ , where  $\mathbf{x}_n \in X \subset \mathcal{R}^l$ , and  $y_n \in \mathcal{R}$ . In the *Recursive Least Squares* (RLS) algorithm, we adopt the parametric class of linear functions, i.e.,  $\mathcal{F} = \{f_{\theta}(\mathbf{x}) = \theta^T \mathbf{x} : \theta \in \mathcal{R}^l\}$ , as the field in which we search for the optimal input-output relationship between  $\mathbf{x}_n$  and  $y_n$ , while the loss function at iteration  $n$  takes the form:

$$\mathcal{L}_n(\theta) = \sum_{i=1}^n |y_i - f_{\theta}(\mathbf{x}_i)|^2 + \lambda \|\theta\|^2, \quad n = 1, 2, \dots, \quad (17.49)$$

for some chosen  $\lambda > 0$ . Observe that the loss function consists of two parts. The first one measures the error between the estimated output,  $f_{\theta}(\mathbf{x}_i)$ , and the actual output,  $y_i$ , at each time instant,  $i$ , up to  $n$ , while the second is a regularization term. The existence of the second term is crucial to the algorithm, as it prevents it from being ill-posed and guards against overfitting. At each time instant,  $n$ , the RLS finds the solution to the minimization problem:

$$\min_{\theta} \mathcal{L}_n(\theta). \quad (17.50)$$

The RLS algorithm, usually, exhibits an order of magnitude faster convergence compared to the LMS, at the cost of higher computational complexity.

The rationale of the RLS algorithm can be described in a significant more compact form, if one adopts a matrix representation. To this end, we define the  $l \times n$  matrices  $X_n = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$  and

the vectors  $\mathbf{y}_n = (y_1, y_2, \dots, y_n)^T \in \mathcal{R}^n$ , for  $n = 1, 2, \dots$ . Then, the loss function (17.49) can be equivalently defined as:

$$\begin{aligned}\mathcal{L}_n(\boldsymbol{\theta}) &= \left\| \mathbf{y}_n - \mathbf{X}_n^T \boldsymbol{\theta} \right\|^2 + \lambda \|\boldsymbol{\theta}\|^2 = (\mathbf{y}_n - \mathbf{X}_n^T \boldsymbol{\theta})^T (\mathbf{y}_n - \mathbf{X}_n^T \boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|^2 \\ &= \|\mathbf{y}_n\|^2 - 2\boldsymbol{\theta}^T \mathbf{X}_n \mathbf{y}_n + \boldsymbol{\theta}^T \mathbf{X}_n \mathbf{X}_n^T \boldsymbol{\theta} + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}.\end{aligned}$$

As this is a strictly convex function, it has a unique minimum, which it is obtained at the point,  $\boldsymbol{\theta}_n := \operatorname{argmin}_{\boldsymbol{\theta}} \{\mathcal{L}_n(\boldsymbol{\theta})\}$ , where its gradient vanishes, i.e.,  $\nabla \mathcal{L}_n(\boldsymbol{\theta}_n) = 0$ . Note, that the gradient of the loss function can be derived using standard differentiation rules:

$$\nabla \mathcal{L}_n(\boldsymbol{\theta}) = -2\mathbf{X}_n \mathbf{y}_n + 2\mathbf{X}_n \mathbf{X}_n^T \boldsymbol{\theta} + 2\boldsymbol{\theta}. \quad (17.51)$$

Equating the gradient to zero, we obtain the solution to the minimization problem (17.50):

$$\boldsymbol{\theta}_n = (\lambda \mathbf{I}_l + \mathbf{X}_n \mathbf{X}_n^T)^{-1} \mathbf{X}_n \mathbf{y}_n, \quad (17.52)$$

where  $\mathbf{I}_l$  is the  $l \times l$  identity matrix. Observe, that this solution includes the inversion of a matrix at each time step, which is usually, a computationally demanding procedure. To overcome this obstacle, the RLS algorithm computes the solution recursively, exploiting the matrix inversion lemma:

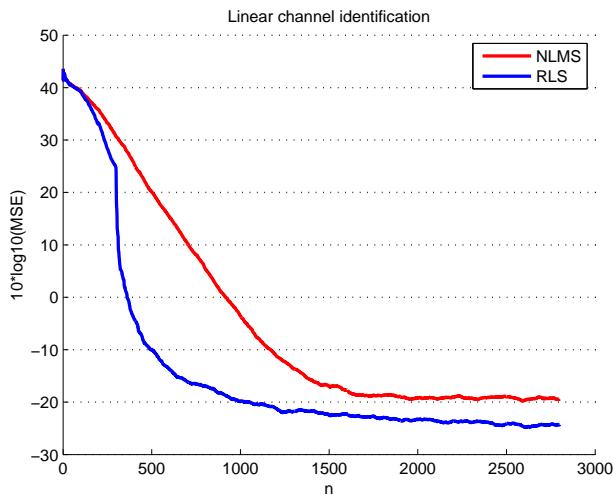
$$(\mathbf{A} + \mathbf{B} \mathbf{C} \mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{C}^{-1} + \mathbf{D} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{D} \mathbf{A}^{-1}. \quad (17.53)$$

Let  $\mathbf{P}_n = \lambda \mathbf{I}_l + \mathbf{x}_n \mathbf{x}_n^T$ . Then, we can easily deduce that  $\mathbf{P}_n = \mathbf{P}_{n-1} + \mathbf{x}_n \mathbf{x}_n^T$ . Thus, using the matrix inversion lemma (17.53) we take:

$$\mathbf{P}_n^{-1} = \left( \mathbf{P}_{n-1} + \mathbf{x}_n \mathbf{x}_n^T \right)^{-1} = \mathbf{P}_{n-1}^{-1} - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1}}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n}.$$

Also, considering that  $\mathbf{X}_n \mathbf{y}_n = \mathbf{X}_{n-1} \mathbf{y}_{n-1} + \mathbf{x}_n y_n$ , the solution  $\boldsymbol{\theta}_n$  becomes

$$\begin{aligned}\boldsymbol{\theta}_n &= \mathbf{P}_n \mathbf{X}_n \mathbf{y}_n = \left( \mathbf{P}_{n-1}^{-1} - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1}}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} \right) (\mathbf{X}_{n-1} \mathbf{y}_{n-1} + \mathbf{x}_n y_n) \\ &= \mathbf{P}_{n-1}^{-1} \mathbf{X}_{n-1} \mathbf{y}_{n-1} + \mathbf{P}_{n-1}^{-1} \mathbf{x}_n y_n - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{P}_{n-1}^{-1} \mathbf{X}_{n-1} \mathbf{y}_{n-1}}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{P}_{n-1}^{-1} \mathbf{x}_n y_n}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} \\ &= \boldsymbol{\theta}_{n-1} - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \boldsymbol{\theta}_{n-1}}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} + \left( \mathbf{P}_{n-1}^{-1} \mathbf{x}_n y_n - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^T \mathbf{x}_n y_n}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} \right) \\ &= \boldsymbol{\theta}_{n-1} - \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \boldsymbol{\theta}_{n-1}}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} + \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n y_n}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} \\ &= \boldsymbol{\theta}_{n-1} + \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} (y_n - \mathbf{x}_n^T \boldsymbol{\theta}_{n-1}).\end{aligned}$$

**FIGURE 17.21**

Learning curves of the normalized LMS ( $\mu = 1$ ) and the RLS ( $\lambda = 0.01$ ) for a filter identification problem (filter length  $l = 200$ ).

Thus, the solution  $\theta_n$  is computed recursively, as

$$\theta_n = \theta_{n-1} + \frac{\mathbf{P}_{n-1}^{-1} \mathbf{x}_n}{1 + \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} e_n, \quad (17.54)$$

where  $e_n = y_n - \theta_{n-1}^T \mathbf{x}_n$  is the estimation error at iteration  $n$ .

As can be seen in Figure 17.21, which considers a 200-taps filter identification problem (where the input signal is a Gaussian random variable with zero mean and unit variance), the RLS learning procedure typically exhibits significantly faster convergence and achieves a lower steady state error compared to the NLMS (in this example the steady state MSE achieved by the RLS is 25% lower than the one achieved by the NLMS). Such a performance is expected for the stationary environment case. The RLS is a Newton-type algorithm and the LMS a gradient descent type. Moreover, concerning the excess error for the LMS, this is due to the non-diminishing value of  $\mu$ , which as we know increases the steady state error.

### 1.17.6.5 Exponentially weighted RLS

The RLS learning rationale utilizes information of all past data at each iteration step. Although this tactic enables RLS to achieve better convergence speed, it may become a burden in time-varying environments. In such cases, it would be preferable to simply “forget” some of the past data, which correspond to an earlier state of the learning task. We can incorporate such an approach to the RLS algorithm by

introducing some weighting factors to the cost function (17.49), i.e.,

$$\mathcal{L}_n(\boldsymbol{\theta}) = \sum_{i=1}^n w(n, i) |y_i - f_{\boldsymbol{\theta}}(\mathbf{x}_i)|^2 + \lambda w(n) \|\boldsymbol{\theta}\|^2, \quad n = 1, 2, \dots, \quad (17.55)$$

where  $w$  is an appropriately chosen weighting function. A typical weighting scenario that is commonly used is the *exponential weighting factor* (or the *forgetting factor*), where we choose  $w(n, i) = w^{n-i}$  and  $w(n) = w^n$ , for some  $0 < w \leq 1$ . Small values of  $w$  (close to 0) imply a strong forgetting mechanism, which makes the algorithm more sensitive to recent samples and might lead to poor performance. Hence, we usually select  $w$  such that it is close to 1. Obviously, for  $w = 1$  we take the ordinary RLS described in 1.17.6.4. In the method of *Exponentially Weighted RLS* (EWRLS) we minimize the cost function

$$\mathcal{L}_n(\boldsymbol{\theta}) = \sum_{i=1}^n w^{n-i} |y_i - \boldsymbol{\theta}^T \mathbf{x}_i|^2 + \lambda w^n \|\boldsymbol{\theta}\|^2, \quad n = 1, 2, \dots \quad (17.56)$$

Using the matrix notation the aforementioned cost function becomes  $\mathcal{L}(\boldsymbol{\theta}) = \left\| \mathbf{W}_n^{\frac{1}{2}} (\mathbf{y}_n - \mathbf{X}_n \boldsymbol{\theta}) \right\|^2 + w^n \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}$ , where  $\mathbf{X}_n$  and  $\mathbf{y}_n$  are defined as in Section 1.17.6.4 and  $\mathbf{W}$  is the diagonal matrix with elements the powers of  $w$  up to  $n-1$ , i.e.,

$$\mathbf{W}_n = \begin{pmatrix} w^{n-1} & 0 & 0 & \dots & 0 \\ 0 & w^{n-2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & & 1 \end{pmatrix}.$$

The solution to the respective minimization problem is

$$\boldsymbol{\theta}_n = (\mathbf{X}_n \mathbf{W}_n \mathbf{X}_n^T + w^n \lambda \mathbf{I}_l)^{-1} \mathbf{X}_n \mathbf{W}_n \mathbf{y}_n. \quad (17.57)$$

Following a similar rationale as in Section 1.17.6.4, it turns out that:

$$\mathbf{P}_n^{-1} = w^{-1} \mathbf{P}_{n-1}^{-1} - \frac{w^{-2} \mathbf{P}_{n-1}^{-1} \mathbf{x}_n \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1}}{1 + w^{-1} \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n},$$

and

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \frac{w^{-1} \mathbf{P}_{n-1}^{-1} \mathbf{x}_n}{1 + w^{-1} \mathbf{x}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{x}_n} e_n.$$

Algorithm 31 describes the EWRLS machinery in details. We test the performance of the EWRLS in a time-varying channel identification problem. The signal (zero-mean unity-variance Gaussian random variable) is fed to a linear channel (length  $K = 200$ ) and then corrupted by white Gaussian noise. After 2000 samples of data, the coefficients of the linear channel undergo a significant change. Figure 17.22 shows the performance of the EWRLS algorithm compared to the standard RLS and the NLMS for

**Algorithm 31.** Exponential Weighted Recursive Least Squares (RLS)

**Require:** The data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , the weighting factor  $w$  and the regularization parameter  $\lambda$ .

- 1: Set  $\mathbf{P} = \lambda \mathbf{I}_l$ ,  $\boldsymbol{\theta} = \mathbf{0}$ .
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3: Compute the learning system's output:  $\hat{y}_n = \boldsymbol{\theta}^T \mathbf{x}_n$ .
- 4: Compute the error:  $e_n = y_n - \hat{y}_n$ .
- 5:  $r = 1 + w^{-1} \mathbf{x}_n^T \mathbf{P}^{-1} \mathbf{x}_n$ .
- 6:  $\mathbf{k} = (rw)^{-1} \mathbf{P}^{-1} \mathbf{x}_n$ .
- 7: Update solution:  $\boldsymbol{\theta} = \boldsymbol{\theta} + e_n \mathbf{k}$ .
- 8: Update the inverse matrix:  $\mathbf{P}^{-1} = w^{-1} \mathbf{P}^{-1} - \mathbf{k} \mathbf{k}^T r$ .

**Output:** The vector  $\boldsymbol{\theta} \in \mathcal{R}^l$ .

- 9: **end for**

various values of the weighting coefficient  $w$ . While the standard RLS fails to adapt to the change of the environment, the EWRLS learns the new channel effectively, at a cost of a slightly larger steady state MSE due to the weighting of the error at each time step. Note that the tracking performance of both the LMS and the RLS algorithms depends on the input signal. For some signals the LMS can track better. This is in contrast to the initial convergence speed, where the RLS always converges faster. [5, 6].

### 1.17.6.6 The Kernel RLS

Similar to the case of the KLMS, in the *Kernel RLS* (KRLS) we estimate the learning system's output by a function  $f$  defined on the RKHS,  $\mathcal{H}$ , which is associated to a positive definite kernel  $\kappa$ . The sequence of examples are transformed via the feature map  $\Phi : X \rightarrow \mathcal{H}$ ,  $\Phi(\mathbf{x}) = \kappa(\cdot, \mathbf{x})$  to generate the input data

$$\{(\Phi(\mathbf{x}_1), y_1), (\Phi(\mathbf{x}_2), y_2), \dots, (\Phi(\mathbf{x}_n), y_n), \dots\}$$

and the employed loss function takes the form

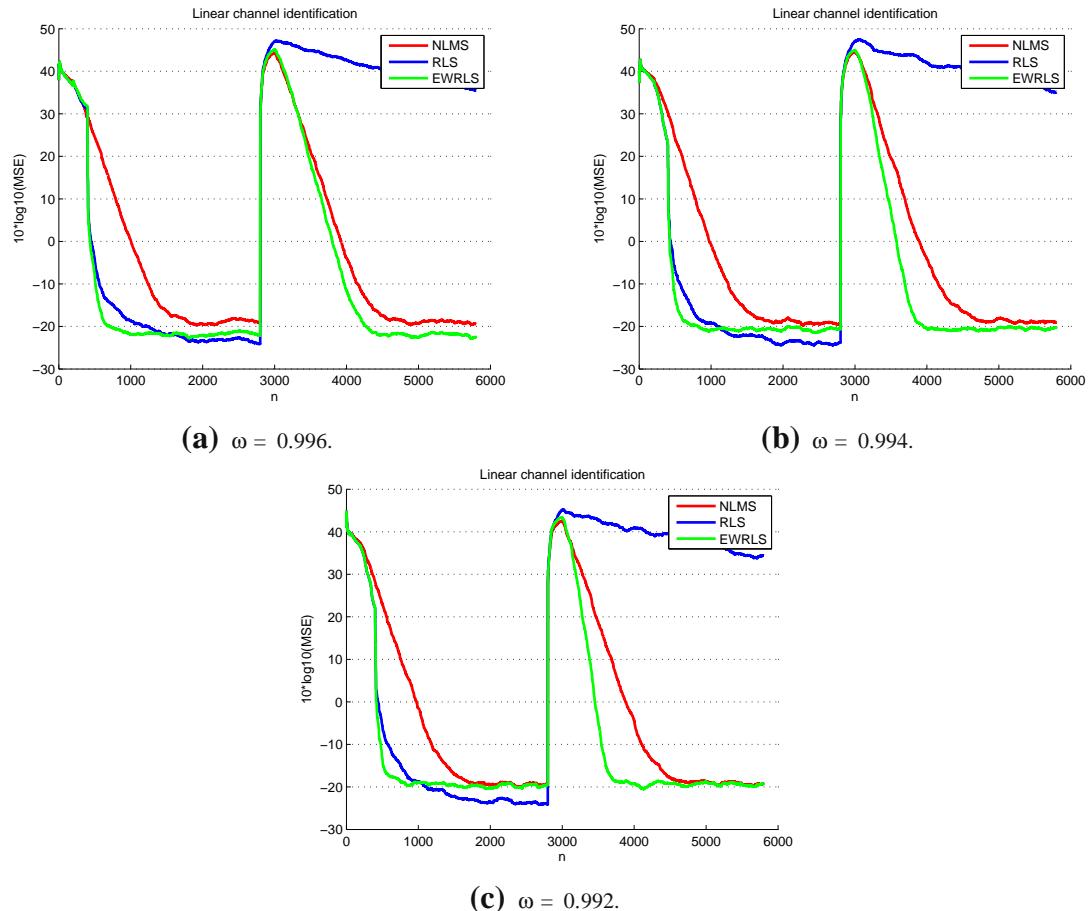
$$\mathcal{L}_n(f) = \sum_{i=1}^n |y_i - \langle \Phi(\mathbf{x}_i), f \rangle_{\mathcal{H}}|^2 + \lambda \|f\|_{\mathcal{H}}^2, \quad n = 1, 2, \dots \quad (17.58)$$

The Representer Theorem 16 ensures that the solution to the minimization task,  $f_n := \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{L}_n(f)$ , lies in the finite dimensional subspace of the span of the  $n$  kernels that are centered on the training points  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ . In other words:

$$f_n = \sum_{i=1}^n \theta_{n,i} \kappa(\cdot, \mathbf{x}_i) \quad (17.59)$$

and thus, exploiting the kernel trick, the estimated output at iteration  $n$  will be given by

$$\hat{y}_n = \langle \Phi(\mathbf{x}_n), f_n \rangle_{\mathcal{H}} = \sum_{i=1}^n \theta_{n,i} \kappa(\mathbf{x}_i, \mathbf{x}_n). \quad (17.60)$$

**FIGURE 17.22**

Learning curves of the normalized LMS ( $\mu = 1$ ), the RLS and EWRLS ( $\lambda = 0.001$ ) for various values of the weighting factor  $w$ , for a time-varying filter identification problem (filter length  $l = 200$ ).

As this expansion grows unbounded while time evolves, analogous to the case of KLMS, a sparsification mechanism, which keeps the most informative training points and discards the rest, is also needed. KRLS's main task is to estimate the (growing) vector  $\theta_n = (\theta_{n,1}, \dots, \theta_{n,n})^T$  in a recursive manner.

The first approach to the KRLS task was presented in [44] by Engel, Mannor, and Meir as a part of Engel's Ph.D. thesis. Their method is built around a specific sparsification strategy, which is called *Approximate Linear Dependency* (ALD). To this end, consider a dictionary of centers  $\mathcal{D}_{n-1} = \{\mathbf{u}_1, \dots, \mathbf{u}_{M_{n-1}}\}$ , of size  $M_{n-1}$ , that keeps some of the past training points. In this setting, we test whether the newly arrived data  $\Phi(\mathbf{x}_n)$  is approximately linearly dependent on the dictionary elements  $\Phi(\mathbf{u}_1), \dots, \Phi(\mathbf{u}_{M_{n-1}})$ . If not,

we add it to the dictionary, otherwise  $\Phi(\mathbf{x}_n)$  is approximated by a linear combination of the dictionary's elements. The specific condition is whether

$$\delta_n = \min_{\mathbf{a}_n} \left\{ \left\| \sum_{m=1}^{M_{n-1}} a_{n,m} \Phi(\mathbf{u}_m) - \Phi(\mathbf{x}_n) \right\|^2 \right\} \leq \epsilon_0, \quad (17.61)$$

where  $\epsilon_0$  is a user-defined parameter. Using matrix-notation, this minimization problem can be equivalently recasted as:

$$\delta_n = \min_{\mathbf{a}_n} \left( \mathbf{a}_n^T \tilde{\mathbf{K}}_{n-1} \mathbf{a}_n - 2\mathbf{a}_n^T \boldsymbol{\beta}_n + \kappa(\mathbf{x}_n, \mathbf{x}_n) \right),$$

where  $(\tilde{\mathbf{K}}_{n-1})_{i,j} = \kappa(\mathbf{u}_i, \mathbf{u}_j)$ , for  $i, j = 1, 2, \dots, M_{n-1}$ , and  $\boldsymbol{\beta}_n = (\kappa(\mathbf{u}_1, \mathbf{x}_n), \kappa(\mathbf{u}_2, \mathbf{x}_n), \dots, \kappa(\mathbf{u}_{M_{n-1}}, \mathbf{x}_n))^T$ . Assuming that  $\tilde{\mathbf{K}}_{n-1}$  is invertible, the solution to this minimization problem is

$$\mathbf{a}_n = \tilde{\mathbf{K}}_{n-1}^{-1} \boldsymbol{\beta}_n, \quad (17.62)$$

while the attained minimum value is  $\delta_n = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \boldsymbol{\beta}_n^T \mathbf{a}_n$ . In the case where  $\delta_n > \epsilon_0$ ,  $\mathbf{x}_n$  is added to the dictionary, which now contains  $M_n = M_{n-1} + 1$  centers. If  $\delta_n \leq \epsilon_0$ , then  $\mathbf{x}_n$  is not included to the dictionary, i.e.,  $\mathcal{D}_n = \mathcal{D}_{n-1}$ ,  $M_n = M_{n-1}$ , and we approximate  $\Phi(\mathbf{x}_n)$  as

$$\Phi(\mathbf{x}_n) \approx \sum_{m=1}^{M_n} a_{n,m} \Phi(\mathbf{u}_m). \quad (17.63)$$

Using (17.63), we can approximate the full kernel matrix  $\mathbf{K}_n$  (i.e.,  $(\mathbf{K}_n)_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ , for  $i, j = 1, \dots, n$ ) as

$$\mathbf{K}_n \approx \mathbf{A}_n \cdot \tilde{\mathbf{K}}_n \cdot \mathbf{A}_n^T, \quad (17.64)$$

where  $\mathbf{A}_n = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)^T$  is the  $M_n \times M_n$  matrix that contains the coefficients of the linear combinations associated with each  $\mathbf{u}_m$ ,  $m = 1, \dots, M_n$ . This is due to the fact that at time index  $n$  each kernel evaluation is computed as

$$\begin{aligned} \kappa(\mathbf{x}_i, \mathbf{x}_j) &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} = \left\langle \sum_{k=1}^{M_n} a_{i,k} \Phi(\mathbf{u}_k), \sum_{l=1}^{M_n} a_{j,l} \Phi(\mathbf{u}_l) \right\rangle_{\mathcal{H}} \\ &= \sum_{k=1}^{M_n} \sum_{l=1}^{M_n} a_{j,l} a_{i,k} \langle \Phi(\mathbf{u}_k), \Phi(\mathbf{u}_l) \rangle_{\mathcal{H}} = \sum_{k=1}^{M_n} \sum_{l=1}^{M_n} a_{j,l} a_{i,k} \kappa(\mathbf{u}_k, \mathbf{u}_l), \end{aligned}$$

for all  $k, l = 1, \dots, M_n$ . Note that  $a_{i,j} = 0$  for all  $j > i$ . This is the kick-off point of KRLS rationale.

In matrix notation, the respective loss of the KRLS minimization task at iteration  $n$  is given by

$$\mathcal{L}_n(\boldsymbol{\theta}) = \|\mathbf{y}_n - \mathbf{K}_n \boldsymbol{\theta}\|^2.$$

As  $\mathbf{K}_n \approx \mathbf{A}_n \cdot \tilde{\mathbf{K}}_n \cdot \mathbf{A}_n^T$ ,  $\mathcal{L}(\boldsymbol{\theta})$  can be recasted as

$$\mathcal{L}_n(\boldsymbol{\theta}) \approx \|\mathbf{y}_n - \mathbf{A}_n \tilde{\mathbf{K}}_n \mathbf{A}_n^T \boldsymbol{\theta}\|^2 = \|\mathbf{y}_n - \mathbf{A}_n \tilde{\mathbf{K}}_n \tilde{\boldsymbol{\theta}}\|^2 = \mathcal{L}_n(\tilde{\boldsymbol{\theta}}),$$

where  $\tilde{\theta} = A_n^T \theta$  is the new parameter to be optimized. Observe that while the original vector  $\theta_n \in \mathcal{R}^n$  is a  $n$ -dimensional vector, the new vector,  $\tilde{\theta}_n \in \mathcal{R}^{M_n}$ , is significantly smaller (depending on the sparsification strategy). The solution to the modified minimization task is

$$\begin{aligned}\tilde{\theta}_n &= (\tilde{K}_n^T A_n^T A_n \tilde{K}_n)^{-1} \tilde{K}_n^T A_n^T y_n \\ &= \tilde{K}_n^{-1} (A_n^T A_n)^{-1} \tilde{K}_n^{-T} \tilde{K}_n^T A_n^T y_n \\ &= K_n^{-1} P_n^{-1} A_n^T y_n,\end{aligned}\tag{17.65}$$

where  $P_n = A_n^T A_n$ , assuming that the inverse matrices exist. We distinguish the two different cases of the ALD sparsification and derive recursive relations for  $\tilde{\theta}_n$  accordingly.

**CASE I.** If  $\delta_n < \epsilon_0$ , then the data point arrived at iteration  $n$ , i.e.,  $(x_n, y_n)$ , is not added to the dictionary. Hence  $D_n = D_{n-1}$ ,  $M_n = M_{n-1}$ ,  $\tilde{K}_n = \tilde{K}_{n-1}$  and the expansion of  $\Phi(x_n)$  as a linear combination of the dictionary's elements is added to  $A_n$ , i.e.,

$$A_n = \begin{pmatrix} A_{n-1} \\ a_n^T \end{pmatrix},\tag{17.66}$$

where  $a_n$  is given by (17.62). Therefore,  $P_n = A_n^T A_n$  can be computed as

$$\begin{aligned}P_n &= A_n^T A_n = (A_{n-1}^T A_n) \begin{pmatrix} A_{n-1} \\ a_n^T \end{pmatrix} \\ &= A_{n-1}^T A_{n-1} + a_n a_n^T = P_{n-1} + a_n a_n^T.\end{aligned}$$

Applying the matrix inversion lemma, we take the recursive relation

$$P_n^{-1} = P_{n-1}^{-1} - \frac{P_{n-1}^{-1} a_n a_n^T P_{n-1}^{-1}}{1 + a_n^T P_{n-1}^{-1} a_n}.\tag{17.67}$$

Substituting in (17.65) and considering that  $A_n^T y_n = A_{n-1}^T y_{n-1} + a_n y_n$  we take

$$\begin{aligned}\tilde{\theta}_n &= \tilde{K}_n^{-1} P_n^{-1} A_n^T y_n \\ &= \tilde{K}_n^{-1} \left( P_{n-1}^{-1} - \frac{P_{n-1}^{-1} a_n a_n^T P_{n-1}^{-1}}{1 + a_n^T P_{n-1}^{-1} a_n} \right) (A_{n-1}^T y_{n-1} + a_n y_n) \\ &= \tilde{\theta}_{n-1} + \frac{\tilde{K}_n^{-1} P_{n-1}^{-1} a_n y_n - \tilde{K}_n^{-1} P_{n-1}^{-1} a_n a_n^T P_{n-1}^{-1} A_{n-1}^T y_{n-1}}{1 + a_n^T P_{n-1}^{-1} a_n} \\ &= \tilde{\theta}_{n-1} + \frac{K_n^{-1} P_{n-1}^{-1} a_n (y_n - a_n^T P_{n-1}^{-1} A_{n-1}^T y_{n-1})}{1 + a_n^T P_{n-1}^{-1} a_n}.\end{aligned}$$

However, relation (17.65) implies that  $\tilde{\mathbf{K}}_{n-1}\tilde{\boldsymbol{\theta}}_{n-1} = \mathbf{P}_{n-1}^{-1}\mathbf{A}_{n-1}^T\mathbf{y}_{n-1}$ . Hence,

$$\begin{aligned}\tilde{\boldsymbol{\theta}}_n &= \tilde{\boldsymbol{\theta}}_{n-1} + \frac{\mathbf{K}_n^{-1}\mathbf{P}_{n-1}^{-1}\mathbf{a}_n(y_n - \mathbf{a}_n^T\tilde{\mathbf{K}}_{n-1}\tilde{\boldsymbol{\theta}}_{n-1})}{1 + \mathbf{a}_n^T\mathbf{P}_{n-1}^{-1}\mathbf{a}_n} \\ &= \tilde{\boldsymbol{\theta}}_{n-1} + \frac{\mathbf{K}_n^{-1}\mathbf{P}_{n-1}^{-1}\mathbf{a}_n(y_n - \boldsymbol{\beta}_n^T\tilde{\boldsymbol{\theta}}_{n-1})}{1 + \mathbf{a}_n^T\mathbf{P}_{n-1}^{-1}\mathbf{a}_n} \\ &= \tilde{\boldsymbol{\theta}}_{n-1} + \frac{\mathbf{K}_n^{-1}\mathbf{P}_{n-1}^{-1}\mathbf{a}_ne_n}{1 + \mathbf{a}_n^T\mathbf{P}_{n-1}^{-1}\mathbf{a}_n},\end{aligned}$$

where the last relations are generated by substituting  $\boldsymbol{\beta}_n^T = \mathbf{a}_n^T\tilde{\mathbf{K}}_{n-1}$  (see relation (17.62)) and considering that the estimated output at time index  $n$  by the learning system is  $\hat{y}_n = \boldsymbol{\beta}_n^T\tilde{\boldsymbol{\theta}}_n = \mathbf{a}_n^T\tilde{\mathbf{K}}_n\tilde{\boldsymbol{\theta}}_n$ .

**CASE II.** If  $\delta_n > \epsilon_0$ , then the data point arrived at iteration  $n$ , i.e.,  $(\mathbf{x}_n, y_n)$ , is added to the dictionary, i.e.,  $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n\}$ ,  $M_n = M_{n-1} + 1$  and the matrices  $\mathbf{A}_n$  and  $\tilde{\mathbf{K}}_n$  are constructed as follows

$$\mathbf{A}_n = \begin{pmatrix} \mathbf{A}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}, \quad \tilde{\mathbf{K}}_n = \begin{pmatrix} \tilde{\mathbf{K}}_{n-1} & \boldsymbol{\beta}_n \\ \boldsymbol{\beta}_n^T & \kappa(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}.$$

Moreover,

$$\mathbf{P}_n = \mathbf{A}_n^T\mathbf{A}_n = \begin{pmatrix} \mathbf{A}_{n-1}^T & \mathbf{0}^T \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{n-1}^T\mathbf{A}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}.$$

Hence, applying the matrix inversion lemma to  $\tilde{\mathbf{K}}_n$  and substituting  $\mathbf{a}_n = \tilde{\mathbf{K}}_{n-1}^{-1}\boldsymbol{\beta}_n$ , we have

$$\tilde{\mathbf{K}}_n^{-1} = \begin{pmatrix} \tilde{\mathbf{K}}_{n-1}^{-1} + \frac{1}{\delta_n}\tilde{\mathbf{K}}_{n-1}^{-1}\boldsymbol{\beta}_n\boldsymbol{\beta}_n^T\tilde{\mathbf{K}}_{n-1}^{-1} & -\frac{1}{\delta_n}\tilde{\mathbf{K}}_{n-1}^{-1}\boldsymbol{\beta}_n \\ -\frac{1}{\delta_n}\boldsymbol{\beta}_n^T\tilde{\mathbf{K}}_n^{-1} & \frac{1}{\delta_n} \end{pmatrix} = \frac{1}{\delta_n} \begin{pmatrix} \delta_n\tilde{\mathbf{K}}_{n-1}^{-1} + \mathbf{a}_n\mathbf{a}_n^T & -\mathbf{a}_n \\ -\mathbf{a}_n^T & 1 \end{pmatrix}.$$

and

$$\mathbf{P}_n^{-1} = \begin{pmatrix} (\mathbf{A}_n^T\mathbf{A}_n)^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{P}_{n-1}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}.$$

Observe that while we replace  $\tilde{\mathbf{K}}_{n-1}^{-1}\boldsymbol{\beta}_n$  by  $\mathbf{a}_n$  using relation (17.62), the actual  $n$ th row of  $\mathbf{A}_n$  is  $(0, 0, \dots, 0, 1)$ . Substituting in (17.65) we take:

$$\begin{aligned}
\tilde{\theta}_n &= \frac{1}{\delta_n} \begin{pmatrix} \delta_n \tilde{K}_{n-1}^{-1} + \mathbf{a}_n \mathbf{a}_n^T & -\mathbf{a}_n \\ -\mathbf{a}_n^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_{n-1}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_{n-1}^T & \mathbf{0}^T \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{y}_{n-1} \\ y_n \end{pmatrix} \\
&= \frac{1}{\delta_n} \begin{pmatrix} \delta_n \tilde{K}_{n-1}^{-1} \mathbf{P}_{n-1}^{-1} \mathbf{A}_{n-1}^T \mathbf{y}_{n-1} + \mathbf{a}_n \mathbf{a}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{A}_{n-1}^T \mathbf{y}_{n-1} - \mathbf{a}_n y_n \\ -\mathbf{a}_n^T \mathbf{P}_{n-1}^{-1} \mathbf{A}_{n-1}^T \mathbf{y}_{n-1} + y_n \end{pmatrix} \\
&= \frac{1}{\delta_n} \begin{pmatrix} \delta_n \tilde{\theta}_{n-1} + \mathbf{a}_n \mathbf{a}_n^T \tilde{K}_{n-1} \tilde{\theta}_{n-1} - \mathbf{a}_n y_n \\ -\mathbf{a}_n^T \tilde{K}_{n-1} \tilde{\theta}_{n-1} + y_n \end{pmatrix} \\
&= \frac{1}{\delta_n} \begin{pmatrix} \delta_n \tilde{\theta}_{n-1} + \mathbf{a}_n \beta_n^T \tilde{\theta}_{n-1} - \mathbf{a}_n y_n \\ -\beta_n^T \tilde{\theta}_{n-1} + y_n \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\theta}_{n-1} - \frac{\mathbf{a}_n}{\delta_n} e_n \\ -\frac{e_n}{\delta_n} \end{pmatrix}.
\end{aligned}$$

The procedure is summarized in Algorithm 32. The KRLS algorithm has complexity  $O(M_n^2)$ , at time  $n$ , where  $M_n$  is the size of the dictionary. Note, that we developed the specific recursive strategy assuming the invertibility of  $\tilde{K}_n$  and  $\mathbf{A}_n^T \mathbf{A}_n$  for each  $n$ . For the case of the RKHS induced by the Gaussian kernel, which is the most popular kernel for practical applications, we can prove that  $\tilde{K}_n$  is indeed invertible. Furthermore, by construction, the matrix  $\mathbf{A}_n$  has full-rank. Hence,  $\mathbf{A}_n^T \mathbf{A}_n$  is strictly positive definite and thus invertible too.

A methodology of similar spirit to the KRLS, which also takes advantage of the loss function (17.58) in the Support Vector Machines (SVM) rationale, can be found in [57–60]. An alternative path to the KRLS was followed in [61], via a formulation based on Gaussian Processes. Such an approach allows for an explicit forgetting factor mechanism to be incorporated in the adaptation flow. An alternative philosophy for forgetting past data in kernel-based adaptive schemes, in the framework of regularization, which can efficiently be implemented via projections, will be discussed in Section 1.17.7.

#### 1.17.6.6.1 Simulation results

In order to study the behavior of the KRLS, we consider some nonlinear channel equalization set-ups similar to the ones used in Section 1.17.6.2.2. In the first experiment, the nonlinear channel consists of a linear filter:

$$t_n = -0.8 \cdot y_n + 0.7 \cdot y_{n-1} - 0.6 \cdot y_{n-2} + 0.4 \cdot y_{n-3}$$

and a memoryless nonlinearity

$$q_n = t_n + 0.08 \cdot t_n^2. \quad (17.68)$$

The input signal that was fed to the channel followed the Gaussian distribution with zero mean and unity variance. At the receiver end of the channels, the signal is corrupted by white Gaussian noise and then observed as  $x_n$ . The level of the noise was set to 15dB. To solve the equalization task, we apply the KRLS-based learning systems to the set of samples  $(\mathbf{x}_n, y_n) := ((x_n, x_{n-1}, \dots, x_{n-l+1}), y_{n-D})$ , where  $l > 0$  is the equalizer's length and  $D$  the equalization time delay. Figure 17.23 shows the performance of KRLS compared to KLMS over a set of 30 training sequences consisting of 5000 samples each. The Quantization sparsification strategy was adopted for the KLMS, as it demonstrates superior performance

**Algorithm 32.** Kernel Recursive Least Squares (KRLS)

---

**Require:** The data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , and the ALD parameter  $\epsilon_0$ .

- 1: Set  $\tilde{\mathbf{K}}^{-1} = [1/\kappa(\mathbf{x}_1, \mathbf{x}_1)], \mathbf{P}^{-1} = [1], \tilde{\boldsymbol{\theta}} = [y_1/\kappa(\mathbf{x}_1, \mathbf{x}_1)], M = 1$ .
- 2: **for**  $n = 2, 3, \dots$  **do**
- 3:   Compute  $\boldsymbol{\beta} = (\kappa(\mathbf{u}_1, \mathbf{x}_n), \dots, \kappa(\mathbf{u}_M, \mathbf{x}_n))^T$ .
- 4:   Compute the learning system's output:  $\hat{y}_n = \boldsymbol{\theta}^T \boldsymbol{\beta}$ .
- 5:   Compute the error:  $e_n = y_n - \hat{y}_n$ .
- 6:   FOR ALD TEST:
- 7:     Compute  $\mathbf{a} = \tilde{\mathbf{K}}^{-1} \boldsymbol{\beta}$ .
- 8:     Compute  $\delta = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \boldsymbol{\beta}^T \mathbf{a}$ .
- 9:     **if**  $\delta > \epsilon_0$  **then**
- 10:       Include the new sample to the dictionary:  $\mathcal{D} = \mathcal{D} \cup \{\mathbf{x}_n\}, M = M + 1$ .
- 11:       Update  $\tilde{\mathbf{K}}^{-1} = \frac{1}{\delta} \begin{pmatrix} \delta \tilde{\mathbf{K}}^{-1} + \mathbf{a} \mathbf{a}^T & -\mathbf{a} \\ -\mathbf{a} & 1 \end{pmatrix}$ .
- 12:       Update  $\mathbf{P}^{-1} = \begin{pmatrix} \mathbf{P}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}$ .
- 13:       Update solution:  $\tilde{\boldsymbol{\theta}} = \begin{pmatrix} \tilde{\boldsymbol{\theta}} - \frac{\mathbf{a}}{\delta} e_n \\ -\frac{e_n}{\delta} \end{pmatrix}$ .
- 14:     **else**
- 15:        $\mathbf{q} = \frac{\mathbf{P}^{-1} \mathbf{a}}{1 + \mathbf{a}^T \mathbf{P}^{-1} \mathbf{a}}$ .
- 16:       Update  $\mathbf{P}^{-1} = \mathbf{P}^{-1} - \mathbf{q} \mathbf{a}^T \mathbf{P}^{-1}$ .
- 17:       Update solution:  $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}} + \mathbf{K}^{-1} \mathbf{q} e_n$ .
- 18:     **end if**
- Output:** The vector  $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^M$  and the dictionary  $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_M\}$ .
- 19: **end for**

---

(see Section 1.17.6.2.2). The sparsification parameters were set to  $\epsilon_0 = 0.1$  for the KRLS, and  $\delta = 4$  for the quantization of KLMS. It is clear that KRLS significantly outperforms KLMS both in terms of convergence and sparsity.

In the second experiment, the linear part of the channel is reduced, i.e.,

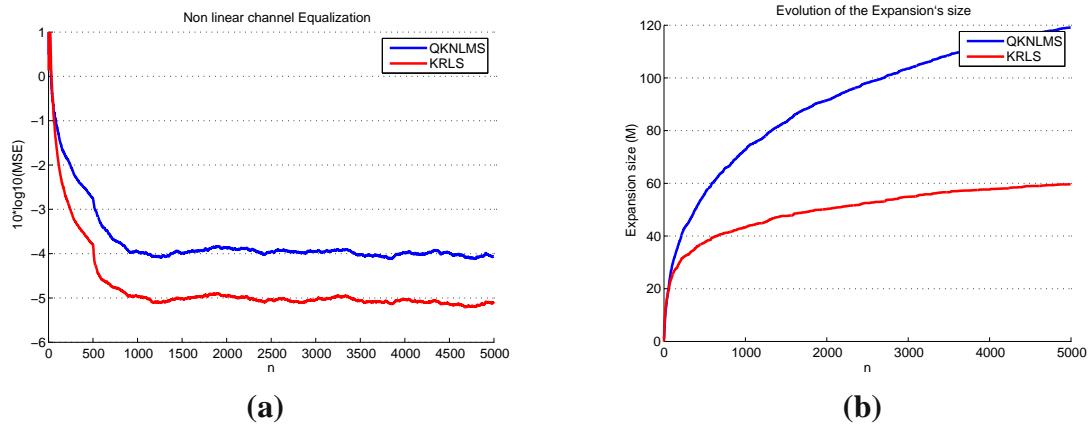
$$t_n = -0.8 \cdot y_n + 0.7 \cdot y_{n-1} \quad (17.69)$$

and the memoryless nonlinearity remains the same. Figure 17.24 shows the performance of the two algorithms over a set of 30 training sequences consisting of 5000 samples each. The parameters used for sparsification were set to  $\epsilon_0 = 0.1$  for the KRLS, and  $\delta = 6$  for the quantization of KLMS.

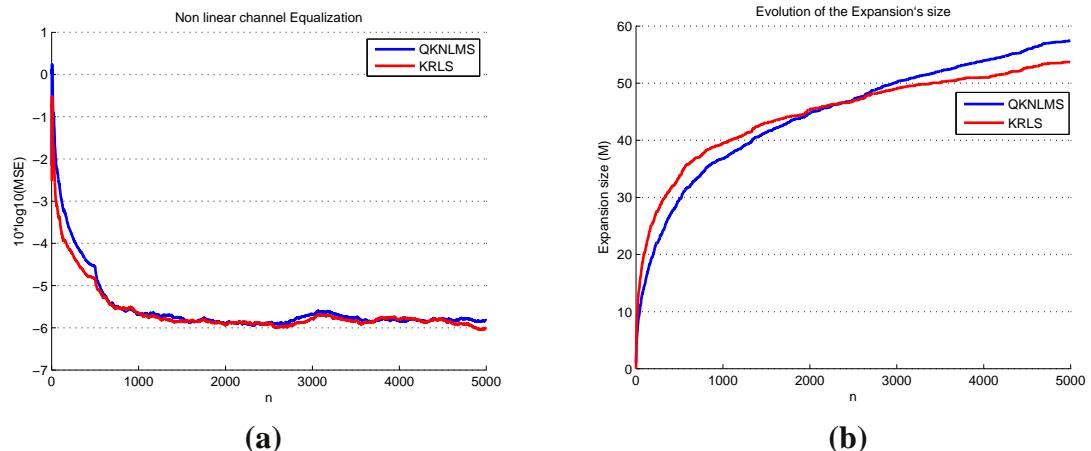
Finally, in the third experiment, the linear part is identical to (17.69), while the non linear part is

$$q_n = t_n + 0.25 \cdot t_n^2 + 0.11 \cdot t_n^3 \quad (17.70)$$

and the memoryless nonlinearity remains the same. Figure 17.25 shows the performance of the two algorithms over a set of 30 training sequences consisting of 5000 samples each, using the same parameters for the sparsification as in the second experiment. In general, KRLS shows the best performance, as

**FIGURE 17.23**

(a) Learning curves of the quantized KNLMS ( $\mu = 1/2$ ), and KRLS based on the ALD sparsification scenario ( $\epsilon_0 = 0.1$ ) using the Gaussian kernel ( $\sigma = 5$ ), for the equalization problem (17.68) (filter length  $l = 5$ , time delay  $D = 2$ ). The quantization size was set to  $\delta = 4$ . (b) Evolution of the expansion's size at each time instant, for each algorithm.

**FIGURE 17.24**

(a) Learning curves of the quantized KNLMS ( $\mu = 1/2$ ), and KRLS based on the ALD sparsification scenario ( $\epsilon_0 = 0.1$ ) using the Gaussian kernel ( $\sigma = 5$ ), for the equalization problem (17.69) (filter length  $l = 5$ , time delay  $D = 2$ ). The quantization size was set to  $\delta = 6$ . (b) Evolution of the expansion's size at each time instant, for each algorithm.

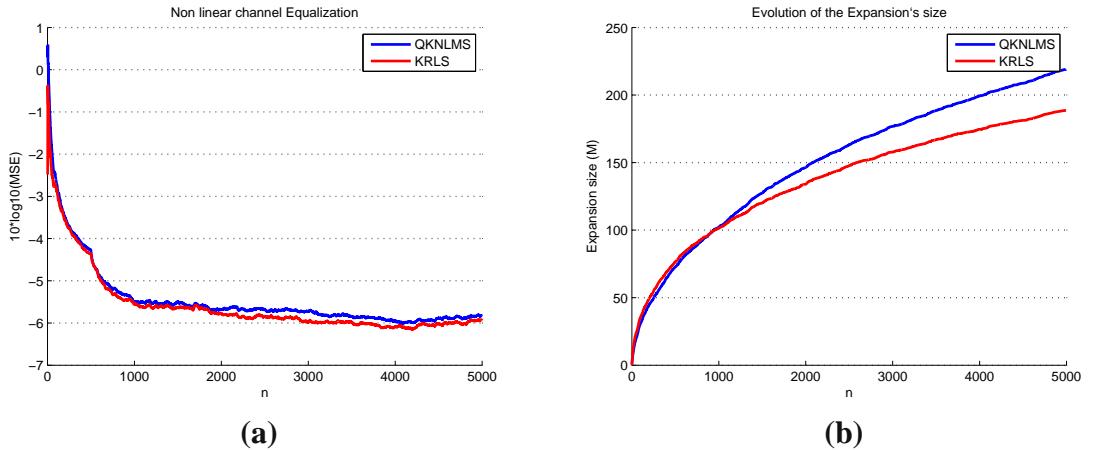


FIGURE 17.25

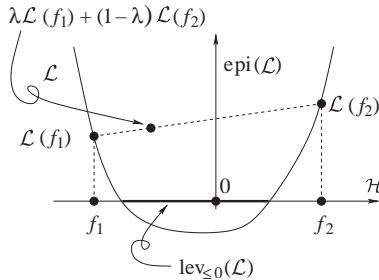
(a) Learning curves of the quantized KNLMS ( $\mu = 1/2$ ), and KRLS based on the ALD sparsification scenario ( $\epsilon_0 = 0.1$ ) using the Gaussian kernel ( $\sigma = 5$ ), for the equalization problem (17.70) (filter length  $l = 5$ , time delay  $D = 2$ ). The quantization size was set to  $\delta = 4$ . (b) Evolution of the expansion's size at each time instant, for each algorithm.

it appears to achieve lower or the same steady state MSE as the QKLMS using somewhat sparser representations. However, there are cases where its performance is quite similar to the QKNLMS. The code for the experiments presented in this section can be found in [<http://bouboulis.mysch.gr/kernels.html>](http://bouboulis.mysch.gr/kernels.html).

## 1.17.7 A convex analytic toolbox for online learning

During the last ten years or so, convex analysis [34, 62] has emerged as the main stage where mathematically sound optimization tools for signal processing and machine learning tasks are developed. This section attempts to give a short introduction on several convex analytic arguments which are central to modern optimization techniques. We do not follow the mainstream of the literature which develops around the classical Lagrange multipliers' methodology [63] or the powerful interior point algorithms [64], but instead, we abide by the less popular, to the engineering community, *operator/mapping* and *fixed point theoretic* framework [65–67], established in general Hilbert spaces. The reason to adopt such a framework is its geometrical nature, generality, and ability to engulf numerous modern tools of convex and variational analysis [62, 65]. Our objective here is to highlight a few fragments of this powerful framework. For more details, and for a deeper view of the subject, the interested reader is referred to [62, 65–67].

In the sequel, the stage of our discussion is a real Hilbert space  $\mathcal{H}$ , equipped with an inner product  $\langle \cdot, \cdot \rangle$ , and with an induced norm  $\|\cdot\| := \sqrt{\langle \cdot, \cdot \rangle}$ . A sequence  $(f_n)_{n \in \mathcal{N}}$  of elements of  $\mathcal{H}$  is said to converge *strongly* to an  $f_* \in \mathcal{H}$ , and it will be denoted as  $f_n \rightarrow f_*$ , if  $\lim_{n \rightarrow \infty} \|f_n - f_*\| = 0$ . The sequence  $(f_n)_{n \in \mathcal{N}}$  is said to converge *weakly* to an  $f_* \in \mathcal{H}$ , and it will be denoted as  $f_n \rightharpoonup f_*$ ,

**FIGURE 17.26**

A convex function  $\mathcal{L}$ , its epigraph, and the lower level set of  $\mathcal{L}$  at height 0.

if  $\forall g \in \mathcal{H}, \lim_{n \rightarrow \infty} \langle f_n - f_*, g \rangle = 0$ . Strong convergence implies the weak one, but, in general, the converse does not hold true in Hilbert spaces. However, it does hold true in the case of Euclidean spaces.

We start, now, with few notions of fundamental importance to convex analysis.

### 1.17.7.1 Closed convex sets and metric projection mappings

**Definition 33 (Convex set, convex function).** A non-empty subset  $C$  of  $\mathcal{H}$  is called *convex* if  $\forall f_1, f_2 \in C$ , and  $\forall \lambda \in [0, 1]$ , the following holds true:  $\lambda f_1 + (1 - \lambda) f_2 \in C$ .

Moreover, a function  $\mathcal{L} : \mathcal{H} \rightarrow \mathbb{R}$  is called *convex* if  $\forall f_1, f_2 \in \mathcal{H}$ , and  $\forall \lambda \in [0, 1]$ ,  $\mathcal{L}(\lambda f_1 + (1 - \lambda) f_2) \leq \lambda \mathcal{L}(f_1) + (1 - \lambda) \mathcal{L}(f_2)$ . The function  $f$  is called *strictly convex* if  $\forall \lambda \in (0, 1)$  and  $\forall f_1, f_2 \in \mathcal{H}$ , such that  $f_1 \neq f_2$ , we have  $\mathcal{L}(\lambda f_1 + (1 - \lambda) f_2) < \lambda \mathcal{L}(f_1) + (1 - \lambda) \mathcal{L}(f_2)$ . Finally, if there exists a  $\beta > 0$  such that  $\forall \lambda \in (0, 1)$  and  $\forall f_1, f_2 \in \mathcal{H}$ , the following holds true:

$$\mathcal{L}(\lambda f_1 + (1 - \lambda) f_2) + \lambda(1 - \lambda) \frac{\beta}{2} \|f_1 - f_2\|^2 \leq \lambda \mathcal{L}(f_1) + (1 - \lambda) \mathcal{L}(f_2),$$

then  $\mathcal{L}$  is called *strongly convex with constant  $\beta$* . ◇

The *epigraph* of a convex function  $\mathcal{L}$  is defined as the set

$$\text{epi}(\mathcal{L}) := \{(h, r) \in \mathcal{H} \times \mathbb{R} : \mathcal{L}(h) \leq r\}.$$

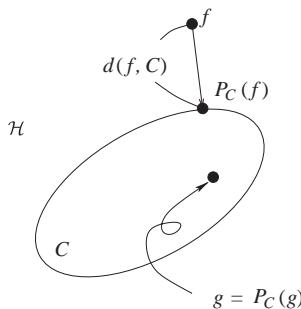
In other words, the epigraph of  $\mathcal{L}$  is the set of all points of  $\mathcal{H} \times \mathbb{R}$  which belong to and lie above the graph of  $\mathcal{L}$ . Moreover, given a real number  $\xi$ , the *lower level set of  $\mathcal{L}$  at height  $\xi$*  is defined as the set

$$\text{lev}_{\leq \xi}(\mathcal{L}) := \{h \in \mathcal{H} : \mathcal{L}(h) \leq \xi\}.$$

For the geometry behind the previous definitions, see Figure 17.26. A simple inspection of Figure 17.26 suggests that  $\mathcal{L}$  is convex if and only if  $\text{epi}(\mathcal{L})$  is convex.

**Definition 34 (The metric distance function).** Given a closed convex set  $C \subset \mathcal{H}$ , the *metric distance function to  $C$*  is defined as follows:

$$\forall f \in \mathcal{H}, \quad d(f, C) := \inf \{\|f - g\| : g \in C\}.$$

**FIGURE 17.27**

A closed convex subset of  $\mathcal{H}$ , and the associated metric projection mapping.

In the case where the set  $C$  is a singleton, i.e., it contains a single element  $C = \{g\}$ , then, clearly, the metric distance  $d(f, C)$  becomes our familiar distance of  $f$  from  $g$ :  $d(f, g)$ .

It can be easily verified by the triangle inequality of  $\|\cdot\|$ , that  $d(\cdot, C)$  is a convex function. Moreover, notice that  $\text{lev}_{\leq 0}(d(\cdot, C)) = C$ .  $\diamond$

**Definition 35 (The metric projection mapping).** Given a non-empty *closed* convex set  $C \subset \mathcal{H}$ , the *metric projection mapping onto*  $C$  is defined as the operator that maps to each  $f \in \mathcal{H}$  the *unique*  $P_C(f) \in C$  such that

$$\|f - P_C(f)\| = d(f, C).$$

In other words, the point  $P_C(f)$  is the unique minimizer of the function  $\|f - g\|$ ,  $g \in C$ . Obviously, in the case where  $g \in C$ , then  $P_C(g) = g$ . For the geometry behind the mapping  $P_C$ , see Figure 17.27.  $\diamond$

Let us give now a couple of examples of closed convex sets in order to illustrate the previous concepts.

**Example 36 (Closed ball).** Given a radius  $\Delta > 0$  and a center  $g_0 \in \mathcal{H}$ , the *closed ball*  $B[g_0, \Delta]$  is defined as the following closed convex set

$$B[g_0, \Delta] := \{g \in \mathcal{H} : \|g - g_0\| \leq \Delta\}.$$

The associated metric projection mapping  $P_{B[g_0, \Delta]}$  can be easily verified to be

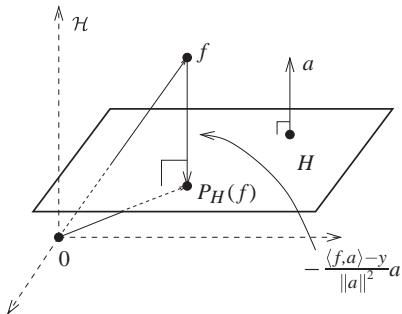
$$P_{B[g_0, \Delta]}(f) = g_0 + \frac{\Delta}{\max\{\Delta, \|f - g_0\|\}}(f - g_0), \quad \forall f \in \mathcal{H}.$$

**Example 37 (Hyperplane).** A *hyperplane*  $H$  is the closed convex subset of  $\mathcal{H}$  which is defined as

$$H := \{g \in \mathcal{H} : \langle a, g \rangle = y\},$$

for some nonzero  $a \in \mathcal{H}$  and some  $y \in \mathcal{R}$ . The (metric) projection mapping  $P_H$  onto  $H$  is given as follows:

$$P_H(f) = f - \frac{\langle f, a \rangle - y}{\|a\|^2}a, \quad \forall f \in \mathcal{H}. \tag{17.71}$$

**FIGURE 17.28**

A hyperplane and its associated projection mapping.

For the geometry behind the previous concepts, see Figure 17.28. Notice that  $a$  stands as the *normal vector* of the hyperplane.

In the special case of an RKHS  $\mathcal{H}$ , which is associated to a kernel function  $\kappa$ , let  $a := \kappa(\mathbf{x}, \cdot)$ , for some  $\mathbf{x} \in \mathcal{R}^l$ . Then, due to the kernel trick, the hyperplane  $H$  obtains a simple form:  $H = \{g \in \mathcal{H} : g(\mathbf{x}) = y\}$ . Moreover, the (metric) projection mapping becomes as follows:

$$P_H(f) = f - \frac{f(\mathbf{x}) - y}{\kappa(\mathbf{x}, \mathbf{x})} \kappa(\mathbf{x}, \cdot), \quad \forall f \in \mathcal{H}. \quad (17.72)$$

The previous example helps us calculate the metric projection mapping onto the following renowned closed convex set.

**Example 38 ((Closed) hyperslab).** A *(closed) hyperslab*  $S[\epsilon]$ , where  $\epsilon > 0$  is a user-defined parameter, is defined as the following closed convex set:

$$S[\epsilon] := \{g \in \mathcal{H} : |\langle g, a \rangle - y| \leq \epsilon\},$$

where  $a$  is a non-zero element of  $\mathcal{H}$ , and  $y \in \mathcal{R}$ . Geometrically, a hyperslab can be visualized as the set of all those points which belong onto and between the following two “parallel” hyperplanes;  $H_{+\epsilon} := \{g \in \mathcal{H} : \langle g, a \rangle = y + \epsilon\}$  and  $H_{-\epsilon} := \{g \in \mathcal{H} : \langle g, a \rangle = y - \epsilon\}$ . Based on this fundamental observation, the metric projection mapping  $P_{S[\epsilon]}$  can be directly calculated by employing the results of Example 37;  $\forall f \in \mathcal{H}$ ,

$$P_{S[\epsilon]}(f) = \begin{cases} P_{H_{+\epsilon}}(f), & \text{if } \langle f, a \rangle > y + \epsilon, \\ f, & \text{if } |\langle f, a \rangle - y| \leq \epsilon, \\ P_{H_{-\epsilon}}(f), & \text{if } \langle f, a \rangle < y - \epsilon. \end{cases} = \begin{cases} f - \frac{\langle f, a \rangle - y - \epsilon}{\|a\|^2} a, & \text{if } \langle f, a \rangle > y + \epsilon, \\ f, & \text{if } |\langle f, a \rangle - y| \leq \epsilon, \\ f - \frac{\langle f, a \rangle - y + \epsilon}{\|a\|^2} a, & \text{if } \langle f, a \rangle < y - \epsilon. \end{cases} \quad (17.73)$$

As in Example 37, in the special case of an RKHS  $\mathcal{H}$ , which is associated to a kernel function  $\kappa$ , let  $a := \kappa(\mathbf{x}, \cdot)$ , for some  $\mathbf{x} \in \mathcal{R}^l$ . Then, under such a scenario, (17.73) takes the following special form:  $\forall f \in \mathcal{H}$ ,

$$P_{S[\epsilon]}(f) = \begin{cases} f - \frac{f(\mathbf{x}) - y - \epsilon}{\kappa(\mathbf{x}, \mathbf{x})} \kappa(\mathbf{x}, \cdot), & \text{if } f(\mathbf{x}) > y + \epsilon, \\ f, & \text{if } |f(\mathbf{x}) - y| \leq \epsilon, \\ f - \frac{f(\mathbf{x}) - y + \epsilon}{\kappa(\mathbf{x}, \mathbf{x})} \kappa(\mathbf{x}, \cdot), & \text{if } f(\mathbf{x}) < y - \epsilon. \end{cases} \quad (17.74)$$

**Example 39 (Affine set).** Assume a positive integer  $m$ , a number of nonzero elements  $\{a_1, a_2, \dots, a_m\}$  of  $\mathcal{H}$ , and a set of real numbers  $\{y_1, y_2, \dots, y_m\}$ . Upon defining the hyperplanes  $H_i := \{g \in \mathcal{H} : \langle a_i, g \rangle = y_i\}$ ,  $\forall i = 1, \dots, m$ , let the set of all common points of the previous hyperplanes:

$$V := \bigcap_{i=1}^m H_i. \quad (17.75)$$

Since the previous  $V$  might be empty, let us generalize it as follows in order to cover also such an unpleasant situation. Define the linear mapping  $A : \mathcal{H} \rightarrow \mathcal{R}^m$  as follows:

$$A(f) := \begin{bmatrix} \langle a_1, f \rangle \\ \vdots \\ \langle a_m, f \rangle \end{bmatrix}, \quad \forall f \in \mathcal{H}.$$

Notice here that in the case where our Hilbert space  $\mathcal{H}$  becomes a Euclidean one, e.g.,  $\mathcal{R}^l$ , and the elements  $\{a_1, a_2, \dots, a_m\}$  become the  $l$ -dimensional vectors  $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ , then the mapping  $A$  is nothing but the following matrix  $\mathbf{A} := [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m]^T$ , where the superscript  $T$  stands for transposition.

If we let  $\mathbf{y} := [y_1, y_2, \dots, y_m]^T$ , then define

$$V := \operatorname{argmin}_{g \in \mathcal{H}} \|\mathbf{y} - A(g)\|^2. \quad (17.76)$$

Notice that if the previous minimum attains the value 0, then the intersection in (17.75) is nonempty. Hence, the Definition (17.76) generalizes the special (17.75) one. It can be shown that  $V$  is a nonempty *affine* set. Notice also here that in the case where our Hilbert space  $\mathcal{H}$  becomes a Euclidean one  $\mathcal{R}^l$ , then the affine set of (17.76) becomes the classical  $V = \operatorname{argmin}_{\theta \in \mathcal{R}^l} \|\mathbf{y} - \mathbf{A}\theta\|^2$ , which is often met in least-squares penalized learning tasks.

Under the previous setting, it can be verified [68] that the metric projection mapping  $P_V$  onto the affine set  $V$  of (17.76) is given as:

$$P_V(f) = f + A^\dagger (\mathbf{y} - A(f)), \quad \forall f \in \mathcal{H}, \quad (17.77)$$

where  $A^\dagger : \mathcal{R}^m \rightarrow \mathcal{H}$  stands for the (Moore-Penrose) pseudoinverse mapping of  $A$ .

Upon defining the adjoint mapping  $A^*$  of  $A$  as the mapping  $A^* : \mathcal{R}^m \rightarrow \mathcal{H}$  which satisfies  $\xi^T A(f) = \langle f, A^*(\xi) \rangle$ ,  $\forall f \in \mathcal{H}, \forall \xi \in \mathcal{R}^m$ , then it can be verified that

$$A^*(\xi) = \sum_{i=1}^m \xi_i a_i, \quad \forall \xi \in \mathcal{R}^m.$$

Notice that in the case where our Hilbert space  $\mathcal{H}$  becomes a Euclidean  $\mathcal{R}^l$  one, then the adjoint mapping is nothing but the transpose of  $A$ , i.e.,  $A^* = A^T$ . A classical result states that  $A^\dagger = A^* (AA^*)^{-1}$  [69]. Notice that the mapping  $AA^* : \mathcal{R}^m \rightarrow \mathcal{R}^m$  is nothing but the Gram matrix

$$AA^* = \begin{bmatrix} \langle a_1, a_1 \rangle & \langle a_1, a_2 \rangle & \cdots & \langle a_1, a_m \rangle \\ \langle a_2, a_1 \rangle & \langle a_2, a_2 \rangle & \cdots & \langle a_2, a_m \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle a_m, a_1 \rangle & \langle a_m, a_2 \rangle & \cdots & \langle a_m, a_m \rangle \end{bmatrix}.$$

In the case where the Hilbert space  $\mathcal{H}$  becomes an RKHS associated to a kernel function  $\kappa$ , and  $a_i := \kappa(\mathbf{x}_i, \cdot)$ , for some  $\mathbf{x}_i \in \mathcal{R}^l$ ,  $\forall i = 1, 2, \dots, m$ , then the previous Gram matrix  $AA^*$  is usually called the *kernel matrix* associated to the set  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ; the  $(i, j)$ th element of  $AA^*$  becomes  $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ .

### 1.17.7.2 The subgradient

**Definition 40 (Subdifferential, subgradient).** Given a convex function  $\mathcal{L}$ , defined on  $\mathcal{H}$ , and a point  $f \in \mathcal{H}$ , the *subdifferential of  $\mathcal{L}$  at  $f$*  is defined as the set-valued operator  $\partial\mathcal{L}$  that maps to  $f$  the set of all the *subgradients of  $\mathcal{L}$  at  $f$* , i.e.,

$$\partial\mathcal{L}(f) := \{g \in \mathcal{H} : \langle g, h - f \rangle + \mathcal{L}(f) \leq \mathcal{L}(h), \forall h \in \mathcal{H}\}. \quad \diamond$$

Let us give now a geometrical point of view to the notion of the subgradient. Fix an  $f \in \mathcal{H}$ , and choose any subgradient  $g \in \partial\mathcal{L}(f)$ . This subgradient defines the following hyperplane in the space  $\mathcal{H} \times \mathcal{R}$ :

$$\mathcal{X}_g := \{(h, r) \in \mathcal{H} \times \mathcal{R} : 0 = \langle g, h - f \rangle + \mathcal{L}(f) - r\}. \quad (17.78)$$

Moreover, the following interesting result can be easily obtained.

**Proposition 41 (Geometric interpretation of the subgradient).** Given a convex function  $\mathcal{L}$ , assume without any loss of generality that  $\text{lev}_{\leq 0}(\mathcal{L}) \neq \emptyset$  (if not, a vertical shift of the  $\mathcal{L}$  downwards by some constant suffices to guarantee the non-emptiness of  $\text{lev}_{\leq 0}(\mathcal{L})$ ). Assume, also, an  $f \in \mathcal{H} \setminus \text{lev}_{\leq 0}(\mathcal{L})$  (see Figure 17.29). Then, for any subgradient  $g$  of  $\mathcal{L}$  at  $f$ , i.e.,  $g \in \partial\mathcal{L}(f)$ , the  $\mathcal{X}_g$ , defined in (17.78), acts as a separating hyperplane between the point  $(f, 0)$  and  $\text{epi}(\mathcal{L})$ .  $\diamond$

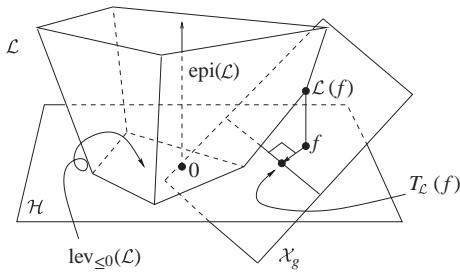
**Proof.** Since  $f \notin \text{lev}_{\leq 0}(\mathcal{L})$ , it is easy to see that  $\mathcal{L}(f) > 0$ . Hence, for the point  $(f, 0)$ ,

$$\langle g, f - f \rangle + \mathcal{L}(f) - 0 = \mathcal{L}(f) > 0. \quad (17.79)$$

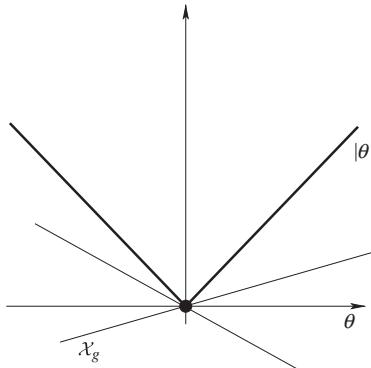
On the other hand, for every  $(h, r) \in \text{epi}(\mathcal{L})$ , the definition of the subgradient suggests that

$$\langle g, h - f \rangle + \mathcal{L}(f) - r \leq \langle g, h - f \rangle + \mathcal{L}(f) - \mathcal{L}(h) \leq 0. \quad (17.80)$$

A simple inspection of (17.79) and (17.80) implies that  $\mathcal{X}_g$  stands as a separating hyperplane between  $(f, 0)$  and  $\text{epi}(\mathcal{L})$ .  $\square$

**FIGURE 17.29**

The hyperplane  $\mathcal{X}_g$  (17.78), generated by a subgradient  $g$  of  $\mathcal{L}$  at  $f$ , stands as a separating hyperplane between the point  $(f, 0)$  and  $\text{epi}(\mathcal{L})$  in the augmented space  $\mathcal{H} \times \mathcal{R}$ .

**FIGURE 17.30**

The graph of the function  $|\cdot|$ , and the supporting hyperplanes generated by the subgradients of  $|\cdot|$  at 0.

Since  $\text{lev}_{\leq 0}(\mathcal{L})$  belongs to  $\text{epi}(\mathcal{L})$ , then  $\mathcal{X}_g$  separates  $f$  and  $\text{lev}_{\leq 0}(\mathcal{L})$  (see Figure 17.29). It can be readily verified that the point  $(f, \mathcal{L}(f))$  belongs to  $\mathcal{X}_g$ . Hence, as it can be also verified by Figure 17.29,  $\mathcal{X}_g$  supports the  $\text{epi}(\mathcal{L})$  at the point  $(f, \mathcal{L}(f))$ .

Let us give now the subdifferentials of some celebrated convex functions.

**Example 42 ( $\ell_1$  norm).** The stage of discussion is the Euclidean  $\mathcal{R}^l$ . Assume, first, the classical case of  $\mathcal{L}(\theta) := |\theta|$ ,  $\theta \in \mathcal{R}$ . Then, the subdifferential of  $\mathcal{L}$  is given as follows:

$$\partial \mathcal{L}(\theta) = \begin{cases} [-1, 1], & \text{if } \theta = 0, \\ \text{sign}(\theta), & \text{if } \theta \neq 0, \end{cases}$$

where  $\text{sign}(\cdot)$  stands for the sign of a real number. For the geometry associated to this cost function see Figure 17.30.

Assume, now, that the function  $\mathcal{L}$  becomes the more involved  $\ell_1$  norm, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}) := \|\boldsymbol{\theta}\|_1 := \sum_{i=1}^l |\theta_i|, \quad \forall \boldsymbol{\theta} \in \mathcal{R}^l.$$

Then, the subdifferential of  $\mathcal{L}$  becomes as follows; given any  $\boldsymbol{\theta} \in \mathcal{R}^l$  define first the index set  $\mathcal{J}_{\boldsymbol{\theta}} := \{j \in \{1, 2, \dots, l\} : \theta_j = 0\}$ . Then, based on  $\mathcal{J}_{\boldsymbol{\theta}}$ , let a set of vectors  $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \dots, \boldsymbol{\tau}_{2^v}$ , where  $v := \text{card}(\mathcal{J}_{\boldsymbol{\theta}})$ , and where the  $j$ th component of the  $i$ th  $\boldsymbol{\tau}_i$  is given by

$$\tau_{ij} := \begin{cases} \text{sign}(\theta_j), & \text{if } j \notin \mathcal{J}_{\boldsymbol{\theta}}, \\ +1 \text{ or } -1, & \text{if } j \in \mathcal{J}_{\boldsymbol{\theta}}. \end{cases}$$

Then,

$$\partial \mathcal{L}(\boldsymbol{\theta}) = \begin{cases} \left\{ (\text{sign}(\theta_1), \dots, \text{sign}(\theta_l))^T \right\}, & \text{if } \boldsymbol{\theta} \neq \mathbf{0} \text{ and } \mathcal{J}_{\boldsymbol{\theta}} = \emptyset, \\ \text{conv}(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \dots, \boldsymbol{\tau}_{2^v}), & \text{if } \boldsymbol{\theta} \neq \mathbf{0} \text{ and } \mathcal{J}_{\boldsymbol{\theta}} \neq \emptyset, \\ \text{conv}(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \dots, \boldsymbol{\tau}_{2^l}), & \text{if } \boldsymbol{\theta} = \mathbf{0}, \end{cases}$$

where  $\text{conv}(\cdot)$  stands for the convex hull of a set [34, 62].  $\diamond$

**Example 43 (Distance function).** The subdifferential of the metric distance function to a closed convex set  $C \subset \mathcal{H}$  is given as follows:

$$\partial d(f, C) = \begin{cases} N_C(f) \cap B[0, 1], & \text{if } f \in C, \\ \left\{ \frac{f - P_C(f)}{d(f, C)} \right\}, & \text{if } f \in \mathcal{H} \setminus C, \end{cases}$$

where  $N_C(f) := \{v \in \mathcal{H} : \langle v, g - f \rangle \leq 0, \forall g \in C\}$ , and  $B[0, 1] := \{f \in \mathcal{H} : \|f\| \leq 1\}$ . Moreover, the function  $d^2(\cdot, C)$  is (Fréchet) differentiable:

$$\partial d^2(f, C) = \nabla d^2(f, C) = \{2(f - P_C(f))\}, \quad \forall f \in \mathcal{H}.$$

### 1.17.7.3 Nonexpansive and quasi-nonexpansive mappings

Let us start with a few fundamental concepts regarding a mapping  $T : \mathcal{H} \rightarrow \mathcal{H}$  which is defined in the Hilbert space  $\mathcal{H}$ .

**Definition 44 (A toolbox of mappings).**

1. The fixed point set of  $T$  is the set of all those points of  $\mathcal{H}$  which are unaffected by  $T$ , i.e.,  $\text{Fix}(T) := \{f \in \mathcal{H} : T(f) = f\}$ .
2. The mapping  $T$  is called  $\gamma$ -Lipschitzian, or  $\gamma$ -Lipschitz continuous, if there exists a  $\gamma > 0$  such that  $\|T(f_1) - T(f_2)\| \leq \gamma \|f_1 - f_2\|, \forall f_1, f_2 \in \mathcal{H}$ .
3. The mapping  $T$  is called nonexpansive if it is 1-Lipschitzian. It is well-known that whenever  $\text{Fix}(T)$  is nonempty, then it is a closed convex subset of  $\mathcal{H}$  [65]. A subclass of nonexpansive mappings are the averaged ones. More specifically, the mapping  $T$  is called  $\alpha$ -averaged nonexpansive, with  $\alpha \in (0, 1)$ , if there exists a nonexpansive mapping  $R : \mathcal{H} \rightarrow \mathcal{H}$  such that  $T = (1 - \alpha)I + \alpha R$ . A celebrated example of an  $\frac{1}{2}$ -averaged nonexpansive mapping is the metric projection mapping  $P_C$  onto a nonempty, closed convex subset  $C$  of  $\mathcal{H}$ . In this case, it can be easily verified that  $\text{Fix}(P_C) = C$ . For an illustration of a nonexpansive mapping, see Figure 17.31.

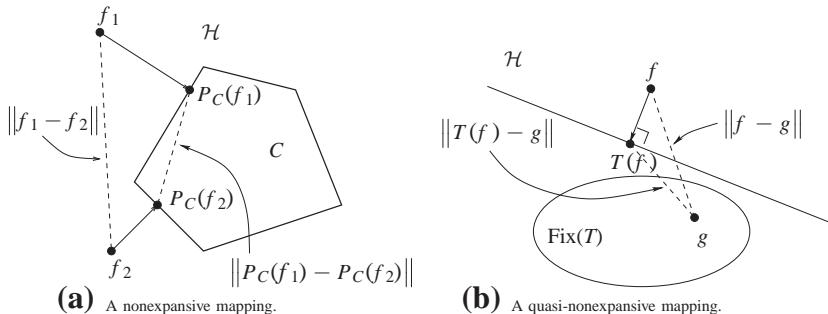


FIGURE 17.31

Illustration of a nonexpansive and a quasi-nonexpansive mapping.

4. The mapping  $T$  is called *strictly contractive* if it is  $\gamma$ -Lipschitzian, with  $\gamma \in (0, 1)$ . Such a mapping is the engine behind the celebrated *Banach-Picard fixed point theorem* [65], which states that the sequence generated by the repetition of  $T$  converges strongly to the *unique* fixed point of  $T$ , i.e., for an arbitrary  $f_0 \in \mathcal{H}$ , the sequence  $f_{n+1} := T(f_n)$ ,  $\forall n$ , converges strongly to  $f_*$ , where  $\text{Fix}(T) = \{f_*\}$ .
5. The mapping  $T$ , with  $\text{Fix}(T) \neq \emptyset$ , is called *quasi-nonexpansive* if

$$\|T(f) - g\| \leq \|f - g\|, \quad \forall f \in \mathcal{H}, \forall g \in \text{Fix}(T).$$

The fixed point set of a quasi-nonexpansive mapping is closed and convex. Every nonexpansive mapping is a quasi-nonexpansive one, but, in general, not vice-versa. The illustration of a quasi-nonexpansive mapping can be found in Figure 17.31b.

◇

The following is a celebrated example of a  $\gamma$ -Lipschitzian mapping.

**Example 45.** Consider a Euclidean space  $\mathcal{R}^l$ , and the following quadratic function:

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{2} \boldsymbol{\theta}^T \mathbf{R} \boldsymbol{\theta} - \mathbf{r}^T \boldsymbol{\theta} + \xi, \quad \forall \boldsymbol{\theta} \in \mathcal{R}^l,$$

where  $\mathbf{R}$  is a positive semidefinite matrix, of dimensions  $l \times l$ , with maximum eigenvalue  $\sigma_{\max} > 0$ , and  $\xi \in \mathcal{R}$ ,  $\mathbf{r} \in \mathcal{R}^l$ .

It can be easily verified that  $\mathcal{L}'(\boldsymbol{\theta}) = \mathbf{R}\boldsymbol{\theta} - \mathbf{r}$ ,  $\forall \boldsymbol{\theta} \in \mathcal{R}^l$ . Then,  $\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathcal{R}^l$ ,

$$\|\mathcal{L}'(\boldsymbol{\theta}_1) - \mathcal{L}'(\boldsymbol{\theta}_2)\| = \|\mathbf{R}(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)\| \leq \|\mathbf{R}\| \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\| = \sigma_{\max} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|,$$

where  $\|\mathbf{R}\| = \sigma_{\max}$  is the spectral norm of  $\mathbf{R}$ . Hence,  $\mathcal{L}'$  is  $\sigma_{\max}$ -Lipschitzian.

As the following theorem suggests, there is a strong link between  $\gamma$ -Lipschitz continuity and nonexpansiveness.

**Fact 46 (The Baillon-Haddad theorem [70,71]).** Assume a differentiable convex function  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R}$ , with its gradient denoted by  $\mathcal{L}' : \mathcal{H} \rightarrow \mathcal{H}$ . Then the following two statements are equivalent.

1.  $\mathcal{L}'$  is  $\gamma$ -Lipschitzian.
2. The mapping  $T := I - \frac{2}{\gamma} \mathcal{L}' : \mathcal{H} \rightarrow \mathcal{H}$  is nonexpansive, where  $I$  stands for the identity mapping in  $\mathcal{H}$ .

The following is an example of an averaged nonexpansive mapping.

**Example 47 (Proximal mapping).** Given a convex continuous<sup>2</sup> function  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R}$ , the *Moreau envelope of index  $\gamma > 0$*  of  $\mathcal{L}$  is the function

$$\gamma \mathcal{L} : \mathcal{H} \rightarrow \mathcal{R} : f \mapsto \inf_{g \in \mathcal{H}} \left( \mathcal{L}(g) + \frac{1}{2\gamma} \|f - g\|^2 \right). \quad (17.81)$$

Then, the *proximal mapping*  $\text{prox}_{\gamma \mathcal{L}}$  is defined as the mapping which maps to an  $f \in \mathcal{H}$  the *unique* minimizer of (17.81) [72–74]. It can be verified that the proximal mapping  $\text{prox}_{\gamma \mathcal{L}}$  is  $\frac{1}{2}$ -averaged nonexpansive with fixed point set  $\text{Fix}(\text{prox}_{\gamma \mathcal{L}}) = \text{argmin}_{f \in \mathcal{H}} \mathcal{L}(f)$  [72, 73].

A well-known example of the proximal mapping is the classical projection operator onto closed convex sets. Indeed, if  $C$  is a non-empty closed convex subset of  $\mathcal{H}$ , then  $\text{prox}_{\gamma \mathcal{L}} = P_C$ ,  $\forall \gamma > 0$ , if the function  $\mathcal{L}$  takes the special form of the *indicator function of  $C$* ,  $\iota_C : \mathcal{H} \rightarrow \mathcal{R} \cup \{+\infty\}$ , which is defined as follows;  $\iota_C(f) = 0$ , if  $f \in C$ , and  $\iota_C(f) = +\infty$ , if  $f \notin C$ .<sup>3</sup> Another celebrated example of a proximal mapping is the well-known soft-thresholding operator (Example 2.16) [72].

Let us give now an example of a quasi-nonexpansive mapping.

**Example 48 (Subgradient projection mapping).** Assume a convex function  $\mathcal{L} : \mathcal{H} \times \mathcal{R}$ , such that its level set of height 0 is nonempty, i.e.,  $\text{lev}_{\leq 0}(\mathcal{L}) \neq \emptyset$ . Then, the subgradient projection mapping  $T_{\mathcal{L}} : \mathcal{H} \rightarrow \mathcal{H}$  with respect to  $\mathcal{L}$  is defined as follows:

$$T_{\mathcal{L}}(f) := \begin{cases} f - \frac{\mathcal{L}(f)}{\|\mathcal{L}'(f)\|^2} \mathcal{L}'(f), & \text{if } f \notin \text{lev}_{\leq 0}(\mathcal{L}), \\ f, & \text{otherwise,} \end{cases}$$

where  $\mathcal{L}'(f)$  is any subgradient taken from the subdifferential  $\partial \mathcal{L}(f)$  at the point  $f$ . For the geometry behind the concept of  $T_{\mathcal{L}}$  see Figure 17.29. It can be verified that  $T_{\mathcal{L}}(f)$  is a quasi-nonexpansive mapping, with  $\text{Fix}(T) = \text{lev}_{\leq 0}(\mathcal{L})$  [65].

**Remark 49.** If we rely again on geometry, it can be seen that in the case where  $f \notin \text{lev}_{\leq 0}(\mathcal{L})$ , then the image of  $f$  under  $T_{\mathcal{L}}$  is nothing but the (metric) projection of  $f$  onto the hyperplane which is defined by the intersection of  $\mathcal{X}_{\mathcal{L}'(f)}$  with  $\mathcal{H}$ ; see Figure 17.29.

**Proof.** In Figure 17.29, the intersection of  $\mathcal{X}_g$ , or better  $\mathcal{X}_{\mathcal{L}'(f)}$ , with  $\mathcal{H}$  is given if we set  $r = 0$  in (17.78). In other words,

$$\begin{aligned} \mathcal{X}_{\mathcal{L}'(f)} \cap \mathcal{H} &= \{h \in \mathcal{H} : 0 = \langle \mathcal{L}'(f), h - f \rangle + \mathcal{L}(f)\} \\ &= \{h \in \mathcal{H} : \langle \mathcal{L}'(f), h \rangle = \langle \mathcal{L}'(f), f \rangle - \mathcal{L}(f)\}. \end{aligned}$$

This is clearly a hyperplane, and by using Example 37 the previous claim is established.  $\square$

<sup>2</sup>In general,  $\mathcal{L}$  is assumed to be lower semicontinuous [65].

<sup>3</sup>Notice that the function  $\iota_C$  is convex and lower semicontinuous.

For the sake of illustration, let us calculate the subgradient projection mappings of several loss functions.

**Example 50.** Assume the loss function

$$\mathcal{L}(f) := d(f, C), \quad \forall f \in \mathcal{H},$$

where  $C$  is a nonempty subset of  $\mathcal{H}$ .

It is easy to see that  $\text{lev}_{\leq 0}(\mathcal{L}) = \text{lev}_{\leq 0}(d(\cdot, C)) = C$ . Hence, in the case where  $f \notin C$ , we have by Example 48 and Example 43 that

$$\begin{aligned} T_{d(\cdot, C)}(f) &= f - \frac{\mathcal{L}(f)}{\|\mathcal{L}'(f)\|^2} \mathcal{L}'(f) \\ &= f - \frac{d(f, C)}{\left\| \frac{f - P_C(f)}{d(f, C)} \right\|^2} \frac{f - P_C(f)}{d(f, C)} \\ &= f - (f - P_C(f)) = P_C(f). \end{aligned} \quad (17.82)$$

In other words, the subgradient projection mapping with respect to the metric distance function is nothing but the classical metric projection mapping  $P_C$ .

◊

**Example 51.** It is interesting to calculate the subgradient projection mapping  $T_{\mathcal{L}}$  with respect to the loss function  $\mathcal{L}(\cdot) = d^2(\cdot, C)$ . By following a procedure similar to the previous one, and by using Example 43, one can verify that

$$T_{d^2(\cdot, C)} = \frac{I + P_C}{2},$$

where  $I$  stands for the identity mapping in  $\mathcal{H}$ .

We stress here the difference between the subgradient projection mappings  $T_{d(\cdot, C)}$  and  $T_{d^2(\cdot, C)}$ . Although both of the loss functions  $d(\cdot, C)$  and  $d^2(\cdot, C)$  have the same level set  $\text{lev}_{\leq 0}(d(\cdot, C)) = \text{lev}_{\leq 0}(d^2(\cdot, C)) = C$ , it seems better to use the *non-differentiable* function  $d(\cdot, C)$  in order to approach  $C$ ; a *single* application of  $T_{d(\cdot, C)}$  takes us directly onto  $C$ , while the application of  $T_{d^2(\cdot, C)}$  covers only half of the distance.

Finally, let us give an example which will be useful in Section 1.17.7.5. The following example introduces the *concurrent* or *parallel* processing of multiple closed convex sets by the single application of a properly defined mapping.

**Example 52.** Assume a number of  $q$  closed convex sets  $\{C_1, C_2, \dots, C_q\}$  such that  $\bigcap_{i=1}^q C_i \neq \emptyset$ . Assume, also, a point  $f \in \mathcal{H}$ , which, for the sake of illustration, is assumed to satisfy  $f \notin \bigcap_{i=1}^q C_i$ . How could we use the concept of the subgradient projection mapping in order to employ concurrently all of  $\{C_1, C_2, \dots, C_q\}$  and to push the point  $f$  closer to  $\bigcap_{i=1}^q C_i$ ?

Since  $f \notin \bigcap_{i=1}^q C_i$ , then there exists an index set  $\mathcal{I}_f$  which indicates those closed convex sets that  $f$  does not belong to, i.e.,

$$\mathcal{I}_f := \{i \in \{1, 2, \dots, q\} : f \notin C_i\}.$$

Given  $\mathcal{I}_f$ , assign to each  $C_i, i \in \mathcal{I}_f$ , a convex weight  $\omega_i$ , i.e.,  $\omega_i \in (0, 1]$  such that  $\sum_{i \in \mathcal{I}_f} \omega_i = 1$ .

Define the function:

$$\mathcal{L}(h) := \sum_{i \in \mathcal{I}_f} \frac{\omega_i d(f, C_i)}{\sum_{j \in \mathcal{I}_f} \omega_j d(f, C_j)} d(h, C_i), \quad \forall h \in \mathcal{H}.$$

Notice here that if we define

$$\beta_i := \frac{\omega_i d(f, C_i)}{\sum_{j \in \mathcal{I}_f} \omega_j d(f, C_j)}, \quad \forall i \in \mathcal{I}_f,$$

then it is easy to verify that  $\mathcal{L}(\cdot) = \sum_{i \in \mathcal{I}_f} \beta_i d(\cdot, C_i)$  with  $\sum_{i \in \mathcal{I}_f} \beta_i = 1$ . In other words,  $\mathcal{L}$  is a convex combination of the convex functions  $d(\cdot, C_i)$ ,  $i \in \mathcal{I}_f$ ; hence, it is convex. Notice, also, that  $\text{lev}_{\leq 0}(\mathcal{L}) = \bigcap_{i \in \mathcal{I}_f} C_i$ .

One can verify by Example 43 that

$$\begin{aligned} \mathcal{L}'(f) &= \sum_{i \in \mathcal{I}_f} \frac{\omega_i d(f, C_i)}{\sum_{j \in \mathcal{I}_f} \omega_j d(f, C_j)} \frac{f - P_{C_i}(f)}{d(f, C_i)} \\ &= \frac{1}{L} \sum_{i \in \mathcal{I}_f} \omega_i (f - P_{C_i}(f)), \end{aligned}$$

where  $L := \sum_{j \in \mathcal{I}_f} \omega_j d(f, C_j)$ . Then, the subgradient projection mapping, applied to  $f$ , becomes as follows:

$$\begin{aligned} T_{\mathcal{L}}(f) &= f - \frac{\frac{1}{L} \sum_{i \in \mathcal{I}_f} \omega_i d^2(f, C_i)}{\frac{1}{L^2} \left\| \sum_{i \in \mathcal{I}_f} \omega_i (f - P_{C_i}(f)) \right\|^2} \frac{1}{L} \sum_{i \in \mathcal{I}_f} \omega_i (f - P_{C_i}(f)) \\ &= f + \mathcal{M} \left( \sum_{i \in \mathcal{I}_f} \omega_i P_{C_i}(f) - f \right), \end{aligned}$$

where

$$\mathcal{M} := \frac{\sum_{i \in \mathcal{I}_f} \omega_i d^2(f, C_i)}{\left\| \sum_{i \in \mathcal{I}_f} \omega_i (f - P_{C_i}(f)) \right\|^2}. \quad (17.83)$$

It becomes clear by the convexity of the function  $\|\cdot\|^2$  that  $\mathcal{M} \geq 1$ .

The *relaxed* version of  $T_{\mathcal{L}}$  is defined as the mapping  $(1 - \mu')I + \mu' T_{\mathcal{L}}$ , where  $\mu' \in (0, 2)$ , and  $I$  is the identity mapping in  $\mathcal{H}$ . Hence, the relaxation of  $T_{\mathcal{L}}$ , applied to the previous  $f$ , becomes as follows:

$$\begin{aligned} (1 - \mu')f + \mu' T_{\mathcal{L}}(f) &= f + \mu' \mathcal{M} \left( \sum_{i \in \mathcal{I}_f} \omega_i P_{C_i}(f) - f \right) \\ &= f + \mu \left( \sum_{i \in \mathcal{I}_f} \omega_i P_{C_i}(f) - f \right), \end{aligned}$$

where  $\mu := \mu' \mathcal{M} \in (0, 2\mathcal{M})$ . Notice that since  $\mathcal{M} \geq 1$ , the parameter  $\mu$  can attain values larger than or equal to 2.

In the case where  $f \in \bigcap_{i=1}^q C_i$ , it is natural to leave  $f$  as it is. To summarize, the previous discussion introduced the following mapping:

$$f \mapsto \begin{cases} f + \mu \left( \sum_{i \in \mathcal{I}_f} \omega_i P_{C_i}(f) - f \right), & \text{if } f \notin \bigcap_{i=1}^q C_i, \\ f, & \text{if } f \in \bigcap_{i=1}^q C_i, \end{cases}$$

where  $\mu \in (0, 2\mathcal{M})$ , with  $\mathcal{M}$  given by (17.83). We will again meet these concepts later on, in Example 67.  $\diamond$

#### 1.17.7.4 The basic algorithmic ingredients for nonsmooth convex batch learning

In this section, we lay the foundations on which the subsequent section of online learning (Section 1.17.7.5) will be based. The objective of this section is to study several basic algorithmic components related to *batch learning*, i.e., the task of solving an estimation problem *after all* the training data have been gathered and made available. Such training data are utilized in order to generate a *fixed* convex loss function  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R}$ , which quantifies the designer's perception of penalty or deviation from the underlying model. If a nonempty closed convex set  $C \subset \mathcal{H}$  is also used as the available *a priori constraint* to the problem at hand, then batch learning breaks down to the following fundamental optimization task:

$$\text{minimize } \mathcal{L}(f) \text{ such that } f \in C \subset \mathcal{H}. \quad (17.84)$$

The only assumption we make on the convex function  $\mathcal{L}$  is to be a continuous one. Neither smoothness, i.e., any form of differentiability, nor strict convexity is assumed for  $\mathcal{L}$ . Such a generality is crucial if one wishes to build algorithmic tools for modern learning tasks; indeed, the  $\ell_1 : \mathcal{R}^l \rightarrow \mathcal{R}$  loss function is a celebrated example of a nonsmooth, nonstrict, convex penalty function which infuses sparsity attributes in a learning task.

The backbone of this section will be the following fundamental recursion; fix arbitrarily an initial point  $f_0 \in \mathcal{H}$  and execute  $\forall n \in \mathcal{N}_*$ ,

$$f_n := P_C(f_{n-1} - \mu_n \mathcal{L}'(f_{n-1})), \quad (17.85)$$

where  $P_C$  stands for the metric projection mapping onto  $C$ ,  $\mathcal{L}'(f_{n-1})$  is the subgradient of  $\mathcal{L}$  at  $f_{n-1}$ , and  $(\mu_n)_{n \in \mathcal{N}_*}$  is a sequence of user-defined, non-negative real numbers, which are usually fixed judiciously in order for the sequence  $(f_n)_{n \in \mathcal{N}}$  to converge in some sense.

Recently, the fundamental recursion (17.85) lent itself to generalizations, and more specifically, to the replacement of the metric projection mapping  $P_C$  of (17.85) with the more general *proximal* mappings (see Example 47), in order to derive the proximal-gradient, aka forward-backward algorithm, and the Douglas-Rachford method [65, 72, 73, 75, 76]. Such a generalization corresponds to the minimization of a sum of convex functions  $\mathcal{L} := \Theta + \psi$ , where  $\Theta$  is a convex function, which accommodates the misfit of the observed data to the model, and  $\psi$  is the proximal function, introduced in order to serve a variety of objectives, depending on the problem at hand. For example,  $\psi$  takes up the role of a regularizer in order to control the “size” of an estimate in classification or regression tasks, or that of a penalty term

which enforces a-priori knowledge, like sparsity, into the design. Such a path, is followed also by the machine learning community in the very recent studies of [77, 78].

**Algorithm 53 (Projected Gradient Method (PGM))** [79, 80]. Assume a nonempty closed convex set  $C \subset \mathcal{H}$ , and a differentiable convex function  $\mathcal{L} : \mathcal{H} \rightarrow \mathcal{R}$ , such that its derivative  $\mathcal{L}' : \mathcal{H} \rightarrow \mathcal{H}$  is  $\gamma$ -Lipschitzian, and  $\operatorname{argmin}_{f \in C} \mathcal{L}(f) \neq \emptyset$ . Then, for any starting point  $f_0 \in \mathcal{H}$ , and any  $\mu \in (0, \frac{2}{\gamma})$ , the sequence  $(f_n)_{n \in \mathcal{N}}$  generated by the *Projected Gradient Method (PGM)*:

$$f_n := P_C(f_{n-1} - \mu \mathcal{L}'(f_{n-1})), \quad \forall n \in \mathcal{N}_*, \quad (17.86)$$

converges weakly to a point in  $\operatorname{argmin}_{f \in C} \mathcal{L}(f)$ .

**Remark 54 (Alternative view of the PGM).** According to the Baillon-Haddad theorem, i.e., Theorem 46, the mapping  $T := I - \frac{2}{\gamma} \mathcal{L}'$  is nonexpansive. Hence, PGM is nothing but the composition of the projection mapping  $P_C$  and the  $\alpha$ -averaged nonexpansive mapping  $\alpha T + (1 - \alpha)I$  (see Definition 44), for  $\alpha \in (0, 1)$ , i.e.,

$$f_n := P_C \left( \alpha \left( I - \frac{2}{\gamma} \mathcal{L}' \right) + (1 - \alpha)I \right) (f_{n-1}), \quad \forall n \in \mathcal{N}_*.$$

Given several scenarios, with respect to  $\mathcal{H}$ ,  $\mathcal{L}$ , and  $C$ , the PGM is hidden behind celebrated algorithmic tools, as the following examples suggest.

**Example 55 (Projected Landweber Method [65]).** Assume a matrix  $A \in \mathcal{R}^{m \times l}$ , a vector  $y \in \mathcal{R}^m$ , and a nonempty closed convex set  $C \subset \mathcal{R}^l$ . For an arbitrarily fixed initial point  $\theta_0 \in \mathcal{R}^l$ , and an  $\mu \in (0, 2/\|A\|^2)$ , where  $\|A\|$  stands for the spectral norm of  $A$ , the *Projected Landweber Method*:

$$\theta_n := P_C \left( \theta_{n-1} - \mu A^T (A\theta_{n-1} - y) \right), \quad \forall n \in \mathcal{N}_*,$$

converges to a point in  $\operatorname{argmin}_{\theta \in C} \frac{1}{2} \|A\theta - y\|^2 \neq \emptyset$ .  $\diamond$

**Proof.** If we define  $\mathcal{L}(\theta) := \frac{1}{2} \|A\theta - y\|^2$ ,  $\forall \theta \in \mathcal{R}^l$ , then it can be easily verified that

$$\mathcal{L}(\theta) = \frac{1}{2} \theta^T A^T A \theta - y^T A \theta + \frac{1}{2} \|y\|^2, \quad \theta \in \mathcal{R}^l.$$

According to Example 45, the derivative  $\mathcal{L}'(\theta) = A^T A \theta - A^T y$ ,  $\forall \theta \in \mathcal{R}^l$ , is  $\sigma_{\max}$ -Lipschitzian, where  $\sigma_{\max}$  is the maximum eigenvalue of  $A^T A$ . Since by definition  $\sigma_{\max} = \|A\|^2$  [81], if we also set  $C := \mathcal{H}$ , then the Projected Landweber Method is a direct application of the PGM to the specific  $\mathcal{L}$ .  $\square$

**Example 56 (Minimum Mean-Squared Error (MMSE) estimation [5, 6]).** Assume the linear model:

$$y = \mathbf{x}^T \boldsymbol{\theta} + \eta, \quad (17.87)$$

where  $\boldsymbol{\theta} \in \mathcal{R}^l$  is a parameter vector,  $y \in \mathcal{R}$  is the observed signal, and  $\mathbf{x} \in \mathcal{R}^l$ ,  $\eta \in \mathcal{R}$  are the vector-valued input signal and the noise random variables, respectively. The correlation matrix of the

input signal  $\mathbf{R} := \mathbb{E}[\mathbf{x}\mathbf{x}^T]$ , where  $\mathbb{E}[\cdot]$  stands for the expectation operator, is obviously a positive semidefinite matrix. In practice, available to us are only realizations of (17.87) via the sequences  $(\mathbf{x}_n, y_n)_{n \in \mathcal{N}_*}$  and  $(\eta_n)_{n \in \mathcal{N}_*}$  which are related to each other via  $y_n = \mathbf{x}_n^T \boldsymbol{\theta} + \eta_n, \forall n \in \mathcal{N}_*$ .

Define the mean square error as follows:

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{2} \mathbb{E}[(y - \mathbf{x}^T \boldsymbol{\theta})^2], \quad \forall \boldsymbol{\theta} \in \mathcal{R}^l.$$

Our objective is to compute an *estimate* of  $\boldsymbol{\theta}$ , by the *least mean squares* rationale:

$$\text{find } \boldsymbol{\theta}_* \in \underset{\boldsymbol{\theta} \in \mathcal{R}^l}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}). \quad (17.88)$$

It is easy to verify that

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \boldsymbol{\theta}^T \mathbf{R} \boldsymbol{\theta} - \mathbb{E}[y \mathbf{x}^T] \boldsymbol{\theta} + \frac{1}{2} \mathbb{E}[y^2] = \frac{1}{2} \boldsymbol{\theta}^T \mathbf{R} \boldsymbol{\theta} - \mathbf{r}^T \boldsymbol{\theta} + \xi,$$

where  $\mathbf{R} := \mathbb{E}[y \mathbf{x}]$  and  $\xi := \frac{1}{2} \mathbb{E}[y^2]$ .

According to Example 45, the function  $\mathcal{L}$  is  $\sigma_{\max}$ -Lipschitzian, where  $\sigma_{\max}$  is the maximum eigenvalue of  $\mathbf{R}$ . Hence, the sequence  $(\boldsymbol{\theta}_n)_{n \in \mathcal{N}}$  generated by the PGM of Algorithm 53:

$$\boldsymbol{\theta}_n := \boldsymbol{\theta}_{n-1} - \mu (\mathbf{R} \boldsymbol{\theta}_{n-1} - \mathbf{r}), \quad \forall n \in \mathcal{N}_*, \quad (17.89)$$

converges to a point  $\boldsymbol{\theta}_* \in \operatorname{argmin}_{\boldsymbol{\theta} \in C} \mathcal{L}(\boldsymbol{\theta})$ , provided that  $\mu \in (0, \frac{2}{\sigma_{\max}})$ . Indeed, this iterative procedure is of paramount importance to the celebrated MMSE and Wiener estimation theory [5, 6]. Moreover, this iteration is the motivation for the classical LMS algorithm (17.34). To see this, notice that (17.89) can be equivalently written  $\forall n \in \mathcal{N}_*$  as:

$$\begin{aligned} \boldsymbol{\theta}_n &= \boldsymbol{\theta}_{n-1} - \mu (\mathbf{R} \boldsymbol{\theta}_{n-1} - \mathbf{r}) \\ &= \boldsymbol{\theta}_{n-1} - \mu \left( \mathbb{E}[\mathbf{x}_n \mathbf{x}_n^T] \boldsymbol{\theta}_{n-1} - \mathbb{E}[y_n \mathbf{x}_n] \right) \\ &= \boldsymbol{\theta}_{n-1} - \mu \mathbb{E}[(\mathbf{x}_n^T \boldsymbol{\theta}_{n-1} - y_n) \mathbf{x}_n]. \end{aligned}$$

It can be readily verified that (17.34) is an approximation of the previous iteration. Notice also that the Lipschitz constant  $\frac{2}{\sigma_{\max}}$  of the  $\mathcal{L}'$  mapping is of central importance for the proof of convergence in the mean of the LMS algorithm (see Section 1.17.6.1).

Let us now go back to (17.85) for the case where  $\mathcal{L}$  is not differentiable. Although there are several strategies to control the step-sizes  $(\mu_n)_{n \in \mathcal{N}}$  in (17.85), in this note we focus on the following method.

**Algorithm 57 (Projected Subgradient Method (PSMa))** [82, 83]. For any  $f_0 \in \mathcal{H}$ , let the following recursion

$$f_n := P_C \left( f_{n-1} - \frac{\mu_n}{\max\{1, \|\mathcal{L}'(f_{n-1})\|\}} \mathcal{L}'(f_{n-1}) \right), \quad n = 1, 2, \dots,$$

(17.90)

where  $\mathcal{L}'(f_{n-1})$  stands for the subgradient of  $\mathcal{L}$  at  $f_{n-1}$ , and the non-negative numbers  $(\mu_n)_{n \in \mathcal{N}_*}$  satisfy the following conditions:

$$\sum_{n=1}^{\infty} \mu_n = \infty, \quad (17.91a)$$

$$\sum_{n=1}^{\infty} \mu_n^2 < \infty. \quad (17.91b)$$

Regarding the previous algorithm, the following hold true.

1. If the previous algorithm generates an infinite sequence  $(f_n)_{n \in \mathcal{N}}$ , then  $\liminf_{n \rightarrow \infty} \mathcal{L}(f_n) = \inf_{f \in C} \mathcal{L}(f)$ .
2. If the set of solutions  $C_{\mathcal{L}_*}$  of (17.84) is nonempty, then either (17.90) stops at some iteration  $n_*$ , in which case  $f_{n_*} \in C_{\mathcal{L}_*}$ , or it generates an infinite sequence which converges *weakly* to some  $f_* \in C_{\mathcal{L}_*}$ .
3. If  $C_{\mathcal{L}_*}$  is empty, then  $(f_n)_{n \in \mathcal{N}}$  is unbounded.

**Remark 58.** The previous convergence results hold also true in the case where the  $\mathcal{L}'(f_n)$  is taken from the  $\epsilon$ -subdifferential ( $\epsilon \geq 0$ ) of  $\mathcal{L}$  at  $f_n$ , which stands as a generalization of the classical subdifferential [62]. It has been observed in [83] that a slight variation of (17.90) is sufficient to guarantee convergence of the sequence of estimates  $(f_n)_{n \in \mathcal{N}}$  in the strong sense.

The unrestricted case of (17.84), i.e., the case where  $C := \mathcal{H}$ , was studied in [84]. In [84], the condition (17.91b) is relaxed to the  $\lim_{n \rightarrow \infty} \mu_n = 0$ , and the recursion is formed as [84]

$$f_n := \begin{cases} f_{n-1} - \frac{\mu_n}{\|\mathcal{L}'(f_{n-1})\|} \mathcal{L}'(f_{n-1}), & \text{if } \mathcal{L}'(f_{n-1}) \neq 0, \\ f_{n-1}, & \text{otherwise.} \end{cases} \quad (17.92)$$

It is proved in [84] that the generating sequence  $(f_n)_{n \in \mathcal{N}}$  is a minimizing one, i.e.,  $\liminf_{n \rightarrow \infty} \mathcal{L}(f_n) = \inf_{f \in C} \mathcal{L}(f)$ , but no results on the convergence of the  $(f_n)_{n \in \mathcal{N}}$  are given. The finite dimensional unconstrained case of (17.85), i.e.,  $\mathcal{H}$  is a Euclidean space and  $C := \mathcal{H}$ , but with different strategies on designing the quantities  $(\mu_n)_{n \in \mathcal{N}_*}$ , was studied in [85, 86].

For the sake of illustration, let us apply the PSMa to a classical learning task, and show that this classical rule falls under the previously generic recursion.

**Example 59 (The Perceptron).** Let two patterns  $\Pi_1, \Pi_2$ , from which a number of finite dimensional observations emanate:  $\mathbf{x}_i \in \mathcal{R}^l$ ,  $i = 1, 2, \dots, N_1$ , from  $\Pi_1$ , and  $\boldsymbol{\xi}_j \in \mathcal{R}^l$ ,  $j = 1, 2, \dots, N_2$ , from  $\Pi_2$ . Assume also a *reproducing kernel* function  $\kappa$  which generates an RKHS  $\mathcal{H}$  of very high, possibly infinite, dimension. The observed data are mapped into  $\mathcal{H}$  by means of the following mapping:  $\mathbf{x}_i \mapsto \kappa(\mathbf{x}_i, \cdot)$ , and  $\boldsymbol{\xi}_j \mapsto \kappa(\boldsymbol{\xi}_j, \cdot)$ .

We assume that the patterns  $\Pi_1, \Pi_2$  are *linearly*, or better, *affinely separable* in the feature space  $\mathcal{H}$ , i.e., there exist  $f \in \mathcal{H}$  and  $\beta \in \mathcal{R}$ , such that

$$\begin{aligned} \langle f, \kappa(\mathbf{x}_i, \cdot) \rangle + \beta &= f(\mathbf{x}_i) + \beta \geq 0, \quad \forall i \in \{1, 2, \dots, N_1\}, \\ \langle f, \kappa(\boldsymbol{\xi}_j, \cdot) \rangle + \beta &= f(\boldsymbol{\xi}_j) + \beta \leq 0, \quad \forall j \in \{1, 2, \dots, N_2\}. \end{aligned} \quad (17.93)$$

Our task becomes that of employing an iterative algorithm in order to find an  $f \in \mathcal{H}$  and a  $\beta \in \mathcal{R}$  which achieve the previous goal. An example of such an iterative procedure is the *perceptron*, which was introduced in [87, 88], in order to solve the task (17.93).

**Proposition 60.** *Assume that there exist  $f \in \mathcal{H}$  and  $\beta \in \mathcal{R}$  such that*

$$\begin{aligned} f(\mathbf{x}_i) + \beta &> 0, \quad \forall i \in \{1, 2, \dots, N_1\}, \\ f(\xi_j) + \beta &< 0, \quad \forall j \in \{1, 2, \dots, N_2\}. \end{aligned}$$

*Then, Algorithm 57 (PSMa) solves the task (17.93) in a finite number of steps.* ◇

**Proof.** See Appendix B. □

According to the studies of [84, 85], the PSMa of Algorithm 57 and its variants exhibit in general a rather slow convergence performance, due to the selection policies of the step-sizes  $(\mu_n)_{n \in \mathcal{N}}$ . To remedy such a situation, Polyak [89] introduced the following algorithm.

**Algorithm 61 (Projected Subgradient Method (PSMb) [89]).** Define, first,  $\mathcal{L}_* := \inf_{f \in C} \mathcal{L}(f)$ . For an arbitrarily fixed  $f_0 \in \mathcal{H}$ , assume the recursion  $\forall n \in \mathcal{N}_*$ :

$$f_n := \begin{cases} P_C \left( f_{n-1} - \mu_n \frac{\mathcal{L}(f_{n-1}) - \mathcal{L}_*}{\|\mathcal{L}'(f_{n-1})\|^2} \mathcal{L}'(f_{n-1}) \right), & \text{if } \mathcal{L}'(f_{n-1}) \neq 0, \\ P_C(f_{n-1}), & \text{otherwise,} \end{cases} \quad (17.94)$$

where  $\mu_n \in (0, 2)$ .

Assume that the set of solutions  $C_{\mathcal{L}_*}$  of (17.84) is nonempty, and let a point  $f_* \in C_{\mathcal{L}_*}$ . If, in addition, there exists an  $\epsilon > 0$  such that  $\mu_n \in [\epsilon, 2 - \epsilon]$ ,  $\forall n$ , and there exists a  $c > 0$  such that  $\|\mathcal{L}'(f)\| \leq c$ ,  $\forall f \in B[f_*, \|f_0 - f_*\|] \cap C$ , where  $B[f_*, \|f_0 - f_*\|]$  stands for the closed ball centered at  $f_*$  with radius  $\|f_0 - f_*\|$ , then

1.  $\lim_{n \rightarrow \infty} \mathcal{L}(f_n) = \mathcal{L}_*$ .
2. The sequence  $(f_n)_{n \in \mathcal{N}}$  converges weakly to a point in  $C_{\mathcal{L}_*}$ . ◇

**Proof.** See Appendix C. □

The following proposition shows that the PSMB, i.e., Algorithm 61, has a clear geometric description as the composition of two projections, a relaxed version of a subgradient one, and a metric one.

**Proposition 62.** *If we assume that the set of all minimizers of the function  $\mathcal{L}$  on  $C$  is nonempty, i.e.,  $C_{\mathcal{L}_*} \neq \emptyset$ , then the PSM Algorithm 61 can be cast in the following equivalent form:*

$$f_n = P_C \left( (1 - \mu_n)I + \mu_n T_{\tilde{\mathcal{L}}} \right) (f_{n-1}), \quad \forall n \in \mathcal{N}_*,$$

where  $I$  stands for the identity mapping in  $\mathcal{H}$ , and the mapping  $T_{\tilde{\mathcal{L}}} : \mathcal{H} \rightarrow \mathcal{H}$  stands for the subgradient projection mapping (see Example 48) with respect to the function  $\tilde{\mathcal{L}}(\cdot) := \mathcal{L}(\cdot) - \mathcal{L}_*$ . ◇

**Proof.** First, notice that the mapping

$$(1 - \mu)I + \mu T_{\tilde{\mathcal{L}}}, \quad \mu \in (0, 2),$$

is called the *relaxed*  $T_{\tilde{\mathcal{L}}}$  [65].

It is also easy to verify that  $\text{lev} \leq 0(\tilde{\mathcal{L}}) = \{f \in \mathcal{H} : \mathcal{L}(f) \leq \mathcal{L}_*\}$ . Since  $C_{\mathcal{L}_*} \neq \emptyset$ , it can be clearly seen that  $\emptyset \neq C_{\mathcal{L}_*} \subset \text{lev} \leq 0(\tilde{\mathcal{L}})$ . Moreover, since  $\tilde{\mathcal{L}}$  is nothing but a vertical shift of  $\mathcal{L}$ , it can be also readily seen that  $\partial \tilde{\mathcal{L}} = \partial \mathcal{L}$ . Under this setting, the claim of the proposition can be easily established by a simple inspection of Example 48.  $\square$

### 1.17.7.5 Algorithmic tools for nonsmooth convex online learning

Based on Section 1.17.7.4, we will present here a variety of algorithmic tools designed for *online learning*, i.e., learning tasks where the training data *keep on flowing sequentially* into the processing unit. As it has already been stated in the introduction of the chapter, such a *continuous* flow is dictated by the need to monitor possible fluctuations or abrupt changes of the surrounding environment, as well as any time-varying a-priori knowledge about the model under study. The *sequential* flow of the training data is dictated by real-time units where the need for processing those data one by one, and not in batches as in Section 1.17.7.4, is the key ingredient for simpler computational operations, and thus, for less consumed power, as well as for easier handling of massive volumes of incoming data.

In order to cast the previous considerations into a mathematical formulation, we let the non-negative integer index  $n$  to account not only for the recursion index, as this was the case in Section 1.17.7.4, but also to denote *time instants*. In other words, the set of all non-negative integers  $\mathcal{N}$  stands also for the entire *time horizon*. Hence, instead of the fixed loss function  $\mathcal{L}$  and the constraint set  $C$  of (17.84), we will deal here with a sequence of loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  and constraint sets  $(C_n)_{n \in \mathcal{N}_*}$ , and the task of (17.84) becomes now that of an *asymptotic* minimization one.

As in Section 1.17.7.4, the loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  are not confined to be either differentiable or strictly convex. In such a way, design freedom is favored against easiness; although the differentiability or strict convexity of  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  might result into elegant and easy-to-implement algorithmic tools, as well as stronger convergence results, our priority is not the mathematical tractability, but the selection of those convex loss functions that are *in agreement* with the available model and the a-priori knowledge.

Our data generation model, in its most general form, is given as follows:

$$y_n = f_*(\mathbf{x}_n) + \eta_n, \quad \forall n \in \mathcal{N}_*, \quad (17.95a)$$

where  $f_*$  is an element of an RKHS  $\mathcal{H}$ , and thus, it is in general a *nonlinear* function  $f_* : \mathcal{R}^l \rightarrow \mathcal{R}$ , and  $(\eta_n)_{n \in \mathcal{N}_*}$  is the noise stochastic process. Notice that due to the reproducing property of the RKHS, the quantity  $f_*(\mathbf{x}_n) = \langle f_*, \kappa(\mathbf{x}_n, \cdot) \rangle$ ; in other words, the  $l$ -dimensional input data  $(\mathbf{x}_n)_{n \in \mathcal{N}_*}$  are mapped into the highly dimensional feature space  $\mathcal{H}$  as  $\mathbf{x}_n \mapsto \kappa(\mathbf{x}_n, \cdot)$ . The real-valued sequence  $(y_n)_{n \in \mathcal{N}_*}$  contains the observed data. For short, the sequence  $(y_n, \mathbf{x}_n)_{n \in \mathcal{N}_*}$  will be called the sequence of *training data* in the sequel. Given these training data, our task is to estimate the *unknown* nonlinear function  $f_*$ .

The nonlinear model of (17.95a) takes a simpler form whenever the kernel behind the feature RKHS  $\mathcal{H}$  is chosen to be the linear one. In such a case,  $\mathcal{H}$  becomes an  $l$ -dimensional Euclidean space, and (17.95a) reduces to the classical *linear* model

$$y_n = \mathbf{x}_n^T \boldsymbol{\theta}_* + \eta_n, \quad \forall n \in \mathcal{N}_*, \quad (17.95b)$$

where  $\boldsymbol{\theta}_* \in \mathcal{R}^l$  is now the unknown vector to be estimated.

The previous training data  $(y_n, \mathbf{x}_n)_{n \in \mathcal{N}_*}$  as well as the underlying model (17.95a) are used to construct the sequence of loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$ . Given also the sequence of closed convex sets  $(C_n)_{n \in \mathcal{N}_*}$ , which depicts the time-varying a-priori information, the recursion which runs beneath almost all of the algorithmic tools developed for convex online learning tasks is a generalization of (17.85) to the current time-adaptive setting; given any initial point  $f_0 \in \mathcal{H}$ , execute

$$f_n := P_{C_n} (f_{n-1} - \mu_n \mathcal{L}'_n(f_{n-1})), \quad \forall n \in \mathcal{N}_*. \quad (17.96)$$

In what follows, and as a special case of (17.96), we provide with a classical time-adaptive algorithm.

**Algorithm 63 (Least Mean Squares (LMS) algorithm [5,6,36,38]).** Regarding (17.95a), assume the loss function

$$\mathcal{L}_n(f) := \frac{1}{2} (y_n - f(\mathbf{x}_n))^2 = \frac{1}{2} (y_n - \langle f, \kappa(\mathbf{x}_n, \cdot) \rangle)^2, \quad \forall f \in \mathcal{H}.$$

Clearly, this loss function is everywhere differentiable, and

$$\mathcal{L}'_n(f) = (f(\mathbf{x}_n) - y_n) \kappa(\mathbf{x}_n, \cdot), \quad \forall f \in \mathcal{H}.$$

Adopting the rationale of the PGM, i.e., Algorithm 53, leads to the *Kernel Least Mean Squares (KLMS)* recursion:

$$\begin{aligned} f_n &:= f_{n-1} - \mu \mathcal{L}'_n(f_{n-1}) \\ &= f_{n-1} - \mu (f_{n-1}(\mathbf{x}_n) - y_n) \kappa(\mathbf{x}_n, \cdot), \quad \forall n \in \mathcal{N}, \end{aligned} \quad (17.97)$$

where  $\mu$  takes values in order to guarantee the convergence of the sequence  $(f_n)_{n \in \mathcal{N}}$  in some sense.

In the case where the RKHS  $\mathcal{H}$  becomes the Euclidean  $\mathcal{R}^l$ , via a linear kernel, the nonlinear model (17.95a) reduces to (17.95a), and the KLMS (17.97) becomes the classical LMS algorithm:

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} - \mu (\mathbf{x}_n^T \boldsymbol{\theta}_{n-1} - y_n) \mathbf{x}_n, \quad \forall n \in \mathcal{N}_*. \quad \diamond$$

Motivated by the geometry behind the batch Algorithm 61, the following online counterpart was introduced in [90,91].

**Algorithm 64 (Adaptive Projected Subgradient Method (APSM) [90,91]).** Assume a sequence of non-negative, continuous, convex loss functions  $(\mathcal{L}_n : \mathcal{H} \rightarrow \mathcal{R})_{n \in \mathcal{N}_*}$ , not necessarily differentiable. Let, also, a nonempty closed convex set  $C \subset \mathcal{H}$ . Then, for an arbitrarily fixed initial point  $f_0$ , the *Adaptive Projected Subgradient Method (APSM)* is formulated as follows;  $\forall n \in \mathcal{N}_*$ ,

$$f_n := \begin{cases} P_C \left( (1 - \mu_n) f_{n-1} + \mu_n \left( f_{n-1} - \frac{\mathcal{L}_n(f_{n-1})}{\|\mathcal{L}'_n(f_{n-1})\|^2} \mathcal{L}'_n(f_{n-1}) \right) \right), & \text{if } \mathcal{L}'_n(f_{n-1}) \neq 0, \\ P_C(f_{n-1}), & \text{if } \mathcal{L}'_n(f_{n-1}) = 0, \end{cases} \quad (17.98)$$

where  $\mu_n \in (0, 2)$ , and  $\mathcal{L}'_n(f_{n-1})$  stands for any subgradient of the function  $\mathcal{L}_n$  at  $f_{n-1}$ ,  $\forall n \in \mathcal{N}_*$ .  $\diamond$

**Algorithm 65.** Adaptive Projected Subgradient Method (APSM)

**Require:** The training data  $((x_n, y_n))_{n \in \mathcal{N}_*}$ , the sequence of non-negative convex loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$ , an RKHS  $\mathcal{H}$ , and a closed convex set  $C$ .

```

1: Choose any  $f_0 \in \mathcal{H}$ .
2: for  $n = 1, 2, \dots$  do
3: Available is the current estimate  $f_{n-1}$ .
4: Choose any  $\mu_n \in (0, 2)$ .
5: if  $\mathcal{L}'_n(f_{n-1}) \neq 0$  then
6:    $f_n := P_C \left( (1 - \mu_n) f_{n-1} + \mu_n \left( f_{n-1} - \frac{\mathcal{L}_n(f_{n-1})}{\|\mathcal{L}'_n(f_{n-1})\|^2} \mathcal{L}'_n(f_{n-1}) \right) \right)$ .
7: else
8:    $f_n := P_C(f_{n-1})$ .
9: end if
10: end for
```

**Output:** The sequence of estimates  $(f_n)_{n \in \mathcal{N}}$ .

**Remark 66.** In order to spread the applicability range of the APSM to scenarios with more involved a-priori information usage than the single projection mapping  $P_C$  in (17.98), the study in [92] extended the APSM to the case where certain nonexpansive mappings take the place of  $P_C$  in (17.98). Moreover, very recently, further generalization has been achieved for the case where the more flexible and general *quasi-nonexpansive* mappings, which include, for example, the proximal ones (see Example 47) as a special case, take up the role of handling the time-varying a-priori information [92–94]. Indeed, such generalizations span the applicability of the APSM [95] from classification [96] and regression tasks [42] in RKHSs, to sparsity-aware signal recovery [97], and sensor network [98] applications. ◇

The following section provides several important versions of the generic scheme of Algorithm 65.

### 1.17.7.6 The kernel adaptive projected subgradient method (KAPSM)

We are now ready to state the kernel APSM algorithm, which together with the KLMS and KRLS, comprise three major online/time adaptive schemes in Signal Processing. Moreover, as we will see, the KAPSM, as it is the case with its linear counterpart, enjoys a number of advantages, including its agility to cope efficiently both with regression as well as with classification problems, and also to allow for straightforward modifications to cope with robustness, when the environment dictates so. The reason behind this agility lies on the freedom of choice for the sequence of convex loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  in Algorithm 65; the nature of the problem, e.g., a classification or a regression task, and the objectives of the designer dictate the choice of  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$ . In order to be more concrete, we start our discussion with Example 67, which incorporates a sequence of specialized loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  in the APSM, but with a wide applicability range. For example, the KNLMS (see (17.41)), as well as the Kernel Affine Projection Algorithm (KAPA) [37, 68], which is a kernelized version of the classical APA [6, 99, 100], become its special cases.

**Example 67 ([101]).** Assume a sequence of closed convex sets  $(C_n)_{n \in \mathcal{N}_*}$ , where each  $C_n$  is associated to a pair of training data  $(y_n, \mathbf{x}_n)$ . Given a positive integer  $q$ , the following discussion will evolve along the lines of the rationale given in Example 52, and it will introduce the *concurrent* or *parallel* processing of multiple closed convex sets or data per time instant  $n$ . Hence, given a point  $f_{n-1} \in \mathcal{H}$ , define, first, the index set

$$\mathcal{I}_{f_{n-1}} := \{i \in \{n-q+1, \dots, n\} : f_{n-1} \notin C_i\}.$$

To each  $i \in \mathcal{I}_{f_{n-1}}$ , we assign a convex weight  $\omega_i^{(n)}$ , i.e.,  $\omega_i^{(n)} \in (0, 1]$  such that  $\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} = 1$ .

Define the function:

$$\mathcal{L}_n(f) := \sum_{i \in \mathcal{I}_{f_{n-1}}} \frac{\omega_i^{(n)} d(f_{n-1}, C_i)}{\sum_{j \in \mathcal{I}_{f_{n-1}}} \omega_j^{(n)} d(f_{n-1}, C_j)} d(f, C_i), \quad \forall f \in \mathcal{H}. \quad (17.99)$$

By mimicking the subgradient projection mapping, the procedure of Example 52, and for some  $\mu'_n \in (0, 2)$ , we obtain

$$\begin{aligned} & (1 - \mu'_n) f_{n-1} + \mu'_n \left( f_{n-1} - \frac{\mathcal{L}_n(f_{n-1})}{\|\mathcal{L}'_n(f_{n-1})\|^2} \mathcal{L}'_n(f_{n-1}) \right) \\ &= f_{n-1} - \mu'_n \frac{\mathcal{L}_n(f_{n-1})}{\|\mathcal{L}'_n(f_{n-1})\|^2} \mathcal{L}'_n(f_{n-1}) \\ &= f_{n-1} + \mu'_n \mathcal{M}'_n \left( \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} P_{C_i}(f_{n-1}) - f_{n-1} \right), \end{aligned}$$

where

$$\mathcal{M}'_n := \frac{\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} d^2(f_{n-1}, C_i)}{\left\| \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} (f_{n-1} - P_{C_i}(f_{n-1})) \right\|^2}.$$

Hence, the APSM for this special case of  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  becomes as follows:

$$f_n = f_{n-1} + \mu_n \left( \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} P_{C_i}(f_{n-1}) - f_{n-1} \right), \quad (17.100a)$$

where the *extrapolation parameter*  $\mu_n \in (0, 2\mathcal{M}_n)$ , with

$$\mathcal{M}_n := \begin{cases} \mathcal{M}'_n, & \text{if } \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} (f_{n-1} - P_{C_i}(f_{n-1})) \neq 0, \\ 1, & \text{otherwise.} \end{cases} \quad (17.100b)$$

As the following example demonstrates, a couple of celebrated Signal Processing and Machine Learning tools stem from (17.100a).

**Example 68 (Kernel Affine Projection Algorithm (KAPA) [6,37,68,99,100] and Kernel Normalized Least Mean Squares (KNLMS) algorithm [5,6,37,102–104]).** Given the sequence of training data  $(y_n, \mathbf{x}_n)_{n \in \mathcal{N}_*}$ , and a user-defined positive integer parameter  $m$ , define the following linear mapping according to Example 39:

$$A_n(f) := \begin{bmatrix} \langle f, \kappa(\mathbf{x}_n, \cdot) \rangle \\ \vdots \\ \langle f, \kappa(\mathbf{x}_{n-m+1}, \cdot) \rangle \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_n) \\ \vdots \\ f(\mathbf{x}_{n-m+1}) \end{bmatrix}, \quad \forall f \in \mathcal{H}, \forall n \in \mathcal{N}_*.$$

Let also  $\mathbf{y}_n := [y_n, \dots, y_{n-m+1}]^T \in \mathcal{R}^m$ , and define the sequence of affine sets:

$$V_n := \operatorname{argmin}_{f \in \mathcal{H}} \| \mathbf{y}_n - A_n(f) \|^2, \quad n \in \mathcal{N}_*,$$

and the sequence of loss functions:

$$\mathcal{L}_n(f) := d(f, V_n), \quad \forall f \in \mathcal{H}, \forall n \in \mathcal{N}_*.$$

Note that this scenario is a special case of the APSM of Example 67, where  $q := 1$ , and  $C_n := V_n, \forall n \in \mathcal{N}_*$ .

Now, motivated by (17.77) and (17.82), and for a sequence of user-defined parameters  $\mu_n \in (0, 2)$ ,  $n \in \mathcal{N}_*$ , we form the following recursive process by means of a relaxed version of the subgradient projection mapping  $T_{\mathcal{L}_n}$  with respect to  $\mathcal{L}_n$ ; for an arbitrary initial point  $f_0 \in \mathcal{H}$ , and  $\forall n \in \mathcal{N}_*$ ,

$$\begin{aligned} f_n &:= (1 - \lambda_n) f_{n-1} + \mu_n T_{\mathcal{L}_n}(f_{n-1}) = (1 - \mu_n) f_{n-1} + \mu_n P_{V_n}(f_{n-1}) \\ &= f_{n-1} + \mu_n (P_{V_n}(f_{n-1}) - f_{n-1}) = f_{n-1} - \mu_n A_n^\dagger(A_n(f_{n-1}) - \mathbf{y}_n), \end{aligned} \quad (17.101)$$

where  $A_n^\dagger$  stands for the pseudoinverse mapping of  $A_n$ . Equation (17.101) is nothing but the Kernel APA [37,68].

In the case where the linear kernel is chosen for  $\mathcal{H}$ , then  $\mathcal{H}$  reduces to a Euclidean  $\mathcal{R}^l$ , and the linear mapping  $A_n$  becomes:

$$A_n(\boldsymbol{\theta}) := \begin{bmatrix} \mathbf{x}_n^T \boldsymbol{\theta} \\ \vdots \\ \mathbf{x}_{n-m+1}^T \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n^T \\ \vdots \\ \mathbf{x}_{n-m+1}^T \end{bmatrix} \boldsymbol{\theta} = \mathbf{A}_n \boldsymbol{\theta}, \quad \forall \boldsymbol{\theta} \in \mathcal{R}^l, \forall n \in \mathcal{N}_*,$$

where  $\mathbf{A}_n := [\mathbf{x}_n, \dots, \mathbf{x}_{n-m+1}]^T$ . Hence, (17.101) reduces to the classical APA [99,100]; for an arbitrarily fixed initial point  $\boldsymbol{\theta}_0$ , for  $\mu_n \in (0, 2)$ , and  $\forall n \in \mathcal{N}_*$ ,

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} - \mu_n \mathbf{A}_n^\dagger(\mathbf{A}_n \boldsymbol{\theta}_{n-1} - \mathbf{y}_n),$$

where  $\mathbf{A}_n^\dagger$  is the Moore-Penrose pseudoinverse of  $\mathbf{A}_n$ .

It is straightforward to verify also that in the case where  $m := 1$  in the KAPA, then the KNLMS algorithm (17.41) is obtained. Thus, KNLMS and its Euclidean version, i.e., the classical NLMS [5, 6, 102, 104], become also special cases of the APSM of Example 67.

It is worthy to notice here that the APA appears to be sensitive when operating in substantially noisy environments [101]. Moreover, it is prone to numerical instability issues due to the need for computation of the Moore-Penrose pseudoinverse in (17.101). However, this is not the case for the APSM. The study [101] showed that the APSM version of Example 67 is resilient to noise, and that no computation of a Moore-Penrose pseudoinverse is needed in order to achieve the same goal as the APA; indeed, given the sequence of training data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , define  $C_n := H_n, \forall n \in \mathcal{N}_*$ , where

$$H_n := \{f \in \mathcal{H} : \langle f, \kappa(\mathbf{x}_n, \cdot) \rangle = y_n\}, \quad \forall n \in \mathcal{N}_*,$$

and set  $q := m$  in Example 67 in order to obtain an algorithmic procedure with the same objective as the APA, but free from the computationally thirsty task of calculating the Moore-Penrose pseudoinverse of a matrix.

Let us tailor now the APSM of Example 67 to suit regression tasks. To this end, we will use a special class of closed convex sets, namely the (closed) hyperslabs of Example 38. Given the sequence of training data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , and a user-defined parameter  $\epsilon > 0$ , let the (closed) hyperslab ( $\forall n \in \mathcal{N}_*$ ):

$$\begin{aligned} S_n[\epsilon] &:= \{f \in \mathcal{H} : |f(\mathbf{x}_n) - y_n| \leq \epsilon\} \\ &= \{f \in \mathcal{H} : |\langle f, \kappa(\mathbf{x}_n, \cdot) \rangle - y_n| \leq \epsilon\}, \end{aligned}$$

be the set that gathers all those points of the RKHS  $\mathcal{H}$  which achieve a fitness to the observed value  $y_n$  of tolerance up to  $\epsilon$ , when evaluated onto  $\mathbf{x}_n$ . To illustrate the way that the APSM operates, let us apply the sequence of the previous hyperslabs  $(S_n[\epsilon])_{n \in \mathcal{N}_*}$  to Example 67, i.e., use the specific  $(S_n[\epsilon])_{n \in \mathcal{N}_*}$  in the place of the closed convex sets  $(C_n)_{n \in \mathcal{N}_*}$  in Example 67. The objective of the APSM, as well as of any other online learning algorithm, is the following; given the training data  $((\mathbf{x}_i, y_i))_{i=1}^n$ , compute an  $n$ -dimensional real-valued vector  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_n)^T$  in order to produce the estimate  $f_n \in \mathcal{H}$  as a linear combination of the elements of the dictionary  $\{\kappa(\mathbf{x}_i, \cdot)\}_{i=1}^n$ , i.e.,

$$f_n = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}_i, \cdot). \tag{17.102}$$

In general, the vector  $\boldsymbol{\theta}$  depends on the time index  $n$ . However, we have suppressed here this dependency for notational convenience. Each learning algorithm has its own way to calculate  $\boldsymbol{\theta}$ ; for example, the APSM of Example 67 follows the way illustrated in Algorithm 1.17.7.6.

The Algorithm 69 is nothing but a realization of Algorithm 65, where  $(S_n[\epsilon])_{n \in \mathcal{N}_*}$  take the place of  $(C_n)_{n \in \mathcal{N}_*}$ , and where calculations take place in the domain of coefficients  $\boldsymbol{\theta}$ , instead of the functional space  $\mathcal{H}$ . To be more rigorous, such a transfer of operations from  $\mathcal{H}$  to the domain of coefficients is due to the linear mapping:

$$\begin{aligned} \text{span}(\{\kappa(\mathbf{x}_i, \cdot)\}_{i=1}^n) &\rightarrow \mathbb{R}^n \\ f_n = \sum_{i=1}^n \theta_i \kappa(\mathbf{x}_i, \cdot) &\mapsto (\theta_1, \dots, \theta_n)^T = \boldsymbol{\theta}. \end{aligned}$$

---

**Algorithm 69.** The APSM of Example 67 with hyperslabs

---

**Require:** The training data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , an RKHS  $\mathcal{H}$ , and an integer  $q > 1$ .

- 1: Set  $\mathcal{D} := \emptyset$ .
  - 2: **for**  $n = 1, 2, \dots$  **do**
  - 3: Insert  $\mathbf{x}_n$  into  $\mathcal{D}$ , i.e., set  $\mathbf{u}_n := \mathbf{x}_n$ , and  $\mathcal{D} := \{\mathbf{u}_1, \dots, \mathbf{u}_{n-1}, \mathbf{u}_n\}$ .
  - 4: Increase the length of  $\boldsymbol{\theta}$  by one, i.e., let  $\boldsymbol{\theta} := (\theta_1, \dots, \theta_{n-1}, \theta_n)^T$ , where  $\theta_n := 0$ .
  - 5: Define  $\mathcal{J} := \{\max\{1, n - q + 1\}, \dots, n\}$ , of length  $q$ , at most.
  - 6: For every  $i \in \mathcal{J}$ , define
$$\beta_i := \begin{cases} \frac{\sum_{j=1}^{n-1} \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) - y_i - \epsilon}{\kappa(\mathbf{u}_j, \mathbf{u}_i)} & \text{if } \sum_{j=1}^{n-1} \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) > y_i + \epsilon, \\ 0, & \text{if } -\epsilon \leq \sum_{j=1}^{n-1} \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) < y_i - \epsilon, \\ \frac{\sum_{j=1}^{n-1} \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) - y_i - \epsilon}{\kappa(\mathbf{u}_j, \mathbf{u}_i)} & \text{if } \sum_{j=1}^{n-1} \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) < y_i - \epsilon. \end{cases}$$
  - 7: Define the index set  $\mathcal{I}_{f_{n-1}} := \{i \in \mathcal{J} : \beta_i \neq 0\}$ .
  - 8: Choose the convex weights  $\omega_i^{(n)} \in (0, 1]$ ,  $\forall i \in \mathcal{I}_{f_{n-1}}$ , such that  $\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} = 1$ .
  - 9: **if**  $\sum_{i,j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j) \neq 0$ , **then**
  - 10: Define  $\mathcal{M}_n := \frac{\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \beta_i^2 \kappa(\mathbf{u}_i, \mathbf{u}_i)}{\sum_{i,j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j)}$ .
  - 11: **else**
  - 12: Let  $\mathcal{M}_n := 1$ .
  - 13: **end if**
  - 14: Choose any  $\mu_n \in (0, 2\mathcal{M}_n)$ .
  - 15: For every  $i \in \mathcal{I}_{f_{n-1}}$ , define  $\theta_i := \theta_i - \mu_n \omega_i^{(n)} \beta_i$ .
  - 16: **end for**
- 

Let us give here a few more comments in order for the reader to grasp the connection between the Algorithms 12 and 13. Given the estimate  $f_{n-1} = \sum_{i=1}^{n-1} \theta_i \kappa(\mathbf{x}_i, \cdot)$ , Step 6 of Algorithm 69 is based on (17.74), and it contains the decisive term in order to calculate  $P_{S_i[\epsilon]}(f_{n-1})$  in Example 67. If the quantity  $\beta_i$  is non-zero, then it follows from (17.74) that  $f_{n-1} \notin S_i[\epsilon]$ ; this is hidden behind the Step 7 of Algorithm 69. Finally, notice that Step 15 refers to (17.100a).

The way to incorporate the hyperslab  $S_n[\epsilon]$  in Example 67 was via the non-negative distance function  $d(\cdot, S_n[\epsilon])$ , as (17.99) clearly suggests. The connection between  $S_n[\epsilon]$  and  $d(\cdot, S_n[\epsilon])$  has been already made clear by Def. 34; indeed,  $\text{lev}_{\leq 0}(d(\cdot, S_n[\epsilon])) = S_n[\epsilon]$ . However, by the freedom that the general form of the APSM in (17.98) infuses into the design, as well as by the way that the loss function is defined in Example 67, one would expect also other candidates of convex loss functions, which take the place of  $d(\cdot, S_n[\epsilon])$  in (17.99), but still maintain  $S_n[\epsilon]$  as their 0th level set, i.e.,  $\text{lev}_{\leq 0}(\cdot)$ . For example, the  $\epsilon$ -insensitive loss:

$$\ell : \mathcal{H} \rightarrow \mathcal{R} : f \mapsto \ell(f) := (|f(\mathbf{x}_n) - y_n| - \epsilon)_+,$$

where  $(\cdot)_+$  stands for the positive part of a real number (see Figure 17.33), is a non-negative convex function that achieves the previous objective. The previous loss function candidate was not a random choice; it has been used extensively in *Support Vector Regression (SVR)* [2, 7, 13], due to its remarkable robustness attributes. The previous discussion illustrates the flexibility of the APSM with respect to the choice of the loss function in (17.98) or in (17.99). Such a choice is often dictated by the nature of the learning task at hand; a sequence of halfspaces were utilized in [96], as a special case of the  $(C_n)_{n \in \mathcal{N}_*}$  in Example 67, in order to attack a classification problem. Moreover, another choice for the sequence of loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  in (17.98) led to a convex analytic alternative for the celebrated RLS in [105], where the need for inverting an autocorrelation matrix is bypassed.

### 1.17.7.6.1 Sparsification

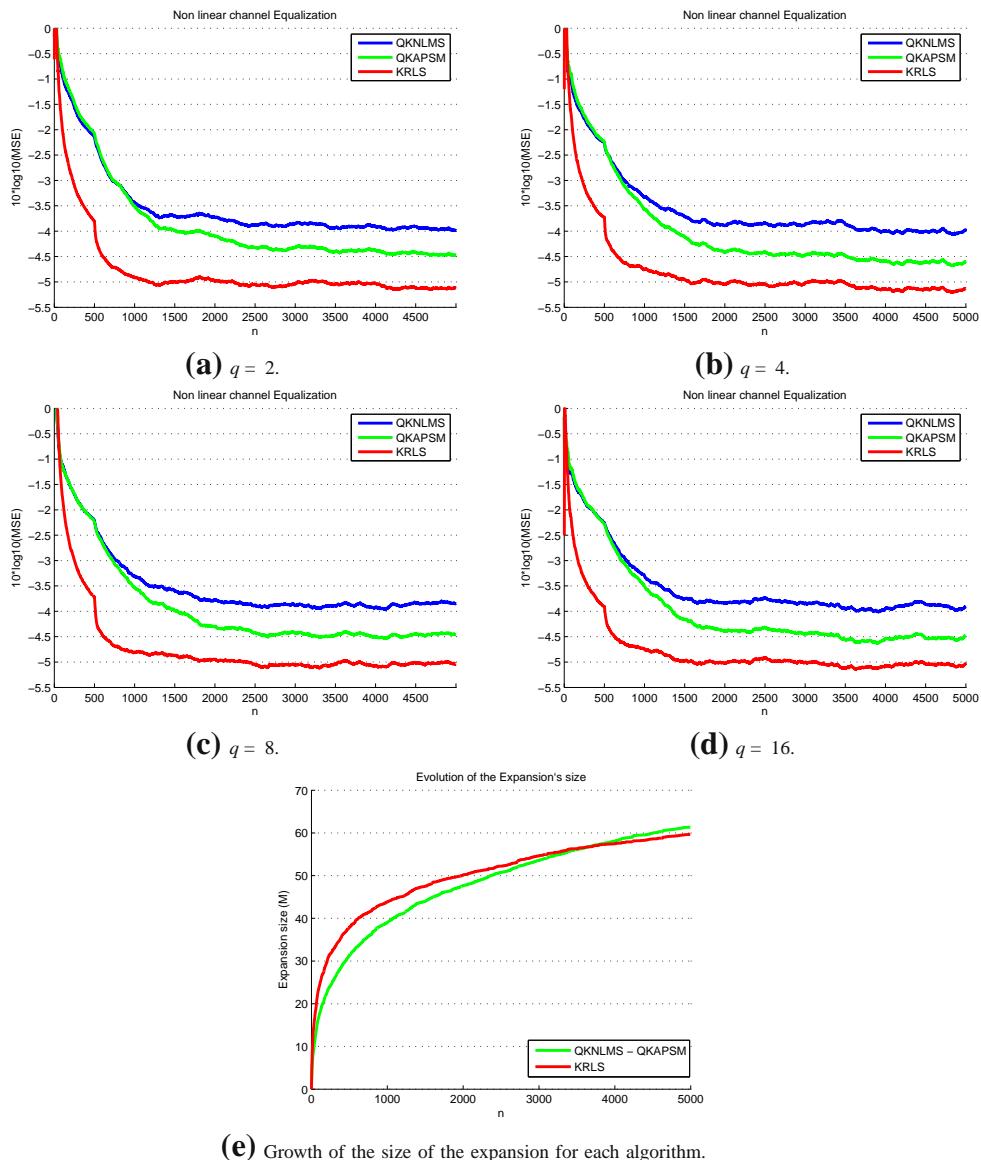
As we have seen in Section 1.17.6.2.1, any online learning algorithm, which operates in a potentially infinite dimensional RKHS, suffers from the large number of incoming training data  $(\mathbf{x}_n, y_n)_{n \in \mathcal{N}_*}$ , since such an abundance of data causes an explosion of the system's computational complexity and memory requirements, via kernel series expansions, like (17.39) and (17.102). This can be also observed in Algorithm 69. Hence, sparsification of the expansions in (17.39) or (17.102) is needed. To make these concepts as concrete as possible, let us elaborate more on the realization of the APSM given in Algorithm 69. In order to enforce sparsification into the algorithm, and motivated by the discussion related to (17.11), the study in [96] imposed an additional constraint to the sequence of estimates  $(f_n)_{n \in \mathcal{N}}$ ; the following closed ball, with center 0 and radius  $\Delta > 0$  (see also Example 36):

$$B[0, \Delta] := \{f \in \mathcal{H} : \|f\| \leq \Delta\},$$

was used in the place of the closed convex set  $C$  in (17.98). It turns out that such a sparsification scheme is equivalent to a forgetting factor mechanism that forgets data in the remote past; see [96] for a more detailed presentation. Since such a forgetting mechanism renders the contribution of “old” training data weak in kernel series expansions like (17.102), it is natural to adopt a buffer of length  $M$  where only the  $M$  more recent training data are kept. For the sake of illustration, this sparsification strategy is summarized by the pseudocode formulation of Algorithm 70. For more advances on the capability of the APSM to incorporate constraints in its formulation, the interested reader is referred to [42, 92, 93].

The main difference between the Algorithms 13 and 14 lies on the previous closed-ball sparsification strategy, which is realized in the latter algorithm. More specifically, this strategy can be seen from Step 15 to Step 29 of Algorithm 70. In Step 15,  $v$  stands for the square of the norm of the term  $f_{n-1} + \mu_n \left( \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} P_{C_i}(f_{n-1}) - f_{n-1} \right)$ , met in (17.100a). The lines from Step 17 to Step 22 refer to the application of the metric projection mapping  $P_{B[0, \Delta]}$ , i.e., Example 36, onto the closed ball constraint  $B[0, \Delta]$ , according to the scheme of the general APSM, depicted in (17.98). Finally, the lines starting from Step 23 to Step 29, ensure that the size of the dictionary stays at most  $M$ , by “throwing away” the “oldest” contribution in  $\mathcal{D}$ . As it was explained in [96], such a brute-force method is justified by the effect that the mapping  $P_{B[0, \Delta]}$  has on the coefficients of  $\theta$ ; it has the tendency to trim down the “oldest” ones.

In principle, any other sparsification scheme, as the ones discussed in Section 1.17.6.2.1, can be used instead of the closed-ball one, seen in Algorithm 70. For example, instead of “throwing away” the “oldest” coefficient of  $\theta$  in Algorithm 70, a more “gentle” approach would be to discard that coefficient

**FIGURE 17.32**

Learning curves for the QKAPSM of Example 67 employing the Quantization sparsification procedure (section 1.17.6.2.1), QKNLMS, and KRLS, and for the same nonlinear equalization task as in Sections 1.17.6.2.2 and 1.17.6.6.1, for various values of the parameter  $q$ . Note that the expansions of QKNLMS and QKAPSM are identical in terms of the training point, as shown in (d). The parameters of each expansion, though, are different.

of  $\theta$  with the least contribution in the kernel series expansion of (17.102). All of these sparsification strategies which rely on the closed ball constraint exhibit a linear computational complexity with respect to the number of elements of  $\mathcal{D}$  [96]. The APSM, combined with a more involved sparsification strategy along the lines of the ALD strategy of the KRLS, can be found in [68]. Finally, inspired by the sparsification scheme of the QKLMS (see Algorithm 27.6.3), a quantized alternative to Algorithm 70 is given by the QKAPSM in Algorithm 71. Here, the size of the dictionary  $\mathcal{D}$  is determined by the quantization parameter  $\delta$ . Although this sparsification strategy shows a computational complexity lower than the one of Algorithm 70, it demonstrates a remarkable performance, as this can be seen by the numerical results of Section 1.17.7.6.2.

---

**Algorithm 70.** The APSM for Example 67 with hyperslabs and the closed-ball sparsification scheme

---

**Require:** The training data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , an RKHS  $\mathcal{H}$ , an integer  $q > 1$ , a radius  $\Delta > 0$ , and an upper bound for the size of the dictionary  $M' > q$ .

- 1: Set  $\mathcal{D} := \emptyset$ ,  $M' := 0$ , and  $\|f_0\| = 0$ .
- 2: **for**  $n = 1, 2, \dots$  **do**
- 3:   Insert  $\mathbf{x}_n$  into  $\mathcal{D}$ , i.e., set  $\mathbf{u}_{M'+1} := \mathbf{x}_n$ , and  $\mathcal{D} := \{\mathbf{u}_1, \dots, \mathbf{u}_{M'}, \mathbf{u}_{M'+1}\}$ .
- 4:   Increase the length of  $\theta$  by one, i.e., let  $\theta := (\theta_1, \dots, \theta_{M'}, \theta_{M'+1})^T$ , where  $\theta_{M'+1} := 0$ .
- 5:   Define  $\mathcal{J} := \{\max\{1, M' - q + 2\}, \dots, M' + 1\}$ , of length  $q$ , at most.
- 6:   For every  $i \in \mathcal{J}$ , define
- 7:     Define the index set  $\mathcal{I}_{f_{n-1}} := \{i \in \mathcal{J} : \beta_i \neq 0\}$ .
- 8:     Choose the convex weights  $\omega_i^{(n)} \in (0, 1]$ ,  $\forall i \in \mathcal{I}_{f_{n-1}}$ , such that  $\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} = 1$ .
- 9:     **if**  $\sum_{i,j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j) \neq 0$ , **then**
- 10:       Define  $\mathcal{M}_n := \frac{\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \beta_i^2 \kappa(\mathbf{u}_i, \mathbf{u}_i)}{\sum_{i,j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j)}$ .
- 11:     **else**
- 12:       Let  $\mathcal{M}_n := 1$ .
- 13:     **end if**
- 14:     Choose any  $\mu_n \in (0, 2\mathcal{M}_n)$ .
- 15:     Define

$$\begin{aligned} v &:= \|f_{n-1}\|^2 + \mu_n^2 \sum_{i,j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j) \\ &\quad - 2\mu_n \sum_{j=1}^{M'} \sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \beta_i \theta_j \kappa(\mathbf{u}_i, \mathbf{u}_j). \end{aligned}$$


---

**Algorithm 70.** (Continued)

---

```

16: For every  $i \in \mathcal{I}_{f_{n-1}}$ , define  $\theta_i := \theta_i - \mu_n \omega_i^{(n)} \beta_i$ .
17: if  $\sqrt{\nu} > \Delta$ , then
18:   For every  $i \in \{1, \dots, M' + 1\}$ , let  $\theta_i := \frac{\Delta}{\sqrt{\nu}} \theta_i$ .
19:   Define  $\|f_n\| := \Delta$ .
20: else
21:   Let  $\|f_n\| := \nu$ .
22: end if
23: if  $M' \geq M$ , then
24:   Let  $\|f_n\|^2 := \|f_n\|^2 + \theta_1^2 \kappa(\mathbf{u}_1, \mathbf{u}_1) - 2\theta_1 \sum_{i=1}^{M'+1} \theta_i \kappa(\mathbf{u}_1, \mathbf{u}_i)$ .
25:   Let  $\mathcal{D} := \mathcal{D} \setminus \{\mathbf{u}_1\}$ , and  $\boldsymbol{\theta} := (\theta_2, \dots, \theta_{M'+1})^T$ .
26:   For every  $i \in \{1, \dots, M'\}$ , let  $\mathbf{u}_i := \mathbf{u}_{i+1}$ , and  $\theta_i := \theta_{i+1}$ .
27: else
28:    $M' := M' + 1$ .
29: end if
30: end for

```

---

**1.17.7.6.2 Simulation results**

In order to validate the APSM of Example 67, we consider the same nonlinear channel equalization scenario to the one used in Sections 1.17.6.2.2 and 1.17.6.6.1. More specifically, the nonlinear channel consists of a linear filter  $t_n = -0.8 \cdot y_n + 0.7 \cdot y_{n-1} - 0.6 \cdot y_{n-2} + 0.4 \cdot y_{n-3}$ , and a memoryless nonlinearity  $q_n = t_n + 0.08 \cdot t_n^2$ ,  $\forall n \in \mathcal{N}_*$ . The parameters for the QKNLMS and the KRLS are identical to the ones used in Sections 1.17.6.2.2 and 1.17.6.6.1. As for the APSM, its QKAPSM version, i.e., Algorithm 71, was employed. For the present scenario, the following parameters were used for the APSM;  $q := 2, 4, 8, 16$ ,  $\mu_n = 0.5$ ,  $\epsilon := 10^{-10}$ ,  $\forall n \in \mathcal{N}_*$ , while the parameter for the sparsification procedure was set to  $\delta = 6$ , for both APSM and QKNLMS (See fig. 17.32).

Notice that the APSM achieves superior performance, when compared to the QKNLMS, while both achieve the same expansion size. This is a natural consequence of the ability of the APSM to activate data reuse or concurrent processing, by letting the parameter  $q$  to take values larger than 1. Recall, by Example 68, that the KNLMS is a byproduct of the APSM for  $q = 1$ . In general, the more the  $q$  is increased, the faster the convergence of the KAPSM becomes. The flexibility offered to the APSM by the theory of convex analysis pays off, also, in the case where the APSM is compared to the classical APA, or KAPA (see Example 68). Even if both APSM and APA employ data reuse, the way the APSM is derived (Example 67) precludes the system from the calculation of a matrix inversion, as is the case for the APA where the computation of a pseudoinverse is necessary. This becomes beneficial in cases where algorithmic stability is the issue. Notice also that the misadjustment level of the APSM is inferior

to the KRLS. Recall, here, that the computational complexity of the KRLS is  $O(M_n^2)$ , as opposed to the  $O(qM_n)$ , where  $M_n$  is the size of the dictionary at the time instant  $n$ . Recall, also, that the computational complexity of the QKNLMS is  $O(M_n)$ .

More on the validation of the APSM, for several learning tasks, can be found in [41–43, 95, 97, 98, 106, 107, 112]. The code for the experiments presented in this section can be found in <<http://bouboulis.mysch.gr/kernels.html>>.

---

**Algorithm 71.** Quantized Kernel APSM (QKAPSM) for Example 67 with hyperslabs

---

**Require:** The training data  $((\mathbf{x}_n, y_n))_{n \in \mathcal{N}_*}$ , an RKHS  $\mathcal{H}$ , an integer  $q > 1$ , and the quantization parameter  $\delta > 0$ .

- 1: Set  $\mathcal{D} := \emptyset$ , and  $M := 0$ .
- 2: **for**  $n = 1, 2, \dots$ , **do**
- 3:   Compute the distance of  $\mathbf{x}_n$  from  $\mathcal{D}$ :  

$$d(\mathbf{x}_n, \mathcal{D}) = \inf_{\mathbf{u}_i \in \mathcal{D}} \|\mathbf{x}_n - \mathbf{u}_i\| = \|\mathbf{x}_n - \mathbf{u}_{l_0}\|, \text{ for some } l_0 \in \{1, \dots, M\}.$$

In the case where there are multiple such  $l_0$ s, then choose the largest one.  
To leave no place for ambiguities, let also  $d(\mathbf{x}_n, \emptyset) := +\infty$ .
- 4:   **if**  $d(\mathbf{x}_n, \mathcal{D}) > \delta$ , **then**
- 5:     Insert  $\mathbf{x}_n$  into  $\mathcal{D}$ , i.e., set  $\mathbf{u}_{M+1} = \mathbf{x}_n$ , and  $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_M, \mathbf{u}_{M+1}\}$ .
- 6:     Increase the length of  $\boldsymbol{\theta}$  by one, i.e., let  $\boldsymbol{\theta} := (\theta_1, \dots, \theta_M, \theta_{M+1})^T$ , where  $\theta_{M+1} := 0$ .
- 7:     Define  $\mathcal{J} := \{\max\{1, M - q + 2\}, \dots, M + 1\}$ .
- 8:   **else**
- 9:     **if**  $l_0 \geq \max\{1, M - q + 1\}$ , **then**
- 10:        $\mathcal{J} := \{\max\{1, M - q + 1\}, \dots, M\}$ .
- 11:     **else**
- 12:        $\mathcal{J} := \{l_0, M - q + 2, \dots, M\}$ .
- 13:     **end if**
- 14:   **end if**
- 15:   For every  $i \in \mathcal{J}$ , define  

$$\beta_i := \begin{cases} \frac{\sum_{j=1}^M \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) - y_i - \epsilon}{\kappa(\mathbf{u}_i, \mathbf{u}_i)}, & \text{if } \sum_{j=1}^M \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) > y_i + \epsilon, \\ 0, & \text{if } -\epsilon \leq \sum_{j=1}^M \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) - y_i \leq \epsilon, \\ \frac{\sum_{j=1}^M \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) - y_i + \epsilon}{\kappa(\mathbf{u}_i, \mathbf{u}_i)}, & \text{if } \sum_{j=1}^M \theta_j \kappa(\mathbf{u}_j, \mathbf{u}_i) < y_i - \epsilon. \end{cases}$$
- 16:   Define the index set  $\mathcal{I}_{f_{n-1}} := \{i \in \mathcal{J} : \beta_i \neq 0\}$ .
- 17:   Choose the convex weights  $\omega_i^{(n)} \in (0, 1], \forall i \in \mathcal{I}_{f_{n-1}}$ , such that  $\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} = 1$ .
- 18:   **if**  $\sum_{i, j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j) \neq 0$ , **then**
- 19:     Define  $\mathcal{M}_n := \frac{\sum_{i \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \beta_i^2 \kappa(\mathbf{u}_i, \mathbf{u}_i)}{\sum_{i, j \in \mathcal{I}_{f_{n-1}}} \omega_i^{(n)} \omega_j^{(n)} \beta_i \beta_j \kappa(\mathbf{u}_i, \mathbf{u}_j)}$ .
- 20:   **else**

---

**Algorithm 71.** (Continued)

---

```

21: Let  $\mathcal{M}_n := 1$ .
22: end if
23: Choose any  $\mu_n \in (0, 2\mathcal{M}_n)$ .
24: For every  $i \in \mathcal{I}_{f_{n-1}}$ , define  $\theta_i := \theta_i - \mu_n \omega_i^{(n)} \beta_i$ .
25: if  $d(\mathbf{x}_n, \mathcal{D}) > \delta$ , then
26:    $M := M + 1$ .
27: end if
```

**Output:** The dictionary  $\mathcal{D} = \{\mathbf{u}_1, \dots, \mathbf{u}_M\}$ , and the real-valued vector  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_M)^T$ , which produce the estimate  $f_n = \sum_{i=1}^M \theta_i \kappa(\mathbf{u}_i, \cdot)$ .

```
28: end for
```

---

## 1.17.8 Related work and applications

The recursions (17.86), (17.90), (17.94), and (17.96), met in the previous section, are the basis of a plethora of algorithmic solutions for signal processing [5, 6, 27, 41–43, 45, 68, 94, 95, 97, 98, 106–113] and machine learning [57–61, 77, 78, 114–128], and, thus, they are the engine behind numerous applications which span from echo cancellation, equalization of linear/nonlinear channels, image processing, and wireless sensor networks, to pattern recognition and classification, social network analysis, bioinformatics, and sparsity-aware learning. This section aims to shedding light on the connection of the convex analytic framework of Section 1.17.7 with several recent works on online learning. We focus mainly on machine learning techniques, since their connection with the framework of Section 1.17.7 is not as much illuminated as is the signal processing side of learning [95, 129, 130].

The recursion (17.96) is the basic ingredient hidden behind the NORMA method of [122]. In [122], the sequence of closed convex sets  $(C_n)_{n \in \mathcal{N}_*}$  is not imposed into the design, i.e., the unconstrained case, and the loss function  $\mathcal{L}_n := \Theta_n + \lambda \|\cdot\|^2$ ,  $\forall n \in \mathcal{N}_*$ , where  $\lambda > 0$  is a regularization parameter. This form of loss functions originates from the very successful Support Vector Machines (SVM) rationale. By assuming Lipschitz continuity for all the loss functions  $(\Theta_n)_{n \in \mathcal{N}_*}$ , a convergence analysis is conducted in [122] in order to derive error bounds whenever the learning step sizes  $(\mu_n)_{n \in \mathcal{N}_*}$  follow the diminishing rule of  $\mu_n \propto \frac{1}{\sqrt{n}}$ ,  $\forall n \in \mathcal{N}_*$ . The SVM rationale and the recursion (17.96) is also hidden behind [126]. The Lipschitz continuity assumption of [122] on  $(\Theta_n)_{n \in \mathcal{N}_*}$  is bypassed, the sequence  $(C_n)_{n \in \mathcal{N}_*}$  is fixed to a single closed convex set, e.g., a closed ball, and the convergence analysis for the error bounds is conducted on the assumption of  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  being a sequence of strongly convex functions (see Definition 33), with  $\mu_n \propto \frac{1}{n}$ ,  $\forall n \in \mathcal{N}_*$ . Notice, here, that the rules for the learning step parameters  $(\mu_n)_{n \in \mathcal{N}_*}$ , adopted in both [122, 126], are very closely related to the discussion on (17.92), and the classical method of [84]. A study on the stability of the basic recursion (17.96), under a stochastic framework, in the case where the loss functions  $(\mathcal{L}_n)_{n \in \mathcal{N}_*}$  are differentiable, is given in the very recent work of [125]. Moreover, by

following the rationale of proximal mappings (see Example 47), as well as the strategies of the forward-backward and the Douglas-Rachford algorithms [65, 72, 73, 75, 76] (see the discussion below (17.85)), generalizations of (17.96) in the online learning context can be found in [77, 78, 110, 113, 124].

The classical LMS recursion (17.34) is studied through the machine learning magnifying glass in [118]. Inner products, in properly defined inner product spaces, are used in order to model estimates. The gradient descent algorithm of [118] is utilized to derive several worst-case bounds of the total sum of the squared prediction errors. A normalized version of the gradient descent algorithm, which is nothing but the direct analogue of the NLMS method (17.37) to general inner product spaces, appears and is analyzed also in [118]. The interaction of the gradient descent rationale with noisy environments was very recently studied in [117], where, in particular, several theoretical results were stated regarding the question of to what extent independently perturbed copies of each training datum assist the gradient descent algorithm in its online learning task.

An algorithm of the same spirit to the KNLMS (see Section 1.17.6.2) is studied in [119]. The kernel series expansion estimate is updated every time instant by projecting the current estimate onto a single halfspace, whenever a classification problem is considered, or a hyperslab (see Example 38), whenever a regression task is studied. These solution sets are defined by the newly arrived training data as the set of all minimizers of properly defined *hinge* loss functions. Variants of this scheme are also studied in [119] by means of a parameter which controls how much aggressive the projection onto the halfspace should be. A generalization of [119], towards using multiple halfspaces at each time instant, similarly to the strategy of Examples 52, 67, and 68, is given in [114]. However, the study of [114] follows a different approach than the one in Examples 52, 67, and 68, by transferring the original problem to the dual domain, by approximating it via a multiple number of simpler problems, and by elaborating inner iterations, at each step of the algorithm, in order to calculate various optimal step parameters.

---

## 1.17.9 Conclusions

This manuscript presented an overview of online learning techniques; both classical and modern. To serve the needs of both the signal processing and machine learning communities, this work has focused on bringing forth the basic principles of a fundamental machine learning tool, the Reproducing Kernel Hilbert Spaces, and on highlighting a theoretical device, mostly known to the signal processing community, i.e., the elegant and geometrically rich convex analytic side of operators/mappings in general Hilbert spaces. Despite its tutorial nature, the manuscript does not spare mathematical rigor; proofs are given in detail in places where we feel that the way to use the previous mathematical tools should be demonstrated. We have also tried to provide with a framework that unites classical and modern stuff; numerous classical online learning techniques stem as special cases of the introduced mathematical framework. Numerical examples have been also given in order to exemplify the common attributes which run along the spine of online learning tasks in both signal processing and machine learning principles, i.e., the nonlinear estimation theory and the classification/regression framework.

## Appendices

### A Key properties of the gaussian kernel

As the Gaussian kernel is the most widely used in applications, we dedicate this section to present some of its most important properties. We begin our study showing that the Gaussian radial basis function is indeed a reproducing kernel. To this end, we introduce some new notions.

**Definition A.1 (Negative Definite Kernel).** Let  $X$  be a set. A function  $\kappa : X \times X \rightarrow \mathcal{R}$  is called a negative definite kernel if it is symmetric, i.e.,  $\kappa(\mathbf{y}, \mathbf{x}) = \kappa(\mathbf{x}, \mathbf{y})$ , and

$$\sum_{n,m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \leq 0,$$

for any  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$ ,  $c_1, \dots, c_N \in \mathcal{R}$ , with  $\sum_{n=1}^N c_n = 0$ , and for all  $N \in \mathcal{N}_*$ .  $\diamond$

Examples of negative kernels are the constant functions and all functions of the form  $-\kappa$ , where  $\kappa$  is a positive definite kernel. Furthermore, the following proposition holds:

**Proposition A.2.** Let  $X$  be a non empty set, the functions  $\psi_k : X \times X \rightarrow \mathcal{R}$  be negative kernels and  $\alpha_k > 0$ , for  $k \in \mathcal{N}_*$ . Then

- Any positive combination of a finite number of negative kernels is also a negative kernel, i.e.,  $\psi = \sum_k \alpha_k \psi_k$ , with  $\alpha_1, \dots, \alpha_n > 0$  is a negative kernel.
- The limit of any converging sequence of negative kernels is also a negative kernel, i.e. if  $\psi(\mathbf{x}, \mathbf{y}) = \lim_k \psi_k(\mathbf{x}, \mathbf{y})$ , for all  $\mathbf{x}, \mathbf{y} \in X$ , then  $\psi$  is a negative kernel.

$\diamond$

**Proof.** For the first part, consider the numbers  $c_1, \dots, c_N$  such that  $\sum_{n=1}^N c_n = 0$ ,  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$  and  $M \in \mathcal{N}_*$ . Then

$$\sum_{n,m=1}^N c_n c_m \sum_{k=1}^M \alpha_k \psi_k(\mathbf{x}_n, \mathbf{x}_m) = \sum_{k=1}^M \alpha_k \sum_{n,m=1}^N c_n c_m \psi_k(\mathbf{x}_n, \mathbf{x}_m) \leq 0.$$

Finally, to prove the second part we take:

$$\sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) = \sum_{n,m=1}^N c_n c_m \lim_k \psi_k(\mathbf{x}_n, \mathbf{x}_m) = \lim_k \sum_{n,m=1}^N c_n c_m \psi_k(\mathbf{x}_n, \mathbf{x}_m) \leq 0.$$

$\square$

**Lemma A.3.** Let  $X$  be a nonempty set,  $V$  be a vector space equipped with an inner product and  $T : X \rightarrow V$ . Then the function

$$\psi(\mathbf{x}, \mathbf{y}) = \|T(\mathbf{x}) - T(\mathbf{y})\|_V^2,$$

is a negative definite kernel on  $X$ .  $\diamond$

**Proof.** Consider the numbers  $c_1, \dots, c_N$  such that  $\sum_{n=1}^N c_n = 0$  and  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$ . Then

$$\begin{aligned}
 \sum_{n,m=1}^N c_n c_m \|T(\mathbf{x}_n) - T(\mathbf{x}_m)\|_V^2 &= \sum_{n,m=1}^N c_n c_m \langle T(\mathbf{x}_n) - T(\mathbf{x}_m), T(\mathbf{x}_n) - T(\mathbf{x}_m) \rangle_V \\
 &= \sum_{n,m=1}^N c_n c_m \left( \|T(\mathbf{x}_n)\|_V^2 + \|T(\mathbf{x}_m)\|_V^2 - \langle T(\mathbf{x}_n), T(\mathbf{x}_m) \rangle_V \right. \\
 &\quad \left. - \langle T(\mathbf{x}_m), T(\mathbf{x}_n) \rangle_V \right) \\
 &= \sum_{m=1}^N c_m \sum_{n=1}^N c_n \|T(\mathbf{x}_n)\|_V^2 + \sum_{n=1}^N c_n \sum_{m=1}^N c_m \|T(\mathbf{x}_m)\|_V^2 \\
 &\quad - \left\langle \sum_{n=1}^N c_n T(\mathbf{x}_n), \sum_{m=1}^N c_m T(\mathbf{x}_m) \right\rangle_V \\
 &\quad - \left\langle \sum_{m=1}^N c_m T(\mathbf{x}_m), \sum_{n=1}^N c_n T(\mathbf{x}_n) \right\rangle_V.
 \end{aligned}$$

As  $\sum_{n=1}^N c_n = 0$ , the first two terms of the summation vanish and we take:

$$\sum_{n,m=1}^N c_n c_m \|T(\mathbf{x}_n) - T(\mathbf{x}_m)\|_V^2 = -2 \left\| \sum_{n=1}^N c_n T(\mathbf{x}_n) \right\|_V^2 \leq 0.$$

Thus  $\psi$  is a negative definite kernel.  $\square$

**Lemma A.4.** Let  $\psi : X \times X \rightarrow \mathcal{R}$  be a function. Fix  $\mathbf{x}_0 \in X$  and define

$$\kappa(\mathbf{x}, \mathbf{y}) = -\psi(\mathbf{x}, \mathbf{y}) + \psi(\mathbf{x}, \mathbf{x}_0) + \psi(\mathbf{x}_0, \mathbf{y}) - \psi(\mathbf{x}_0, \mathbf{x}_0).$$

Then  $\psi$  is a negative definite kernel if and only if  $\kappa$  is a positive definite kernel.  $\diamond$

**Proof.** Let  $\mathbf{x}_1, \dots, \mathbf{x}_N \in X$ . For the if part, consider the numbers  $c_1, \dots, c_N$  such that  $\sum_{n=1}^N c_n = 0$ . Then

$$\begin{aligned}
 \sum_{n,m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{x}_m) &= - \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_0) \\
 &\quad + \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_0, \mathbf{x}_m) - \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_0, \mathbf{x}_0) \\
 &= - \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^N c_m \sum_{n=1}^N c_n \psi(\mathbf{x}_n, \mathbf{x}_0) \\
 &\quad + \sum_{n=1}^N c_n \sum_{m=1}^N c_m \psi(\mathbf{x}_0, \mathbf{x}_m) - \sum_{m=1}^N c_m \sum_{n=1}^N c_n \psi(\mathbf{x}_0, \mathbf{x}_0).
 \end{aligned}$$

As  $\sum_{n=1}^N c_n = 0$  and  $\sum_{n,m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{y}_m) \geq 0$ , we take that  $\sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{y}_m) \leq 0$ . Thus  $\psi$  is a negative definite kernel.

For the converse, take  $c_1, \dots, c_N \in \mathcal{R}$  and define  $c_0 = -\sum_{n=1}^N c_n$ . By this simple trick, we generate the numbers  $c_0, c_1, \dots, c_N \in \mathcal{R}$ , which have the property  $\sum_{n=0}^N c_n = c_0 + \sum_{n=1}^N c_n = 0$ . As  $\psi$  is a negative definite kernel, we take that  $\sum_{n,m=0}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) \leq 0$ , for any  $\mathbf{x}_0 \in X$ . Thus,

$$\begin{aligned} \sum_{n,m=0}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) &= \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) + \sum_{m=1}^N c_0 c_m \psi(\mathbf{x}_0, \mathbf{x}_m) \\ &\quad + \sum_{n=1}^N c_0 c_n \psi(\mathbf{x}_n, \mathbf{x}_0) + c_0^2 \psi(\mathbf{x}_0, \mathbf{x}_0) \\ &= \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_m) - \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_0, \mathbf{x}_m) \\ &\quad - \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_n, \mathbf{x}_0) + \sum_{n,m=1}^N c_n c_m \psi(\mathbf{x}_0, \mathbf{x}_0) \\ &= \sum_{n,m=1}^N c_n c_m (\psi(\mathbf{x}_n, \mathbf{x}_m) - \psi(\mathbf{x}_0, \mathbf{x}_m) - \psi(\mathbf{x}_n, \mathbf{x}_0) + \psi(\mathbf{x}_0, \mathbf{x}_0)) \\ &= - \sum_{n,m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{x}_m). \end{aligned}$$

Thus  $\sum_{n,m=1}^N c_n c_m \kappa(\mathbf{x}_n, \mathbf{x}_m) \geq 0$  and  $\kappa$  is a positive definite kernel.  $\square$

**Theorem A.5 (Schoenberg).** *Let  $X$  be a nonempty set and  $\psi : X \times X \rightarrow \mathcal{R}$ . The function  $\psi$  is a negative kernel if and only if  $\exp(-t\psi)$  is a positive definite kernel for all  $t \geq 0$ .*  $\diamond$

**Proof.** For the if part, recall that

$$\psi(\mathbf{x}, \mathbf{y}) = \lim_{t \rightarrow 0} \frac{1 - \exp(-t\psi(\mathbf{x}, \mathbf{y}))}{t}.$$

As  $\exp(-t\psi)$  is positive definite,  $-\exp(-t\psi)$  is negative definite and the result follows from Proposition A.2. It suffices to prove the converse for  $t = 1$ , as if  $\psi$  is a negative definite kernel so is  $t\psi$ , for any  $t \geq 0$ . Take  $\mathbf{x}_0 \in X$  and define the positive definite kernel  $\kappa(\mathbf{x}, \mathbf{y}) = -\psi(\mathbf{x}, \mathbf{y}) + \psi(\mathbf{x}, \mathbf{x}_0) + \psi(\mathbf{x}_0, \mathbf{y}) - \psi(\mathbf{x}_0, \mathbf{x}_0)$  (Lemma A.4). Then

$$e^{-\psi(\mathbf{x}, \mathbf{y})} = e^{-\psi(\mathbf{x}, \mathbf{x}_0)} e^{\kappa(\mathbf{x}, \mathbf{y})} e^{-\psi(\mathbf{x}_0, \mathbf{y})} e^{\kappa(\mathbf{x}_0, \mathbf{x}_0)}.$$

Let  $f(\mathbf{x}) = e^{-\psi(\mathbf{x}, \mathbf{x}_0)}$ . Then, as  $\psi$  is a negative kernel and therefore symmetric, one can readily prove that the last relation can be rewritten as

$$e^{-\psi(\mathbf{x}, \mathbf{y})} = e^{\kappa(\mathbf{x}_0, \mathbf{x}_0)} \cdot f(\mathbf{x}) e^{\kappa(\mathbf{x}, \mathbf{y})} f(\mathbf{y}).$$

Since  $e^{\kappa(x_0, x_0)}$  is a positive number, employing the properties of positive kernels given in Section 1.17.5.5, we conclude that  $e^{-\psi(x, y)}$  is a positive definite kernel.  $\square$

**Corollary A.6.** *The Gaussian radial basis function is a reproducing kernel.*

Although all properties of positive kernels do not apply to negative kernels as well (for example the product of negative kernels is not a negative kernel), there are some other operations that preserve negativity.

**Proposition A.7.** *Let  $\psi : X \times X \rightarrow \mathcal{R}$  be negative definite. In this case:*

1. *If  $\psi(x, x) \geq 0$ , for all  $x \in X$ , then  $\psi^p(x, y)$  is negative definite for any  $0 < p \leq 1$ .*
2. *If  $\psi(x, x) \geq 0$ , for all  $x \in X$ , then  $\log(1 + \psi(x, y))$  is negative definite.*
3. *If  $\psi : X \times X \rightarrow (0, +\infty)$ , then  $\log \psi(y, x)$  is negative definite.*  $\diamond$

**Proof.** We give a brief description of the proofs.

1. We use the formula:

$$\psi(x, y)^p = \frac{p}{\Gamma(1-p)} \int_0^\infty t^{-p-1} (1 - e^{-t\psi(x, y)}) dt,$$

where the Gamma function is given by  $\Gamma(z) = \int_0^\infty e^{-t} t^z dt$ . As  $e^{-t\psi(x, y)}$  is positive definite (Theorem A.5) and  $t^{-p-1}$  is a positive number, it is not difficult to prove that the expression inside the integral is negative definite for all  $t > 0$ .

2. Similarly, we use the formula:

$$\log(1 + \psi(x, y)) = \int_0^\infty \frac{e^{-t}}{t} (1 - e^{-t\psi(x, y)}) dt.$$

3. For any  $c > 0$ ,  $\log(\psi(x, y) + 1/c) = \log(1 + c\psi(x, y)) - \log(c)$ . We can prove that the second part is negative definite. Then, by taking the limit as  $c \rightarrow 0$ , one completes the proof.  $\square$

As a direct consequence, one can prove that since  $\|x - y\|^2$  is a negative kernel, so is  $\|x - y\|^{2p}$ , for any  $0 < p \leq 1$ . Thus, for any  $0 < p \leq 2$ ,  $\|x - y\|^p$  is a negative kernel and  $\exp(-t\|x - y\|^p)$  is a positive kernel for any any  $t > 0$ . Therefore, for  $p = 2$  we take another proof of the positivity of the Gaussian radial basis function. In addition, for  $p = 1$  one concludes that the Laplacian radial basis function is also a positive kernel. Moreover, for the Gaussian kernel the following important property has been proved.

**Theorem A.8 (Full rank of the Gaussian RBF Gram matrices).** *Suppose that  $x_1, \dots, x_N \subset X$  are distinct points and  $\sigma \neq 0$ . The Gram matrix given by  $K = (K_{n,m})$ , where*

$$K_{n,m} = \exp\left(-\frac{\|x_n - x_m\|^2}{2\sigma^2}\right),$$

*has full rank.*  $\diamond$

As a consequence, for any choice of discrete points  $\mathbf{x}_1, \dots, \mathbf{x}_N$ , we have that  $\sum_{m=1}^N a_m \kappa(\mathbf{x}_n, \mathbf{x}_m) = 0$ , for all  $n = 1, 2, \dots, N$ , if and only if  $a_1 = \dots = a_N = 0$ . However, observe that for any  $a_1, \dots, a_N$

$$\sum_{m=1}^N a_m \kappa(\mathbf{x}_n, \mathbf{x}_m) = \sum_{m=1}^N a_m \langle \kappa(\cdot, \mathbf{x}_m), \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = \left\langle \sum_{m=1}^N a_m \kappa(\cdot, \mathbf{x}_m), \kappa(\cdot, \mathbf{x}_n) \right\rangle_{\mathcal{H}} = \langle f, \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}},$$

where  $f = \sum_{m=1}^N a_m \kappa(\cdot, \mathbf{x}_m) \in \text{span}\{\kappa(\cdot, \mathbf{x}_m), m = 1, \dots, N\}$ . In addition, if for an  $f \in \text{span}\{\kappa(\cdot, \mathbf{x}_m), m = 1, \dots, N\}$  we have that  $\langle f, \kappa(\cdot, \mathbf{x}_n) \rangle_{\mathcal{H}} = 0$  for all  $n = 1, \dots, N$ , if and only if  $f = 0$ . Hence, if  $f$  is orthogonal to all  $\Phi(\mathbf{x}_n)$ , then  $f = 0$ . We conclude that  $f = \sum_{m=1}^N a_m \kappa(\cdot, \mathbf{x}_m) = 0$  if and only if  $a_1 = \dots = a_N = 0$ . Therefore, the points  $\Phi(\mathbf{x}_m) = \kappa(\cdot, \mathbf{x}_m)$ ,  $m = 1, \dots, N$ , are linearly independent, provided that no two  $\mathbf{x}_m$  are the same. *Hence, a Gaussian kernel defined on a domain of infinite cardinality, produces a feature space of infinite dimension.* Moreover, the Gram matrices defined by Gaussian kernels are always strictly positive definite and invertible.

In addition, for every  $\mathbf{x}, \mathbf{y} \in X$  we have that  $\kappa(\mathbf{x}, \mathbf{x}) = 1$  and  $\kappa(\mathbf{x}, \mathbf{y}) \geq 0$ . This means that all  $\mathbf{x} \in X$  are mapped through the feature map  $\Phi$  to points lying in the surface of the unit sphere of the RKHS  $\mathcal{H}$  and that the angle between any two mapped points  $\Phi(\mathbf{x})$  and  $\Phi(\mathbf{y})$  is between  $0^\circ$  and  $90^\circ$  degrees. We conclude this section with the following two important formulas, which hold for the case of the RKHS induced by the Gaussian kernel. For the norm of  $f \in \mathcal{H}$ , one can prove that:

$$\|f\|_{\mathcal{H}}^2 = \int_X \sum_n \frac{\sigma^{2n}}{n! 2^n} (O^n f(\mathbf{x}))^2 d\mathbf{x}, \quad (17.103)$$

with  $O^{2n} = \Delta^n$  and  $O^{2n+1} = \nabla \Delta^n$ ,  $\Delta$  being the Laplacian and  $\nabla$  the gradient operator. The implication of this is that a regularization term of the form  $\|f\|_{\mathcal{H}}^2$  (which is usually adopted in practice) "penalizes" the derivatives of the minimizer. This results to a very smooth solution of the regularized risk minimization problem. Finally, the Fourier transform of the Gaussian kernel  $\kappa_\sigma$  is given by

$$F[\kappa](\omega) = |\sigma| \exp\left(-\frac{\sigma^2 \omega^2}{2}\right). \quad (17.104)$$

## B Proof of Proposition 60

First, let us define the stage of discussion. For the sake of convenience, we bind  $f$ s and  $\beta$ s together. To this end, define the Cartesian product

$$\mathsf{H} := \mathcal{H} \times \mathcal{R} := \{(f, \beta) : f \in \mathcal{H}, \beta \in \mathcal{R}\}.$$

To save space, let us denote every  $(f, \beta) \in \mathsf{H}$  by  $\mathbf{f}$ . Further, to equip  $\mathsf{H}$  with an inner product, define  $\forall \mathbf{f}_1 := (f, \beta), \forall \mathbf{f}_2 := (g, \gamma) \in \mathsf{H}$ ,

$$\langle \mathbf{f}_1, \mathbf{f}_2 \rangle_{\mathsf{H}} = \langle (f, \beta), (g, \gamma) \rangle_{\mathsf{H}} := \langle f, g \rangle + \beta \gamma.$$

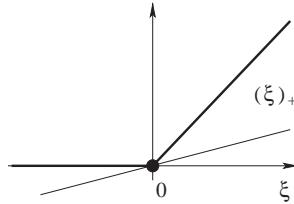


FIGURE 17.33

The positive part of a real number is not a differentiable function.

Under this definition, the task (17.93) obtains the following equivalent formulation:

$$\text{find } f \in H \text{ such that } \begin{cases} \langle f, k_{x_i} \rangle_H \geq 0, & \forall i \in \{1, 2, \dots, N_1\}, \\ \langle f, k_{\xi_j} \rangle_H \leq 0, & \forall j \in \{1, 2, \dots, N_2\}, \end{cases} \quad (17.105)$$

where we have defined  $k_{x_i} := (\kappa(x_i, \cdot), 1) \in H$  and  $k_{\xi_j} := (\kappa(\xi_j, \cdot), 1) \in H$ .

Let us develop this formulation a little further. Given a parameter vector  $k_z \in H$ , define the following closed convex set, namely, the *closed halfspace*:

$$H_{k_z}^- := \{f \in H : \langle f, k_z \rangle_H \leq 0\}.$$

Under this setting, (17.105) takes the following equivalent form:

$$\text{find } f \in \left( \bigcap_{i=1}^{N_1} H_{-k_{x_i}}^- \right) \cap \left( \bigcap_{j=1}^{N_2} H_{k_{\xi_j}}^- \right). \quad (17.106)$$

Now, let us define our *perceptron loss/penalty function*  $\mathcal{L} : H \rightarrow \mathcal{R}$  as follows:

$$\mathcal{L}(f) := \sum_{i=1}^{N_1} (\langle f, -k_{x_i} \rangle_H)_+ + \sum_{j=1}^{N_2} (\langle f, k_{\xi_j} \rangle_H)_+, \quad (17.107)$$

where the function  $(\cdot)_+$  stands for the *positive part* of a real number; see Figure 17.33.

The reasoning behind the definition of (17.107) is based on the following rationale. Since our goal is to compute an  $f$  which satisfies (17.106), we have to penalize somehow those  $f$  that do not. To this end, and without any loss of generality, assume an  $f$  and an  $i_0$  such that  $f \notin H_{-k_{x_{i_0}}}^-$ . By the definition of  $H_{-k_{x_{i_0}}}^-$ , we have  $\langle f, -k_{x_{i_0}} \rangle_H > 0$ . In such a way, the loss function  $\mathcal{L}$  scores a penalty, and the term  $\langle f, -k_{x_{i_0}} \rangle_H$  appears in (17.107). Moreover, the more  $f$  lies further from  $H_{-k_{x_{i_0}}}^-$ , i.e., the larger the quantity  $\langle f, -k_{x_{i_0}} \rangle_H$  is, the larger the penalty becomes. On the contrary, in the case where  $f \in H_{-k_{x_{i_0}}}^-$ , i.e.,  $\langle f, -k_{x_{i_0}} \rangle_H \leq 0$ , there is no penalty in (17.107) with respect to  $H_{-k_{x_{i_0}}}^-$ .

It can be easily seen by Figure 17.33 that the function  $l(\cdot) := (\cdot)_+$  is convex, with the following subdifferential:

$$\partial l(\theta) = \begin{cases} \{0\}, & \text{if } \theta < 0, \\ [0, 1], & \text{if } \theta = 0, \\ \{1\}, & \text{if } \theta > 0. \end{cases} \quad (17.108)$$

If we also define the *linear mapping*  $A_{k_z} : \mathcal{H} \rightarrow \mathcal{R} : f \mapsto A_{k_z}(f) := \langle f, k_z \rangle_{\mathcal{H}}$ , where  $k_z$  is a parameter vector, then we can readily verify that (17.107) can be given by the following equivalent form:

$$\mathcal{L}(f) := \sum_{i=1}^{N_1} l\left(A_{-k_{x_i}}(f)\right) + \sum_{j=1}^{N_2} l\left(A_{k_{y_j}}(f)\right),$$

i.e.,  $\mathcal{L}$  is the sum of a number of compositions of the convex function  $l(\cdot)$  with the linear mappings  $A_{-k_{x_i}}, A_{k_{y_j}}$ ; hence it is convex. An inspection of (17.107) shows that

$$\text{lev}_{\leq 0}(\mathcal{L}) = \left( \bigcap_{i=1}^{N_1} H_{-k_{x_i}}^- \right) \cap \left( \bigcap_{j=1}^{N_2} H_{k_{y_j}}^- \right).$$

In the sequel, we will show that Algorithm 57 solves the task (17.93), when the function  $\mathcal{L}$  is chosen as previously. To this end, let us recall here the definition of the interior of a set; a point  $g \in \text{int}(\text{lev}_{\leq 0}(\mathcal{L}))$  iff there exists a positive  $\delta > 0$  such that the open ball centered at  $g$  with a radius  $\delta$  belongs to  $\text{lev}_{\leq 0}(\mathcal{L})$ , i.e.,  $B(g, \delta) := \{f \in \mathcal{H} : \|g - f\| < \delta\} \subset \text{lev}_{\leq 0}(\mathcal{L})$ . Since the boundaries of the closed halfspaces  $H_{-k_{x_i}}^-$  and  $H_{k_{y_j}}^-$  are all those  $f$  such that  $\langle f, -k_{x_i} \rangle = 0$  and  $\langle f, k_{y_j} \rangle = 0$ , respectively, the assumption of Proposition 60 is equivalent to

$$\text{int}(\text{lev}_{\leq 0}(\mathcal{L})) \neq \emptyset. \quad (17.109)$$

Based on the definition of  $\mathcal{L}$ , we also have that

$$\mathcal{L}(f) = \sum_{i: f \notin H_{-k_{x_i}}^-} \langle f, -k_{x_i} \rangle_{\mathcal{H}} + \sum_{j: f \notin H_{k_{y_j}}^-} \langle f, k_{y_j} \rangle_{\mathcal{H}}, \quad \forall f \notin \text{lev}_{\leq 0}(\mathcal{L}).$$

Clearly, in such a case, the function is differentiable:

$$\mathcal{L}'(f) = - \sum_{i: f \notin H_{-k_{x_i}}^-} k_{x_i} + \sum_{j: f \notin H_{k_{y_j}}^-} k_{y_j}, \quad \forall f \notin \text{lev}_{\leq 0}(\mathcal{L}). \quad (17.110)$$

Hence,

$$\begin{aligned} \|\mathcal{L}'(f)\|_{\mathcal{H}} &\leq \sum_{i: f \notin H_{-k_{x_i}}^-} \|k_{x_i}\|_{\mathcal{H}} + \sum_{j: f \notin H_{k_{y_j}}^-} \|k_{y_j}\|_{\mathcal{H}} \\ &\leq \sum_{i=1}^{N_1} \|k_{x_i}\|_{\mathcal{H}} + \sum_{j=1}^{N_2} \|k_{y_j}\|_{\mathcal{H}} := K, \quad \forall f \notin \text{lev}_{\leq 0}(\mathcal{L}). \end{aligned} \quad (17.111)$$

Let us apply, now, the PSMa Algorithm 57 to the loss function of (17.107). Notice that in this case, the minimization task is unconstrained, i.e.,  $C := \mathbb{H}$ . In what follows, we assume that  $\mathbf{f}_{n-1} \notin \text{lev}_{\leq 0}(\mathcal{L})$ , since otherwise, the PSMa Algorithm 57 stalls to a point which is the solution of (17.106). Under the previous setting, for any initial point  $\mathbf{f}_0$ , for any  $\mathbf{f}_* \in \text{int}(\text{lev}_{\leq 0}(\mathcal{L}))$ , and for any iteration index  $n$ , we have:

$$\begin{aligned}\|\mathbf{f}_n - \mathbf{f}_*\|_{\mathbb{H}}^2 &= \|\mathbf{f}_{n-1} - \mu_n \mathcal{L}'(\mathbf{f}_{n-1}) - \mathbf{f}_*\|_{\mathbb{H}}^2 \\ &= \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2 - 2\mu_n \langle \mathbf{f}_{n-1} - \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} \\ &= \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2 - 2\mu_n \langle \mathbf{f}_{n-1}, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} \\ &\quad + 2\mu_n \langle \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}}.\end{aligned}$$

Notice, now, by (17.110) and the fact that  $\mathbf{f}_{n-1} \notin \text{lev}_{\leq 0}(\mathcal{L})$  that

$$\langle \mathbf{f}_{n-1}, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} = \sum_{i: \mathbf{f}_{n-1} \notin H_{-\mathbf{k}_{\mathbf{x}_i}}^-} \langle \mathbf{f}_{n-1}, -\mathbf{k}_{\mathbf{x}_i} \rangle_{\mathbb{H}} + \sum_{j: \mathbf{f}_{n-1} \notin H_{\mathbf{k}_{\mathbf{\xi}_j}}^-} \langle \mathbf{f}_{n-1}, \mathbf{k}_{\mathbf{\xi}_j} \rangle_{\mathbb{H}}, \quad (17.112a)$$

$$\langle \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} = \sum_{i: \mathbf{f}_{n-1} \notin H_{-\mathbf{k}_{\mathbf{x}_i}}^-} \langle \mathbf{f}_*, -\mathbf{k}_{\mathbf{x}_i} \rangle_{\mathbb{H}} + \sum_{j: \mathbf{f}_{n-1} \notin H_{\mathbf{k}_{\mathbf{\xi}_j}}^-} \langle \mathbf{f}_*, \mathbf{k}_{\mathbf{\xi}_j} \rangle_{\mathbb{H}}. \quad (17.112b)$$

It can be readily verified by (17.112a) that  $\langle \mathbf{f}_{n-1}, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} > 0$ , so that

$$\|\mathbf{f}_n - \mathbf{f}_*\|_{\mathbb{H}}^2 \leq \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2 + 2\mu_n \langle \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}}.$$

Now, let us focus on the term  $\langle \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}}$ . Due to assumption (17.109), there exists an  $\epsilon > 0$  such that

$$\begin{aligned}\forall i = 1, 2, \dots, N_1, \quad \langle \mathbf{f}_*, -\mathbf{k}_{\mathbf{x}_i} \rangle_{\mathbb{H}} &\leq -\epsilon < 0, \\ \forall j = 1, 2, \dots, N_2, \quad \langle \mathbf{f}_*, \mathbf{k}_{\mathbf{\xi}_j} \rangle_{\mathbb{H}} &\leq -\epsilon < 0.\end{aligned}$$

It becomes clear, then, by (17.112b) that

$$\langle \mathbf{f}_*, \mathcal{L}'(\mathbf{f}_{n-1}) \rangle_{\mathbb{H}} \leq -\epsilon < 0.$$

For this reason, we end up in

$$\begin{aligned}\|\mathbf{f}_n - \mathbf{f}_*\|_{\mathbb{H}}^2 &\leq \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2 - 2\epsilon\mu_n \\ &= \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \frac{\mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2}{\max^2 \{1, \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}\}} - \frac{2\mu_n\epsilon}{\max \{1, \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}\}} \\ &\leq \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \frac{\mu_n^2 \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2}{\|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}^2} - \frac{2\mu_n\epsilon}{\max \{1, \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}\}} \\ &\leq \|\mathbf{f}_{n-1} - \mathbf{f}_*\|_{\mathbb{H}}^2 + \mu_n^2 - \frac{2\mu_n\epsilon}{\max \{1, \|\mathcal{L}'(\mathbf{f}_{n-1})\|_{\mathbb{H}}\}},\end{aligned} \quad (17.113)$$

where we used the definition of  $\mu_n$  and the quantity  $K$  introduced in (17.111). A repetition of the telescoping (17.113) gives the following:

$$\begin{aligned}\|f_n - f_*\|_H^2 &\leq \|f_0 - f_*\|_H^2 + \sum_{k=1}^n \mu_k^2 - \frac{2\epsilon}{\max\{1, K\}} \sum_{k=1}^n \mu_k \\ &\leq \|f_0 - f_*\|_H^2 + \sum_{n=1}^{\infty} \mu_n^2 - \frac{2\epsilon}{\max\{1, K\}} \sum_{k=1}^n \mu_k.\end{aligned}\quad (17.114)$$

Since  $\sum_{n=1}^{\infty} \mu_n = \infty$ , there exists an  $n_0 \in \mathcal{N}_*$  such that  $\forall n \geq n_0$ ,

$$\sum_{k=1}^n \mu_k \geq \frac{\max\{1, K\}}{2\epsilon} \left( \|f_0 - f_*\|_H^2 + \sum_{n=1}^{\infty} \mu_n^2 \right).$$

This result and (17.114) imply that  $\forall n \geq n_0$ ,

$$0 \leq \|f_n - f_*\|_H^2 \leq 0,$$

i.e.,  $\forall n \geq n_0$ ,  $f_n = f_*$ . This completes the proof.

## C Proof of convergence for algorithm 61

Since the set of all minimizers,  $C_{\mathcal{L}_*}$ , of (17.84) is nonempty, let us choose any  $f_* \in C_{\mathcal{L}_*}$ . Then, if we let  $v_n := \mu_n \frac{\mathcal{L}(f_{n-1}) - \mathcal{L}_*}{\|\mathcal{L}'(f_{n-1})\|^2}$ , we observe that

$$\begin{aligned}\|f_n - f_*\|^2 &= \|P_C(f_{n-1} - v_n \mathcal{L}'(f_{n-1})) - P_C(f_*)\|^2 \leq \|f_{n-1} - f_* - v_n \mathcal{L}'(f_{n-1})\|^2 \\ &= \|f_{n-1} - f_*\|^2 + v_n^2 \|\mathcal{L}'(f_{n-1})\|^2 - 2v_n \langle f_{n-1} - f_*, \mathcal{L}'(f_{n-1}) \rangle \\ &\leq \|f_{n-1} - f_*\|^2 + v_n^2 \|\mathcal{L}'(f_{n-1})\|^2 - 2v_n (\mathcal{L}(f_{n-1}) - \mathcal{L}(f_*)) \\ &= \|f_{n-1} - f_*\|^2 + \mu_n^2 \frac{(\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2}{\|\mathcal{L}'(f_{n-1})\|^2} - 2\mu_n \frac{(\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2}{\|\mathcal{L}'(f_{n-1})\|^2} \\ &= \|f_{n-1} - f_*\|^2 - \mu_n(2 - \mu_n) \frac{(\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2}{\|\mathcal{L}'(f_{n-1})\|^2} \\ &\leq \|f_{n-1} - f_*\|^2 - \epsilon^2 \frac{(\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2}{\|\mathcal{L}'(f_{n-1})\|^2},\end{aligned}$$

or in other words,

$$\epsilon^2 \frac{(\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2}{\|\mathcal{L}'(f_{n-1})\|^2} \leq \|f_{n-1} - f_*\|^2 - \|f_n - f_*\|^2. \quad (17.115)$$

Hence,  $\|f_n - f_*\| \leq \|f_{n-1} - f_*\|$ , which implies that the  $\lim_{n \rightarrow \infty} \|f_n - f_*\|^2$  exists. Since the previous analysis was conducted for any point  $f_* \in C_{\mathcal{L}_*}$ , we obtain that given any  $f_* \in C_{\mathcal{L}_*}$ , there exists a  $\delta_{f_*} \geq 0$  such that

$$\lim_{n \rightarrow \infty} \|f_n - f_*\| = \delta_{f_*}. \quad (17.116)$$

Due to the fact that  $(\|f_n - f_*\|^2)_{n \in \mathcal{N}}$  is convergent, it is also a Cauchy sequence, which suggests, in turn, that

$$\lim_{n \rightarrow \infty} (\|f_{n-1} - f_*\|^2 - \|f_n - f_*\|^2) = 0. \quad (17.117)$$

Moreover, due to the monotonicity of the sequence  $(\|f_n - f_*\|^2)_{n \in \mathcal{N}}$ , we observe that  $\|f_n - f_*\| \leq \|f_0 - f_*\|$ , or in other words,  $f_n \in B[f_*, \|f_0 - f_*\|] \cap C$ , and hence  $\|\mathcal{L}'(f_n)\| \leq c, \forall n$ . Based on this observation, we have by (17.115) that

$$\frac{\epsilon^2}{c^2} (\mathcal{L}(f_{n-1}) - \mathcal{L}_*)^2 \leq \|f_{n-1} - f_*\|^2 - \|f_n - f_*\|^2.$$

Hence, by (17.117),  $\lim_{n \rightarrow \infty} \mathcal{L}(f_n) = \mathcal{L}_* = \min_{f \in C} \mathcal{L}(f)$ .

Since  $(\|f_n - f_*\|)_{n \in \mathcal{N}}$  is bounded, so is  $(f_n)_{n \in \mathcal{N}}$ . Thus,  $(f_n)_{n \in \mathcal{N}}$  is weakly compact [65], and necessarily possesses at least one sequentially weak cluster point. Moreover,  $(f_n)_{n \in \mathcal{N}}$  lies in  $C$ , which is closed and convex, and thus weakly closed, which implies that the cluster point  $\tilde{f}_*$  belongs to  $C$ . Since  $\mathcal{L}$  is weakly lower semicontinuous [65], we finally obtain that

$$\mathcal{L}_* \leq \mathcal{L}(\tilde{f}_*) \leq \liminf_{k \rightarrow \infty} \mathcal{L}(f_{n_k}) = \lim_{k \rightarrow \infty} \mathcal{L}(f_{n_k}) = \lim_{n \rightarrow \infty} \mathcal{L}(f_n) = \mathcal{L}_*,$$

i.e.,  $\tilde{f}_* \in C_{\mathcal{L}_*}$ .

Let us show, now, that the sequence  $(f_n)_{n \in \mathcal{N}}$  converges weakly to a point in  $C_{\mathcal{L}_*}$ . We have already shown previously that the set of sequentially weak cluster points  $\mathcal{W}((f_n)_{n \in \mathcal{N}})$  of  $(f_n)_{n \in \mathcal{N}}$  is non-empty, and that  $\mathcal{W}((f_n)_{n \in \mathcal{N}}) \subset C_{\mathcal{L}_*}$ . To derive a contradiction, let two points  $\tilde{f}_{*1}, \tilde{f}_{*2} \in \mathcal{W}((f_n)_{n \in \mathcal{N}})$  such that  $\tilde{f}_{*1} \neq \tilde{f}_{*2}$ . Since both of  $\tilde{f}_{*1}, \tilde{f}_{*2}$  are sequentially weak cluster points, there exist two subsequences  $(f_{n_k})_{k \in \mathcal{N}}$  and  $(f_{m_k})_{k \in \mathcal{N}}$  such that  $f_{n_k} \rightharpoonup \tilde{f}_{*1}$  and  $f_{m_k} \rightharpoonup \tilde{f}_{*2}$ . By (17.116),

$$\lim_{n \rightarrow \infty} \|f_n - \tilde{f}_{*1}\| = \delta_{\tilde{f}_{*1}}, \quad \lim_{n \rightarrow \infty} \|f_n - \tilde{f}_{*2}\| = \delta_{\tilde{f}_{*2}}.$$

Without any loss of generality, assume that  $\delta_{\tilde{f}_{*1}} \leq \delta_{\tilde{f}_{*2}}$ . Then, notice that

$$\begin{aligned} \delta_{\tilde{f}_{*2}}^2 &= \lim_{n \rightarrow \infty} \|f_n - \tilde{f}_{*2}\|^2 = \lim_{k \rightarrow \infty} \|f_{m_k} - \tilde{f}_{*2}\|^2 = \lim_{k \rightarrow \infty} \|f_{m_k} - \tilde{f}_{*1} + \tilde{f}_{*1} - \tilde{f}_{*2}\|^2 \\ &= \lim_{k \rightarrow \infty} \left( \|f_{m_k} - \tilde{f}_{*1}\|^2 + \|\tilde{f}_{*1} - \tilde{f}_{*2}\|^2 + 2 \langle f_{m_k} - \tilde{f}_{*1}, \tilde{f}_{*1} - \tilde{f}_{*2} \rangle \right) \\ &= \delta_{\tilde{f}_{*1}}^2 + \|\tilde{f}_{*1} - \tilde{f}_{*2}\|^2 + 2 \langle \tilde{f}_{*2} - \tilde{f}_{*1}, \tilde{f}_{*1} - \tilde{f}_{*2} \rangle \\ &= \delta_{\tilde{f}_{*1}}^2 + \|\tilde{f}_{*1} - \tilde{f}_{*2}\|^2 - 2 \|\tilde{f}_{*1} - \tilde{f}_{*2}\|^2 = \delta_{\tilde{f}_{*1}}^2 - \|\tilde{f}_{*1} - \tilde{f}_{*2}\|^2 < \delta_{\tilde{f}_{*1}}^2 \leq \delta_{\tilde{f}_{*2}}^2, \end{aligned}$$

which is evidently a contradiction. Hence, there is only one sequentially weak cluster point of  $(f_n)_{n \in \mathcal{N}}$ , which implies that  $(f_n)_{n \in \mathcal{N}}$  converges weakly to that point. This completes the proof.

*Relevant Theory:* Signal Processing Theory

See this Volume, [Chapter 12](#) Adaptive Filters

---

## References

- [1] C.M. Bishop, Pattern Recognition and Machine Learning, corr. second printing ed., Springer, 2007.
- [2] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth ed., Academic Press, 2009.
- [3] N. Aronszajn, Theory of reproducing kernels, Trans. Am. Math. Soc. 68 (3) (1950) 337–404.
- [4] P.S.R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation, third ed., Springer, 2008.
- [5] S. Haykin, Adaptive Filter Theory, third ed., Prentice-Hall, New Jersey, 1996.
- [6] A.H. Sayed, Fundamentals of Adaptive Filtering, John Wiley & Sons, New Jersey, 2003.
- [7] V.N. Vapnik, The Nature of Statistical Learning Theory, second ed., Springer, 2000.
- [8] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed., Springer, 2009.
- [9] E.J. Candès, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
- [10] J.M. Lee, Introduction to Smooth Manifolds, Springer, 2003.
- [11] M.A. Aizerman, E.M. Braverman, L.I. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Autom. Remote Control 25 (1964) 821–837.
- [12] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, 1992, pp. 144–152.
- [13] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2001.
- [14] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equation, Philos. Trans. Roy. Soc. Lond. 209 (1909) 415–446.
- [15] J. Mercer, Sturm-Liouville series of normal functions in the theory of integral equations, Philos. Trans. Roy. Soc. Lond. Ser. A 211 (1911) 111–198.
- [16] E.H. Moore, On properly positive hermitian matrices, Bull. Am. Math. Soc. 23 (1916) 59.
- [17] E.H. Moore, General analysis, Part I, Memoirs of the American Philosophical Society, 1935.
- [18] E.H. Moore, General analysis, Part II, Memoirs of the American Philosophical Society, 1939.
- [19] S. Bochner, Vorlesungen Ueber Fouriersche Integrale, 1932.
- [20] S. Bochner, Hilbert distances and positive definite functions, Ann. Math. 42 (1941) 647–656.
- [21] S. Zaremba, L'equation biharmonique et une classe remarquable de fonctions fondamentales harmoniques, Bulletin international de l' Academie des sciences de Cracovie, 1907, pp. 147–196.
- [22] S. Zaremba, Sur le calcul numerique des fonctions demandees dans le probleme de dirichlet et le probleme hydrodynamique, Bulletin international de l' Academie des sciences de Cracovie, 1908.
- [23] S. Bergman, Ueber die entwicklung der harmonischen funktionen der ebene und des raumes nach orthogonalfunktionen, Math. Ann. 86 (1922) 238–271.
- [24] V.I. Paulsen, An Introduction to the Theory of Reproducing Kernel Hilbert Spaces, September 2009 (Notes).
- [25] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
- [26] G.S. Kimeldorf, G. Wahba, Some results on Tchebycheffian spline functions, J. Math. Anal. Appl. 33 (1971) 82–95.
- [27] P. Bouboulis, K. Slavakis, S. Theodoridis, Adaptive kernel-based image denoising employing semi-parametric regularization, IEEE Trans. Image Process. 19 (6) (2010) 1465–1479.

- [28] A.L. Yuille, N.M. Grzywacz, A mathematical analysis of the motion coherence theory, *Int. J. Comput. Vis.* 3 (2) (1989) 155–175.
- [29] I.J. Schoenberg, Positive definite functions on spheres, *Duke Math. J.* 9 (1942) 96–108.
- [30] A.V. Balakrishnan, *Applied Functional Analysis*, Springer, 1976.
- [31] L. Debnath, P. Mikusinski, *Introduction to Hilbert Spaces with Applications*, second ed., Academic Press, 1999.
- [32] S. Lang, *Real and Functional Analysis*, third ed., Springer, 1993.
- [33] L.A. Liusternik, V.J. Sobolev, *Elements of Functional Analysis*, Frederick Ungar Publishing Co., 1961.
- [34] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
- [35] J. van Tiel, *Convex Analysis: An Introductory Text*, John Wiley & Sons Ltd., 1984.
- [36] B. Widrow, M.E. Hoff Jr., Adaptive switching circuits, in: *IRE WESCON Conv. Rec.*, 1960, pp. 96–104 (pt. 4).
- [37] W. Liu, J. Príncipe, S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley, Hoboken, New Jersey, 2010.
- [38] W. Liu, P. Pokharel, J. Príncipe, The kernel least mean squares algorithm, *IEEE Trans. Signal Process.* 56 (2) (2008) 543–554.
- [39] C. Richard, J. Bermudez, P. Honeine, Online prediction of time series data with kernels, *IEEE Trans. Signal Process.* 57 (3) (2009) 1058–1067.
- [40] B. Chen, S. Zhao, P. Zhu, J. Príncipe, Quantized kernel least mean square algorithm, *IEEE Trans. Neural Networks Learn. Syst.* 23 (1) (2012) 22–32.
- [41] P. Boulopoulis, K. Slavakis, S. Theodoridis, Adaptive learning in complex reproducing kernel Hilbert spaces employing Wirtinger's subgradients, *IEEE Trans. Neural Networks Learn. Syst.* 23 (3) (2012) 425–438.
- [42] K. Slavakis, S. Theodoridis, I. Yamada, Adaptive constrained learning in reproducing kernel Hilbert spaces: the robust beamforming case, *IEEE Trans. Signal Process.* 57 (12) (2009) 4744–4764.
- [43] K. Slavakis, P. Boulopoulis, S. Theodoridis, Adaptive multiregression in reproducing kernel Hilbert spaces: the multiaccess MIMO channel case, *IEEE Trans. Neural Networks Learn. Syst.* 23 (2) (2012) 260–276.
- [44] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, *IEEE Trans. Signal Process.* 52 (8) (2004) 2275–2285.
- [45] D.J. Sebald, J.A. Bucklew, Support vector machine techniques for nonlinear equalization, *IEEE Trans. Signal Process.* 48 (2000) 3217–3226.
- [46] K. Kreutz-Delgado, The complex gradient operator and the  $\mathbb{CR}$ -calculus, 2009.  
[<http://arxiv.org/abs/0906.4835>](http://arxiv.org/abs/0906.4835)
- [47] H. Li, Complex-valued adaptive signal processing using wirtinger calculus and its application to independent component analysis, PhD Thesis, University of Maryland, 2008.
- [48] B. Picinbono, On circularity, *IEEE Trans. Signal Process.* 42 (12) (1994) 3473–3482.
- [49] B. Picinbono, P. Chevalier, Widely linear estimation with complex data, *IEEE Trans. Signal Process.* 43 (8) (1995) 2030–2033.
- [50] B. Picinbono, P. Bondon, Second order statistics of complex signals, *IEEE Trans. Signal Process.* 45 (2) (1997) 411–420.
- [51] P. Boulopoulis, S. Theodoridis, M. Mavroforakis, The augmented complex kernel LMS, *IEEE Trans. Signal Process.* (2012).
- [52] T. Adali, H. Li, Complex-valued adaptive signal processing, *Adaptive Signal Processing: Next Generation Solutions*, Wiley, NJ, 2010, pp. 1–74.
- [53] D. Mandic, V. Goh, *Complex Valued Nonlinear Adaptive Filters*, Wiley, 2009.
- [54] P. Boulopoulis, S. Theodoridis, Extension of Wirtinger calculus in RKH spaces and the complex kernel LMS, in: *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, Kittil, Finland, 2010.

- [55] P. Bouboulis, S. Theodoridis, Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS, *IEEE Trans. Signal Process.* 59 (3) (2011) 964–978.
- [56] A. Kuh, D. Mandic, Applications of complex augmented kernels to wind profile prediction, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2009, pp. 3581–3584.
- [57] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez, A.R. Figueiras-Vidal, Weighted least squares training of support vectors classifiers which leads to compact and adaptive schemes, *IEEE Trans. Neural Networks* 12 (5) (2001) 1047–1059.
- [58] J.A.K. Suykens, J. de Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing* 48 (2002) 85–105.
- [59] J.A.K. Suykens, T. van Gestel, J. de Brabanter, B. de Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.
- [60] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.* 9 (1999) 293–300.
- [61] S. van Vaerenbergh, M. Lazaro-Gredilla, I. Santamaría, Kernel recursive least-squares tracker for time-varying regression, *IEEE Trans. Neural Networks Learn. Syst.* 23 (8) (2012) 1313–1326.
- [62] R.T. Rockafellar, R.J.-B. Wets, *Variational Analysis*, Springer, Berlin, 2004.
- [63] D.P. Bertsekas, A. Nedic, A.E. Ozdaglar, *Convex Analysis and Optimization*, Athena Scientific, 2003.
- [64] M. Todd, Semidefinite optimization, *Acta Numer.* 10 (2001) 515–560.
- [65] H.H. Bauschke, P.L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.
- [66] K. Goebel, S. Reich, *Uniform Convexity, Hyperbolic Geometry, and Nonexpansive Mappings*, Pure and Applied Mathematics, vol. 83, Marcel Dekker, New York, 1984.
- [67] K. Goebel, W.A. Kirk, *Topics in Metric Fixed Point Theory*, Cambridge Studies Advanced Mathematics, vol. 28, Cambridge University Press, 1990.
- [68] K. Slavakis, S. Theodoridis, Sliding window generalized kernel affine projection algorithm using projection mappings, *EURASIP J. Adv. Signal Process.* 2008 (2008) 16.
- [69] D.G. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Sons, New York, 1969.
- [70] J.-B. Baillon, G. Haddad, Quelques propriétés des opérateurs angle-bornés et n-cycliquement monotones, *Isr. J. Math.* 26 (1977) 137–150.
- [71] H.H. Bauschke, P.L. Combettes, The Baillon-Haddad theorem revisited, *J. Convex Anal.* 17 (2010) 781–787.
- [72] P.L. Combettes, V.R. Wajs, Signal recovery by proximal forward-backward splitting, *Multiscale Model. Simul.* 4 (2005) 1168–1200.
- [73] P.L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, in: *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer-Verlag, 2011.
- [74] J.-J. Moreau, Fonctions convexes duales et points proximaux dans un espace Hilbertien, *Acad. Sci. Paris Sér. A Math.* 255 (1962) 2897–2899.
- [75] P.L. Combettes, J.-C. Pesquet, A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery, *IEEE J. Sel. Top. Signal Process.* 1 (4) (2007) 564–574.
- [76] J. Douglas Jr., H.H. Rachford Jr., On the numerical solution of heat conduction problems in two and three space variables, *Trans. Am. Math. Soc.* 82 (1956) 421–439.
- [77] J. Duchi, Y. Singer, Efficient online and batch learning using forward backward splitting, *J. Mach. Learn. Res.* 10 (2009) 2899–2934.
- [78] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.* 12 (2011) 2121–2159.
- [79] A.A. Goldstein, Convex programming in Hilbert space, *Bull. Am. Math. Soc.* 70 (5) (1964) 709–710.

- [80] E.S. Levitin, B.T. Polyak, Constrained minimization methods, *Zh. učebn. Mat. mat. Fiz.* 6 (5) (1966) 787–823.
- [81] R.A. Horn, C.R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.
- [82] Y.I. Alber, A.N. Iusem, M.V. Solodov, On the projected subgradient method for nonsmooth convex optimization in a Hilbert space, *Math. Program.* 81 (1998) 23–35.
- [83] P.-E. Maingé, Strong convergence of projected subgradient methods for nonsmooth and nonstrictly convex minimization, *Set-Valued Anal.* 16 (2008) 899–912.
- [84] B.T. Polyak, A general method for solving extremal problems, *Dokl. Akad. Nauk. SSSR* 174 (1) 1967 33–36.
- [85] Y.M. Ermol'ev, Methods for solving non-linear extremal problems, *Kibernetika* 4 (1966) 1–17.
- [86] N.Z. Shor, On the structure of algorithms for the numerical solution of optimal planning and design problems, PhD Thesis, Cybernetics Institute, Academy of Sciences, Kiev, 1964.
- [87] A.B. Novikoff, On convergence proofs on perceptrons, in: *Symposium on the Mathematical Theory of Automata*, 1962, pp. 615–622.
- [88] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, *Psychol. Rev.* 65 (1958) 386–408.
- [89] B.T. Polyak, Minimization of unsmooth functionals, *Zh. učebn. Mat. mat. Fiz.* 9 (3) (1969) 509–521.
- [90] I. Yamada, Adaptive projected subgradient method: a unified view for projection based adaptive algorithms, *J. IEICE* 86 (8) (2003) 654–658 (in Japanese).
- [91] I. Yamada, N. Ogura, Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions, *Numer. Funct. Anal. Optim.* 25 (7/8) (2004) 593–617.
- [92] K. Slavakis, I. Yamada, N. Ogura, The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings, *Numer. Funct. Anal. Optim.* 27 (7/8) (2006) 905–930.
- [93] K. Slavakis, I. Yamada, The adaptive projected subgradient method constrained by families of quasi-nonexpansive mappings and its application to online learning, submitted for publication. <<http://arxiv.org/abs/1008.5231>>.
- [94] M. Yukawa, K. Slavakis, I. Yamada, Multi-domain adaptive learning based on feasibility splitting and adaptive projected subgradient method, *IEICE Trans. Fundam.* E93-A (2) (2010) 456–466.
- [95] S. Theodoridis, K. Slavakis, I. Yamada, Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks, *IEEE Signal Process. Mag.* 28 (1) (2011) 97–123.
- [96] K. Slavakis, S. Theodoridis, I. Yamada, Online kernel-based classification using adaptive projection algorithms, *IEEE Trans. Signal Process.* 56 (7) (2008) 2781–2796.
- [97] Y. Kopsinis, K. Slavakis, S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted  $\ell_1$  balls, *IEEE Trans. Signal Process.* 59 (3) (2011) 936–952.
- [98] S. Chouvardas, K. Slavakis, S. Theodoridis, Adaptive robust distributed learning in diffusion sensor networks, *IEEE Trans. Signal Process.* 59 (10) (2011) 4692–4707.
- [99] K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties, *IEICE Trans.* 67-A (5) (1984) 126–132 (in Japanese).
- [100] T. Hinamoto, S. Maekawa, Extended theory of learning identification, *Trans. IEE Jpn.* 95-C (10) (1975) 227–234 (in Japanese).
- [101] I. Yamada, K. Slavakis, K. Yamada, An efficient robust adaptive filtering algorithm based on parallel subgradient projection techniques, *IEEE Trans. Signal Process.* 50 (5) (2002) 1091–1101.
- [102] A.E. Albert, L.A. Gardner, *Stochastic Approximation and Nonlinear Regression*, MIT Press, 1967.
- [103] A.V. Malipatil, Y.-F. Huang, S. Andra, K. Bennett, Kernelized set-membership approach to nonlinear adaptive filtering, in: *Proceedings of IEEE ICASSP*, IEEE, 2005, pp. 149–152.
- [104] J. Nagumo, J. Noda, A learning method for system identification, *IEEE Trans. Autom. Control* 12 (3) (1967) 282–287.

- [105] K. Slavakis, Y. Kopsinis, S. Theodoridis, Revisiting adaptive least-squares estimation and application to online sparse signal recovery, in: Proceedings of the IEEE ICASSP, Prague, Czech Republic, 2011, pp. 4292–4295.
- [106] R. Cavalcante, I. Yamada, Flexible peak-to-average power ratio reduction scheme for OFDM systems by the adaptive projected subgradient method, *IEEE Trans. Signal Process.* 57 (4) (2009) 1456–1468.
- [107] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Generalized thresholding sparsity-aware algorithm for low complexity online learning, in: Proceedings of the IEEE ICASSP, Kyoto, Japan, 2012, pp. 3277–3280.
- [108] C. Breining, P. Dreiseitel, E. Hänsler, A. Mader, B. Nitsch, H. Puder, T. Schertler, G. Schmidt, J. Tilp, Acoustic echo control—An application of very-high-order adaptive filters, *IEEE Signal Process. Mag.* 16 (4) (1999) 42–69.
- [109] M. Martínez-Ramón, J.L. Rojo-Álvarez, G. Camps-Valls, C.G. Christodoulou, Kernel antenna array processing, *IEEE Trans. Antennas Propag.* 55 (3) (2007) 642–650.
- [110] Y. Murakami, M. Yamagishi, M. Yukawa, I. Yamada, A sparse adaptive filtering using time-varying soft-thresholding techniques, in: Proceedings of the IEEE ICASSP, Dallas, USA, March 2010, pp. 3734–3737.
- [111] S. Werner, P.S.R. Diniz, Set-membership affine projection algorithm, *IEEE Signal Process. Lett.* 8 (8) (2001) 231–235.
- [112] M. Yukawa, I. Yamada, Efficient adaptive stereo echo canceling schemes based on simultaneous use of multiple state data, *IEICE Trans. Fundam. E87-A* (8) (2004) 1949–1957.
- [113] M. Yukawa, Multi-kernel adaptive filtering, *IEEE Trans. Signal Process.*, in press.
- [114] Y. Amit, S. Shalev-Shwartz, Y. Singer, Online learning of complex prediction problems using simultaneous projections, *J. Mach. Learn. Res.* 9 (2008) 1399–1435.
- [115] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, *J. Mach. Learn. Res.* 6 (2005) 1579–1619.
- [116] L. Bottou, Y. LeCun, Large scale online learning, in: Advances in Neural Information Processing Systems (NIPS), MIT Press, Cambridge, MA, 2004.
- [117] N. Cesa-Bianchi, S. Shalev-Shwartz, O. Shamir, Online learning of noisy data, *IEEE Trans. Inform. Theory* 57 (12) (2011) 7907–7931.
- [118] N. Cesa-Bianchi, P.M. Long, M.K. Warmuth, Worst-case quadratic loss bounds for prediction using linear functions and gradient descent, *IEEE Trans. Neural Networks* 7 (3) (1996) 604–619.
- [119] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, *J. Mach. Learn. Res.* 7 (2006) 551–585.
- [120] A. Elisseeff, M. Pontil, Leave-one-out error and stability of learning algorithms with applications, in: Advances in Learning Theory: Methods, Models and Applications, IOS Press, 2003.
- [121] M. Herbster, M. Pontil, Prediction on a graph with the perceptron, in: Advances in Neural Information Processing Systems (NIPS), MIT Press, Cambridge, MA, 2006, pp. 577–584.
- [122] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, *IEEE Trans. Signal Process.* 52 (8) (2004) 2165–2176.
- [123] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, *J. Mach. Learn. Res.* 10 (2009) 777–801.
- [124] A.F.T. Martins, N.A. Smith, E.P. Xing, P.M.Q. Aguiar, M.A.T. Figueiredo, Online learning of structured predictors with multiple kernels, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 2011.
- [125] T. Poggio, S. Voinea, L. Rosasco, Online learning, stability, and stochastic gradient descent, September 2011. <<http://arxiv.org/abs/1105.4701v3>>.
- [126] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: primal estimated sub-gradient solver for SVM, *Math. Program.* 127 (1) (2011) 3–30.

- [127] S. Shalev-Shwartz, Online learning and online convex optimization, *Found. Trends Mach. Learn.* 4 (2) (2012) 107–194.
- [128] Y. Ying, M. Pontil, Online gradient descent learning algorithms, *Found. Comput. Math.* 8 (5) (2008) 561–596.
- [129] P.L. Combettes, The foundations of set theoretic estimation, *Proc. IEEE* 81 (2) (1993) 182–208.
- [130] D.C. Youla, H. Webb, Image restoration by the method of convex projections: Part 1—Theory, *IEEE Trans. Med. Imag.* MI-1 (1982) 81–94.

# Introduction to Probabilistic Graphical Models

# 18

Franz Pernkopf, Robert Peharz, and Sebastian Tschiatschek

Graz University of Technology, Laboratory of Signal Processing and Speech Communication Inffeldgasse 16c,  
8010 Graz, Austria

---

## Nomenclature

$X, Y, Z, Q, C$	Random variable (RV)
$x, y, z, q, w, c$	Instantiation of random variable
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{O}, \mathbf{Q}, \mathbf{H}$	Set of random variables
$\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{o}, \mathbf{q}, \mathbf{h}, \mathbf{w}$	Set of instantiations of random variables
$\mathbf{val}(X)$	Set of values (states) of random variable $X$
$\mathbf{sp}(X) =  \mathbf{val}(X) $	Number of states of random variable $X$
$P_X(x)$	Probability density function (PDF) of a continuous random variable $X$
$P_X(x) = P(X = x)$	Probability mass function (PMF) of a discrete random variable $X$
$P(X, Y)$	Joint probability distribution of random variable $X$ and $Y$
$P(X Y)$	Conditional probability distribution (CPD) of $X$ conditioned on $Y$
$\Omega$	Outcome space composed of a set of events
$\omega_i$	Elementary event
$\Sigma$	Event space
$\theta$	Parameters
$\boldsymbol{\theta}, \Theta$	Set of parameters
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of nonnegative real numbers
$\mathbb{E}[X]$	Expected value of random variable $X$
$\mathbb{1}_{\{A\}}$	Indicator function; equals 1 if statement $A$ is true and 0 otherwise
$\mathcal{N}(\cdot   \mu; \sigma^2)$	Gaussian distribution with mean $\mu$ and variance $\sigma^2$
$\mathcal{B}$	Bayesian network (BN)
$\mathcal{M}$	Markov network (MN)
$\mathcal{F}$	Factor graph (FG)
$\mathcal{D}$	Set of i.i.d. samples
$\mathcal{C}$	Class of models, e.g., Bayesian networks

$\mathcal{L}(\cdot; \cdot)$	Likelihood function (LH)
$\ell(\cdot, \cdot)$	Log-likelihood function
$\text{KL}(P_A    P_B)$	Kullback-Leibler (KL) divergence between distribution $P_A$ and $P_B$
$L(\cdot, \cdot)$	Lagrangian function
$\nu$	Lagrangian multiplier
$\text{Dir}(\cdot; \alpha)$	Dirichlet distribution parametrized by the vector $\alpha$
$B(\alpha)$	Beta-function parametrized by the vector $\alpha$
$\text{CR}(\cdot)$	Classification rate (CR)
$\text{CLL}(\cdot; \cdot)$	Conditional log likelihood (CLL)
$\text{MM}(\cdot; \cdot)$	Margin objective function (MM)
<b>G</b>	Generator matrix of a linear block code
<b>H</b>	Parity-check matrix of a linear block code
$\mathcal{A}$	Finite alphabet of a code
$\mathcal{G}$	Directed or undirected graph
<b>E</b>	Set of edges
$\text{Pa}_{\mathcal{G}}(X)$	Set of all parents of $X$ in graph $\mathcal{G}$
$\text{Ch}_{\mathcal{G}}(X)$	Set of all children of $X$ in graph $\mathcal{G}$
$\text{Nb}_{\mathcal{G}}(X)$	Set of neighbors of $X$ in graph $\mathcal{G}$
$\text{Anc}_{\mathcal{G}}(X)$	Set of ancestors of $X$ in graph $\mathcal{G}$
$\text{Desc}_{\mathcal{G}}(X)$	Set of descendants of $X$ in graph $\mathcal{G}$
$\text{NonDesc}_{\mathcal{G}}(X)$	Set of nondescendants of $X$ in graph $\mathcal{G}$
<b>C</b>	Clique
<b>S</b>	Separator set
$\Psi_{\mathbf{C}}$	Nonnegative potential functions over the maximal cliques <b>C</b>
$f_i$	Positive function of a factor graph $\mathcal{F}$
<b>F</b>	Set of factor nodes
$\mu_{X \rightarrow Y}(y)$	Message from node $X$ to node $Y$
$\alpha(\cdot)$	Forward probabilities
$\beta(\cdot)$	Backward probabilities

---

### 1.18.1 Introduction

Machine learning and pattern recognition are elementary building blocks of *intelligent systems*. The aim is to capture relevant phenomena hidden in empirical data using computational and statistical methods. Hence, the task is to automatically recognize and identify complex patterns in data which are further cast into a model. A prerequisite is that the learned model generalizes well in order to provide useful information for new data samples. To account for this, machine learning relies on many fields in science such as statistics and probability calculus, optimization methods, cognitive science, computer science, et cetera.

Learning can be divided into supervised, unsupervised, and reinforcement learning problems. For supervised learning the training data comprises of feature vectors which contain an abstract description of the object and their corresponding target or class label. If the desired target value is continuous, then this is referred to as *regression*. In the case where the input vector is assigned to a finite number of categories this is called *classification*. In unsupervised learning, the training data consists of a set of feature vectors without corresponding target value. The aim is to discover groups/clusters within the data or to model the distribution of the data, i.e., representations of the data are modeled. In *reinforcement* learning, an agent should perform actions in an environment so that it maximizes a long-term reward. The learning algorithm cannot access a finite set of labeled samples for training as in supervised classification. It must discover which actions of the agent result in an optimal long-term reward. Typically, the actions do not only affect an immediate reward but also have impact on future actions, i.e., taking the best reward at each time step is insufficient for the global optimal solution.

Probabilistic graphical models (PGMs) [1] are important in all three learning problems and have turned out to be the method of choice for modeling uncertainty in many areas such as computer vision, speech processing, time-series and sequential data modeling, cognitive science, bioinformatics, probabilistic robotics, signal processing, communications and error-correcting coding theory, and in the area of artificial intelligence in general. One reason is that this common modeling framework allows to transfer concepts and ideas among different application areas. Another reason stated in [2] is that *common-sense reasoning is naturally embedded within the syntax of probability calculus*. PGMs combine probability theory and graph theory. They offer a compact graph-based representation of joint probability distributions exploiting conditional independencies among the random variables. Conditional independence assumptions alleviate the computational burden for model learning and inference. Hence, graphical models provide techniques to deal with two inherent problems throughout applied mathematics and engineering, namely, uncertainty and complexity [3]. Many well-known statistical models, e.g., (dynamic) Bayesian networks, mixture models, factor analysis, hidden Markov models (HMMs), factorial HMMs, Kalman filters, Boltzmann machines, the Ising model, amongst others, can be represented by graphical models. The framework of graphical models provides techniques for inference (e.g., *sum product algorithm*, also known as *belief propagation or message passing*) [2] and learning [1].<sup>1</sup> In this article, three popular representations of graphical models are presented: Markov networks (MNs) (also known as undirected graphical models (UGMs) or Markov random fields (MRFs)), Bayesian networks (BNs) (also known as directed graphical models (DGMs) or belief networks) and factor graphs (FGs).

The research in PGMs has focused on two major fields which both are tightly connected to advanced optimization methods:

1. *Learning of graphical models*: Recent advances in structure learning are in finding structures satisfying some optimality conditions. This is in contrast to search heuristics which in general do not provide any guarantees. Current research in learning the parameters goes beyond classical maximum likelihood parameter learning which belongs to *generative learning*. Popular objectives are the conditional likelihood or a probabilistic formulation of the margin. Optimizing these objectives can lead to better classification performance, particularly when the class conditional distributions

---

<sup>1</sup>In Bayesian statistics, learning and inference are the same, i.e., learning is inference of the parameters and structure.

poorly approximate the true distribution [4]. This is often referred to as *discriminative learning* with the aim of optimizing the model for one inference scenario, namely classification.

2. *Efficient inference:* Inference means answering queries using the PGM. In more technical language, inference deals with determining the marginal or most likely configuration of random variables given observed variables using the joint distribution modeled as PGM. Generally, exact inference is not tractable in large and dense graphical models and, therefore, approximate inference techniques are needed. Over the past decades, tractable approximate inference has been one of the most challenging and active research fields.

This tutorial is organized as follows: In Section 1.18.3 we present three types of representations, namely BNs, MNs, and FGs. Then in Sections 1.18.4 and 1.18.5 different learning approaches and inference methods are discussed. In Section 1.18.6, we present typical examples for each of the three graphical model representations. Sections 1.18.7 and 1.18.8 provide references to implementations and data sets. Finally, Section 1.18.9 concludes the tutorial providing a brief perspective on advanced methods and future trends.

In this tutorial article we presume that the reader is familiar with elementary probability calculus and fundamental linear algebra concepts. This introduction to probabilistic graphical models is necessarily incomplete due to the vast amount of methods developed over the last decades. Nevertheless, we hope to provide the reader with an easily accessible and interesting introduction to the field of PGMs. For additional details on any of the covered subjects the interested reader is referred to the cited literature.

## 1.18.2 Preliminaries

In this section we introduce parts of the notation used throughout this tutorial. Further, we review the basics of probability theory and present elementary definitions from graph theory.

### 1.18.2.1 Probability theory

In this section we present a short review of the most important concepts from probability theory. For a more detailed introduction we refer the reader to [1,5,6]. We start with some basic definitions.

**Definition 1 (Outcome Space, Elementary Events).** An *outcome space*  $\Omega$  is a non-empty set. The elements of  $\Omega$  are called *elementary events*.

**Example 1 (Outcome Space, Elementary Events).** We can define an outcome space of a die game according to  $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\}$ . The elementary event  $\omega_i$  denotes the outcome “die shows number  $i$ .”

We further define the notion of an event space.

**Definition 2 (Event Space, Events).** Given an outcome space  $\Omega$ , an *event space*  $\Sigma$  over  $\Omega$  is a set of subsets of  $\Omega$ , satisfying the following properties:

- $\emptyset \in \Sigma$  and  $\Omega \in \Sigma$ .
- If  $\sigma_1 \in \Sigma$  and  $\sigma_2 \in \Sigma$ , then also  $\sigma_1 \cup \sigma_2 \in \Sigma$ .

- If  $\sigma \in \Sigma$ , then also  $\Omega \setminus \sigma \in \Sigma$ .

The elements of  $\Sigma$  are called *events*.  $\Omega$  is called the *certain event* and  $\emptyset$  is the *impossible event*.

**Example 2 (Event Space, Events).** Given the outcome space of the die game  $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\}$  of Example 1, we give three examples of possible event spaces:

- The event space  $\Sigma_1 = \{\emptyset, \{\omega_1, \omega_3, \omega_5\}, \{\omega_2, \omega_4, \omega_6\}, \Omega\}$  contains the impossible event, the certain event, and the two events “outcome is odd” and “outcome is even.”
- The event space  $\Sigma_2 = \{\emptyset, \{\omega_1, \omega_2, \omega_3\}, \{\omega_4, \omega_5, \omega_6\}, \Omega\}$  contains the impossible event, the certain event, and the two events “outcome is smaller or equal 3” and “outcome is larger than 3.”
- The event space  $\Sigma_3 = 2^\Omega$ , where  $2^\Omega$  is the power set of  $\Omega$ , contains all subsets of  $\Omega$ , i.e., all possible combinations of elementary events.

We are now ready to define probability distributions.

**Definition 3 (Probability Distribution, Probability Space).** Let  $\Omega$  be an outcome space and  $\Sigma$  an event space over  $\Omega$ . A *probability distribution* over  $(\Omega, \Sigma)$  is a function  $P : \Sigma \mapsto \mathbb{R}$  satisfying the following properties:

- $P(\sigma) \geq 0, \forall \sigma \in \Sigma$ .
- $P(\Omega) = 1$ .
- If  $\sigma_i \cap \sigma_j = \emptyset$ , then  $P(\sigma_i \cup \sigma_j) = P(\sigma_i) + P(\sigma_j), \forall \sigma_i, \sigma_j \in \Sigma, i \neq j$ .

The triplet  $(\Omega, \Sigma, P)$  is called a *probability space*.

**Example 3 (Probability Distribution).** Let  $\Sigma_3 = 2^\Omega$  be the event space from Example 2. We can define a probability distribution by setting  $P(\omega_1) = P(\omega_2) = P(\omega_3) = P(\omega_4) = P(\omega_5) = P(\omega_6) = \frac{1}{6}$ , i.e., we assume a fair die. Since  $P$  has to be a probability distribution, it follows that  $P(\sigma) = \frac{|\sigma|}{6}, \forall \sigma \in \Sigma_3$ , where  $|\sigma|$  denotes the cardinality of  $\sigma$ .

Throughout the article, we use events and probability distributions defined via random variables, which are characterized by their cumulative distribution functions.

**Definition 4 (Random Variable, Cumulative Distribution Function (CDF)).** Let  $(\Omega, \Sigma, P)$  be a probability space and  $\mathbb{S}$  a measurable space (in this tutorial we assume  $\mathbb{S} = \mathbb{R}$ ). A *random variable* (RV)  $X$  is a function  $X : \Omega \mapsto \mathbb{S}$ , where “ $X \leq x$ ” =  $\{\omega \in \Omega : X(\omega) \leq x\}$  is an event for every  $x \in \mathbb{S}$ , and for the events “ $X = -\infty$ ” =  $\{\omega \in \Omega : X(\omega) = -\infty\}$  and “ $X = \infty$ ” =  $\{\omega \in \Omega : X(\omega) = \infty\}$  it must hold that  $P(X = -\infty) = P(X = \infty) = 0$ . The probability of the event “ $X \leq x$ ” is the cumulative distribution function (CDF)

$$F_X(x) = P(X \leq x). \quad (18.1)$$

More generally, let  $\mathbf{X} = \{X_1, X_2, \dots, X_N\}$  be an ordered set of  $N$  RVs and  $\mathbf{x}$  a corresponding set of  $N$  values from  $\mathbb{S}$ . The *joint CDF* of  $\mathbf{X}$  is defined as

$$F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1 \cap X_2 \leq x_2 \cap \dots \cap X_N \leq x_N). \quad (18.2)$$

The image of an RV  $X$  is denoted as  $\text{val}(X) = \{x \in \mathbb{S} | \exists \omega \in \Omega : X(\omega) = x\}$ . We say that  $X$  takes values out of  $\text{val}(X)$ . Similarly, the set of values which can be assumed by a set of random variables  $\mathbf{X}$  is denoted as  $\text{val}(\mathbf{X})$ .

When we are only interested in events connected with RVs, the CDF fully characterizes the underlying probability distribution. If the CDF of an RV  $X$  is continuous, then we say that  $X$  is a *continuous* RV. For continuous RVs we define the probability density function.

**Definition 5 (Probability Density Function (PDF)).** Let  $F_X(x)$  be the CDF of a continuous RV  $X$ . The *probability density function* (PDF) of  $X$  is defined as

$$P_X(x) = \frac{dF_X(x)}{dx}. \quad (18.3)$$

If  $P_X(x)$  exists, then the CDF of  $X$  is given as

$$F_X(x) = \int_{-\infty}^x P_X(x) dx. \quad (18.4)$$

More generally, let  $F_{\mathbf{X}}(\mathbf{x})$  be the CDF of a set of continuous RVs  $\mathbf{X}$ . The *joint* PDF of  $\mathbf{X}$  is defined as

$$P_{\mathbf{X}}(\mathbf{x}) = \frac{\partial^N F_{\mathbf{X}}(\mathbf{x})}{\partial x_1 \dots \partial x_N}. \quad (18.5)$$

If  $P_{\mathbf{X}}(\mathbf{x})$  exists, then the CDF of  $\mathbf{X}$  is given as

$$F_{\mathbf{X}}(\mathbf{x}) = \int_{-\infty}^{x_1} \dots \int_{-\infty}^{x_N} P_{\mathbf{X}}(\mathbf{x}) dx_1 \dots dx_N. \quad (18.6)$$

The PDF, if it exists, is an equivalent representation of the CDF, i.e., the PDF also represents the underlying probability distribution. It follows from the definition of probability distributions, that a PDF is nonnegative and integrates to one, i.e.,  $P_{\mathbf{X}}(\mathbf{x}) \geq 0, \forall \mathbf{x}$ , and  $\int P_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = 1$ .

A RV is called *discrete* when the CDF of  $X$  is piecewise constant and has a finite number of discontinuities. For discrete RVs we define the probability mass function.

**Definition 6 (Probability Mass Function (PMF)).** Let  $X$  be a discrete RV. The probability mass function (PMF) of  $X$  is a function  $P_X : \text{val}(X) \mapsto \mathbb{R}$  and defined as

$$P_X(x) = P(X = x), x \in \text{val}(X). \quad (18.7)$$

More generally, let  $\mathbf{X}$  be a set of discrete RVs. The *joint* PMF of  $\mathbf{X}$  is a function  $P_{\mathbf{X}} : \text{val}(\mathbf{X}) \mapsto \mathbb{R}$  and defined as

$$P_{\mathbf{X}}(\mathbf{x}) = P(X_1 = x_1 \cap X_2 = x_2 \cap \dots \cap X_N = x_N), \mathbf{x} = \{x_1, x_2, \dots, x_N\} \in \text{val}(\mathbf{X}). \quad (18.8)$$

It follows from the definition of probability distributions, that a PMF is nonnegative and sums to one, i.e.,  $P_{\mathbf{X}}(\mathbf{x}) \geq 0, \forall \mathbf{x} \in \text{val}(\mathbf{X})$ , and  $\sum_{\mathbf{x} \in \text{val}(\mathbf{X})} P_{\mathbf{X}}(\mathbf{x}) = 1$ . Unlike the PDF, a PMF always exists in the case of discrete RVs.

We use the same symbol  $P_{\mathbf{X}}(\mathbf{x})$  to denote PDFs and PMFs throughout the manuscript, since most of the discussion holds for both objects. From now on, we assume that  $P_{\mathbf{X}}(\mathbf{x})$  exists and fully represents the underlying probability distribution, and with slight abuse of notation, we refer to  $P_{\mathbf{X}}(\mathbf{x})$  itself as

distribution. Usually we omit the subscripts of PDFs/PMFs, i.e., we use the shorthand  $P(\mathbf{x}) = P_{\mathbf{X}}(\mathbf{x})$ . Furthermore, we use the notation  $P(\mathbf{X})$  for RVs,  $P(\mathbf{x})$  for instantiations  $\mathbf{x}$  of RVs  $\mathbf{X}$ ,  $P(\mathbf{X} \cup \mathbf{Y}) = P(\mathbf{X}, \mathbf{Y})$ , and  $P(\{X\} \cup \mathbf{Y}) = P(X, \mathbf{Y})$ . Throughout this tutorial, we assume that  $\mathbf{X} = \{X_1, \dots, X_N\}$ . Unless stated otherwise, the variables  $X_1, \dots, X_N$  are assumed to be discrete and the number of possible states of variable  $X_i$  is denoted as  $\text{sp}(X_i) = |\text{val}(X_i)|$ . Similarly, for a set of RVs  $\mathbf{X}$ , the total number of possible assignments is given as

$$\text{sp}(\mathbf{X}) = \prod_{i=1}^N \text{sp}(X_i). \quad (18.9)$$

When  $\mathbf{Y}$  is a subset of  $\mathbf{X}$ , and  $\mathbf{x}$  is a set of values for  $\mathbf{X}$ , then  $\mathbf{x}(\mathbf{Y})$  denotes the set of values corresponding to  $\mathbf{Y}$ .

An important quantity of RVs is the expected value.

**Definition 7 (Expected Value).** Let  $X$  be an RV with distribution  $P(X)$ . The expected value of  $X$  is defined as

$$\mathbb{E}[X] = \begin{cases} \sum_{x \in \text{val}(X)} x P(x) & \text{if } X \text{ is discrete,} \\ \int_{\text{val}(X)} x P(x) dx & \text{if } X \text{ is continuous.} \end{cases} \quad (18.10)$$

More generally, the expected value of a function  $f : \text{val}(X) \mapsto \mathbb{R}$  is defined as

$$\mathbb{E}[f(X)] = \begin{cases} \sum_{x \in \text{val}(X)} f(x) P(x) & \text{if } X \text{ is discrete,} \\ \int_{\text{val}(X)} f(x) P(x) dx & \text{if } X \text{ is continuous.} \end{cases} \quad (18.11)$$

Similarly, we can define expectations over sets of RVs.

When the joint distribution  $P(\mathbf{X})$  is given, one is often interested in the distribution  $P(\mathbf{Y})$  of a subset  $\mathbf{Y} \subseteq \mathbf{X}$  of RVs. Also, one often considers the distributions of subsets  $\mathbf{Y} \subseteq \mathbf{X}$ , conditioned on the values of the remaining variables  $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$ .

**Definition 8 (Marginal Distribution, Conditional Distribution).** Let  $P(\mathbf{X})$  be a distribution over a set of RVs  $\mathbf{X}$ . Further let  $\mathbf{Y} \subseteq \mathbf{X}$ , and  $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$ , i.e.,  $\mathbf{X} = \mathbf{Y} \cup \mathbf{Z}$  and  $P(\mathbf{X}) = P(\mathbf{Y}, \mathbf{Z})$ . Assuming discrete RVs, the *marginal distribution*  $P(\mathbf{Y})$  over  $\mathbf{Y}$  is given as

$$P(\mathbf{Y}) = \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z}) = \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(\mathbf{Y}, \mathbf{z}), \quad (18.12)$$

and the *conditional distribution*  $P(\mathbf{Y}|\mathbf{Z})$  over  $\mathbf{Y}$ , conditioned on  $\mathbf{Z}$ , is given as

$$P(\mathbf{Y}|\mathbf{Z}) = \frac{P(\mathbf{Y}, \mathbf{Z})}{P(\mathbf{Z})} = \frac{P(\mathbf{Y}, \mathbf{Z})}{\sum_{\mathbf{y} \in \text{val}(\mathbf{Y})} P(\mathbf{y}, \mathbf{Z})}. \quad (18.13)$$

In the case of continuous RVs, summations are replaced by integration in a straightforward manner.

The definition of the marginal and conditional distribution leads to an important rule, which allows to invert the “direction of conditioning.” Using (18.13), it is easily seen that for two sets of discrete RVs

$\mathbf{Y}$  and  $\mathbf{Z}$  the following holds:

$$P(\mathbf{Y}, \mathbf{Z}) = P(\mathbf{Y}, \mathbf{Z}), \quad (18.14)$$

$$P(\mathbf{Y}|\mathbf{Z})P(\mathbf{Z}) = P(\mathbf{Z}|\mathbf{Y})P(\mathbf{Y}), \quad (18.15)$$

$$P(\mathbf{Y}|\mathbf{Z}) = \frac{P(\mathbf{Z}|\mathbf{Y})P(\mathbf{Y})}{P(\mathbf{Z})} = \frac{P(\mathbf{Z}|\mathbf{Y})P(\mathbf{Y})}{\sum_{\mathbf{y} \in \text{val}(\mathbf{Y})} P(\mathbf{Z}|\mathbf{y})P(\mathbf{y})}. \quad (18.16)$$

Equation (18.16) is known as the *Bayes' rule* or the rule of *inverse probability*. For continuous RVs, the summation is replaced by an integral.

From the conditional distribution follows the concept of statistical independence. Informally, two RVs  $X$  and  $Y$  are independent, if knowing the state of one variable, i.e., conditioning on this variable, does not change the distribution over the other variable. Formally, statistical independence and conditional statistical independence are defined as follows.

**Definition 9 (Statistical Independence, Conditional Statistical Independence).** Let  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{Z}$  be mutually disjoint sets of RVs.  $\mathbf{X}$  and  $\mathbf{Y}$  are *conditionally statistically independent* given  $\mathbf{Z}$ , annotated as  $\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}$ , if and only if the following equivalent conditions hold:

- $P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})$ .
- $P(\mathbf{Y}|\mathbf{X}, \mathbf{Z}) = P(\mathbf{Y}|\mathbf{Z})$ .
- $P(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z})$ .

For the special case  $\mathbf{Z} = \emptyset$ , we say that  $\mathbf{X}$  and  $\mathbf{Y}$  are *statistically independent*, annotated as  $\mathbf{X} \perp \mathbf{Y}$ .

Furthermore, using conditional distributions, there are many ways to factorize an arbitrary joint distribution  $P(\mathbf{X})$ , i.e., any joint probability over  $\mathbf{X} = \{X_1, \dots, X_N\}$  can be written as

$$P(\mathbf{X}) = P(X_N|X_{N-1}, X_{N-2}, \dots, X_1)P(X_{N-1}|X_{N-2}, \dots, X_1) \dots P(X_2|X_1)P(X_1) \quad (18.17)$$

$$= P(X_1) \prod_{i=2}^N P(X_i|X_{i-1}, \dots, X_1). \quad (18.18)$$

This is called the *chain rule of probability*. The chain rule holds for any permutation of the indexes of the RVs  $\mathbf{X}$ .

### 1.18.2.2 Graph theory

In the context of PGMs *graphs* are used to represent probability distributions in an intuitive and easily readable way. Formally, a graph is defined as follows:

**Definition 10 (Graph, Vertex, Edge).** A graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is a tuple consisting of a set of *vertices*, also denoted as *nodes*,  $\mathbf{X}$  and a set of *edges*  $\mathbf{E}$ .

In PGMs each RV corresponds to exactly one node, and vice versa. That is, there is a one-to-one relationship between RVs and nodes. Therefore, we use these terms equivalently.

The edges of the graph define specific relationships between these variables. Any two nodes  $X_i$  and  $X_j$ ,  $i \neq j$ , in a graph can be connected by either a *directed* or an *undirected* edge. An undirected edge between node  $i$  and  $j$  is denoted as  $(X_i - X_j)$ , and a directed edge from node  $X_i$  to  $X_j$  as  $(X_i \rightarrow X_j)$ .

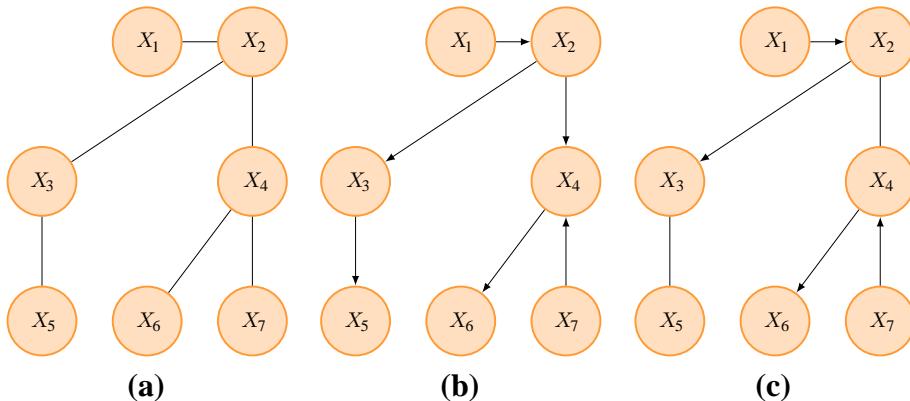


FIGURE 18.1

Graphical representation of different graph types. (a) Undirected graph. (b) Directed graph. (c) Mixed graph.

While an undirected edge is a symmetric relationship, i.e., the edge  $(X_i - X_j)$  is the same as the edge  $(X_j - X_i)$ , a directed edge represents an asymmetric relationship. Note that we do not allow these two types of edges to exist between any pair of nodes simultaneously and that there must not be an edge connecting some node with itself.

Depending on the types of edges in a graph we define the following three graph types:

**Definition 11 (Directed, Undirected, and Mixed Graph).** A graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is *directed (undirected)* if all edges  $e \in \mathbf{E}$  are directed (undirected). Further, a graph is said to be a *mixed graph* if the set of edges contains undirected and directed edges.

At this point we want to introduce a graphical representation of PGMs; every node is represented by a circle and edges are depicted as lines, in the case of undirected edges, or as arrows, in the case of directed edges, connecting the corresponding nodes.

**Example 4 (Graphical representation of graphs).** Consider Figure 18.1. For all shown graphs  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  the set of nodes is  $\mathbf{X} = \{X_1, \dots, X_7\}$ . The corresponding edge sets are

- a.  $\mathbf{E} = \{(X_1 - X_2), (X_2 - X_3), (X_2 - X_4), (X_3 - X_5), (X_4 - X_6), (X_4 - X_7)\}$ ,
- b.  $\mathbf{E} = \{(X_1 \rightarrow X_2), (X_2 \rightarrow X_3), (X_2 \rightarrow X_4), (X_3 \rightarrow X_5), (X_4 \rightarrow X_6), (X_7 \rightarrow X_4)\}$ , and
- c.  $\mathbf{E} = \{(X_1 \rightarrow X_2), (X_2 \rightarrow X_3), (X_2 - X_4), (X_3 - X_5), (X_4 \rightarrow X_6), (X_7 \rightarrow X_4)\}$ .

In graphs we can define relationships between variables. In directed and mixed graphs we introduce a parent–child relationship:

**Definition 12 (Parent and Child).** Let  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  be a directed or mixed graph and  $X_i, X_j \in \mathbf{X}$ ,  $i \neq j$ . If  $(X_i \rightarrow X_j) \in \mathbf{E}$ , then  $X_i$  is a *parent* of  $X_j$  and  $X_j$  is a *child* of  $X_i$ . The set  $\mathbf{Pa}_{\mathcal{G}}(X_j)$  consists of all parents of  $X_j$  and the set  $\mathbf{Ch}_{\mathcal{G}}(X_i)$  contains all children of  $X_i$ .

**Example 5 (Parents and Children).** In Figure 18.1b  $X_3$  is a child of  $X_2$  and  $X_2$  is a parent of  $X_3$ . Further, the set  $\text{Pa}_{\mathcal{G}}(X_3) = \{X_2\}$ , as the only parent of  $X_3$  is  $X_2$ . The set of all children of  $X_2$  is  $\text{Ch}_{\mathcal{G}}(X_2) = \{X_3, X_4\}$ .

For undirected and mixed graphs we define the notion of neighborhood:

**Definition 13 (Neighbor).** Let  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  be an undirected or mixed graph and  $X_i, X_j \in \mathbf{X}, i \neq j$ . If  $(X_i - X_j) \in \mathbf{E}$ , then  $X_i$  is said to be a neighbor of  $X_j$ . The set of all neighbors of  $X_i$  is denoted as  $\text{Nb}_{\mathcal{G}}(X_i)$ , i.e.,

$$\text{Nb}_{\mathcal{G}}(X_i) = \{X_j : (X_i - X_j) \in \mathbf{E}, X_j \in \mathbf{X}\}. \quad (18.19)$$

**Example 6 (Neighborhood).** In Figure 18.1a  $X_1, X_3$  and  $X_4$  are neighbors of  $X_2$ . As these are all neighbors of  $X_2$  in the graph,

$$\text{Nb}_{\mathcal{G}}(X_2) = \{X_1, X_3, X_4\}. \quad (18.20)$$

While edges represent *direct* relationships between two nodes, the notion of *paths* and *trails* describes *indirect* relationships across several nodes:

**Definition 14 (Path, Directed Path, Trail).** Let  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  be a graph. A sequence of nodes  $Q = (X_1, \dots, X_n)$ , with  $X_1, \dots, X_n \in \mathbf{X}$  is a *path* from  $X_1$  to  $X_n$  if

$$\forall i \in \{1, \dots, n-1\} : (X_i \rightarrow X_{i+1}) \in \mathbf{E} \vee (X_i - X_{i+1}) \in \mathbf{E}. \quad (18.21)$$

A path is *directed* if

$$\exists i \in \{1, \dots, n-1\} : (X_i \rightarrow X_{i+1}) \in \mathbf{E}, \quad (18.22)$$

i.e., if there is at least one directed edge on the path.

The sequence of nodes  $Q$  is a *trail* if

$$\forall i \in \{1, \dots, n-1\} : (X_i \rightarrow X_{i+1}) \in \mathbf{E} \vee (X_{i+1} \rightarrow X_i) \in \mathbf{E} \vee (X_i - X_{i+1}) \in \mathbf{E}, \quad (18.23)$$

i.e., if  $Q$  is a path from  $X_1$  to  $X_n$  replacing all directed edges by undirected edges.

**Example 7 (Paths).** Again, consider Figure 18.1.

- a. In Figure 18.1a, the sequence  $(X_2, X_4, X_6)$  is a path from  $X_2$  to  $X_6$ , and there exists a path between all pairs of distinct nodes.
- b. In Figure 18.1b,  $(X_1, X_2, X_4)$  is a directed path from  $X_1$  to  $X_4$ , but there does not exist a path from  $X_6$  to  $X_7$  because of the directed edges. But,  $X_6$  and  $X_7$  are connected by the trail  $(X_6, X_4, X_7)$ .
- c. In Figure 18.1c, the sequence of nodes  $(X_7, X_4, X_2)$  is a directed path, but again, there does not exist a path from  $X_6$  to  $X_7$ .

An important class of undirected graphs are *trees*:

**Definition 15 (Tree).** An undirected graph  $\mathcal{G}$  is a *tree* if any two distinct nodes are connected by exactly one path.

**Example 8 (Tree).** Figure 18.1a represents a tree.

In directed graphs we can identify parts of the graph that appear *before* and *after* a certain node:

**Definition 16 (Ancestor, Descendant).** In a directed graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  with  $X_i, X_j \in \mathbf{X}$ ,  $i \neq j$ , the node  $X_i$  is an *ancestor* of  $X_j$  if there exists a directed path from  $X_i$  to  $X_j$ . In this case,  $X_j$  is also denoted as a *descendant* of  $X_i$ .

The set of all ancestors of some node  $X_j \in \mathbf{X}$  is denoted as  $\text{Anc}_{\mathcal{G}}(X_j)$ , i.e.,

$$\text{Anc}_{\mathcal{G}}(X_j) = \{X_i | X_i \text{ is an ancestor of } X_j\}. \quad (18.24)$$

Similarly, the set of all descendants of some node  $X_i \in \mathbf{X}$  is denoted as  $\text{Desc}_{\mathcal{G}}(X_i)$ , i.e.,

$$\text{Desc}_{\mathcal{G}}(X_i) = \{X_j | X_j \text{ is a descendant of } X_i\}. \quad (18.25)$$

We further define the nondescendants of  $X_i$  as

$$\text{NonDesc}_{\mathcal{G}}(X_i) = \mathbf{X} \setminus (\text{Desc}_{\mathcal{G}}(X_i) \cup \{X_i\} \cup \text{Pa}_{\mathcal{G}}(X_i)), \quad (18.26)$$

i.e., the nondescendants of  $X_i$  are all nodes in the graph that are neither descendants of  $X_i$ , nor the node  $X_i$  itself or the parents  $\text{Pa}_{\mathcal{G}}(X_i)$  of this node.

A special type of paths are *directed cycles*:

**Definition 17 (Directed cycle).** Let  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  be a graph,  $X_i \in \mathbf{X}$ , and let  $P$  be a directed path from  $X_i$  to  $X_i$ . Then  $P$  is denoted as a *directed cycle*.

Using these notion, we can define an important type of directed graphs:

**Definition 18 (Directed acyclic graphs).** A graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is a *directed acyclic graph* (DAG) if it contains no directed cycles.

**Example 9 (DAG).** An example of a DAG is given in Figure 18.1b.

Sometimes, we will consider graphs that are *contained* in a *larger* graph:

**Definition 19 (Subgraph, Supergraph).** The graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is a *subgraph* of  $\mathcal{G}' = (\mathbf{X}', \mathbf{E}')$ , if and only if  $\mathbf{X} \subseteq \mathbf{X}'$  and  $\mathbf{E} \subseteq \mathbf{E}'$ . Further,  $\mathcal{G}'$  is a *supergraph* of  $\mathcal{G}$  if and only if  $\mathcal{G}$  is a subgraph of  $\mathcal{G}'$ .

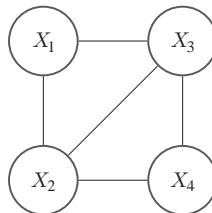
For some of our considerations we need to identify special subsets of the nodes of an undirected graph:

**Definition 20 (Clique, Maximal Clique).** In an undirected graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  a set of nodes  $\mathbf{C} \subseteq \mathbf{X}$  is a *clique* if there exists an edge between all pairs of nodes in the subset  $\mathbf{C}$ , i.e., if

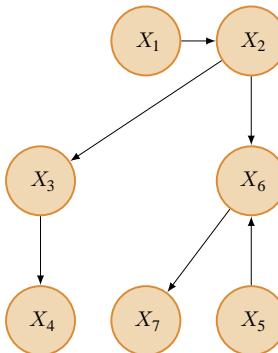
$$\forall C_i, C_j \in \mathbf{C}, i \neq j : (C_i - C_j) \in \mathbf{E}. \quad (18.27)$$

The clique  $\mathbf{C}$  is a *maximal clique* if adding any node  $X \in \mathbf{X} \setminus \mathbf{C}$  makes it no longer a clique.

**Example 10 (Cliques, Maximal Cliques).** Consider Figure 18.2. The set of nodes  $\{X_2, X_3, X_4\}$  is a maximal clique as  $\{X_1, X_2, X_3, X_4\}$  is no clique. The set  $\{X_1, X_3\}$  is a clique that is not maximal as  $\{X_1, X_2, X_3\}$  is also a clique.

**FIGURE 18.2**

Clique and maximal clique.

**FIGURE 18.3**

Topologically ordered graph.

The notion of cliques is closely related to that of *complete* (sub) graphs:

**Definition 21 (Complete Graph).** The undirected graph  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is *complete* if every pair of distinct nodes is connected by an edge, i.e., if

$$\forall X_i, X_j \in \mathbf{X}, i \neq j : (X_i - X_j) \in \mathbf{E}. \quad (18.28)$$

The set  $\mathbf{C} \subseteq \mathbf{X}$  is a clique if the *induced* graph  $\mathcal{G}' = (\mathbf{C}, \mathbf{E}')$  is complete, where

$$\mathbf{E}' = \{(X_i - X_j) \in \mathbf{E} : X_i, X_j \in \mathbf{C}, i \neq j\}. \quad (18.29)$$

At the end of this section, we introduce the notion of a topological ordering of the nodes in a directed graph:

**Definition 22 (Topological Ordering).** Let  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  be a directed graph and  $\mathbf{X} = \{X_1, \dots, X_N\}$ . The nodes  $\mathbf{X}$  are *topologically ordered* if for any edge  $(X_i \rightarrow X_j) \in \mathbf{E}$  also  $i \leq j$ .

**Example 11.** The nodes of the graph in Figure 18.3 are topologically ordered, while the nodes of the graph in Figure 18.1b are not.

### 1.18.3 Representations

In this section we introduce three types of PGMs, namely *Bayesian networks* in Section 1.18.3.1, *Markov networks* in Section 1.18.3.2 and *factor graphs* in Section 1.18.3.3. Each type has its specific advantages and disadvantages, captures different aspects of probabilistic models, and is preferred in a certain application domain. Finally, we conclude the section by an example demonstrating how Markov chains can be represented in each of the three types of PGMs.

#### 1.18.3.1 Bayesian networks (BNs)

Bayesian networks are directed graphical models. A *dynamic BN* is a BN unrolled over time. This essentially means that at each time slice the BN has the same structure [7]. There are two equivalent ways to define BNs, namely (a) by *factorization properties*, or by *conditional independencies*. We introduce BNs by their factorization properties and show that this is equivalent to a set of conditional independence assumptions in Section 1.18.3.1.2.

##### 1.18.3.1.1 Definition and factorization properties

A BN is defined as follows:

**Definition 23 (Bayesian Network).** A *Bayesian network*  $\mathcal{B}$  is a tuple  $(\mathcal{G}, \{P_{X_1}, \dots, P_{X_N}\})$ , where  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is a DAG, each node  $X_i$  corresponds to an RV, and  $P_{X_1}, \dots, P_{X_N}$  are conditional probability distributions (CPDs) associated with the nodes of the graph. Each of these CPDs has the form

$$P_{X_i}(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i)). \quad (18.30)$$

The BN  $\mathcal{B}$  defines the joint probability distribution  $P_{\mathcal{B}}(X_1, \dots, X_N)$  according to

$$P_{\mathcal{B}}(X_1, \dots, X_N) = \prod_{i=1}^N P_{X_i}(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i)). \quad (18.31)$$

We will use the shorthand notation  $P(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i))$  for  $P_{X_i}(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i))$  when the meaning is clear from the context. Usually each conditional probability distribution  $P_{X_i}$  is specified by a parameter set  $\theta^i$ , and we collect all the parameters into the set  $\Theta = \{\theta^1, \dots, \theta^N\}$  and use the terms  $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \dots, P_{X_N}\})$  and  $\mathcal{B} = (\mathcal{G}, \Theta)$  equivalently. Similarly, in the case of parameterized distributions, we write  $P(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i), \theta^i)$  for  $P_{X_i}(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i))$ .

**Example 12.** Consider the BN in Figure 18.3. The joint probability distribution  $P_{\mathcal{B}}(X_1, \dots, X_7)$  factorizes according to the graph structure as

$$P_{\mathcal{B}}(X_1, \dots, X_7) = \prod_{i=1}^7 P(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i)) \quad (18.32)$$

$$= P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)P(X_5)P(X_6|X_2, X_5)P(X_7|X_6) \quad (18.33)$$

Assuming discrete RVs with  $r = \text{sp}(X_i)$  states, a total of  $(r - 1) + (r - 1)r + (r - 1)r + (r - 1)r + (r - 1) + (r - 1)r^2 + (r - 1)r = 2(r - 1) + 4(r - 1)r + (r - 1)r^2$  parameters are required to represent  $P_{\mathcal{B}}(X_1, \dots, X_7)$ . In contrast, a non-factored representation of  $P_{\mathcal{B}}(X_1, \dots, X_7)$  requires  $r^7 - 1$  parameters. Hence, the representation of a joint probability distribution as BN can dramatically reduce the number of parameters. This is true particularly when the factorization of the joint distribution implies conditional independencies among the variables. The number of parameters for the conditional probability of  $X_i$  given its parents increases exponentially with the number of parents, i.e.,  $(\text{sp}(X_i) - 1)\text{sp}(\text{Pa}_{\mathcal{G}}(X_i))$ . The factorization of the joint probability distribution is beneficial for learning and inference as shown in Section 1.18.4 and Section 1.18.5, respectively.

In the sequel, we give two commonly used examples of parametric BNs, one with discrete nodes, and one with conditional Gaussian probabilities.

**Example 13 (Discrete Variables).** Assume a BN  $\mathcal{B}$  with discrete variables  $X_1, \dots, X_N$ , where each  $X_i$  takes one out of a finite set of states. For simplicity, we identify the states of  $X_i$  with natural numbers in  $\{1, \dots, \text{sp}(X_i)\}$ . In this case, the conditional distributions  $P(X_i | \text{Pa}_{\mathcal{G}}(X_i), \theta^i)$  are PMFs, parameterized by  $\theta^i$  such that

$$P(X_i = j | \text{Pa}_{\mathcal{G}}(X_i) = \mathbf{h}, \theta^i) = \theta_{j|\mathbf{h}}^i, \quad (18.34)$$

where  $\mathbf{h}$  is the instantiation of the parent variables  $\text{Pa}_{\mathcal{G}}(X_i)$  and  $j$  the instantiation of  $X_i$ . Note that the parameters  $\theta^i$  must satisfy

$$\theta_{j|\mathbf{h}}^i \geq 0, \quad \forall j \in \text{val}(X_i), \mathbf{h} \in \text{val}(\text{Pa}_{\mathcal{G}}(X_i)) \text{ and} \quad (18.35)$$

$$\sum_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i = 1, \quad \forall \mathbf{h} \in \text{val}(\text{Pa}_{\mathcal{G}}(X_i)), \quad (18.36)$$

in order to specify valid probability distributions. Therefore, the CPDs corresponding to the  $i$ th variable can be stored in a table with  $(\text{sp}(X_i) - 1) \cdot \text{sp}(\text{Pa}_{\mathcal{G}}(X_i))$  entries.

The BN distribution in Eq. (18.31) becomes

$$P_{\mathcal{B}}(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^N \prod_{j=1}^{\text{sp}(X_i)} \prod_{\mathbf{h} \in \text{val}(\text{Pa}_{\mathcal{G}}(X_i))} \left( \theta_{j|\mathbf{h}}^i \right)^{u_{j|\mathbf{h}}^i}, \quad (18.37)$$

where  $u_{j|\mathbf{h}}^i = \mathbb{1}_{\{x_i=j \text{ and } \mathbf{x}(\text{Pa}_{\mathcal{G}}(X_i))=\mathbf{h}\}}$ . The symbol  $\mathbb{1}_{\{A\}}$  denotes the indicator function which equals 1 if the statement  $A$  is true and 0 otherwise.

**Example 14 (Conditional Gaussian Variables).** In this example we assume that the variables of a BN  $\mathcal{B}$  are real-valued and that the CPDs are specified by Gaussian distributions, with the mean depending linearly on the value of the parents, and shared variance  $\sigma^2$ . That is, the CPD of node  $X_i$ , is given as

$$P(X_i | \text{Pa}_{\mathcal{G}}(X_i), \theta^i) = \mathcal{N} \left( X_i \left| \sum_{X_k \in \text{Pa}_{\mathcal{G}}(X_i)} \alpha_{i,k} x_k + \beta_i; \sigma^2 \right. \right), \quad (18.38)$$

where

$$\mathcal{N}(X|\mu; \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right) \quad (18.39)$$

denotes the Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ . Here, the parameter set  $\boldsymbol{\theta}^i = \{\boldsymbol{\alpha}_i, \beta_i\}$  contains the linear weights  $\boldsymbol{\alpha}_i = (\alpha_{i,1}, \dots, \alpha_{i,|\text{Pa}_{\mathcal{G}}(X_i)|})$  and the bias  $\beta_i$ . It can be shown that in this case the BN distribution in Eq. (18.31) is a multivariate Gaussian distribution, with mean vector  $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_N)$  and covariance matrix  $\boldsymbol{\Sigma}$ , which are recursively given as [4]:

$$\mu_i = \mathbb{E}[X_i] = \sum_{X_k \in \text{Pa}_{\mathcal{G}}(X_i)} \alpha_{i,k} \mu_k + \beta_i, \quad (18.40)$$

$$\begin{aligned} \Sigma_{i,j} &= \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])^T] \\ &= \sum_{X_k \in \text{Pa}_{\mathcal{G}}(X_j)} \alpha_{j,k} \Sigma_{i,k} + I_{i,j} \sigma^2, \end{aligned} \quad (18.41)$$

where  $I$  denotes the identity matrix,  $\Sigma_{i,j}$  the entry in  $\boldsymbol{\Sigma}$  in the  $i$ th row and  $j$ th column.

### 1.18.3.1.2 Conditional independence properties

BNs provide means for representing conditional independence properties of a probability distribution. Exploiting conditional independencies among variables supports efficient inference and reduces complexity. The factorization properties of the joint distribution of BNs as given in the last section imply a set of conditional independencies [1]:

**Theorem 1 (Local Conditional Independencies).** *Let  $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \dots, P_{X_N}\})$  be a BN and  $P_{\mathcal{B}}$  the joint probability defined by  $\mathcal{B}$ . Then,  $P_{\mathcal{B}}$  satisfies the local independencies*

$$X_i \perp \text{NonDesc}_{\mathcal{G}}(X_i) | \text{Pa}_{\mathcal{G}}(X_i) \quad \forall i, \quad (18.42)$$

i.e., any variable  $X_i$  is conditional independent of its nondescendants given its parents.

**Proof.** To prove the stated conditional independence property, it suffices to show that

$$P_{\mathcal{B}}(X_i | \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i)) = P_{\mathcal{B}}(X_i | \text{Pa}_{\mathcal{G}}(X_i)). \quad (18.43)$$

The conditional distribution  $P_{\mathcal{B}}(X_i | \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i))$  can be written as

$$P_{\mathcal{B}}(X_i | \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i)) = \frac{P_{\mathcal{B}}(X_i, \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i))}{P_{\mathcal{B}}(\text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i))}. \quad (18.44)$$

The numerator is

$$P_{\mathcal{B}}(X_i, \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i)) = \sum_{\text{Desc}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_i, \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i), \text{Desc}_{\mathcal{G}}(X_i)) \quad (18.45)$$

$$= \sum_{\text{Desc}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_i | \text{Pa}_{\mathcal{G}}(X_i)) \prod_{X_j \in \text{Desc}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_j | \text{Pa}_{\mathcal{G}}(X_j)) \cdot \\ \prod_{X_k \in \text{NonDesc}_{\mathcal{G}}(X_i) \cup \text{Pa}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_k | \text{Pa}_{\mathcal{G}}(X_k)) \quad (18.46)$$

$$= P_{\mathcal{B}}(X_i | \text{Pa}_{\mathcal{G}}(X_i)) \underbrace{\prod_{X_k \in \text{NonDesc}_{\mathcal{G}}(X_i) \cup \text{Pa}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_k | \text{Pa}_{\mathcal{G}}(X_k))}_{=A} \cdot \\ \underbrace{\sum_{\text{Desc}_{\mathcal{G}}(X_i)} \prod_{X_j \in \text{Desc}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_j | \text{Pa}_{\mathcal{G}}(X_j))}_{=1} \quad (18.47)$$

In the same way, the denominator is given as

$$P_{\mathcal{B}}(\text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i)) = \underbrace{\prod_{X_k \in \text{NonDesc}_{\mathcal{G}}(X_i) \cup \text{Pa}_{\mathcal{G}}(X_i)} P_{\mathcal{B}}(X_k | \text{Pa}_{\mathcal{G}}(X_k))}_{=A} \quad (18.48)$$

Consequently, we obtain

$$P_{\mathcal{B}}(X_i | \text{NonDesc}_{\mathcal{G}}(X_i), \text{Pa}_{\mathcal{G}}(X_i)) = P_{\mathcal{B}}(X_i | \text{Pa}_{\mathcal{G}}(X_i)) \quad (18.49)$$

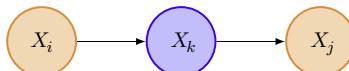
and the desired statement follows.  $\square$

Theorem 1 could have also been used as a basis for the definition of BNs, as, without proof, the local conditional independence assumptions from the Theorem imply the factorization properties of  $P_{\mathcal{B}}$  given in Eq. (18.31).

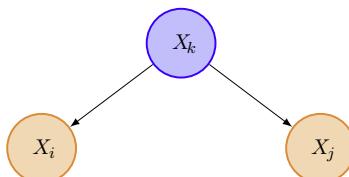
Additionally to the local conditional independencies, several other conditional independencies hold in BNs [1]. Some of them can be read off the graph by the concept of *d-separation*. Before turning to *d-separation*, we shortly introduce the notion of *context-specific independence*. In BNs with discrete variables the conditional distributions  $P(X_i | \text{Pa}_{\mathcal{G}}(X_i), \theta^i)$  are represented as conditional probability tables (CPTs). Regularities in these CPTs can render RV  $X_i$  independent of some parents conditioned on specific values of other parent variables. An example of context-specific independence is given in [1,8].

**Definition 24 (*d-separation* [2]).** Let  $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \dots, P_{X_N}\})$  be a BN,  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ , and  $\mathbf{X} = \{X_1, \dots, X_N\}$ . Two RVs  $X_i$  and  $X_j$ ,  $i \neq j$ , are *d-separated* if for all trails between  $X_i$  and  $X_j$  there is an intermediate variable  $X_k$ ,  $i \neq k, j \neq k$ , such that

- the connection is *serial* or *diverging* and the state of  $X_k$  is observed, or

**FIGURE 18.4**

Serial connection.

**FIGURE 18.5**

Diverging connection.

- the connection is *converging* and neither the state of  $X_k$  nor the state of any descendant of  $X_k$  is observed,

where the term connection means the path corresponding to the trail.

We now introduce three simple graphs illustrating the underlying concept of the definition, i.e., the different connection types and the thereby implied conditional independencies.

*Serial connection:* A BN  $\mathcal{B}$  showing a so called *serial connection* is presented in Figure 18.4. According to Eq. (18.31) the corresponding joint distribution is given as

$$P_{\mathcal{B}}(X_i, X_k, X_j) = P(X_i)P(X_k|X_i)P(X_j|X_k). \quad (18.50)$$

We now condition the joint probability on  $X_k$ , i.e., we assume  $X_k$  to be observed. Then, by probability calculus,

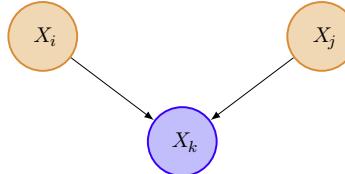
$$P_{\mathcal{B}}(X_i, X_j|X_k) = \overbrace{\frac{P(X_i)P(X_k|X_i)}{P_{\mathcal{B}}(X_k)}}^{=P_{\mathcal{B}}(X_i, X_k)} P(X_j|X_k), \quad (18.51)$$

$$= P_{\mathcal{B}}(X_i|X_k)P(X_j|X_k), \quad (18.52)$$

where  $P_{\mathcal{B}}(X_i, X_j|X_k)$ ,  $P_{\mathcal{B}}(X_i|X_k)$ , and  $P_{\mathcal{B}}(X_j|X_k)$  are conditional and marginal distributions resulting from the joint distribution  $P_{\mathcal{B}}(X_i, X_k, X_j)$ . That is,  $X_i$  and  $X_j$  are conditionally independent given  $X_k$ , i.e.,  $X_i \perp X_j|X_k$ . However, if  $X_k$  is not observed, the RVs  $X_i$  and  $X_j$  are dependent in general.

*Diverging connection:* A diverging connection is shown in Figure 18.5. Again, by exploiting the factorization properties of the BN according to (18.31), we obtain the joint probability

$$P_{\mathcal{B}}(X_i, X_k, X_j) = P(X_k)P(X_i|X_k)P(X_j|X_k). \quad (18.53)$$

**FIGURE 18.6**

Converging connection.

Conditioning this probability on  $X_k$  yields

$$P_{\mathcal{B}}(X_i, X_j | X_k) = \frac{\overbrace{P(X_k) P(X_i | X_k)}^{=P_{\mathcal{B}}(X_i, X_k)} P(X_j | X_k)}{P_{\mathcal{B}}(X_k)}, \quad (18.54)$$

$$= P_{\mathcal{B}}(X_i | X_k) P(X_j | X_k). \quad (18.55)$$

Hence,  $X_i \perp X_j | X_k$ . That is, when  $X_k$  is observed, then the variables  $X_i$  and  $X_j$  are conditionally independent given  $X_k$ , while, when  $X_k$  is not observed, they are dependent in general.

*Converging connection:* A converging connection is illustrated in Figure 18.6. The represented joint distribution is given as

$$P_{\mathcal{B}}(X_i, X_k, X_j) = P(X_i) P(X_j) P(X_k | X_i, X_j). \quad (18.56)$$

In contrast to the two cases before, the RVs  $X_i$  and  $X_j$  are a-priori independent since

$$\sum_{x_k \in \text{val}X_k} P(X_k = x_k | X_i, X_j) = 1. \quad (18.57)$$

That is

$$P_{\mathcal{B}}(X_i, X_j) = P(X_i) P(X_j), \quad (18.58)$$

which can be stated as  $X_i \perp X_j$ . To this end, consider the probability  $P_{\mathcal{B}}(X_i, X_j | X_k)$  which can be computed as

$$P_{\mathcal{B}}(X_i, X_j | X_k) = \frac{P(X_i) P(X_j) P(X_k | X_i, X_j)}{P_{\mathcal{B}}(X_k)}. \quad (18.59)$$

In general, this does not factorize in probabilities over  $X_i$  and  $X_j$ . Hence, the RVs  $X_i$  and  $X_j$  are independent if  $X_k$  is not observed but become dependent upon observation of variable  $X_k$  (or any descendant of  $X_k$ ). This is known as the *Berkson's paradox* [9] or the *explaining away phenomenon*.

Without proof, the  $d$ -separation Theorem expresses the conditional independence properties that follow from two sets of variables being  $d$ -separated (a proof is provided in [10]):

**Theorem 2 ( $d$ -separation [2]).** *Let  $\mathcal{B} = (\mathcal{G}, \{P_{X_1}, \dots, P_{X_N}\})$  be a BN and  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$ . Furthermore, let  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{D}$  be mutually disjoint subsets of  $\mathbf{X}$ , where the variables in  $\mathbf{D}$  are observed. If every pair of nodes  $A \in \mathbf{A}$  and  $B \in \mathbf{B}$  are  $d$ -separated, then*

$$\mathbf{A} \perp \mathbf{B} | \mathbf{D}, \quad (18.60)$$

that is  $\mathbf{A}$  and  $\mathbf{B}$  are conditionally independent given  $\mathbf{D}$ .

### 1.18.3.2 Markov networks (MNs)

*Markov networks*, also known as *Markov Random Fields* (MRFs) or *undirected graphical models*, are represented by undirected graphs. As in BNs, an MN encodes certain factorization and conditional independence properties of the joint probability distribution.

#### 1.18.3.2.1 Definition and factorization properties

**Definition 25 (Markov Network).** A *Markov network* is a tuple  $\mathcal{M} = (\mathcal{G}, \{\Psi_{\mathbf{C}_1}, \dots, \Psi_{\mathbf{C}_L}\})$ , where  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  is an undirected graph with maximal cliques  $\mathbf{C}_1, \dots, \mathbf{C}_L$ . The nodes  $\mathbf{X}$  correspond to RVs and  $\Psi_{\mathbf{C}_i} : \text{val}(\mathbf{C}_i) \rightarrow \mathbb{R}_+$  are nonnegative functions, called *factors* or *potentials*, over the maximal cliques of the graph  $\mathcal{G}$ .

The MN defines a probability distribution according to

$$P_{\mathcal{M}}(X_1, \dots, X_N) = \frac{1}{Z} \prod_{l=1}^L \Psi_{\mathbf{C}_l}(\mathbf{C}_l), \quad (18.61)$$

where  $Z$  is a normalization constant given as

$$Z = \sum_{\mathbf{x} \in \text{val}(\mathbf{X})} \prod_{l=1}^L \Psi_{\mathbf{C}_l}(\mathbf{x}(\mathbf{C}_l)). \quad (18.62)$$

Often,  $Z$  is also referred to as *partition function*.

**Example 15 (Markov Network).** Consider the MN shown in Figure 18.7. The maximal cliques in this graph are surrounded by dotted boxes and are given as

$$\mathbf{C}_1 = \{X_1, X_2, X_3\}, \quad (18.63)$$

$$\mathbf{C}_2 = \{X_3, X_4\}, \quad \text{and} \quad (18.64)$$

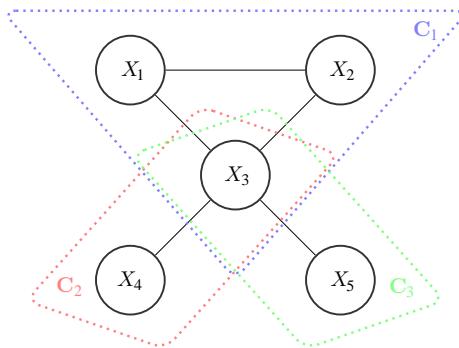
$$\mathbf{C}_3 = \{X_3, X_5\}. \quad (18.65)$$

Hence, the joint probability distribution of the represented model is

$$P_{\mathcal{M}}(X_1, \dots, X_5) = \frac{1}{Z} \Psi_{\mathbf{C}_1}(X_1, X_2, X_3) \Psi_{\mathbf{C}_2}(X_3, X_4) \Psi_{\mathbf{C}_3}(X_3, X_5). \quad (18.66)$$

#### 1.18.3.2.2 Conditional independence properties

Similarly as in BNs, MNs define a set of conditional independencies. In the following we present three conditional independence properties implied by MNs with strictly positive factors. If the factors are not strictly positive, the conditional independence properties stated below are not equivalent in general. Details regarding this are provided in [1].

**FIGURE 18.7**

Example of an MN.

The joint probability distributions represented by MNs with strictly positive factors satisfy the following equivalent conditional independencies:

- i. *Local Markov property*: Any RV  $X_i$  is conditionally independent from all other RVs given its neighbors. Formally,

$$X_i \perp (\mathbf{X} \setminus (\mathbf{Nb}_{\mathcal{G}}(X_i) \cup \{X_i\})) | \mathbf{Nb}_{\mathcal{G}}(X_i). \quad (18.67)$$

- ii. *Pairwise Markov property*: Any two non-adjacent RVs are conditionally independent given all other RVs, i.e.,

$$X_i \perp X_j | (\mathbf{X} \setminus \{X_i, X_j\}), \quad \text{if } (i - j) \notin \mathbf{E}. \quad (18.68)$$

- iii. *Global Markov property*: Let  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{D}$  be mutually disjoint subsets of  $\mathbf{X}$ , and the variables in  $\mathbf{D}$  are observed. Then,

$$\mathbf{A} \perp \mathbf{B} | \mathbf{D}, \quad (18.69)$$

if every path from any node in  $\mathbf{A}$  to any node in  $\mathbf{B}$  passes through  $\mathbf{D}$ . In this case, the set  $\mathbf{D}$  is also denoted as *separating subset*.

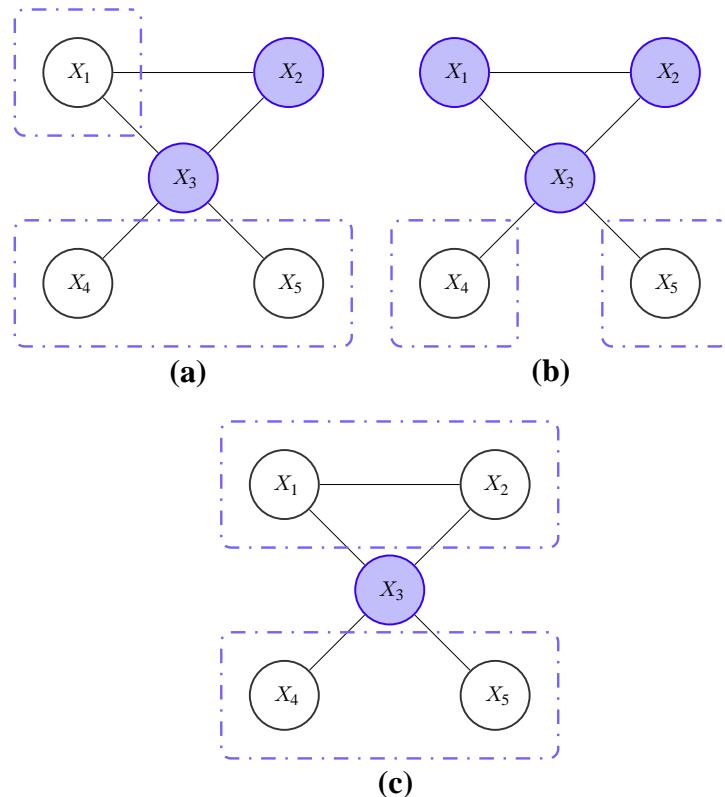
We illustrate these conditional independence statements in an example.

**Example 16 (Conditional Independencies in an MN).** Again, consider the MN shown in Figure 18.7. Then, for example,

- a. by the *local Markov property*,  $X_1$  is conditionally independent from the RVs  $X_4$  and  $X_5$  given its neighbors  $X_2$  and  $X_3$ , i.e.,

$$X_1 \perp \{X_4, X_5\} | \{X_2, X_3\}. \quad (18.70)$$

This is illustrated in Figure 18.8a.

**FIGURE 18.8**

Conditional independencies in MNs. (a) Local Markov property. (b) Pairwise Markov property. (c) Global Markov property.

- b.** by the *pairwise Markov property*,  $X_4$  and  $X_5$  are conditionally independent given  $X_1$ ,  $X_2$ , and  $X_3$ . Formally,

$$X_4 \perp X_5 | \{X_1, X_2, X_3\}. \quad (18.71)$$

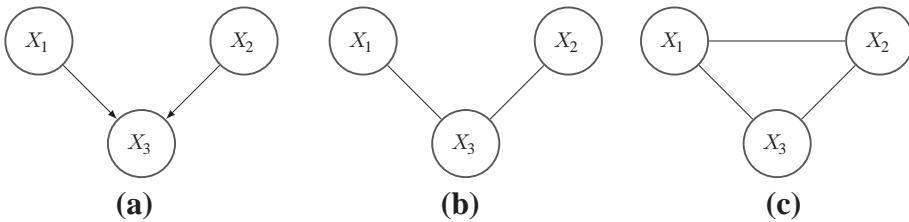
This is illustrated in Figure 18.8b.

- c.** by the *global Markov property*,  $X_1$  and  $X_2$  are conditionally independent of  $X_4$  and  $X_5$  given  $X_3$ , i.e.,

$$\{X_1, X_2\} \perp \{X_4, X_5\} | X_3. \quad (18.72)$$

This is illustrated in Figure 18.8c.

*Difference between BNs and MNs:* As introduced in the last sections, both BNs and MNs imply certain conditional independence statements. One might ask if any BN and its conditional independence

**FIGURE 18.9**

Representation of conditional independencies in BNs by MNs. (a) BN to be represented as an MN. (b) Candidate MN. (c) Another candidate MN.

properties can be represented by an MN and vice versa. Two simple examples demonstrate, that this is not the case:

**Example 17 (Conditional Independencies in BNs and MNs).** Consider the PGM with RVs \$X\_1\$, \$X\_2\$, and \$X\_3\$ represented by the BN in Figure 18.9a. We want to represent this BN by an MN that captures all the conditional independence properties of the BN:

- An appropriate MN must contain the edges \$(X\_1 - X\_3)\$ and \$(X\_2 - X\_3)\$. Otherwise, \$X\_3\$ would not depend on the state of \$X\_1\$ and \$X\_2\$ as in the BN. Such an MN is shown in Figure 18.9b. However, while in the BN in general \$X\_1 \not\perp\!\!\!\perp X\_2 | X\_3\$ the MN satisfies \$X\_1 \perp\!\!\!\perp X\_2 | X\_3\$ by the pairwise Markov property.
- Hence, we additionally need to add the edge \$(X\_1 - X\_2)\$ to the MN resulting in the MN represented in Figure 18.9c. However, now the MN does in general not satisfy that \$X\_1 \perp\!\!\!\perp X\_2\$ while the BN does.

Consequently, the conditional independencies of the BN in Figure 18.9a cannot be represented by an MN.

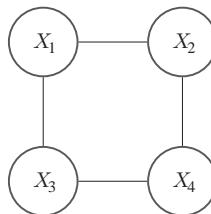
**Example 18 (Conditional Independencies in BNs and MNs).** Consider the MN in Figure 18.10. We want to represent this MN by a BN capturing all the independence properties that hold in the MN. By the pairwise Markov property \$X\_1\$ is conditionally independent from \$X\_4\$ given \$X\_2\$ and \$X\_3\$, and \$X\_2\$ is conditionally independent from \$X\_3\$ given \$X\_1\$ and \$X\_4\$, i.e.,

$$X_1 \perp\!\!\!\perp X_4 | \{X_2, X_3\}, \quad \text{and} \tag{18.73}$$

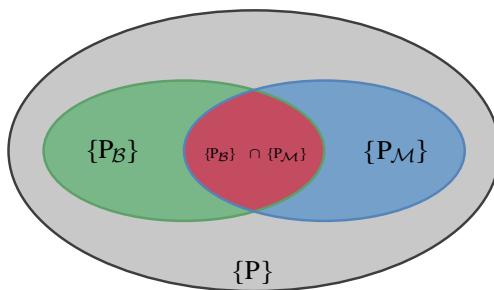
$$X_2 \perp\!\!\!\perp X_3 | \{X_1, X_4\}. \tag{18.74}$$

However, these conditional independencies cannot be encoded in a BN. The interested reader can find a proof for this in [1].

Nevertheless, there exist distributions which can be represented by both MNs and BNs. An example is the Markov chain, cf. Section 1.18.3.4. This can be illustrated graphically by the notion of *perfect maps*. A graph is a perfect map for a probability distribution if it represents all conditional independence properties of the distribution, and vice versa. Now, consider probability distributions over a given set of variables. Then, the set of probability distributions \$\{P\_B\}\$ for which some BN is a perfect map is a subset of all probability distributions \$\{P\}\$. Similarly, the set of probability distributions \$\{P\_M\}\$ for which some

**FIGURE 18.10**

Representation of conditional independencies in MNs by BNs.

**FIGURE 18.11**

Venn diagram depicting the set of all probability distributions  $\{P\}$  over a given set of variables, and the sets of probability distributions  $\{P_B\}$  and  $\{P_M\}$  for which some BN or some MN are a perfect map, respectively.

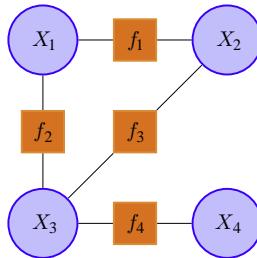
MN is a perfect map is also a subset of all probability distributions. Additionally, there are probability distributions  $\{P_B\} \cap \{P_M\}$  for which both BNs and MNs are perfect maps. This is illustrated by the Venn diagram in Figure 18.11 [4].

### 1.18.3.3 Factor graphs (FGs)

As last type of probabilistic model representation we introduce *factor graphs* (FGs) which explicitly represent the factorization properties of a function and have initially been used in the coding and image processing community.

#### 1.18.3.3.1 Definition

FGs are defined using the notion of bipartite graphs. These are graphs  $\mathcal{G} = (\mathbf{X}, \mathbf{E})$  for which the set of nodes can be divided into two disjoint subsets such that there are only edges between nodes from one subset to nodes of the other subset, as follows:

**FIGURE 18.12**

Example of an FG.

**Definition 26 (Factor Graph).** A *factor graph* is a tuple  $\mathcal{F} = (\mathcal{G}, \{f_1, \dots, f_L\})$ , where

- $\mathcal{G} = (\mathbf{V}, \mathbf{E})$  is an undirected bipartite graph such that  $\mathbf{V} = \mathbf{X} \cup \mathbf{F}$ , where  $\mathbf{X}$ , and  $\mathbf{F}$  are disjoint, and the nodes  $\mathbf{X} = \{X_1, \dots, X_N\}$  correspond to RVs. The nodes  $\mathbf{X}$  are *variable nodes*, while the nodes  $\mathbf{F}$  are *factor nodes*.
- Further,  $f_1, \dots, f_L$  are positive functions and the number of functions  $L$  equals the number of nodes in  $\mathbf{F} = \{F_1, \dots, F_L\}$ . Each node  $F_i$  is associated with the corresponding function  $f_i$  and each  $f_i$  is a function of the neighboring variables of  $F_i$ , i.e.,  $f_i = f_i(\mathbf{Nb}_{\mathcal{G}}(F_i))$ .

The factor graph  $\mathcal{F}$  defines the joint probability distribution

$$P_{\mathcal{F}}(X_1, \dots, X_N) = \frac{1}{Z} \prod_{l=1}^L f_l(\mathbf{Nb}_{\mathcal{G}}(F_l)), \quad (18.75)$$

where  $Z$  is the normalization constant given as

$$Z = \sum_{\mathbf{x} \in \text{val}(\mathbf{X})} \prod_{l=1}^L f_l(\mathbf{x}(\mathbf{Nb}_{\mathcal{G}}(F_l))). \quad (18.76)$$

As each factor node corresponds to a factor we use both terms equivalently. We illustrate the definition of FGs by the following example:

**Example 19 (Factor Graph).** Consider Figure 18.12. The given factor graph  $\mathcal{F}$  is  $\mathcal{F} = (\mathcal{G}, \{f_1, \dots, f_4\})$ , where the nodes of  $\mathcal{G}$  are the union of the set of variable nodes  $\{X_1, \dots, X_4\}$  and the set of factor nodes  $\{F_1, \dots, F_4\}$ . The functions corresponding to these factor nodes are  $f_1(X_1, X_2)$ ,  $f_2(X_1, X_3)$ ,  $f_3(X_2, X_3)$ , and  $f_4(X_3, X_4)$ . The represented joint probability density is

$$P_{\mathcal{F}}(X_1, \dots, X_4) = \frac{1}{Z} f_1(X_1, X_2) f_2(X_1, X_3) f_3(X_2, X_3) f_4(X_3, X_4), \quad (18.77)$$

where  $Z$  is the normalization constant as given in the definition of FGs.

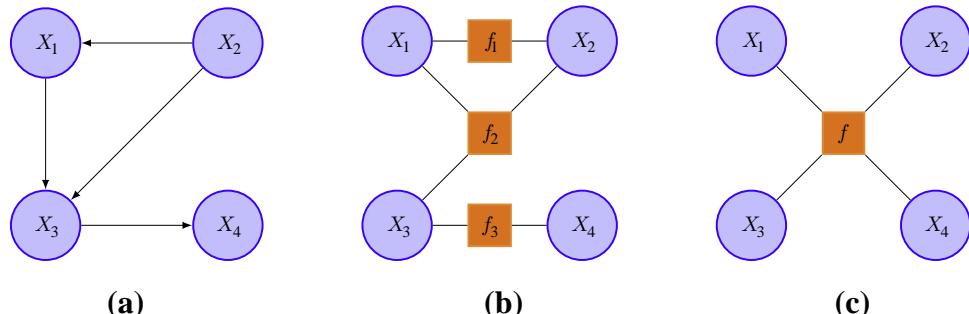


FIGURE 18.13

Conversion of a BN to an FG. (a) BN to be converted to an FG. (b) FG for representing the BN. (c) Alternative FG for representing the BN.

### **1.18.3.3.2 BNs and MNs as FGs**

FGs allow the representation of both BNs and MNs. However, note that there is in general no unique transformation from a BN/MN to an FG. We illustrate the transformation of a BN or MN to an FG in the following two examples:

**Example 20 (Conversion of a BN to an FG).** The BN shown in Figure 18.13a can be represented as the FG in Figure 18.13b where the factors  $f_1$ ,  $f_2$ ,  $f_3$  are given as

$$f_1(X_1, X_2) \equiv P_{X_2}(X_2)P_{X_1}(X_1|X_2), \quad (18.78)$$

$$f_2(X_1, X_2, X_3) \equiv P_{X_3}(X_3|X_1, X_2), \quad (18.79)$$

$$f_3(X_3, X_4) = P_{X_4}(X_4 | X_3). \quad (18.80)$$

In this way, the factor graph represents the probability distribution

$$P_{\mathcal{F}}(X_1, \dots, X_4) = f_1(X_1, X_2) f_2(X_1, X_2, X_3) f_3(X_3, X_4), \quad (18.81)$$

$$= P_{X_2}(X_2)P_{X_1}(X_1|X_2)P_{X_3}(X_3|X_1, X_2)P_{X_4}(X_4|X_3), \quad (18.82)$$

which is exactly the one given by the BN. Alternatively, the BN could also be represented by the FG shown in Figure 18.13c, where the factor  $f$  is given as

$$f(X_1, \dots, X_4) = P_{X_2}(X_2)P_{X_1}(X_1|X_2)P_{X_3}(X_3|X_1, X_2)P_{X_4}(X_4|X_3). \quad (18.83)$$

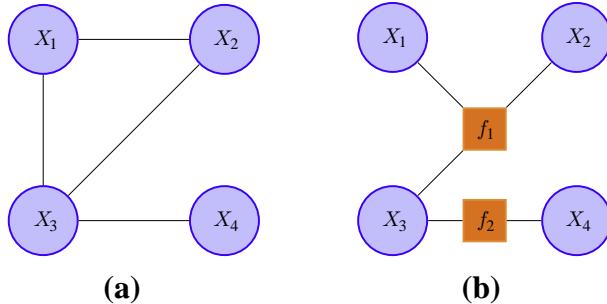
**Example 21 (Conversion of an MN to an FG).** The MN in Figure 18.14a represents the factorization

$$P_M(X_1, \dots, X_4) \equiv \Psi_{C^+}(X_1, X_2, X_3)\Psi_{C^+}(X_3, X_4). \quad (18.84)$$

where the maximal cliques are  $\mathbf{C}_1 = \{X_1, X_2, X_3\}$  and  $\mathbf{C}_2 = \{X_3, X_4\}$ , respectively. This factorization can be represented by the FG in Figure 18.14b, where

$$f_1(X_1, X_2, X_3) \equiv \Psi_{C_1}(X_1, X_2, X_3), \quad \text{and} \quad (18.85)$$

$$f_2(X_3, X_4) \equiv \Psi_{C_2}(X_3, X_4), \quad (18.86)$$

**FIGURE 18.14**

Conversion of an MN to an FG. (a) MN to be converted to an FG. (b) FG for representing the MN.

### 1.18.3.4 Markov chains

To conclude the section on PGM representations we consider Markov chains (MCs), and how they can be represented in each of the three types of PGMs. An MC is defined as follows:

**Definition 27 (Markov chain).** Let \$X\_1, \dots, X\_N\$ be a sequence of RVs. These RVs form a *Markov chain* if they satisfy

$$P(X_i | X_1, \dots, X_{i-1}) = P(X_i | X_{i-1}), \quad \forall i. \quad (18.87)$$

This is called *Markov property*.

In an MC the RVs preceding some given RV \$X\_i\$ are conditionally independent of all RVs succeeding \$X\_i\$, i.e.,

$$\{X_1, \dots, X_{i-1}\} \perp \{X_{i+1}, \dots, X_N\} | X_i. \quad (18.88)$$

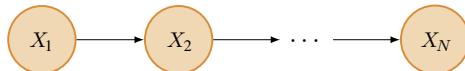
Furthermore, since any joint probability distribution \$P(X\_1, \dots, X\_N)\$ factorizes as

$$P(X_1, \dots, X_N) = \prod_{i=1}^N P(X_i | X_{i-1}, \dots, X_1), \quad (18.89)$$

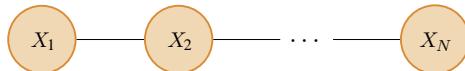
this simplifies due to the Markov property to

$$P(X_1, \dots, X_N) = \prod_{i=1}^N P(X_i | X_{i-1}). \quad (18.90)$$

We can represent MCs as BNs, MNs and FGs, as presented in the following:

**FIGURE 18.15**

BN modeling an MC.

**FIGURE 18.16**

MN modeling an MC.

- *Bayesian network:* A BN  $\mathcal{B}$  modeling an MC is shown in Figure 18.15. The represented joint distribution is

$$P_{\mathcal{B}}(X_1, \dots, X_N) = \prod_{i=1}^N P(X_i | \text{Pa}_{\mathcal{G}}(X_i)), \quad (18.91)$$

$$= P_{\mathcal{B}}(X_1, \dots, X_N) = P(X_1) \prod_{i=2}^N P(X_i | X_{i-1}), \quad (18.92)$$

where the only parent of  $X_i$  is  $X_{i-1}$ .

- *Markov network:* The MN  $\mathcal{M}$  for representing an MC in Figure 18.16 has the maximal cliques  $\mathbf{C}_i = \{X_i, X_{i+1}\}$  for  $i = 1, \dots, N - 1$ . If we define the factor  $\Psi_{\mathbf{C}_i}$  to represent the function  $P(X_{i+1}|X_i)$  for  $i = 2, \dots, N - 1$  and  $\Psi_{\mathbf{C}_1} = P(X_1)P(X_2|X_1)$ , then the MN specifies the joint probability distribution of an MC, i.e.,

$$P_{\mathcal{M}}(X_1, \dots, X_N) = \prod_{l=1}^{N-1} \Psi_{\mathbf{C}_l}(\mathbf{C}_l), \quad (18.93)$$

$$= P(X_1) \prod_{i=2}^N P(X_i | X_{i-1}). \quad (18.94)$$

- *Factor graph:* Finally, the FG  $\mathcal{F}$  in Figure 18.17 with factors  $f_1 = P(X_1)$  and  $f_i = P(X_i | X_{i-1})$  for  $i = 2, \dots, N$  specifies the distribution of an MC, i.e.,

$$P_{\mathcal{F}}(X_1, \dots, X_N) = \prod_{l=1}^N f_l(\text{Nb}_{\mathcal{G}}(F_l)), \quad (18.95)$$

$$= P(X_1) \prod_{i=2}^N P(X_i | X_{i-1}). \quad (18.96)$$

**FIGURE 18.17**

FG modeling an MC.

### 1.18.4 Learning

Our goal is now to determine a PGM which models the variables of interest. One possibility is to specify the PGM by hand. Especially BNs are suited to be specified by a domain expert, since the directed edges can be treated as causal directions and local probability tables can be elicited relatively easily. However, the more attractive approach might be to automatically determine a PGM from a set of training data. Assume we want to find a PGM for a set of random variables  $\mathbf{X} = \{X_1, \dots, X_N\}$  which are distributed according to an unknown joint distribution  $P^*(\mathbf{X})$ . Further, assume that we have a collection of  $L$  independent and identically distributed (i.i.d.) samples  $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(L)}\}$ , drawn from distribution  $P^*(\mathbf{X})$ . In this section we assume *complete* data, i.e., each data case  $\mathbf{x}^{(i)}$  contains values of all variables in  $\mathbf{X}$ . For incomplete data, learning becomes much harder and is typically tackled using *expectation-maximization* (EM) methods [11], or related techniques. The task of learning a PGM can be described as follows: given  $\mathcal{D}$ , return a PGM which approximates  $P^*(\mathbf{X})$  best. This type of learning is known as *generative learning*, since we aim to model the process which generated the data. Generative learning can be seen as a “multi-purpose” tool, since we directly strive for the primary object of interest: the joint distribution  $P^*(\mathbf{X})$ . All other quantities of the model which might be interesting, such as marginal distributions of single variables/sets of variables, conditional distributions, or expectations, can be derived from the joint distribution. However, it is rarely possible to exactly retrieve  $P^*(\mathbf{X})$ , especially when using a finite sample  $\mathcal{D}$ . This can be obstructive, when the ultimate goal is not to retrieve  $P^*(\mathbf{X})$ , but to use the PGM in a specialized way. An example for this is when we want to use the PGM as classifier, i.e., we want to infer the value of a class variable, given the values of the remaining variables. In the classification context we are primarily interested in minimizing the loss resulting from erroneous decisions, not in modeling the overall generative process. It can be easily shown that knowledge of  $P^*(\mathbf{X})$  leads to optimal classification, and therefore generative learning seems to be suitable to learn classifiers. However, a somewhat surprising but consistently occurring phenomenon in machine learning literature is that so-called *discriminative* learning methods, which refrain from modeling  $P^*(\mathbf{X})$ , but directly address the classification problem, often yield better classification results than the generative approach. The discriminative paradigm is discussed in Section 1.18.4.3. For now, let us focus on the generative approach.

#### 1.18.4.1 Principles of generative learning

Let us assume a specific class of models  $\mathcal{C}$ , e.g., the class of all BNs, or the class of all MNs. We define a prior distribution  $P(\mathcal{C})$  over the model class, which represents our preference for certain models. For example, if  $\mathcal{C}$  is the class of all BNs, we may prefer networks with fewer edges, and therefore define a prior which decreases with the number of edges. Furthermore, let  $\mathcal{D}$  be a set of i.i.d. training samples

drawn from the true distribution  $P^*(\mathbf{X})$ . In the generative approach, we are interested in the posterior probability distribution over the model class, conditioned on the given data. Using Bayes' law, this distribution is

$$P(\mathcal{C}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{C})P(\mathcal{C})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\mathcal{C})P(\mathcal{C})}{\int_{\mathcal{C}} P(\mathcal{D}|M')P(M')dM'} \propto P(\mathcal{D}|\mathcal{C})P(\mathcal{C}). \quad (18.97)$$

The data generation probability  $P(\mathcal{D}|\mathcal{C})$  is widely known as *likelihood*  $\mathcal{L}(\mathcal{C}; \mathcal{D})$ . The only difference between  $P(\mathcal{D}|\mathcal{C})$  and  $\mathcal{L}(\mathcal{C}; \mathcal{D})$  is that the likelihood is interpreted as a function of  $\mathcal{C}$ , while the data generation probability is viewed as a probability of  $\mathcal{D}$ . In (18.97) we see, that  $P(\mathcal{C}|\mathcal{D})$  is proportional to the product of prior  $P(\mathcal{C})$  and likelihood  $\mathcal{L}(\mathcal{C}; \mathcal{D})$ , i.e., the posterior  $P(\mathcal{C}|\mathcal{D})$  represents an “updated” version of the prior  $P(\mathcal{C})$ , incorporating the likelihood  $\mathcal{L}(\mathcal{C}; \mathcal{D})$ . There exist two general approaches to use the posterior distribution for learning:

- i. *Maximum a-posteriori (MAP) approach:* This approach aims to find the most probable model  $M_{\text{MAP}} \in \mathcal{C}$  for given data, i.e.,

$$M_{\text{MAP}} = \arg \max_{M \in \mathcal{C}} P(M|\mathcal{D}), \quad (18.98)$$

$$= \arg \max_{M \in \mathcal{C}} \mathcal{L}(M; \mathcal{D})P(M). \quad (18.99)$$

In the MAP approach we accept the single model  $M_{\text{MAP}}$  as best explanation for the data and consequently use it for further tasks.

- ii. *Bayesian approach:* In the Bayesian approach, we refrain to decide for a single model out of  $\mathcal{C}$ . Instead, we maintain the uncertainty about which model might be the “true” one and keep the whole model class  $\mathcal{C}$ , together with the posterior distribution  $P(\mathcal{C}|\mathcal{D})$ .

To illustrate the difference of these two approaches, let us assume we are interested in the probability of an unseen sample  $\mathbf{x}'$ , after observing  $\mathcal{D}$ . The MAP approach yields

$$P(\mathbf{X} = \mathbf{x}'|\mathcal{D}) = P(\mathbf{x}'|M_{\text{MAP}}), \quad (18.100)$$

while the Bayesian approach results in

$$P(\mathbf{X} = \mathbf{x}'|\mathcal{D}) = \int_{\mathcal{C}} P(\mathbf{x}'|M)P(M|\mathcal{D})dM. \quad (18.101)$$

The Bayesian approach proceeds more carefully and treats the uncertainty by marginalizing over all possible models. The MAP and the Bayesian approach often agree in the large sample limit, since for large  $L$  the mass of the posterior  $P(\mathcal{C}|\mathcal{D})$  is typically concentrated on a single model. For small sample sizes, the Bayesian approach typically surpasses the MAP approach. However, it is rarely possible to solve the integral in (18.101) exactly, which is the main drawback of the Bayesian approach. Typically, (18.101) has to be approximated, e.g., by integrating (or summing) over a small portion of all models.

In the special case when the prior distribution  $P(\mathcal{C})$  is flat, i.e., no model is preferred over the other, the posterior  $P(\mathcal{C}|\mathcal{D})$  is proportional to the likelihood  $\mathcal{L}(\mathcal{C}; \mathcal{D})$  and the MAP solution coincides with the *maximum-likelihood* (ML) solution, i.e.,

$$M_{\text{MAP}} = M_{\text{ML}} = \arg \max_{M \in \mathcal{C}} \mathcal{L}(M; \mathcal{D}). \quad (18.102)$$

Since the data samples in  $\mathcal{D}$  are i.i.d., the likelihood is given as

$$\mathcal{L}(\mathcal{C}; \mathcal{D}) = P(\mathcal{D}|\mathcal{C}) = \prod_{l=1}^L P(\mathbf{x}^{(l)}|\mathcal{C}). \quad (18.103)$$

In practise, it is often convenient and numerically more stable to work with the *log-likelihood*

$$\log \mathcal{L}(\mathcal{C}; \mathcal{D}) = \sum_{l=1}^L \log P(\mathbf{x}^{(l)}|\mathcal{C}). \quad (18.104)$$

Since the logarithm is an increasing function, maximizing the log-likelihood is equivalent to maximizing the likelihood. Intuitively, (log-) likelihood is a measure of “goodness of fit,” i.e., a model with maximum likelihood is considered as best explanation for the training data. Furthermore, there is an interesting connection between likelihood and information theory; For the sample  $\mathcal{D}$ , the empirical data distribution is defined as

$$P_{\mathcal{D}}(\mathbf{x}) = \frac{1}{L} \sum_{l=1}^L \mathbb{1}_{\{\mathbf{x}=\mathbf{x}^{(l)}\}}, \quad (18.105)$$

where  $\mathbb{1}_{\{\cdot\}}$  is the indicator function. It is easily shown that maximizing the likelihood is equivalent to minimize the Kullback-Leibler (KL) divergence between empirical distribution and model distribution, which is given as

$$\text{KL}(P_{\mathcal{D}}||P(\cdot|\mathcal{C})) = \sum_{\mathbf{x} \in \text{var}(\mathbf{X})} P_{\mathcal{D}}(\mathbf{x}) \log \frac{P_{\mathcal{D}}(\mathbf{x})}{P(\mathbf{x}|\mathcal{C})}. \quad (18.106)$$

The KL divergence is a dissimilarity measure of the argument distributions. In particular,  $\text{KL}(P||Q) \geq 0$  and  $\text{KL}(P||Q) = 0$  if and only if  $P(\mathbf{X})$  and  $Q(\mathbf{X})$  are identical. Furthermore,  $\text{KL}(P||Q)$  measures the minimal average coding overhead, for encoding data using the distribution  $Q(\mathbf{X})$ , while the data is actually distributed according to  $P(\mathbf{X})$  [5].

### 1.18.4.2 Generative learning of Bayesian networks

We now specify the model class  $\mathcal{C}$ , and concentrate on the problem of learning BNs from data. As discussed in Section 1.18.3.1, a BN  $\mathcal{B} = (\mathcal{G}, \Theta)$  defines a distribution over the model variables  $\mathbf{X}$  according to

$$P_{\mathcal{B}}(\mathbf{X}) = \prod_{i=1}^N P(X_i | \text{Pa}_{\mathcal{G}}(X_i), \boldsymbol{\theta}^i), \quad (18.107)$$

which is a product over local probability distributions, one for each variable  $X_i$ . Equation (18.107) highlights the two aspects of a BN: the structure  $\mathcal{G}$ , which determines the parents for each variable, and the parameters  $\Theta = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots, \boldsymbol{\theta}^N\}$  which parameterize the local distributions. Although  $\mathcal{G}$  and  $\Theta$  are coupled, e.g., the number of parameters in  $\boldsymbol{\theta}^i$  depends on  $\mathcal{G}$ , an isolated consideration of  $\mathcal{G}$  and  $\Theta$  is possible and reasonable. This leads to the two tasks of learning BNs: parameter learning and structure learning. For parameter learning, we assume that the network structure  $\mathcal{G}$  is fixed, either

because a domain expert specified the independence relationships among the model variables, or because a structure learning algorithm found an appropriate structure. For fixed structure  $\mathcal{G}$ , we introduce the shorthand  $\mathbf{Pa}_i = \mathbf{Pa}_{\mathcal{G}}(X_i)$  and  $\mathbf{x}_{\mathbf{Pa}_i} = \mathbf{x}(\mathbf{Pa}_{\mathcal{G}}(X_i))$ . In the next section we discuss ML parameter learning, and in Section 1.18.4.2.2 we introduce a full Bayesian approach for BN parameter learning. In Section 1.18.4.2.3 we address the problem of learning also the structure  $\mathcal{G}$  from data.

### 1.18.4.2.1 Maximum likelihood parameter learning

For an i.i.d. sample  $\mathcal{D}$ , the log-likelihood of a BN model is given as

$$\log \mathcal{L}(\mathcal{B}; \mathcal{D}) = \sum_{l=1}^L \sum_{i=1}^N \log P(x_i^{(l)} | \mathbf{x}_{\mathbf{Pa}_i}^{(l)}, \boldsymbol{\theta}^i) = \sum_{i=1}^N \ell(\boldsymbol{\theta}^i, \mathcal{D}). \quad (18.108)$$

Equation (18.108) shows that the log-likelihood decomposes, i.e.,  $\log \mathcal{L}(\mathcal{B}; \mathcal{D})$  equals to a sum over local terms, one for each variable, given as

$$\ell(\boldsymbol{\theta}^i, \mathcal{D}) = \sum_{l=1}^L \log P(x_i^{(l)} | \mathbf{x}_{\mathbf{Pa}_i}^{(l)}, \boldsymbol{\theta}^i). \quad (18.109)$$

Thus the overall BN log-likelihood is maximized by maximizing the individual local terms w.r.t. the local parameters  $\boldsymbol{\theta}^i$ . This holds true as long as each local term  $\ell(\boldsymbol{\theta}^i; \mathcal{D})$  only depends on  $\boldsymbol{\theta}^i$ . If there exists a coupling between the parameters, as for instance introduced by parameter tying, we loose this decomposition property.

*Maximum likelihood parameter learning for discrete variables:* If the variables  $\mathbf{X}$  are discrete, the most general CPDs are discrete distributions (cf. Example 13):

$$P(X_i = j | \mathbf{Pa}_i = \mathbf{h}, \boldsymbol{\theta}^i) = \theta_{j|\mathbf{h}}^i. \quad (18.110)$$

In order to maximize the likelihood, we aim to individually maximize the local terms  $\ell(\boldsymbol{\theta}^i; \mathcal{D})$  (18.109), which are given as

$$\ell(\boldsymbol{\theta}^i; \mathcal{D}) = \sum_{l=1}^L \log \left( \prod_{j=1}^{\text{sp}(X_i)} \prod_{\mathbf{h} \in \text{val}(\mathbf{Pa}_i)} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^{i,l} \right), \quad (18.111)$$

$$= \sum_{\mathbf{h}} \sum_{l: \mathbf{x}_l \in \mathcal{D}_{\mathbf{h}}} \sum_{j=1}^{\text{sp}(X_i)} u_{j|\mathbf{h}}^{i,l} \log \theta_{j|\mathbf{h}}^i, \quad (18.112)$$

$$= \sum_{\mathbf{h}} \log \mathcal{L}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i; \mathcal{D}_{\mathbf{h}}). \quad (18.113)$$

Here  $u_{j|\mathbf{h}}^{i,l} = \mathbb{1}_{\{x_i^{(l)} = j \text{ and } \mathbf{x}_{\mathbf{Pa}_i}^{(l)} = \mathbf{h}\}}$ . The data set  $\mathcal{D}_{\mathbf{h}}$  is a subset of the training data  $\mathcal{D}$ , containing those samples  $\mathbf{x}^{(l)}$  where  $\mathbf{x}_{\mathbf{Pa}_i}^{(l)} = \mathbf{h}$ . In (18.112) sums are interchanged and some terms which would be

evaluated to zero, are discarded. In (18.113) the local terms  $\ell(\boldsymbol{\theta}^i; \mathcal{D})$  decompose to a sum of local log-likelihoods

$$\log \mathcal{L}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i; \mathcal{D}_{\mathbf{h}}) = \sum_{l: \mathbf{x}_l \in \mathcal{D}_{\mathbf{h}}} \sum_{j=1}^{\text{sp}(X_i)} u_{j|\mathbf{h}}^{i,l} \log \theta_{j|\mathbf{h}}^i. \quad (18.114)$$

Inserting this result into (18.108), we see that the BN log-likelihood is given as a sum of local log-likelihoods:

$$\log \mathcal{L}(\mathcal{B}; \mathcal{D}) = \sum_{i=1}^N \sum_{\mathbf{h} \in \text{val}(\mathbf{Pa}_i)} \log \mathcal{L}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i; \mathcal{D}_{\mathbf{h}}) \quad (18.115)$$

Since each  $\theta_{\cdot|\mathbf{h}}^i$  only occurs in a single summand in (18.115), the overall log-likelihood is maximized by individually maximizing each local log-likelihood. Note that  $\theta_{\cdot|\mathbf{h}}^i$  are the parameters of  $X_i$  conditioned on a particular  $\mathbf{h}$ , i.e., they are the parameters of single dimensional discrete distributions. Finding these ML parameters is easy. For notational simplicity, we derive the ML estimate for a generic variable  $Y$  with data  $\mathcal{D}_Y = \{y^{(1)}, \dots, y^{(L)}\}$  and generic parameters  $\boldsymbol{\theta}^Y = (\theta_1^Y, \dots, \theta_{\text{sp}(Y)}^Y)$ . The ML estimate  $\hat{\boldsymbol{\theta}}^Y$  is found by solving the optimization problem:

$$\underset{\boldsymbol{\theta}^Y}{\text{maximize}} \sum_{l=1}^L \sum_{j=1}^{\text{sp}(Y)} u_j^l \log \theta_j^Y, \quad (18.116)$$

$$\text{s.t. } \sum_{j=1}^{\text{sp}(Y)} \theta_j^Y = 1. \quad (18.117)$$

Here  $u_j^l$  is the indicator function  $\mathbb{1}_{\{y^{(l)}=j\}}$ . The Lagrangian function of this problem is given as

$$L(\boldsymbol{\theta}^Y, v) = \sum_{l=1}^L \sum_{j=1}^{\text{sp}(Y)} u_j^l \log \theta_j^Y + v \left( 1 - \sum_{j=1}^{\text{sp}(Y)} \theta_j^Y \right). \quad (18.118)$$

Since the objective in (18.116) is concave and we have a linear equality constraint (18.117), stationarity of  $L(\boldsymbol{\theta}^Y, v)$  is necessary and sufficient for optimality [12], i.e., the constraint (18.117) has to hold and the partial derivatives w.r.t.  $\theta_j^Y$  have to vanish:

$$\frac{\partial L(\boldsymbol{\theta}^Y, v)}{\partial \theta_j^Y} = \sum_{l=1}^L u_j^l \frac{1}{\theta_j^Y} - v \stackrel{!}{=} 0. \quad (18.119)$$

We see that the optimal parameters have the form

$$\hat{\theta}_j^Y = \frac{\sum_{l=1}^L u_j^l}{v} = \frac{n_j}{v}, \quad (18.120)$$

where  $n_j = \sum_{l=1}^L u_j^l$  is the number of occurrences of  $Y = j$  in the data set  $\mathcal{D}_Y$ . Since (18.117) has to hold, we see that  $v$  has to be equal to  $\sum_{j=1}^{\text{sp}(Y)} n_j$ , and therefore the ML parameters are simply the relative frequency counts according to  $\mathcal{D}_Y$ :

$$\hat{\theta}_j^Y = \frac{n_j}{\sum_{j'=1}^{\text{sp}(Y)} n_{j'}}. \quad (18.121)$$

Applying this result to  $\theta_{|\mathbf{h}}^i$ , we obtain the ML parameters as

$$\hat{\theta}_{j|\mathbf{h}}^i = \frac{\sum_{l=1}^L u_{j|\mathbf{h}}^{i,l}}{\sum_{j'=1}^{\text{sp}(X_j)} \sum_{l=1}^L u_{j'|\mathbf{h}}^{i,l}}. \quad (18.122)$$

This estimate is well-defined if there is at least one sample in the subset  $\mathcal{D}_{\mathbf{h}}$ . If there is no sample in  $\mathcal{D}_{\mathbf{h}}$ , an arbitrary distribution (e.g., the uniform distribution) can be assumed, since in this case  $\theta_{|\mathbf{h}}^i$  does not contribute to the likelihood.

*Maximum likelihood parameter learning for conditional Gaussian variables:* Now let us assume that the variables are real-valued, and that the CPDs are conditional Gaussians (cf. Example 14), i.e.,

$$P(X_i|\mathbf{Pa}_i, \theta^i) = \mathcal{N}\left(X_i \mid \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i} + \beta_i; \sigma^2\right), \quad (18.123)$$

where we assume a fixed shared variance  $\sigma^2$ . Again, in order to maximize the BN likelihood, we aim to individually maximize the local terms  $\ell(\theta^i; \mathcal{D})$  in (18.109). Inserting the logarithm of (18.123) in (18.109) yields

$$\ell(\theta^i; \mathcal{D}) = \sum_{l=1}^L \left[ -\frac{1}{2} \log(2\pi\sigma^2) - \frac{(x_i^{(l)} - \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \beta_i)^2}{2\sigma^2} \right] \quad (18.124)$$

$$\begin{aligned} &= -\frac{1}{2\sigma^2} \sum_{l=1}^L \left[ x_i^{(l)2} + \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i + \beta_i^2 - 2x_i^{(l)} \boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \right. \\ &\quad \left. - 2x_i^{(l)} \beta_i + 2\boldsymbol{\alpha}_i^T \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \beta_i \right] + \text{const.}, \end{aligned} \quad (18.125)$$

where additive terms which are independent of  $\boldsymbol{\alpha}_i$  and  $\beta_i$  are summarized in a constant. The overall log-likelihood of the BN is maximized by individually maximizing (18.124) w.r.t.  $\theta^i = \{\boldsymbol{\alpha}_i, \beta_i\}$ . The function  $\ell(\theta^i; \mathcal{D})$  is the negative square of an affine function and therefore concave in  $\boldsymbol{\alpha}_i$  and  $\beta_i$  [12]. Thus, a maximum can be found by setting the partial derivatives to zero. For  $\beta_i$  we get:

$$\frac{\partial \ell(\theta^i; \mathcal{D})}{\partial \beta_i} = -\frac{1}{2\sigma^2} \sum_{l=1}^L \left[ 2\beta_i - 2x_i^{(l)} + 2\mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i \right] \stackrel{!}{=} 0. \quad (18.126)$$

Hence the ML parameter  $\hat{\beta}_i$  is given as

$$\hat{\beta}_i = \frac{1}{L} \sum_{l=1}^L \left[ x_i^{(l)} - \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i \right] = \hat{x}_i - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \boldsymbol{\alpha}_i, \quad (18.127)$$

where  $\hat{x}_i$  and  $\hat{\mathbf{x}}_{\mathbf{Pa}_i}$  are the sample means of  $X_i$  and  $\mathbf{Pa}_i$ , respectively. For  $\boldsymbol{\alpha}_i$  we have

$$\nabla_{\boldsymbol{\alpha}_i} \ell(\boldsymbol{\theta}^i; \mathcal{D}) = -\frac{1}{2\sigma^2} \sum_{l=1}^L \left[ 2\mathbf{x}_{\mathbf{Pa}_i}^{(l)} \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} \boldsymbol{\alpha}_i - 2x_i^{(l)} \mathbf{x}_{\mathbf{Pa}_i}^{(l)} + 2\mathbf{x}_{\mathbf{Pa}_i}^{(l)} \beta_i \right] \stackrel{!}{=} 0. \quad (18.128)$$

Using  $\hat{\beta}_i$  in (18.128), we get the ML parameters  $\hat{\boldsymbol{\alpha}}_i$  by the following chain of equations:

$$\begin{aligned} \sum_{l=1}^L \left[ \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i &= \sum_{l=1}^L \left[ \left( x_i^{(l)} - \hat{x}_i \right) \mathbf{x}_{\mathbf{Pa}_i}^{(l)} \right], \\ \sum_{l=1}^L \left[ \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} + \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i &= \sum_{l=1}^L \left[ \left( x_i^{(l)} - \hat{x}_i \right) \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} + \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \right], \end{aligned} \quad (18.129)$$

$$\frac{1}{L-1} \sum_{l=1}^L \left[ \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) \right] \hat{\boldsymbol{\alpha}}_i = \frac{1}{L-1} \sum_{l=1}^L \left[ \left( x_i^{(l)} - \hat{x}_i \right) \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)} - \hat{\mathbf{x}}_{\mathbf{Pa}_i} \right) \right], \quad (18.131)$$

$$\hat{\mathbf{C}}_{\mathbf{Pa}} \hat{\boldsymbol{\alpha}}_i = \hat{\mathbf{c}}, \quad (18.132)$$

$$\hat{\boldsymbol{\alpha}}_i = \hat{\mathbf{C}}_{\mathbf{Pa}}^{-1} \hat{\mathbf{c}}. \quad (18.133)$$

Here  $\hat{\mathbf{C}}_{\mathbf{Pa}}$  is the estimated covariance matrix of the parent variables, and  $\hat{\mathbf{c}}$  is the estimated covariance vector between parents and the child. In (18.131) we used the fact that  $\sum_{l=1}^L \mathbf{v} \left( \mathbf{x}_{\mathbf{Pa}_i}^{(l)T} - \hat{\mathbf{x}}_{\mathbf{Pa}_i}^T \right) = \mathbf{0}$  and  $\sum_{l=1}^L \left( x_i^{(l)} - \hat{x}_i \right) \mathbf{v} = \mathbf{0}$ , for any vector  $\mathbf{v}$  which does not depend on  $i$ .

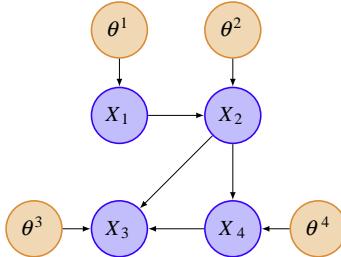
#### 1.18.4.2.2 Bayesian parameter learning

In a Bayesian approach, we do not assume that there exists a single set of “true” parameters, but maintain the uncertainty about the parameters. Therefore, we explicitly include the parameters into the model as in Figure 18.18, where a BN over variables  $X_1, \dots, X_4$  is augmented with their local CPD parameters  $\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^4$ . Learning in the Bayesian setting means to perform inference over parameters; in the Bayesian view, there is actually no difference between learning and inference.

Note that there are no edges between any  $\boldsymbol{\theta}^i$  and  $\boldsymbol{\theta}^j$ , which means that we assume parameter independence. This kind of independence is called *global parameter independence* (as opposed to *local parameter independence*, discussed below) [1,13]. Global parameter independence, although reasonable in many cases, does not necessarily hold in every context. However, in the case of global parameter independence any prior over  $\Theta = \{\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^N\}$  factorizes according to

$$P(\Theta) = \prod_{i=1}^N P(\boldsymbol{\theta}^i). \quad (18.134)$$

Furthermore, if all model variables  $X_1, \dots, X_N$  are observed, then the local parameters  $\boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^N$  are pairwise  $d$ -separated, and thus independent. Furthermore, the same is true for multiple i.i.d. samples,

**FIGURE 18.18**

Bayesian network over variables  $X_1, \dots, X_4$ , augmented with local CPD parameters  $\theta^1, \dots, \theta^4$ .

i.e., for complete data sets  $\mathcal{D}$ . In this case, if global (a-priori) parameter independence holds, also a-posteriori parameter independence holds, and the posterior over  $\Theta$  factorizes according to

$$P(\Theta|\mathcal{D}) = \prod_{i=1}^N P(\theta^i|\mathcal{D}). \quad (18.135)$$

Combining (18.135) with (18.107), we obtain the model posterior as

$$P_B(\mathbf{X}, \Theta|\mathcal{D}) = P_B(\mathbf{X}|\Theta)P(\Theta|\mathcal{D}), \quad (18.136)$$

$$= \left( \prod_{i=1}^N P(X_i|\mathbf{Pa}_i, \theta^i) \right) \left( \prod_{i=1}^N P(\theta^i|\mathcal{D}) \right), \quad (18.137)$$

$$= \prod_{i=1}^N P(X_i|\mathbf{Pa}_i, \theta^i)P(\theta^i|\mathcal{D}). \quad (18.138)$$

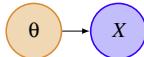
Since one is usually interested in the data posterior  $P(\mathbf{X}|\mathcal{D})$ , one has to marginalize over  $\Theta$ :

$$P(\mathbf{X}|\mathcal{D}) = \int_{\Theta} P(\mathbf{X}, \Theta|\mathcal{D})d\Theta, \quad (18.139)$$

$$= \int_{\theta^1} \int_{\theta^2} \cdots \int_{\theta^N} \left( \prod_{i=1}^N P(X_i|\mathbf{Pa}_i, \theta^i)P(\theta^i|\mathcal{D}) \right) d\theta^1 d\theta^2 \cdots d\theta^N, \quad (18.140)$$

$$= \prod_{i=1}^N \left( \int_{\theta^i} P(X_i|\mathbf{Pa}_i, \theta^i)P(\theta^i|\mathcal{D})d\theta^i \right). \quad (18.141)$$

The exchange of integrals and products in (18.141) is possible since each term in the product in (18.140) depends only on a single  $\theta^i$ . Hence, under the assumption of global parameter independence and completely observed data, the integral over the entire parameter space in (18.139) factorizes into a product of several simpler integrals (18.141).

**FIGURE 18.19**

Simple Bayesian network containing a single variable  $X$ , augmented with parameters  $\theta$ .

*Bayesian parameter learning for discrete variables:* We now return to the example, where all variables of the BN are discrete. As a basic example, and in order to introduce concepts which are required later, let us consider a simple BN with only a single variable, i.e.,  $N = 1$  and set  $X = X_1$  and  $J = \text{sp}(X)$ . The probability of  $X$  being in state  $j$  is

$$P(X = j|\theta) = \theta_j. \quad (18.142)$$

Figure 18.19 shows the corresponding BN, where  $\theta$  is explicitly included. The parameters can be represented as a vector  $\theta = (\theta_1, \dots, \theta_J)^T$  of RVs, which has to be an element of the standard simplex  $\mathcal{S} = \left\{ \theta \mid \theta_j \geq 0, \forall j, \sum_{j=1}^J \theta_j = 1 \right\}$ . We need to specify a prior distribution over  $\theta$ , where a natural choice is the Dirichlet distribution, defined as

$$\text{Dir}(\theta; \alpha) = \begin{cases} \frac{1}{B(\alpha)} \prod_{j=1}^J \theta_j^{\alpha_j - 1} & \text{if } \theta \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases} \quad (18.143)$$

The Dirichlet distribution is parameterized by the vector  $\alpha = (\alpha_1, \dots, \alpha_J)^T$  with  $\alpha_j > 0$  for all  $j$ . The term  $\frac{1}{B(\alpha)}$  is the normalization constant of the distribution, where the *beta-function*  $B(\alpha)$  is given as

$$B(\alpha) = \int_{\mathcal{S}} \prod_{j=1}^J \theta_j^{\alpha_j - 1} d\theta = \frac{\prod_{j=1}^J \Gamma(\alpha_j)}{\Gamma\left(\sum_{j=1}^J \alpha_j\right)}. \quad (18.144)$$

Here  $\Gamma(\cdot)$  is the *gamma function*, i.e., a continuous generalization of the factorial function. In the special case that  $\alpha_j = 1, \forall j$ , the Dirichlet distribution is uniform over all  $\theta \in \mathcal{S}$ .

Assuming a Dirichlet prior over  $\theta$ , the distribution of the BN in Figure 18.19 is given as  $P(X, \theta) = P_B(X|\theta)\text{Dir}(\theta; \alpha)$ . Usually one is interested in the posterior over  $X$ , which is obtained by marginalizing over  $\theta$ :

$$P(X = j') = \int_{\theta} P(X|\theta)\text{Dir}(\theta; \alpha)d\theta, \quad (18.145)$$

$$= \int_{\mathcal{S}} \theta_{j'} \frac{1}{B(\alpha)} \prod_{j=1}^J \theta_j^{\alpha_j - 1} d\theta, \quad (18.146)$$

$$= \frac{1}{B(\boldsymbol{\alpha})} \frac{\Gamma(\alpha_{j'} + 1) \prod_{j \neq j'} \Gamma(\alpha_j)}{\Gamma\left(1 + \sum_{j=1}^J \alpha_j\right)}, \quad (18.147)$$

$$= \frac{B(\boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \frac{\alpha_{j'}}{\sum_{j=1}^J \alpha_j}, \quad (18.148)$$

$$= \frac{\alpha_{j'}}{\sum_{j=1}^J \alpha_j}. \quad (18.149)$$

The integral in (18.146) can be solved using (18.144), and (18.148) follows from the properties of the gamma function. We see that (18.148) has the form of relative frequency counts, exactly as the ML estimator in (18.122). Thus, the Dirichlet parameters  $\boldsymbol{\alpha}$  gain the interpretation of a-priori pseudo-data-counts, which represent our belief about  $\boldsymbol{\theta}$ . Now assume that we have an i.i.d. sample  $\mathcal{D} = (x^{(1)}, \dots, x^{(L)})$  of variable  $X$ . Learning in the Bayesian framework means to incorporate the information given by  $\mathcal{D}$ , i.e., to modify our belief about  $\boldsymbol{\theta}$ . Formally, we want to obtain the posterior distribution over  $\boldsymbol{\theta}$ :

$$P(\boldsymbol{\theta}|\mathcal{D}) \propto P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta}) = \left( \prod_{l=1}^L P(x^{(l)}|\boldsymbol{\theta}) \right) \text{Dir}(\boldsymbol{\theta}; \boldsymbol{\alpha}), \quad (18.150)$$

$$= \left( \prod_{l=1}^L \prod_{j=1}^J \theta_j^{u_j^l} \right) \left( \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^J \theta_j^{\alpha_j - 1} \right), \quad (18.151)$$

$$= \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^J \theta_j^{n_j + \alpha_j - 1}, \quad (18.152)$$

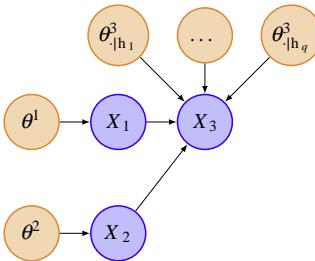
where  $u_j^l = \mathbb{1}_{\{x^{(l)}=j\}}$  and  $n_j = \sum_{l=1}^L u_j^l$ . Note that (18.152) has again the functional form of a Dirichlet distribution, although not correctly normalized. Hence, we can conclude that the posterior  $P(\boldsymbol{\theta}|\mathcal{D})$  is also a Dirichlet distribution. After normalizing, we obtain

$$P(\boldsymbol{\theta}|\mathcal{D}) = \text{Dir}(\boldsymbol{\theta}; \boldsymbol{\alpha} + \mathbf{n}), \quad (18.153)$$

where  $\mathbf{n} = (n_1, n_2, \dots, n_J)^T$ . The property that the posterior is from the same family of distributions as the prior, comes from the fact that the Dirichlet distribution is a so-called *conjugate prior* [4] for the discrete distribution. The intuitive interpretation of (18.153) is that the distribution over  $\boldsymbol{\theta}$  is updated by adding the real data counts  $\mathbf{n}$  to our a-priori pseudo-counts  $\boldsymbol{\alpha}$ . Replacing the prior  $P(\boldsymbol{\theta})$  with the posterior  $P(\boldsymbol{\theta}|\mathcal{D})$  in 18.145–18.148 yields

$$P(X = j'|\mathcal{D}) = \frac{\alpha_{j'} + n'_j}{\sum_{j=1}^J (\alpha_j + n_j)}. \quad (18.154)$$

Now we extend the discussion to general BNs. First, we interpret the CPD parameters  $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots, \boldsymbol{\theta}^N\}$  as RVs, i.e., we define a prior  $P(\boldsymbol{\Theta})$ . We assume global parameter independence,

**FIGURE 18.20**

Bayesian network over variables  $X_1, X_2, X_3, X_4$ .

such that the prior takes the form  $P(\Theta) = \prod_{i=1}^N P(\theta^i)$ . Note that  $\theta^i$  can be interpreted as a matrix of RVs, containing one column  $\theta_{\cdot|\mathbf{h}}^i$  for each possible assignment  $\mathbf{h}$  of the parent variables  $\mathbf{Pa}_i$ . We can additionally assume that the columns of  $\theta^i$  are a-priori independent, i.e., that the prior over  $\theta^i$  factorizes as

$$P(\theta^i) = \prod_{\mathbf{h} \in \text{val}(\mathbf{Pa}_i)} P(\theta_{\cdot|\mathbf{h}}^i). \quad (18.155)$$

This independence assumption is called *local parameter independence*, as in contrast to global parameter independence [1, 13]. Similar, as global parameter independence, local parameter independence is often a reasonable assumption, but has to be considered with care. For a specific assignment  $\mathbf{h}$ , the vector  $\theta_{\cdot|\mathbf{h}}^i = (\theta_{1|\mathbf{h}}^i, \dots, \theta_{\text{sp}(X_i)|\mathbf{h}}^i)^T$  represents a single (conditional) discrete distribution over  $X_i$ . As in the basic example with only one variable, we can assume a Dirichlet prior for  $\theta_{\cdot|\mathbf{h}}^i$ , each with its own parameters  $\alpha_{\mathbf{h}}^i = (\alpha_{1|\mathbf{h}}^i, \dots, \alpha_{\text{sp}(X_i)|\mathbf{h}}^i)$ . Together with the assumption of global and local parameter independence, the prior over the whole parameter set  $\Theta$  is then given as

$$P(\Theta) = \prod_{i=1}^N \prod_{\mathbf{h}} \text{Dir}(\theta_{\cdot|\mathbf{h}}^i; \alpha_{\mathbf{h}}^i). \quad (18.156)$$

For global parameter independence, we observed that the parameters  $\theta^1, \dots, \theta^N$  remain independent, when conditioned on a completely observed data set (cf. (18.134) and (18.135)) since the parameters are  $d$ -separated by the observed nodes  $X_1, \dots, X_N$ . For local parameter independence, it does not happen that any two sets of parameters  $\theta_{\cdot|\mathbf{h}}^i$  and  $\theta_{\cdot|\mathbf{h}'}^j$ ,  $\mathbf{h} \neq \mathbf{h}'$ , are  $d$ -separated.

**Example 22.** Consider Figure 18.20 and let  $q = \text{sp}(X_1)\text{sp}(X_2)$  be the total number of parent configurations for node  $X_3$ . The parameter sets  $\theta_{\cdot|\mathbf{h}_1}^3, \dots, \theta_{\cdot|\mathbf{h}_q}^3$  are not  $d$ -separated, since  $X_3$  is observed.

However, note that  $d$ -separation is sufficient for conditional independence, but not necessary. This means that even if two sets of nodes are not  $d$ -separated, they still can turn out to be independent. Fortunately, this is the case here. To see this, assume that local (a-priori) parameter independence holds, and that all nodes  $X_1, \dots, X_N$  are observed. For some arbitrary node  $X_i$ , the posterior parameter distribution is

given as

$$P(\boldsymbol{\theta}^i | \mathbf{X}) = P(\boldsymbol{\theta}^i | X_i, \mathbf{Pa}_i) = \frac{P(X_i | \boldsymbol{\theta}^i, \mathbf{Pa}_i) P(\boldsymbol{\theta}^i | \mathbf{Pa}_i)}{P(X_i | \mathbf{Pa}_i)}, \quad (18.157)$$

$$= \frac{P(X_i | \boldsymbol{\theta}_{\cdot|\mathbf{Pa}_i}^i, \mathbf{Pa}_i) \prod_{\mathbf{h}} P(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i)}{P(X_i | \mathbf{Pa}_i)}, \quad (18.158)$$

$$= \frac{\prod_{\mathbf{h}} f(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i, \mathbf{X})}{P(X_i | \mathbf{Pa}_i)}, \quad (18.159)$$

where

$$f(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i, \mathbf{X}) = \begin{cases} P(X_i | \boldsymbol{\theta}_{\cdot|\mathbf{h}}^i, \mathbf{Pa}_i) P(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i) & \text{if } \mathbf{h} = \mathbf{Pa}_i, \text{ and} \\ P(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i) & \text{otherwise.} \end{cases} \quad (18.160)$$

We see that the posterior parameter distribution factorizes over the parent configurations, i.e., that the parameters  $\boldsymbol{\theta}_{\cdot|\mathbf{h}_1}^i, \dots, \boldsymbol{\theta}_{\cdot|\mathbf{h}_q}^i, q = \text{sp}(\mathbf{Pa}_i)$ , remain independent when conditioned on  $\mathbf{X}$ . This conclusion comes from two observations in (18.158): (i) All  $\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i$  except  $\boldsymbol{\theta}_{\cdot|\mathbf{Pa}_i}^i$  are independent from  $X_i$ , when  $\mathbf{Pa}_i$  is given, and (ii)  $\boldsymbol{\theta}^i$  is a-priori independent from  $\mathbf{Pa}_i$ .

Furthermore, local a-posteriori parameter independence also holds for multiple observed i.i.d. samples  $\mathcal{D}$ , i.e.,

$$P(\boldsymbol{\theta}^i | \mathcal{D}) = \prod_{\mathbf{h}} P(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \mathcal{D}). \quad (18.161)$$

Since we assumed Dirichlet priors, which are conjugate priors for discrete distributions, the posterior over  $\boldsymbol{\theta}^i$  takes the form (cf. (18.153)),

$$P(\boldsymbol{\theta}^i | \mathcal{D}) = \prod_{\mathbf{h}} \text{Dir}\left(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i\right), \quad (18.162)$$

where  $\mathbf{n}_{\mathbf{h}}^i = (n_{1|\mathbf{h}}^i, n_{2|\mathbf{h}}^i, \dots, n_{\text{sp}(X_i)|\mathbf{h}}^i)^T$ , and  $n_{j|\mathbf{h}}^i = \sum_{l=1}^L u_{j|\mathbf{h}}^{i,l}$ . This result can now be applied to (18.141), where we showed (under the assumption of global parameter independence), that

$$P(\mathbf{X} | \mathcal{D}) = \prod_{i=1}^N \left( \int_{\boldsymbol{\theta}^i} P(X_i | \mathbf{Pa}_i, \boldsymbol{\theta}^i) P(\boldsymbol{\theta}^i | \mathcal{D}) d\boldsymbol{\theta}^i \right). \quad (18.163)$$

Inserting (18.162) in (18.163), we obtain

$$P(\mathbf{X} = \mathbf{x} | \mathcal{D}) = \prod_{i=1}^N \left( \int_{\boldsymbol{\theta}^i} \left( \prod_{j=1}^{\text{sp}(X_i)} \prod_{\mathbf{h}} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^i \right) \left( \prod_{\mathbf{h}} \text{Dir}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i) \right) d\boldsymbol{\theta}^i \right), \quad (18.164)$$

$$= \prod_{i=1}^N \left( \int_{\boldsymbol{\theta}^i} \prod_{\mathbf{h}} \left[ \left( \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^i \right) \text{Dir}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i) \right] d\boldsymbol{\theta}^i \right), \quad (18.165)$$

$$= \prod_{i=1}^N \left( \int_{\boldsymbol{\theta}_{\cdot|\mathbf{h}_1}^i} \cdots \int_{\boldsymbol{\theta}_{\cdot|\mathbf{h}_{q_i}}^i} \prod_{\mathbf{h}} \left[ \left( \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^i \right) \text{Dir}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i) \right] d\boldsymbol{\theta}_{\cdot|\mathbf{h}_1}^i \cdots d\boldsymbol{\theta}_{\cdot|\mathbf{h}_{q_i}}^i \right), \quad (18.166)$$

$$= \prod_{i=1}^N \prod_{\mathbf{h}} \left( \int_{\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i} \left( \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^i \right) \text{Dir}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i) d\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i \right), \quad (18.167)$$

where  $q_i = \text{sp}(\mathbf{Pa}_i)$ . The integration over the entire parameter space factorizes into a product of many simpler integrals. Each integral in (18.167) has the same form as in (18.145) and is given as

$$\int_{\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i} \left( \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^i \right) \text{Dir}(\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i | \boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i) d\boldsymbol{\theta}_{\cdot|\mathbf{h}}^i = \begin{cases} \frac{\alpha_{x_i|\mathbf{h}}^i + n_{x_i|\mathbf{h}}^i}{\sum_{j=1}^{\text{sp}(X_i)} (\alpha_{j|\mathbf{h}}^i + n_{j|\mathbf{h}}^i)} & \text{if } \mathbf{h} = \mathbf{x}_{\mathbf{Pa}_i}, \\ 1 & \text{otherwise.} \end{cases} \quad (18.168)$$

Therefore, under the assumptions of global and local parameter independence and using Dirichlet parameter priors, the Bayesian approach is feasible for BNs with discrete variables and reduces to a product over closed-form integrations. This result can be seen as the Bayesian analog to (18.115), where we showed that the BN likelihood is maximized by solving many independent closed-form ML problems.

#### 1.18.4.2.3 Learning the structure of Bayesian networks

So far we have discussed parameter learning techniques for BNs, i.e., we assumed that the BN structure  $\mathcal{G}$  is fixed and learned the parameters  $\boldsymbol{\Theta}$ . The goal is now to also learn the structure  $\mathcal{G}$  from data, where we restrict the discussion to discrete variables from now on. Generally there are two main approaches to structure learning:

- i. *Constraint-based approach:* In this approach, we aim to find a structure  $\mathcal{G}$  which represents the same conditional independencies as present in the data. While being theoretically sound, these methods heavily rely on statistical independence tests performed on the data, which might be inaccurate and usually render the approach unsuitable in practise. Therefore, we do not discuss constraint-based approaches here, but refer the interested reader to [1, 14, 15], and references therein.

- ii. *Scoring-based approach:* Here we aim to find a graph which represents a “suitable” distribution to represent the true distribution  $P^*(\mathbf{X})$ , where “suitability” is measured by a scoring function score  $\mathcal{G}$ . There are two key issues here: Defining an appropriate scoring function, and developing a search algorithm which finds a score-maximizing structure  $\mathcal{G}$ .

A natural choice for the scoring function would be the log-likelihood, i.e., the aim to maximize  $\log \mathcal{L}(\mathcal{B}; \mathcal{D}) = \log \mathcal{L}(\mathcal{G}, \Theta; \mathcal{D})$ , w.r.t.  $\mathcal{G}$  and  $\Theta$ . Using the ML parameter estimates  $\hat{\Theta}$  from Section 1.18.4.2.1, this problem reduces to maximize

$$\log \mathcal{L}(\mathcal{G}, \hat{\Theta}; \mathcal{D}) = \sum_{l=1}^L \log P(\mathbf{X} = \mathbf{x}^{(l)} | \mathcal{G}, \hat{\Theta}), \quad (18.169)$$

w.r.t.  $\mathcal{G}$ . However, the likelihood score in (18.169) is in general not suitable for structure learning. When  $\mathcal{G}$  is a subgraph of  $\mathcal{G}'$ , it is easy to show that a BN defined over  $\mathcal{G}'$  contains all distributions which can also be represented by a BN defined over  $\mathcal{G}$ . Consequently, the supergraph  $\mathcal{G}'$  will always have at least the same likelihood-score as  $\mathcal{G}$ , and furthermore, any fully connected graph will also be a maximizer of (18.169). We see that likelihood prefers more complex models and leads to overfitting. Note that overfitting stems from using a too flexible model class for a finite data set, and that the flexibility of a model is reflected by the number of free parameters. The number of parameters  $T(\mathcal{G})$  in a BN is given as

$$T(\mathcal{G}) = \sum_{i=1}^N (\mathbf{sp}(X_i) - 1)(\mathbf{sp}(\mathbf{Pa}_{\mathcal{G}}(X_i))), \quad (18.170)$$

which grows exponentially with the maximal number of parents. The likelihood-score merely aims to fit the training data, but is blind to model complexity.

As a remedy, we can modify the pure likelihood-score in order to account for model complexity. One way delivers the minimum description length (MDL) principle [16], which aims to find the shortest, most compact representation of the data  $\mathcal{D}$ . As discussed in Section 1.18.4.1, maximizing likelihood is equivalent to minimizing  $\text{KL}(P_{\mathcal{D}} || P(\cdot | \mathcal{C}))$ , i.e., the KL divergence between the empirical data distribution and the model distribution. The KL divergence measures the average coding overhead for encoding the data  $\mathcal{D}$  using  $P(\mathbf{X} | \mathcal{C})$  instead of  $P_{\mathcal{D}}(\mathbf{X})$ . The MDL principle additionally accounts for the description length of the model, which is measured by the number of parameters. For BNs with discrete variables, the following MDL score was derived [17, 18]:

$$\text{MDL}(\mathcal{G}) = - \sum_{l=1}^L \log P(\mathbf{x}^{(l)} | \mathcal{G}, \Theta_{\text{ML}}) + \frac{\log L}{2} T(\mathcal{G}), \quad (18.171)$$

which is a negative score, since we aim to find a graph  $\mathcal{G}$  minimizing  $\text{MDL}(\mathcal{G})$ . The MDL score is simply the negative likelihood score (18.169), plus a penalization term for the number of parameters. In typical applications, the likelihood decreases linearly with  $L$ , while the penalization term in (18.171) grows logarithmically. Therefore, for large  $L$ , the likelihood dominates the penalization term and the MDL score becomes equivalent to likelihood. However, when  $L$  is small, then the penalization term significantly influences (18.171), resulting in a preference for BNs with fewer parameters.

An alternative way to avoid overfitting is delivered by a Bayesian approach. The main problem with the likelihood score (18.169) is that it is not aware of model complexity, i.e., the number of free parameters. While with more densely connected graphs  $\mathcal{G}$  the dimensionality of the parameter space  $\Theta$  increases exponentially, nevertheless only a single point estimate is used from this space, i.e., the ML parameters  $\Theta_{\text{ML}}$ . Therefore, the ML approach can be described as overly optimistic, trusting in the single “true” parameters  $\Theta_{\text{ML}}$ . In contrast to the ML approach, the Bayesian approach (cf. Section 1.18.4.2.2) uses the parameter space in its entirety. We introduce a prior distribution  $P(\Theta)$  over the parameters, and marginalize over  $\Theta$ , leading to the Bayesian score

$$\text{Bayes}(\mathcal{G}) = P(\mathcal{D}|\mathcal{G}) = \int_{\Theta} P(\mathcal{D}|\Theta, \mathcal{G}) P(\Theta) d\Theta, \quad (18.172)$$

$$= \int_{\Theta} \prod_{l=1}^L P(\mathbf{x}^{(l)}|\Theta, \mathcal{G}) P(\Theta) d\Theta. \quad (18.173)$$

We see that this score actually represents a semi-Bayesian approach: While being Bayesian about  $\Theta$ , we still follow the ML principle for the structure  $\mathcal{G}$ . A Bayesian approach, which requires approximation techniques, is provided in [19]. Assuming global and local parameter independence, and using Dirichlet priors for the local parameters (cf. Section 1.18.4.2.2) leads to the Bayesian-Dirichlet score (BD) [20, 21]

$$\text{BD}(\mathcal{G}) = \int_{\Theta} \left( \prod_{l=1}^L \prod_{i=1}^N \prod_{j=1}^{\text{sp}(X_i)} \prod_{\mathbf{h}} \theta_{j|\mathbf{h}}^i u_{j|\mathbf{h}}^{i,l} \right) \left( \prod_{i=1}^N \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}_{\mathbf{h}}^i)} \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i \alpha_{j|\mathbf{h}}^{i-1} \right) d\Theta, \quad (18.174)$$

$$= \int_{\Theta} \prod_{i=1}^N \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}_{\mathbf{h}}^i)} \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i \alpha_{j|\mathbf{h}}^{i-1} n_{j|\mathbf{h}}^{i-1} d\Theta, \quad (18.175)$$

$$= \prod_{i=1}^N \prod_{\mathbf{h}} \frac{1}{B(\boldsymbol{\alpha}_{\mathbf{h}}^i)} \left( \int_{\theta_{|\mathbf{h}}^i}^{\infty} \prod_{j=1}^{\text{sp}(X_i)} \theta_{j|\mathbf{h}}^i \alpha_{j|\mathbf{h}}^{i-1} n_{j|\mathbf{h}}^{i-1} d\theta_{|\mathbf{h}}^i \right), \quad (18.176)$$

$$= \prod_{i=1}^N \prod_{\mathbf{h}} \frac{B(\boldsymbol{\alpha}_{\mathbf{h}}^i + \mathbf{n}_{\mathbf{h}}^i)}{B(\boldsymbol{\alpha}_{\mathbf{h}}^i)}, \quad (18.177)$$

$$= \prod_{i=1}^N \prod_{\mathbf{h}} \frac{\Gamma\left(\sum_{j=1}^{\text{sp}(X_i)} \alpha_{j|\mathbf{h}}^i\right)}{\Gamma\left(\sum_{j=1}^{\text{sp}(X_i)} \alpha_{j|\mathbf{h}}^i + n_{j|\mathbf{h}}^i\right)} \prod_{j=1}^{\text{sp}(X_i)} \frac{\Gamma\left(\alpha_{j|\mathbf{h}}^i + n_{j|\mathbf{h}}^i\right)}{\Gamma\left(\alpha_{j|\mathbf{h}}^i\right)}. \quad (18.178)$$

In (18.176) we can interchange integrals with products, since each term only depends on a single  $\theta_{|\mathbf{h}}^i$ . The solution of the integrals is given in (18.144). Heckerman et al. [13] pointed out that the parameters  $\boldsymbol{\alpha}_{\mathbf{h}}^i$  can not be chosen arbitrarily, when one desires a consistent, likelihood-equivalent score, meaning that the same score is assigned for two networks representing the same set of independence assumptions. They show how such parameters can be constructed, leading to the likelihood-equivalent Bayesian-Dirichlet

score (BDe). A simple choice for  $\alpha_{\mathbf{h}}^i$ , already proposed in [20], is

$$\alpha_{j|\mathbf{h}}^i = \frac{A}{\mathbf{sp}(X_i)\mathbf{sp}(\mathbf{Pa}_{\mathcal{G}}(X_i))}, \quad (18.179)$$

where  $A$  is the equivalent sample size, i.e., the number of virtual a-priori samples (a typical value would be  $A = 1$ ). These parameters result from the a-priori assumption of an uniform distribution over the state space of  $\mathbf{X}$ . Consequently, the BDe score using parameters according (18.179) is called BDe uniform score (BDeu).

Furthermore, there is a connection between the BD score and the MDL score: It can be shown [1], that for  $L \rightarrow \infty$ , the logarithm of the BD score converges to the so-called Bayesian information criterion score (BIC)

$$\text{BIC}(\mathcal{G}) = \sum_{l=1}^L \log P(\mathbf{x}^{(l)}|\mathcal{G}, \Theta_{\text{ML}}) - \frac{\log N}{2} T(\mathcal{G}) + \text{const.}, \quad (18.180)$$

which is the negative MDL score (18.171), up to some constant. Thus, although derived from two different perspectives, the MDL score and the BIC score are equivalent.

We now turn to the problem of finding a graph  $\mathcal{G}$  which maximizes a particular score. All the scores discussed so far, namely log-likelihood (18.169), MDL (18.171) and BIC (18.180), and (the logarithm of) the BD/BDe score (18.178) can be written in the form

$$\text{score}(\mathcal{G}) = \sum_{i=1}^N \text{score}_i(X_i, \mathbf{Pa}_{\mathcal{G}}(X_i)), \quad (18.181)$$

which means that the global network score decomposes into a sum of local scores, depending only on the individual variables and their parents. This property is very important to find efficient structure learning algorithms. For example, if an algorithm decides to add a parent to some variable  $X_i$ , then the overall score is affected only via the the change of a single additive term. Therefore, decomposability of the scoring function alleviates the problem of finding a score-maximizing graph  $\mathcal{G}$ . Unfortunately, even for decomposable scores the problem of finding a score-maximizing BN structure  $\mathcal{G}$  is NP-hard in general, even if the maximum number of parents is restricted to  $K \geq 2$  [22,23]. For  $K = 1$ , i.e., when the BN is constrained to be a directed tree, the Chow-Liu algorithm [24] learns the optimal network structure in polynomial time. Here the structure learning problem is reduced to a maximum spanning tree problem, where the edge weights are given by mutual information estimates. It is shown [24] that this is equivalent to maximizing the log-likelihood over the class of BNs with a directed tree structure. The class of directed trees is typically restrictive enough to avoid overfitting, i.e., the likelihood score is an appropriate choice in this case.

For learning more complicated structures, there exist a large variety of approximative techniques. One of the first general search algorithms was the K2 algorithm [21] which relies on an initial variable ordering, and greedily adds parents for each node, in order to increase the BD score. A more general scheme is hill-climbing (HC) [13,25], which does not rely on an initial variable ordering. HC starts with an unconnected graph  $\mathcal{G}$ . In each iteration, the current graph  $\mathcal{G}$  is replaced with a neighboring graph  $\mathcal{G}'$  with maximal score, if  $\text{score}(\mathcal{G}') > \text{score}(\mathcal{G})$ . Neighboring graphs are those graphs which

can be created from the current graph by single edge-insertions, edge-deletions and edge-reversals, maintaining acyclicity constraints. When no neighboring graph has higher score than the current graph, the algorithm stops. HC greedily increases the score, and will typically terminate in a local maximum. HC can be combined with tabu-search [26] in order to escape local maxima: When the algorithm is trapped in a local maximum, then the last several iterations leading to the local maximum are reverted and marked as forbidden. In this way, the algorithm is forced to use alternative iterations, possibly leading to a better local maximum. A related method is simulated annealing (SA) which randomly generates a neighbor solution in each step. If the new solution has higher score than the current one, it is immediately accepted. However, also a solution with lower score is accepted with a certain probability. While this probability is high in the beginning, leading to a large exploration over the search space, it is successively reduced according to a cooling schedule. At the end of the search process, SA only accepts score-increasing solutions. [25] applied SA for BN structure learning. While these heuristic methods perform a greedy search in the space of all DAGs, [27] proposed to greedily search over the space of equivalence classes. An equivalence class contains all BN structures which represent the same set of independence assumptions. A related approach can be found in [28], which proposes to perform a search over possible variable orderings. This approach uses the fact, as already mentioned in [21], that for a given variable ordering the optimal structure can be found in polynomial time.

Recently, several exact approaches for finding globally optimal structures have been proposed. Methods based on dynamic programming [29–31] have exponential time and memory requirements, which restricts their application to approximately 30 variables. Furthermore, there are branch-and-bound methods [32–34] which cast the combinatorial structure learning problem to a relaxed continuous problem. The relaxed continuous problem is successively divided (branched) into sub-problems, until the solutions of the relaxed sub-problems coincide with feasible structures, i.e., DAGs. The scores of the relaxed solutions provide an upper bound of the achievable score, and each feasible solution provides a lower bound of this score. Branches whose upper bound is lower than the currently best lower bound can consequently be pruned. Although branch-and-bound methods also have an exponential runtime, they provide two key advantages: (i) They maintain (after some initial time) a feasible solution, i.e., they can be prematurely terminated, returning the currently best solution. Methods based on dynamic programming do not offer this possibility; (ii) They provide upper and lower bounds on the maximal score, which represents a worst-case certificate of sub-optimality.

### 1.18.4.3 Discriminative learning in Bayesian networks

So far, we discussed generative learning of BNs, aiming to retrieve the underlying true distribution  $P^*(\mathbf{X})$  from a sample  $\mathcal{D} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)})$ . Now, let us assume that our final goal is classification, i.e., we aim to predict the value of a class variable  $C \in \mathbf{X}$ , given the values of the features  $\mathbf{Z} = \mathbf{X} \setminus \{C\}$ . Without loss of generality, we can assume that  $C$  is  $X_1$ , and we renumber the remaining variables, such that  $\mathbf{X} = \{C, \mathbf{Z}\} = \{C, X_1, X_2, \dots, X_{N-1}\}$ . In classification, we are interested in finding a decision function  $f(\mathbf{Z})$ , which can take values in  $\{1, \dots, \text{sp}(C)\}$ , such that the expected classification rate

$$\text{CR}(f) = \mathbb{E}_{P^*}[\mathbf{1}_{\{f(\mathbf{z})=c\}}] \quad (18.182)$$

$$= \sum_{\{c, \mathbf{z}\} \in \text{val}(\{C, \mathbf{Z}\})} P^*(c, \mathbf{z}) \mathbf{1}_{\{f(\mathbf{z})=c\}} \quad (18.183)$$

is maximized. It is easily shown that a function  $f^*(\mathbf{Z})$  maximizing (18.182) is given by [35]

$$f^*(\mathbf{Z}) = \arg \max_{c \in \text{val}(\mathcal{C})} P^*(c|\mathbf{Z}) = \arg \max_{c \in \text{val}(\mathcal{C})} P^*(c, \mathbf{Z}). \quad (18.184)$$

Therefore, since the generative approach aims to retrieve  $P^*(\mathbf{X})$ , we are in principle able to learn an optimal classifier, when (i) we use a sufficiently expressive model, (ii) we have a sufficient amount of training data, and (iii) we train the model using a generative approach, such as ML. Classifiers trained this way are called *generative classifiers*. Generative classifiers are amenable to interpretation and for incorporating prior information, and are flexible to versatile inference scenarios. However, we rarely have sufficient training data, and typically we restrict the model complexity, partly to avoid overfitting (which occurs due to a lack of training data), partly due to computational reasons. Therefore, although theoretically sound, generative classifiers usually do not yield the best classification results.

Discriminative approaches, on the other hand, directly represent aspects of the model which are important for classification accuracy. For example, some approaches model the class posterior probability  $P(C|\mathbf{Z})$ , while others, such as support vector machines (SVMs) [36, 37] or neural networks [4, 38], model information about the decision function, without using a probabilistic model. In each case, a discriminative objective does not necessarily agree with accurate joint distribution estimates, but rather aims for a low classification error on the training data. Discriminative models are usually restricted to the mapping from observed input features  $\mathbf{Z}$  to the unknown class output variable  $C$ , while inference of  $\mathbf{Z}$  for given  $C$  is impossible or unreasonable. There are several reasons for using discriminative rather than generative classifiers, one of which is that the classification problem should be solved most simply and directly, and never via a more general problem such as the intermediate step of estimating the joint distribution [39]. Discriminative classifiers typically lead to better classification performance, particularly when the class conditional distributions poorly approximate the true distribution [4]. The superior performance of discriminative classifiers has been reported in many application domains.

When using PGMs for classification, both parameters and structure can be learned using either generative or discriminative objectives [40]. In the generative approach, the central object is the parameter posterior  $P(\Theta|\mathcal{D})$ , yielding the MAP, ML and the Bayesian approach. In the discriminative approach, we use alternative objective functions such as conditional log-likelihood (CLL), classification rate (CR), or margin, which can be applied for both structure learning and for parameter learning. Unfortunately, while essentially all generative scores (likelihood, MDL/BIC, BDe) decompose into a sum over the nodes, it turns out that this does not happen for discriminative scores. Therefore, although many of the structure learning algorithms discussed for the generative case can also be applied for discriminative scores, the problem becomes computationally more expensive in the discriminative case. We restrict our discussion to BN, since most of the literature for discriminative learning is concerned about this type of PGMs.

*Conditional log-likelihood (CLL):* The log-likelihood over a model class  $\mathcal{C}$  can be written as

$$\log \mathcal{L}(\mathcal{C}; \mathcal{D}) = \sum_{l=1}^L \log P(\mathbf{x}^{(l)}|\mathcal{C}), \quad (18.185)$$

$$= \sum_{l=1}^L \log P(\mathbf{c}^{(l)}|\mathbf{z}^{(l)}, \mathcal{C}) + \sum_{l=1}^L \log P(\mathbf{z}^{(l)}|\mathcal{C}), \quad (18.186)$$

$$= \text{CLL}(\mathcal{C}; \mathcal{D}) + \log \mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}}), \quad (18.187)$$

where the log-likelihood decomposes into two terms. The term  $\log \mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}})$  is the log-likelihood of the attributes  $\mathbf{Z}$ , where the data set  $\mathcal{D}_{\mathbf{Z}}$  contains only the samples of the features  $\mathbf{Z}$ , without the values of  $C$ . The term  $\text{CLL}(\mathcal{C}; \mathcal{D})$  is called *conditional log-likelihood*, which is the log-likelihood of the class variable  $C$ , conditioned on the respective values of  $\mathbf{Z}$ . While  $\text{CLL}(\mathcal{C}; \mathcal{D})$  reflects how well some model  $M \in \mathcal{C}$  fits the empirical conditional distribution  $P_{\mathcal{D}}(C|\mathbf{Z})$ , the term  $\mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}})$  measures the fitness of  $M$  to represent the empirical distribution over the features  $P_{\mathcal{D}_{\mathbf{Z}}}(\mathbf{Z})$ . For discriminative learning, we are primarily interested in optimizing  $\text{CLL}(\mathcal{C}; \mathcal{D})$ , but not  $\mathcal{L}(\mathcal{C}; \mathcal{D}_{\mathbf{Z}})$ . Therefore, for discriminative learning we discard the second term in (18.187) and maximize the conditional likelihood:

$$\text{CLL}(\mathcal{C}; \mathcal{D}) = \sum_{l=1}^L \left[ \log P(c^{(l)}, \mathbf{z}^{(l)} | \mathcal{C}) - \log \sum_{c=1}^{|C|} P(c, \mathbf{z}^{(l)} | \mathcal{C}) \right]. \quad (18.188)$$

Suppose we found a model  $M^*$  maximizing the CLL, and we wish to classify a new sample  $\mathbf{z}$ . The decision function according to  $M^*$  is then

$$f_{M^*}(\mathbf{z}) = \arg \max_{c \in \text{val}(C)} P(c | \mathbf{z}, M^*). \quad (18.189)$$

However, although CLL is connected with classification rate, maximizing CLL is not the same as maximizing the classification rate on the training set. Friedman [41] shows that improving CLL even can be obstructive for classification. This can be understood in a wider context, when our goal is not to maximize the number of correct classifications, but to make an optimal decision based on the classification result. As an example given in [4], assume a system which classifies if a patient has a lethal disease, based on some medical measurements. There are 4 different combinations of the state of the patient and the system diagnosis, which have very different consequences: the scenario that a healthy patient is diagnosed as ill is not as disastrous as the scenario when an ill patient is diagnosed as healthy. For such a system it is not enough to return a 0/1-decision, but additionally a value of confidence has to be provided. These requirements are naturally met by maximum CLL learning. Generally, one can define a loss-function  $L(k, l)$  which represents the assumed loss when a sample with class  $c = k$  is classified as  $c = l$ . Maximizing CLL is tightly connected with minimizing the expected loss for a loss-function  $L(k, l)$  satisfying  $L(k, k) = 0$ , and  $L(k, l) \geq 0, k \neq l$ .

In [42], an algorithm for learning the BN parameters for a fixed structure  $\mathcal{G}$  was proposed. This algorithm uses gradient ascend to increase the CLL, while maintaining parameter feasibility, i.e., non-negativity and sum-to-one constraints. Although CLL is a concave function, these constraints are non-convex, such that the algorithm generally finds only a local maximum of the CLL. However, in [43, 44] it is shown that for certain network structures the parameter constraints can be neglected, yielding a convex optimization problem and thus a globally optimal solution. More precisely, the network structure  $\mathcal{G}$  has to fulfill the condition:

$$\forall Z \in \mathbf{Ch}_{\mathcal{G}}(C) : \exists X \in \mathbf{Pa}_{\mathcal{G}}(Z) : \mathbf{Pa}_{\mathcal{G}}(Z) \subseteq \mathbf{Pa}_{\mathcal{G}}(X) \cup \{X\}. \quad (18.190)$$

In words, every class child  $Z_i$  must have a parent  $X$ , which together with its own parents  $\mathbf{Pa}_{\mathcal{G}}(X)$  contain all parents  $\mathbf{Pa}_{\mathcal{G}}(Z_i)$ . It is shown that for structures fulfilling this condition, always a correctly

normalized BN with the same CLL objective value as for the unconstrained solution can be found. This implies that the obtained BN parameters globally maximize the CLL. In [45], a greedy hill-climbing algorithm for BN structure learning is proposed using CLL as scoring function.

*Empirical classification rate (CR):* Often one is interested in the special case of 0/1-loss, where each correctly classified sample has loss 0, while each misclassified sample causes a loss of 1. As pointed out in [41], maximum CLL training does not directly correspond to optimal 0/1-loss. An objective directly reflecting the 0/1-loss is the empirical classification rate

$$\text{CR}(\mathcal{C}; \mathcal{D}) = \frac{1}{L} \sum_{l=1}^L \mathbb{1}_{\{c^{(l)} = \arg \max_{c \in \text{val}(\mathcal{C})} P(c|\mathbf{z}^{(l)}, \mathcal{C})\}}.$$

Parameter learning using CR is hard, due to the non-differentiability and non-convexity of the function. In [40, 46], CR was used for greedy hill-climbing structure learning.

*Maximum margin (MM):* The probabilistic multi-class margin [47] for the  $l$ th sample can be expressed as

$$d_{\mathcal{C}}^{(l)} = \min_{c \neq c^{(l)}} \frac{P(c^{(l)}|\mathbf{z}^{(l)}, \mathcal{C})}{P(c|\mathbf{z}^{(l)}, \mathcal{C})} = \frac{P(c^{(l)}, \mathbf{z}^{(l)}|\mathcal{C})}{\max_{c \neq c^{(l)}} P(c, \mathbf{z}^{(l)}|\mathcal{C})}. \quad (18.191)$$

If  $d_{\mathcal{C}}^{(l)} > 1$ , then sample  $l$  is correctly classified and vice versa. The magnitude of  $d_{\mathcal{C}}^{(l)}$  is related to the confidence of the classifier about its decision. Taking the logarithm, we obtain

$$\log d_{\mathcal{C}}^{(l)} = \log P(c^{(l)}, \mathbf{z}^{(l)}|\mathcal{C}) - \max_{c \neq c^{(l)}} (\log P(c, \mathbf{z}^{(l)}|\mathcal{C})). \quad (18.192)$$

Usually, the maximum margin approach maximizes the margin of the sample with the smallest margin for a separable classification problem [37], i.e., we use the objective function

$$\text{MM}(\mathcal{C}; \mathcal{D}) = \min_{l=1, \dots, L} \log d_{\mathcal{C}}^{(l)}. \quad (18.193)$$

For non-separable problems, this is relaxed by introducing a so-called soft margin, which has been used for parameter learning [47, 48] and for structure learning [49, 50].

## 1.18.5 Inference

In the last section the basics of learning PGMs have been introduced. Now, assuming that the process of learning has already been performed, i.e., the model structure and parameterization are established, we move onto the task of *inference*. This task deals with assessing the marginal and/or most likely configuration of variables given observations. For this, the set of RVs  $\mathbf{X}$  in the PGM is partitioned into three mutually disjoint subsets  $\mathbf{O}$ ,  $\mathbf{Q}$  and  $\mathbf{H}$ , i.e.,  $\mathbf{X} = \mathbf{O} \cup \mathbf{Q} \cup \mathbf{H}$  and  $\mathbf{O} \cap \mathbf{Q} = \mathbf{O} \cap \mathbf{H} = \mathbf{Q} \cap \mathbf{H} = \emptyset$ . Here,  $\mathbf{O}$  denotes the set of *observed nodes*, i.e., the evidence variables,  $\mathbf{Q}$  denotes the set of *query variables*, and  $\mathbf{H}$  refers to the set of nodes which belong neither to  $\mathbf{O}$  or  $\mathbf{Q}$ , also known as latent or hidden variables.

There are two basic types of inference queries:

- i. *Marginalization query*: This first type of query infers the marginal distribution of the query variables conditioned on the observation  $\mathbf{O}$ , i.e., it computes

$$P(\mathbf{Q}|\mathbf{O} = \mathbf{o}) = \frac{P(\mathbf{Q}, \mathbf{O} = \mathbf{o})}{P(\mathbf{O} = \mathbf{o})}. \quad (18.194)$$

The terms  $P(\mathbf{Q}, \mathbf{O} = \mathbf{o})$  and  $P(\mathbf{O} = \mathbf{o})$  can be determined by marginalization over  $\mathbf{H}$  and  $\mathbf{H} \cup \mathbf{Q}$  of the joint probability  $P(\mathbf{X}) = P(\mathbf{O}, \mathbf{Q}, \mathbf{H})$  represented by the PGM, respectively. That is,

$$P(\mathbf{Q}, \mathbf{O} = \mathbf{o}) = \sum_{\mathbf{h} \in \text{val}(\mathbf{H})} P(\mathbf{O} = \mathbf{o}, \mathbf{Q}, \mathbf{H} = \mathbf{h}), \quad \text{and} \quad (18.195)$$

$$P(\mathbf{O} = \mathbf{o}) = \sum_{\mathbf{q} \in \text{val}(\mathbf{Q})} P(\mathbf{O} = \mathbf{o}, \mathbf{Q} = \mathbf{q}). \quad (18.196)$$

**Example 23.** Assume a PGM used by a car insurance company to calculate the insurance fees for individuals. Then, a query could ask for the probabilities of certain damage sums (this corresponds to query variables) arising for the case of a driver with age 25 and no children (this corresponds to evidence variables). Other variables in the model, such as e.g., the number of accidents in the past, are marginalized out since they are not observed.

- ii. *Maximum a posteriori query*: This query determines the most likely instantiation of the query variables given some evidence, i.e., it computes

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} P(\mathbf{Q} = \mathbf{q} | \mathbf{O} = \mathbf{o}). \quad (18.197)$$

This is equivalent to

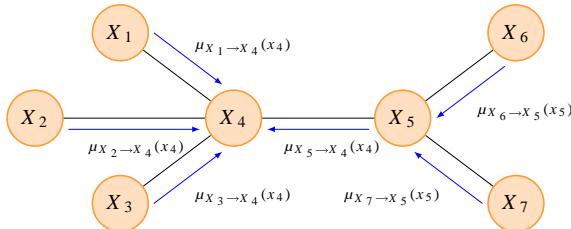
$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \text{val}(\mathbf{H})} P(\mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h} | \mathbf{O} = \mathbf{o}), \quad (18.198)$$

$$= \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \text{val}(\mathbf{H})} P(\mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h}, \mathbf{O} = \mathbf{o}). \quad (18.199)$$

Note, that the most likely instantiation of the joint variables  $\mathbf{q}^*$  does in general not correspond to the states obtained by selecting the most likely instantiation for all variables in  $\mathbf{Q}$  individually. An example for this is shown in [1], Chapter 2.

**Example 24.** In hidden Markov models (HMMs) this amounts to finding the most likely state sequence  $\mathbf{q}^*$  explaining the given observation sequence. This state sequence is determined by applying the Viterbi algorithm as shown in Section 1.18.6.1.

While both inference queries can in principle be answered by directly evaluating the sums in the corresponding equations, this approach becomes intractable for PGMs with many variables. This is due to the number of summands growing exponentially with the number of variables. Therefore, more

**FIGURE 18.21**

Message passing in a tree-shaped MN  $\mathcal{M}$  to determine the marginal  $P_{\mathcal{M}}(X_4)$ .

efficient inference methods exploiting the structure of the underlying PGMs have been developed; The exact solution to (i) can be found using the *sum-product algorithm*, also known as *belief propagation* or *message passing algorithm* [2, 51], assuming tree-structured graphical models, and (ii) can be calculated by applying the *max-product* or in the log-domain the *max-sum* algorithm. In this tutorial, the focus is on marginalization queries  $P(\mathbf{Q}|\mathbf{O})$ . Details on MAP queries are provided in [1], Chapter 9.

The remainder of this section is structured as follows: In Section 1.18.5.1 the sum-product algorithm is derived and the junction tree algorithm for exact inference in arbitrary graphs is introduced. In cases, where exact inference is computationally too demanding, approximate inference techniques can be used. A short introduction to these techniques is given in Section 1.18.5.2.

### 1.18.5.1 Exact inference

Given some evidence  $\mathbf{o} \in \text{val}(\mathbf{O})$ , the direct calculation of the marginal in (18.196) of the joint probability distribution

$$P(\mathbf{O} = \mathbf{o}) = \sum_{\mathbf{q} \in \text{val}(\mathbf{Q})} \sum_{\mathbf{h} \in \text{val}(\mathbf{H})} P(\mathbf{O} = \mathbf{o}, \mathbf{Q} = \mathbf{q}, \mathbf{H} = \mathbf{h}) \quad (18.200)$$

expands to  $\prod_{i=1}^{|\mathbf{Q}|} \text{sp}(\mathbf{Q}_i) \prod_{i=1}^{|\mathbf{H}|} \text{sp}(\mathbf{H}_i)$  summations. Hence, assuming that each variable can be in one of  $k$  states the evaluation of Eq. (18.200) requires  $\mathcal{O}(k^{|\mathbf{Q}|+|\mathbf{H}|})$  summations, i.e., the evaluation of a sum with exponentially many terms in the number of variables. This renders the direct computation intractable for large values of  $k$  and many variables. Fortunately, the underlying graph structure, and the thereby introduced factorization of the joint probability, can be exploited. This is demonstrated in the following example:

**Example 25.** Consider the MN  $\mathcal{M}$  in Figure 18.21. The joint probability factorizes according to the maximal cliques  $\mathbf{C}_1 = \{X_1, X_4\}$ ,  $\mathbf{C}_2 = \{X_2, X_4\}$ ,  $\mathbf{C}_3 = \{X_3, X_4\}$ ,  $\mathbf{C}_4 = \{X_4, X_5\}$ ,  $\mathbf{C}_5 = \{X_5, X_6\}$ , and  $\mathbf{C}_6 = \{X_5, X_7\}$  as

$$P_{\mathcal{M}}(X_1, X_2, X_3, X_4, X_5, X_6, X_7) = \frac{1}{Z} \Psi_{\mathbf{C}_1}(X_1, X_4) \Psi_{\mathbf{C}_2}(X_2, X_4) \Psi_{\mathbf{C}_3}(X_3, X_4) \cdot \quad (18.201)$$

$$\Psi_{\mathbf{C}_4}(X_4, X_5) \Psi_{\mathbf{C}_5}(X_5, X_6) \Psi_{\mathbf{C}_6}(X_5, X_7). \quad (18.202)$$

Suppose, we are interested in determining the marginal probability  $P_{\mathcal{M}}(X_4)$ . Then,

$$\begin{aligned}
 P_{\mathcal{M}}(X_4) &= \sum_{x_1} \cdots \sum_{x_3} \sum_{x_5} \cdots \sum_{x_7} P_{\mathcal{M}}(X_1, X_2, X_3, X_4, X_5, X_6, X_7) \\
 &= \frac{1}{Z} \left[ \underbrace{\sum_{x_1 \in \text{val}(X_1)} \Psi_{C_1}(x_1, x_4)}_{=\mu_{X_1 \rightarrow X_4}(x_4)} \right] \left[ \underbrace{\sum_{x_2 \in \text{val}(X_2)} \Psi_{C_2}(x_2, x_4)}_{=\mu_{X_2 \rightarrow X_4}(x_4)} \right] \left[ \underbrace{\sum_{x_3 \in \text{val}(X_3)} \Psi_{C_3}(x_3, x_4)}_{=\mu_{X_3 \rightarrow X_4}(x_4)} \right] \\
 &\quad \cdot \left[ \underbrace{\sum_{x_5 \in \text{val}(X_5)} \Psi_{C_4}(x_4, x_5)}_{=\mu_{X_5 \rightarrow X_4}(x_4)} \left[ \underbrace{\sum_{x_6 \in \text{val}(X_6)} \Psi_{C_5}(x_5, x_6)}_{=\mu_{X_6 \rightarrow X_5}(x_5)} \right] \left[ \underbrace{\sum_{x_7 \in \text{val}(X_7)} \Psi_{C_6}(x_5, x_7)}_{=\mu_{X_7 \rightarrow X_5}(x_5)} \right] \right] \\
 &= \frac{1}{Z} \mu_{X_1 \rightarrow X_4}(x_4) \mu_{X_2 \rightarrow X_4}(x_4) \mu_{X_3 \rightarrow X_4}(x_4) \mu_{X_5 \rightarrow X_4}(x_4),
 \end{aligned}$$

where the terms  $\mu_{X_i \rightarrow X_j}(x_j)$  denote so-called *messages* from  $X_i$  to  $X_j$ . The grouping of the factors with the corresponding sums essentially exploits the distributive law [52] and reduces the computational burden. In this example, we require  $\mathcal{O}(k^2)$  summations and multiplications assuming that  $\text{sp}(X_i) = k$  for all variables  $X_i$ . Direct calculation of  $P_{\mathcal{M}}(X_4)$  would instead require  $\mathcal{O}(k^{N-1})$  arithmetic operations, where  $N = 7$ . The normalization constant  $Z$  can be determined by summing the product of incoming messages over  $x_4$ , i.e.,

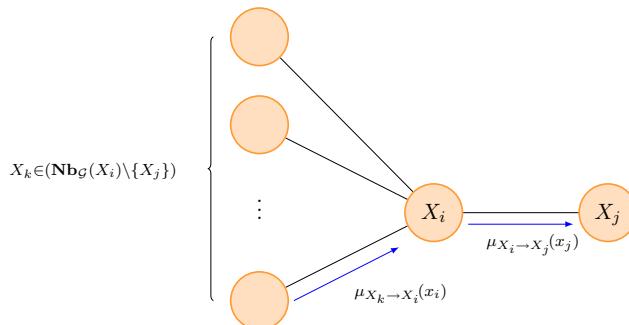
$$Z = \sum_{x_4 \in \text{val}(X_4)} \mu_{X_1 \rightarrow X_4}(x_4) \mu_{X_2 \rightarrow X_4}(x_4) \mu_{X_3 \rightarrow X_4}(x_4) \mu_{X_5 \rightarrow X_4}(x_4). \quad (18.203)$$

In the sequel, we generalize the example above to pairwise MNs, while inference in more general MNs, BNs and FGs is covered later in this section. This generalization leads to the *sum-product rule*: Messages  $\mu_{X_i \rightarrow X_j}(x_j)$  are sent on each edge  $(X_i - X_j) \in \mathbf{E}$  from  $X_i$  to  $X_j$ . These messages are updated according to

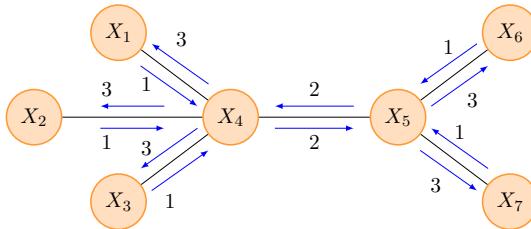
$$\mu_{X_i \rightarrow X_j}(x_j) = \sum_{x_i \in \text{val}(X_i)} \Psi_{\{X_i, X_j\}}(x_i, x_j) \prod_{X_k \in (\text{Nb}_{\mathcal{G}}(X_i) \setminus \{X_j\})} \mu_{X_k \rightarrow X_i}(x_i), \quad (18.204)$$

i.e., the new message  $\mu_{X_i \rightarrow X_j}(x_j)$  is calculated as the product of the *incoming* messages with the potential function  $\Psi_{\{X_i, X_j\}}$  and summation over all variables except  $X_j$ . This message update is sketched in Figure 18.22.

The *sum-product algorithm* schedules the messages such that an updated message is sent from  $X_i$  to  $X_j$  when  $X_i$  has received the messages from all its neighbors except  $X_j$ , i.e., from all nodes in  $\text{Nb}_{\mathcal{G}}(X_i) \setminus \{X_j\}$  [53].

**FIGURE 18.22**

Sum-product update rule.

**FIGURE 18.23**

Message update of the sum-product algorithm.

**Example 26.** This message update schedule of the sum-product algorithm for our previous example is illustrated in Figure 18.23. The sent messages are illustrated as arrows in the figure, where the numbers next to the arrows indicate the iteration count at which message updates are sent to neighboring nodes. After three iterations the updates converge for all nodes in the graph.

The sum-product algorithm is *exact* if the MN is free of cycles [2], i.e., if it is a tree. *Exact* means that the marginals  $P(X_i)$  (also called *beliefs*) obtained from the sum-product algorithm according to

$$P(X_i) = \frac{1}{Z} \prod_{X_k \in \text{Nb}_{\mathcal{G}}(X_i)} \mu_{X_k \rightarrow X_i}(x_i) \quad (18.205)$$

are guaranteed to converge to the true marginals. Again, considering the graph in Figure 18.23, each clique factor  $\Psi_C$  includes two variables. Consequently, the complexity of the sum-product algorithm is  $\mathcal{O}(k^2)$  assuming that  $\text{sp}(X_i) = k$  for all variables  $X_i$ .

So far, we are able to obtain the marginals  $P(X_i)$  for any  $X_i \in \mathbf{Q}$  for tree-shaped MNs. However, the aim is to determine the marginal distribution over the query variable  $Q_i \in \mathbf{Q}$  conditioned on the evidence  $\mathbf{O} = \mathbf{o}$ . This requires the computation of  $P(Q_i, \mathbf{o})$ , cf. Eq. (18.194) assuming only one query

variable. The evidence is introduced to each clique factor  $\Psi_C$  that depends on evidence nodes, i.e., to each  $\Psi_C$  for which there is an  $O_i \in \mathbf{C}$  that is also in  $\mathbf{O}$ . Therefore, we identify all factors  $\Psi_C$  with  $O_i \in \mathbf{C}$  for any  $O_i \in \mathbf{O}$  and determine the new factor by multiplying  $\Psi_C$  with the indicator function  $\mathbb{1}_{\{O_i=o_i\}}$ , i.e.,  $\Psi_C(\cdot)$  is modified according to  $\Psi_C(\cdot)\mathbb{1}_{\{O_i=o_i\}}$ . In other words, factors are set to zero for instantiations inconsistent with  $\mathbf{O} = \mathbf{o}$ . Interestingly, it turns out that running the sum-product algorithm on the updated factors leads to an unnormalized representation of  $P(Q_i, \mathbf{o})$ . The regular  $P(Q_i|\mathbf{o})$  is obtained by normalizing  $P(Q_i, \mathbf{o})$  by  $Z = \sum_{q_i \in \text{val}(Q_i)} P(Q_i = q_i, \mathbf{o})$  [53,54].

*Message Passing in FGs:* For FGs, the sum-product algorithm for determining the marginal distribution of the variable  $X_i$ , given all the observations can be formulated analogously [55]. The observations are absorbed into the factors instead of the clique factors. Neighboring nodes communicate with each other: whenever a node, either a variable or factor node, has received all messages except on one edge, it sends a new message to its neighbor over this edge. At the beginning, messages from variable nodes are initialized to 1. The message from variable node  $X$  to factor node  $F$  is

$$\mu_{X \rightarrow F}(x) = \prod_{F_k \in (\mathbf{Nb}_G(X) \setminus \{F\})} \mu_{F_k \rightarrow X}(x), \quad (18.206)$$

while a message from factor node  $F$  to variable node  $X$  is computed as

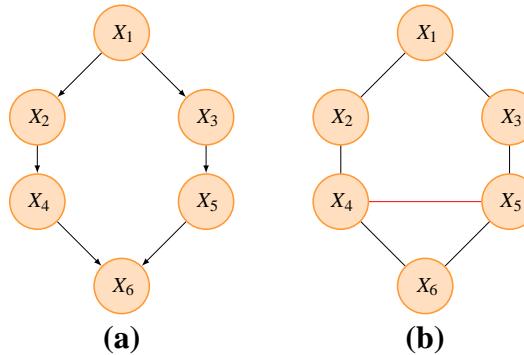
$$\mu_{F \rightarrow X}(x) = \sum_{\mathbf{Nb}_G(F) \setminus \{X\}} \left( f(\mathbf{Nb}_G(F)) \prod_{X_k \in (\mathbf{Nb}_G(F) \setminus \{X\})} \mu_{X_k \rightarrow F}(X_k) \right). \quad (18.207)$$

The marginal at each variable node can be determined as  $P(X_i) = \frac{1}{Z} \prod_{F \in \mathbf{Nb}_G(X_i)} \mu_{F \rightarrow X_i}(X_i)$ , where  $Z$  is a normalization constant.

So far, we have shown how to efficiently compute the marginals of a factorized joint probability distribution given some evidence in the case of tree-structured MNs and FGs. Generally, the sum-product algorithm is exact only if performed on a graphical model that is a tree. The remaining question is how to perform inference in BNs and arbitrarily structured BNs and MNs?

*Message passing in arbitrarily structured BNs and MNs:* Inference in BNs can be performed by first transforming the BN into an MN. Inference is then performed on this transformed model. The transformation of the model is known as *moralization*. Further, any arbitrary directed (and undirected) graphical model can be transformed into an undirected tree (*junction tree* or *clique tree*) where each node in the tree corresponds to a subset of variables (cliques) [3,53,54,56–59]. This is done with the goal of performing inference on this tree, exploiting that inference on trees is exact. The transformation of an arbitrary graph to a junction tree involves the following three steps, where the first step is only required when transforming a BN to an MN:

- i. *Moralization:* In the case of a directed graphical model, undirected edges are added between all pairs of parents  $\mathbf{Pa}_G(X_i)$  having a common child  $X_i$  for all  $X_i \in \mathbf{X}$ . This is known as moralization [54]. The *moral graph* is then obtained by dropping the directions of the edges. Moralization essentially results in a loss of conditional independence statements represented by the original graph. However, this is necessary for representing the original factorization by the undirected graph.

**FIGURE 18.24**

Moralization example. (a) BN to be transformed to an MN. (b) Moral graph; the added edge is red.

**Example 27 (Moralization).** The joint probability of the BN  $\mathcal{B}$  in Figure 18.24a is

$$P_{\mathcal{B}}(X_1, \dots, X_6) = P_{X_1}(X_1)P_{X_2}(X_2|X_1)P_{X_3}(X_3|X_1)P_{X_4}(X_4|X_2)P_{X_5}(X_5|X_3)P_{X_6}(X_6|X_4, X_5). \quad (18.208)$$

The corresponding moral graph is shown in Figure 18.24b. It factorizes as a product of the factors of the maximal cliques  $\mathbf{C}_1 = \{X_1, X_2\}$ ,  $\mathbf{C}_2 = \{X_1, X_3\}$ ,  $\mathbf{C}_3 = \{X_2, X_4\}$ ,  $\mathbf{C}_4 = \{X_3, X_5\}$ , and  $\mathbf{C}_5 = \{X_4, X_5, X_6\}$  as

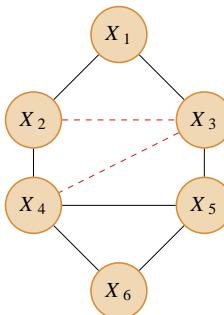
$$P_{\mathcal{M}}(X_1, \dots, X_6) = \Psi_{\mathbf{C}_1}(X_1, X_2)\Psi_{\mathbf{C}_2}(X_1, X_3)\Psi_{\mathbf{C}_3}(X_2, X_4)\Psi_{\mathbf{C}_4}(X_3, X_5)\Psi_{\mathbf{C}_5}(X_4, X_5, X_6). \quad (18.209)$$

Each CPD  $P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$  of the BN is mapped into the factor of exactly one clique, i.e., each clique factor is initialized to one and then each  $P_{X_i}(X_i|\mathbf{Pa}_{\mathcal{G}}(X_i))$  is multiplied into a clique factor which contains  $\{X_i\} \cup \mathbf{Pa}_{\mathcal{G}}(X_i)$ . In this example, this results in the following factors:  $\Psi_{\mathbf{C}_1}(X_1, X_2) = P_{X_1}(X_1)P_{X_2}(X_2|X_1)$ ,  $\Psi_{\mathbf{C}_2}(X_1, X_3) = P_{X_3}(X_3|X_1)$ ,  $\Psi_{\mathbf{C}_3}(X_2, X_4) = P_{X_4}(X_4|X_2)$ ,  $\Psi_{\mathbf{C}_4}(X_3, X_5) = P_{X_5}(X_5|X_3)$ , and  $\Psi_{\mathbf{C}_5}(X_4, X_5, X_6) = P_{X_6}(X_6|X_4, X_5)$ . The normalization factor of  $P_{\mathcal{M}}(X_1, \dots, X_6)$  in this case is  $Z = 1$ .

- ii. *Triangulation:* The next step in converting an arbitrary MN into a junction tree is the triangulation of the moral graph. The *triangulated graph* is sometimes called *chordal graph*.

**Definition 28 (Triangulation [54]).** A graph  $\mathcal{G}$  is *triangulated* if there is no cycle of length  $\geq 4$  without an edge joining two non-neighboring nodes.

The triangulated graph can be found by the *elimination algorithm* [1] that eliminates one variable from the graph in each iteration: Assuming a given *elimination order* of the variables the first node from the ordering is selected and in the moral graph all pairs of neighbors of that node are connected. The resulting graph is collected in a set of graphs  $\mathcal{G}_e$ . Afterwards, this node and all its edges are removed from the graph. Then, the next node in the elimination ordering is selected for elimination. We proceed in the same way as above until all nodes have been eliminated. The graph resulting from the union

**FIGURE 18.25**

Triangulated graph; added edges are dashed.

of all the graphs in  $\mathcal{G}_e$  is triangulated. Depending on the ordering, there typically exist many different triangulations. For efficient inference, we would like to choose a ordering such that e.g., the state-space size of the largest clique is minimal. Finding the optimal elimination ordering is NP-hard [1]. Greedy algorithms for determining an ordering so that the induced graph is close to optimal are summarized in [1], Chapter 9.

**Example 28 (Triangulation).** An example of a triangulated graph obtained from Figure 18.24b with elimination ordering  $X_6, X_5, X_4, X_3, X_2, X_1$  is shown in Figure 18.25. The elimination of node  $X_5$  and  $X_4$  introduces the edges  $(X_3 - X_4)$  and  $(X_2 - X_3)$ , respectively. The dashed edges are added to triangulate the moral graph.

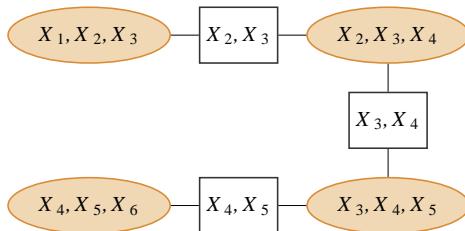
Each maximal clique of the moral graph is contained in the triangulated graph  $\mathcal{G}_t$ . Using a similar mechanism as in Example 27, we can map the factors of the moral graph to the factors of the triangulated graph. The cliques in  $\mathcal{G}_t$  are maximal cliques and

$$P_{\mathcal{M}}(\mathbf{X}) = \frac{1}{Z} \prod_{\mathbf{C} \in \mathcal{G}_t} \Psi_{\mathbf{C}}(\mathbf{C}). \quad (18.210)$$

For efficiency, the cliques of the triangulated graph can be identified during variable elimination; A node which is eliminated from the moral graph forms a clique with all neighboring nodes, when connecting all pairs of neighbors. If this clique is not contained in a previously determined maximal clique, it is a maximal clique in  $\mathcal{G}_t$ .

**Example 29.** The maximal cliques of the triangulated graph in Figure 18.25 are  $\{X_4, X_5, X_6\}$ ,  $\{X_3, X_4, X_5\}$ ,  $\{X_2, X_3, X_4\}$ , and  $\{X_1, X_2, X_3\}$ .

- iii. *Building a junction tree:* In the last step, the maximal cliques are treated as nodes and are connected to form a tree. The junction tree has to satisfy the *running intersection property*, i.e., an RV  $X_i$  present in two different cliques  $\mathbf{C}_i$  and  $\mathbf{C}_j$ ,  $i \neq j$ , has to be also in each clique on the path

**FIGURE 18.26**

Junction tree.

connecting these two cliques.<sup>2</sup> The cliques in the junction tree are connected by a *separator set*  $\mathbf{S} = \mathbf{C}_i \cap \mathbf{C}_j$ . The number of variables in  $\mathbf{S}$  determines the edge weight for joining these two cliques in the junction tree. These weights can be used to find the junction tree using the *maximum spanning tree* algorithm [60]. The tree with maximal sum of the weights is guaranteed to be a junction tree satisfying the running intersection property.<sup>3</sup>

**Example 30 (Junction Tree).** A junction tree of the example in Figure 18.25 with separators (square nodes) is shown in Figure 18.26.

Once the junction tree is determined, message passing between the cliques is performed for inference. This essentially amounts to the sum-product algorithm. First, the potential of a separator  $\Psi_{\mathbf{S}}(\mathbf{S})$  is initialized to unity. The clique potential of neighboring cliques  $\mathbf{C}_i$  and  $\mathbf{C}_j$  is  $\Psi_{\mathbf{C}_i}(\mathbf{C}_i)$  and  $\Psi_{\mathbf{C}_j}(\mathbf{C}_j)$ , respectively. The message passed from  $\mathbf{C}_i$  to  $\mathbf{C}_j$  is determined as

$$\begin{aligned}\tilde{\Psi}_{\mathbf{S}}(\mathbf{S}) &\leftarrow \sum_{\mathbf{C}_i \setminus \mathbf{S}} \Psi_{\mathbf{C}_i}(\mathbf{C}_i), \quad \text{and} \\ \Psi_{\mathbf{C}_j}(\mathbf{C}_j) &\leftarrow \frac{\tilde{\Psi}_{\mathbf{S}}(\mathbf{S})}{\Psi_{\mathbf{S}}(\mathbf{S})} \Psi_{\mathbf{C}_j}(\mathbf{C}_j),\end{aligned}$$

where the sum is over all variables in  $\mathbf{C}_i$  which are not in  $\mathbf{S}$ .

The message passing schedule is similar as for tree-structured models, i.e., a clique sends a message to a neighbor after receiving all the messages from its other neighbors. Furthermore, the introduction of evidence in the junction tree can be performed likewise as above. After each node has sent and received a message, the junction tree is *consistent* and the marginals  $P(Q_i, \mathbf{o})$  can be computed by identifying the clique potential  $\Psi_{\mathbf{C}_i}(\mathbf{C}_i)$  which contains  $Q_i$  and summing over  $\mathbf{C}_i \setminus \{Q_i\}$  as

$$P(Q_i, \mathbf{o}) = \sum_{\mathbf{C}_i \setminus \{Q_i\}} \Psi_{\mathbf{C}_i}(\mathbf{C}_i).$$

<sup>2</sup>This property is important for the message passing algorithm to achieve local consistency between cliques. Local consistency means that  $\Psi_{\mathbf{S}}(\mathbf{S}) = \sum_{\mathbf{C}_i \setminus \mathbf{S}} \Psi_{\mathbf{C}_i}(\mathbf{C}_i)$  for any  $\mathbf{C}_i$  and neighboring separator set  $\mathbf{S}$ . Because of this property, local consistency implies global consistency [1].

<sup>3</sup>A proof of this result is provided in [61].

The computational complexity of exact inference in discrete networks is exponential in the size of the largest clique (the width of the tree<sup>4</sup>). Hence, there is much interest in finding an optimal triangulation. Nevertheless, exact inference is intractable in large and dense PGMs and approximation algorithms are necessary.

### 1.18.5.2 Approximate inference

In the past decade, the research in PGMs has focused on the development of efficient *approximate inference* algorithms. While exact inference algorithms provide a satisfactory solution to many inference and learning problems, there are large-scale problems where a recourse to approximate inference is inevitable. In the following, we provide a brief overview of the most popular approximate inference methods. Details can be found in [1, 4, 62].

#### 1.18.5.2.1 Variational methods

The underlying idea in variational methods [4, 62–65] is the approximation of the posterior probability distribution  $P(\mathbf{Q}|\mathbf{O})$ , assuming  $\mathbf{H} = \emptyset$ , by a tractable surrogate distribution  $g(\mathbf{Q})$ , where we assume that the evaluation of  $P(\mathbf{Q}, \mathbf{O})$  is computational less expensive than that of  $P(\mathbf{Q}|\mathbf{O})$  and  $P(\mathbf{O})$ . The logarithm of  $P(\mathbf{O})$  can be decomposed as

$$\ln P(\mathbf{O}) = \underbrace{\sum_{\mathbf{q}} g(\mathbf{q}) \ln \left[ \frac{P(\mathbf{O}, \mathbf{q})}{g(\mathbf{q})} \right]}_{\text{LB}(g)} + \underbrace{\left\{ - \sum_{\mathbf{q}} g(\mathbf{q}) \ln \left[ \frac{P(\mathbf{q}|\mathbf{O})}{g(\mathbf{q})} \right] \right\}}_{\text{KL}(g||P)},$$

where the term  $\text{LB}(g)$  denotes a lower bound for  $\ln P(\mathbf{O})$ , i.e.,  $\text{LB}(g) \leq \ln P(\mathbf{O})$ , and  $\text{KL}(g||P)$  denotes the Kullback-Leibler divergence between  $g(\mathbf{Q})$  and  $P(\mathbf{Q}|\mathbf{O})$ . Since  $\ln P(\mathbf{O})$  does not depend on  $g(\mathbf{q})$ , and  $\text{LB}(g)$  and  $\text{KL}(g||P)$  are non-negative, maximizing  $\text{LB}(g)$  amounts to minimizing  $\text{KL}(g||P)$ . Hence, maximizing  $\text{LB}(g)$  with respect to  $g(\mathbf{q})$  results in the best approximation of the posterior  $P(\mathbf{Q}|\mathbf{O})$ .

Alternatively, the lower bound can be determined as

$$\ln P(\mathbf{O}) = \ln \sum_{\mathbf{q}} P(\mathbf{q}, \mathbf{O}) = \ln \sum_{\mathbf{q}} g(\mathbf{q}) \frac{P(\mathbf{q}, \mathbf{O})}{g(\mathbf{q})} \geq \sum_{\mathbf{q}} g(\mathbf{q}) \ln \left[ \frac{P(\mathbf{O}, \mathbf{q})}{g(\mathbf{q})} \right] = \text{LB}(g), \quad (18.211)$$

where we applied Jensen's inequality [5]. Furthermore, the lower bound  $\text{LB}(g)$  can be derived by using convex duality theory [65].

In variational approaches,  $g(\mathbf{Q})$  is restricted to simple tractable distributions. The simplest possibility, also called *mean-field approximation*, assumes that  $g(\mathbf{Q})$  factorizes as

$$g(\mathbf{Q}) = \prod_{i=1}^{|\mathbf{Q}|} g_i(Q_i). \quad (18.212)$$

---

<sup>4</sup>The tree-width of a graph is defined as the size (i.e., number of variables) of the largest clique of the moralized and triangulated directed graph minus one.

The variational inference approach can be combined with exact inference applied on parts of the graph. In particular, exact inference is performed on tractable substructures. This is called *structured variational approximation* [63], where much of the structure of the original graph is preserved.

### 1.18.5.2.2 Loopy message passing

While message passing algorithms can be shown to be exact on PGMs with tree structure, convergence of these algorithms on arbitrary graphs with cycles (loops) is not guaranteed. Moreover, even after convergence, the resulting solution might be only an approximation of the exact solution. Surprisingly, message passing on *loopy graphs* often converges to stable posterior/marginal probabilities. The most significant breakthrough came with the insight that for certain graph structures the fixed points of the message passing algorithm are actually the stationary points of the Bethe free energy [66]. This clarified the nature of message passing and led to more efficient algorithms. Further, this established a bond to a large body of physics literature and generalized belief propagation (GBP) algorithms [67] were developed. The GBP algorithms operate on regions of nodes and messages are passed between these regions. In contrast to GBP, Yuille [68] proposes a concave-convex procedure to directly optimize the Bethe free energy. Experiments on spin class configurations show that this method is stable, converges rapidly, and leads to even better solutions than message passing and GBP in some cases.

Convergence of loopy message passing has been experimentally confirmed for many applications (the maybe most popular being “turbo codes” [69, 70]). There has also been a lot of theoretical work investigating convergence properties of loopy message passing [67, 71–73]. Recently, theoretical conditions guaranteeing convergence of loopy message passing to a unique fixed point have been proposed in [74]. Minka [75] showed that many of the proposed approximate inference approaches, such as variational message passing, loopy message passing, expectation propagation amongst others can be viewed as instances of minimizing particular information divergences.

### 1.18.5.2.3 Sampling Methods

Sampling methods are computational tractable methods aiming at computing the quantities of interest by means of Monte Carlo procedures. One of the simplest of such methods is known as *importance sampling* and *sampling importance resampling* [76] for estimating expectations of functions. There are severe limitations of importance sampling in high dimensional sample spaces. However, Markov Chain Monte Carlo methods scale well with the dimensionality. Special cases are Gibbs sampling and the *Metropolis-Hastings* algorithm (see, e.g., [77, 78] for an introduction). One of the most prominent application of Monte Carlo methods (i.e., sequential importance sampling) are *particle filters* [79, 80] for online, non-linear and non-Gaussian tracking, where the posterior distribution is represented by a finite set of particles (samples). Particle filters generalize traditional Kalman filters which assume that the evolution of the state space is linear and that probability densities are Gaussian. Classical particle filters solve the inference task of computing the posterior  $P(\mathbf{Q}|\mathbf{O} = \mathbf{o})$  for a limited family of graph structures such as Kalman filters. Recently, Koller et al., Sudderth et al., and Ihler and McAllester [81–83] extended particle methods to solve inference problems defined on general graphical model structures.

## 1.18.6 Applications

PGMs have a long tradition in modeling uncertainty in intelligent systems and are important in many application areas such as computer vision [84–87], speech processing [88, 89], time-series and sequential modeling [90], cognitive science [91], bioinformatics [92], probabilistic robotics [93], signal processing, communications and error-correcting coding theory [55, 70, 76, 94], and in the area of artificial intelligence.

In this tutorial, we restrict ourselves to present some typical applications for each of the three PGM representations, namely for BNs, MNs and FGs. The first application is about dynamic BNs for time-series modeling of speech signals, followed by BNs for expert systems. Then, we present MRFs for image analysis, and FGs for decoding error-correcting codes.

### 1.18.6.1 Dynamic BNs for speech processing and time-series modeling

Speech recognition is the task of retrieving the sequence of words from a speech recording. The main components of an automatic speech recognition (ASR) system are:

1. *Feature extraction*: A feature sequence  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$  is determined by extracting  $T$  feature vectors from the audio signal. Commonly, mel-cepstral coefficients, delta, and double delta features [89] are derived from overlapping windows of  $\sim 25$  ms length of the speech signal.
2. *Recognition*: Based on the extracted features, a sequence of  $N$  words  $\mathbf{w} = (w_1, \dots, w_N)$  is determined.

The most likely word sequence  $\mathbf{w}^*$  given some sequence of feature vectors can be determined by the posterior probability  $P(\mathbf{w}|\mathbf{X})$  as

$$\mathbf{w}^* = \arg \max_{\mathbf{w} \in \mathcal{W}} P(\mathbf{w}|\mathbf{X}), \quad (18.213)$$

$$= \arg \max_{\mathbf{w} \in \mathcal{W}} \frac{P(\mathbf{X}|\mathbf{w}) P(\mathbf{w})}{P(\mathbf{X})}, \quad (18.214)$$

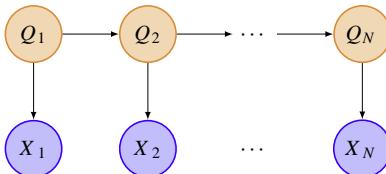
$$= \arg \max_{\mathbf{w} \in \mathcal{W}} P(\mathbf{X}|\mathbf{w}) P(\mathbf{w}), \quad (18.215)$$

where  $\mathcal{W}$  denotes the set of all possible word sequences. The *acoustic model* probability  $P(\mathbf{X}|\mathbf{w})$  for an utterance  $\mathbf{w}$  can be modeled as the concatenation of HMMs representing acoustic units such as words. In practice, it might even be beneficial to model subunits of words, e.g., bi- or triphones<sup>5</sup> or syllable units.

An HMM is a (hidden) Markov chain over states  $\{Q_1, \dots, Q_N\}$  and observations  $\{X_1, \dots, X_N\}$ . The states generate the observations such that observation  $X_i$  stochastically depends only on state  $Q_i$ , i.e.,

$$P(X_i|Q_1, \dots, Q_N, X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_N) = P(X_i|Q_i). \quad (18.216)$$

<sup>5</sup>In linguistics, a biphone (triphone) is a sequence of two (three) phonemes.

**FIGURE 18.27**

Hidden Markov Model.

The number of states of  $Q_i$  and  $X_i$  (considering discrete observations) is the same for each  $i \in \{1, \dots, N\}$ . An HMM can be represented as *dynamic BN* shown in Figure 18.27. In the case of Gaussian distributed state variables and observation probabilities  $P(X_i|Q_i)$ , this model structure is known as *Kalman filter*.

In the context of HMMs, we usually have to deal with three fundamental problems: (i) Evaluation problem; (ii) Decoding problem; and (iii) Learning (estimation) problem. Approaches to each of these problems for discrete observation variables are discussed in the following. The observations  $\{X_1, \dots, X_N\}$  are collected in  $\mathbf{X}$  and the states  $\{Q_1, \dots, Q_N\}$  in  $\mathbf{Q}$ .

- i. *Evaluation problem:* The aim is to determine the likelihood  $P(\mathbf{X}|\Theta)$  for an observation sequence  $\mathbf{X}$  given the model parameters  $\Theta$ . This likelihood is typically used in sequence classification tasks. The joint probability distribution<sup>6</sup> of the HMM in Figure 18.27 is

$$P_B(\mathbf{X}, \mathbf{Q}|\Theta) = P(Q_1|\theta^1)P(X_1|Q_1, \theta^3)\prod_{i=2}^N P(Q_i|Q_{i-1}, \theta^2)P(X_i|Q_i, \theta^3), \quad (18.217)$$

where  $\theta^1$ ,  $\theta^2$ , and  $\theta^3$  are known as prior, transition, and observation (emission) probabilities, respectively. The parameters  $\theta^2$  and  $\theta^3$  are assumed to be constant for all  $i$ . The likelihood  $P(\mathbf{X}|\Theta)$  can be obtained by marginalizing over all possible state sequences according to

$$P(\mathbf{X}|\Theta) = \sum_{\mathbf{q} \in \text{val}(\mathbf{Q})} P_B(\mathbf{X}, \mathbf{Q} = \mathbf{q}|\Theta), \quad (18.218)$$

i.e., determining  $P(\mathbf{X}|\Theta)$  is a simple marginalization query. A naive summation over all state sequences would be exponential in the length of the sequence  $N$ . Fortunately, the HMM has a tree structure which can be exploited to compute the likelihood  $P(\mathbf{X}|\Theta)$  efficiently. Similar as in Example 25, we can rearrange the sums to the relevant terms in  $P_B(\mathbf{X}, \mathbf{Q}|\Theta)$ , i.e., we can determine  $P(\mathbf{X}|\Theta)$  recursively. Hence, neglecting the parameters  $\Theta$ , we have

---

<sup>6</sup>In implementations on computers the joint probability distribution is usually computed in the log-domain where the products turn into sums. This alleviates numerical problems.

$$P(\mathbf{X}|\Theta) = \sum_{q_1 \in \text{val}(\mathcal{Q}_1)} P(q_1) P(X_1|q_1) \sum_{q_2 \in \text{val}(\mathcal{Q}_2)} P(q_2|q_1) P(X_2|q_2) \cdot \\ \sum_{q_3 \in \text{val}(\mathcal{Q}_3)} P(q_3|q_2) P(X_3|q_3) \cdots \quad (18.219)$$

$$\cdots \sum_{q_{N-1} \in \text{val}(\mathcal{Q}_{N-1})} P(q_{N-1}|q_{N-2}) P(X_{N-1}|q_{N-1}) \underbrace{\sum_{q_N \in \text{val}(\mathcal{Q}_N)} P(q_N|q_{N-1}) P(X_N|q_N)}_{= \sum_{q_N \in \text{val}(\mathcal{Q}_N)} P(q_N, X_N|q_{N-1}) = P(X_N|q_{N-1}) = \beta(q_{N-1})} \\ = \underbrace{\sum_{q_{N-1} \in \text{val}(\mathcal{Q}_{N-1})} P(q_{N-1}, X_{N-1}|q_{N-2}) \beta(q_{N-1}) = P(X_{N-1}, X_N|q_{N-2}) = \beta(q_{N-2})}_{(18.220)}$$

where we defined the *backward* probability recursively as

$$\beta(Q_i = q_i) = P(X_{i+1}, \dots, X_N | Q_i = q_i), \quad (18.221)$$

$$= \sum_{q_{i+1} \in \text{val}(Q_{i+1})} \beta(Q_{i+1} = q_{i+1}) \cdot$$

$$P(Q_{i+1} = q_{i+1} | Q_i = q_i) P(X_{i+1} | Q_{i+1} = q_{i+1}). \quad (18.222)$$

The recursion for computing  $\beta(Q_{N-1})$  backwards to  $\beta(Q_1)$  is initialized with  $\beta(Q_N) = 1$ . The likelihood  $P(\mathbf{X}|\Theta)$  can be determined from  $\beta(Q_1 = q_1)$  according to

$$P(\mathbf{X}|\Theta) = \sum_{q_1 \in \text{val}(\mathcal{Q}_1)} P(Q_1 = q_1) P(X_1 | Q_1 = q_1) \beta(Q_1 = q_1), \quad (18.223)$$

$$= \sum_{q_1 \in \text{val}(\mathcal{Q}_1)} P(Q_1 = q_1) P(X_1 | Q_1 = q_1) P(X_2, \dots, X_N | Q_1 = q_1), \quad (18.224)$$

$$= \sum_{q_1 \in \text{val}(\mathcal{Q}_1)} P(Q_1 = q_1, X_1, \dots, X_N). \quad (18.225)$$

This recursive algorithm reduces the computational complexity from  $\mathcal{O}(N \cdot \text{sp}(\mathcal{Q})^N)$  to  $\mathcal{O}(N \cdot \text{sp}(\mathcal{Q})^2)$ .

In a similar manner, *forward* probabilities  $\alpha(Q_i = q_i)$  can be defined recursively as

$$\alpha(Q_i = q_i) = P(X_1, \dots, X_i, Q_i = q_i), \quad (18.226)$$

$$= \sum_{q_{i-1} \in \text{val}(Q_{i-1})} \alpha(Q_{i-1} = q_{i-1}) \cdot$$

$$P(Q_i = q_i | Q_{i-1} = q_{i-1}) P(X_i | Q_i = q_i), \quad (18.227)$$

$$= \sum_{q_{i-1} \in \text{val}(Q_{i-1})} P(X_1, \dots, X_{i-1}, Q_{i-1} = q_{i-1}) \cdot$$

$$P(Q_i = q_i | Q_{i-1} = q_{i-1}) P(X_i | Q_i = q_i), \quad (18.228)$$

$$= \sum_{q_{i-1} \in \text{val}(Q_{i-1})} P(X_1, \dots, X_i, Q_{i-1} = q_{i-1}, Q_i = q_i). \quad (18.229)$$

After initializing  $\alpha(Q_1)$  to  $\alpha(Q_1) = P(Q_1)P(X_1|Q_1)$  the forward probabilities can be computed for  $i = 2, \dots, N$  and the likelihood  $P(\mathbf{X}|\Theta)$  is determined as

$$P(\mathbf{X}|\Theta) = \sum_{q_N \in \text{val}(Q_N)} \alpha(Q_N = q_N) = \sum_{q_N \in \text{val}(Q_N)} P(X_1, \dots, X_N, Q_N = q_N). \quad (18.230)$$

Interestingly,  $P(\mathbf{X}|\Theta)$  can be determined as

$$P(\mathbf{X}|\Theta) = \sum_{q_i \in \text{val}(Q_i)} \alpha(Q_i = q_i) \beta(Q_i = q_i), \quad (18.231)$$

$$= \sum_{q_i \in \text{val}(Q_i)} P(X_1, \dots, X_i, Q_i = q_i) P(X_{i+1}, \dots, X_N | Q_i = q_i), \quad (18.232)$$

$$= \sum_{q_i \in \text{val}(Q_i)} P(\mathbf{X}, Q_i = q_i). \quad (18.233)$$

Computing  $P(\mathbf{X}|\Theta)$  using  $\alpha(Q_i)$  and/or  $\beta(Q_i)$  is computationally efficient and is known as *forward/backward algorithm* [89]. Both algorithms are an instance of the more general sum-product algorithm.

- ii. *Decoding problem:* Decoding is concerned with finding the state sequence which best explains a given observation sequence  $\mathbf{X}$  in model  $\Theta$ . This relates decoding to the maximum a posteriori query, cf. Section 1.18.5. In other words, we are interested in the most likely instantiation of  $\mathbf{Q}$  given  $\mathbf{X}$  and the model  $\Theta$ , i.e.,

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} P(\mathbf{Q} = \mathbf{q} | \mathbf{X}, \Theta) = \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} \frac{P(\mathbf{Q} = \mathbf{q}, \mathbf{X} | \Theta)}{P(\mathbf{X} | \Theta)}. \quad (18.234)$$

Since  $P(\mathbf{X}|\Theta)$  is the same for every state sequences, we obtain

$$\mathbf{q}^* = \arg \max_{\mathbf{q} \in \text{val}(\mathbf{Q})} P(\mathbf{Q} = \mathbf{q}, \mathbf{X} | \Theta). \quad (18.235)$$

The Viterbi algorithm is an efficient approach for determining  $\mathbf{q}^*$  in HMMs. Similarly as the forward probabilities, we define  $\delta(Q_i = q)$  as

$$\delta(Q_i = q_i) = \max_{q_1 \in \text{val}(Q_1), \dots, q_{i-1} \in \text{val}(Q_{i-1})} P(Q_1 = q_1, \dots, Q_{i-1} = q_{i-1}, Q_i = q_i, X_1, \dots, X_i | \Theta). \quad (18.236)$$

This term describes the largest probability for the partial observation sequence  $X_1, \dots, X_i$  in model  $\Theta$  along a single state sequence which ends at variable  $Q_i$  in state  $q_i$ . Similar observations as for the evaluation problem apply and we can express the computation of  $\delta(Q_i = q_i)$  recursively as

$$\delta(Q_i = q_i) = \max_{\tilde{q} \in \text{val}(Q_{i-1})} [\delta(Q_{i-1} = \tilde{q}) P(Q_i = q_i | Q_{i-1} = \tilde{q})] P(X_i | Q_i = q_i). \quad (18.237)$$

This equation is closely related to the computation of  $\alpha(Q_i)$  in (18.227), i.e., the sum over all states in  $Q_{i-1}$  is replaced by the maximum operator. At the end of the recursion  $\delta(Q_N = q_N)$  gives the

largest probability along a single state sequence for  $\mathbf{X}$  and  $\Theta$ . Since we are interested in the most likely state sequence  $\mathbf{q}^*$ , we have to memorize for each  $q_i$  the best previous state  $q_{i-1}$  in  $\phi(Q_i = q)$  according to

$$\phi(Q_i = q_i) = \arg \max_{\tilde{q} \in \text{val}(Q_{i-1})} [\delta(Q_{i-1} = \tilde{q}) P(Q_i = q_i | Q_{i-1} = \tilde{q})]. \quad (18.238)$$

This enables to extract  $\mathbf{q}^*$  once  $\delta(Q_i = q_i)$  and  $\phi(Q_i = q_i)$  have been determined for  $1 \leq i \leq N$ . The Viterbi algorithm consists of the following steps:

(a) *Initialization:*

$$\delta(Q_1 = q) = P(Q_1 = q) P(X_1 | Q_1 = q) \quad \forall q \in \text{val}(Q), \quad (18.239)$$

$$\phi(Q_1 = q) = 0 \quad \forall q \in \text{val}(Q). \quad (18.240)$$

(b) *Recursion:*

$$\begin{aligned} \delta(Q_i = q) &= \max_{\tilde{q} \in \text{val}(Q_{i-1})} [\delta(Q_{i-1} = \tilde{q}) P(Q_i = q | Q_{i-1} = \tilde{q})] \cdot \\ &\quad P(X_i | Q_i = q), \quad \forall q \in \text{val}(Q), i = 2, \dots, N, \end{aligned} \quad (18.241)$$

$$\begin{aligned} \phi(Q_i = q) &= \arg \max_{\tilde{q} \in \text{val}(Q_{i-1})} [\delta(Q_{i-1} = \tilde{q}) P(Q_i = q | Q_{i-1} = \tilde{q})], \\ &\quad \forall q \in \text{val}(Q), i = 2, \dots, N. \end{aligned} \quad (18.242)$$

(c) *Termination:*

$$P(\mathbf{X}, \mathbf{Q} = \mathbf{q}^* | \Theta) = \max_{q \in \text{val}(Q_N)} \delta(Q_N = q), \quad (18.243)$$

$$q_N^* = \arg \max_{q \in \text{val}(Q_N)} \delta(Q_N = q). \quad (18.244)$$

(d) *Backtracking:*

$$q_i^* = \phi(Q_{i+1} = q_{i+1}^*) \quad i = N - 1, \dots, 1. \quad (18.245)$$

iii. *Learning problem:* The focus of learning is determining the parameters  $\Theta$  of an HMM given observation sequences  $\mathbf{X}^{(l)}$ . Usually, there is a set of  $L$  training sequences  $\mathcal{D} = \{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(L)}\}$ , where  $\mathbf{X}^{(l)} = \{x_1^{(l)}, \dots, x_{N_l}^{(l)}\}$ . The parameters  $\Theta$  in HMMs are determined by maximum likelihood estimation, i.e.,

$$\Theta_{\text{ML}} = \arg \max_{\Theta} \log \mathcal{L}(\Theta; \mathcal{D}), \quad (18.246)$$

where

$$\log \mathcal{L}(\Theta; \mathcal{D}) = \log \sum_{\mathbf{q} \in \text{val}(\mathbf{Q})} P_{\mathcal{B}}(\mathbf{X}, \mathbf{Q} = \mathbf{q} | \Theta), \quad (18.247)$$

$$= \log \sum_{\mathbf{q} \in \text{val}(\mathbf{Q})} P(Q_1 = q_1 | \theta^1) P(X_1 | Q_1 = q_1, \theta^3). \quad (18.248)$$

$$\prod_{i=2}^N P(Q_i = q_i | Q_{i-1} = q_{i-1}, \theta^2) P(X_i | Q_i = q_i, \theta^3). \quad (18.248)$$

So the aim is to maximize the log-likelihood  $\log \mathcal{L}(\Theta; \mathcal{D})$  with respect to  $\Theta$  given  $\mathcal{D}$ .

In HMMs we have to determine the parameters

$$P(Q_1 = q | \theta^1) = \theta_q^1 \quad \forall q \in \text{val}(Q), \quad (18.249)$$

$$P(Q_i = q | Q_{i-1} = \tilde{q}, \theta^2) = \theta_{q|\tilde{q}}^2 \quad \forall q, \tilde{q} \in \text{val}(Q), \quad (18.250)$$

$$P(X_i = j | Q_i = q, \theta^3) = \theta_{j|q}^3 \quad \forall j \in \text{val}(X), q \in \text{val}(Q). \quad (18.251)$$

ML parameter learning for discrete variables derived in Section 1.18.4.2.1 essentially amounts to counting and normalizing. In particular, we have

$$\hat{\theta}_q^1 = \frac{\sum_{l=1}^L u_q^{1,l}}{\sum_{j=1}^{\text{sp}(Q)} \sum_{l=1}^L u_j^{1,l}}, \quad (18.252)$$

$$\hat{\theta}_{q|\tilde{q}}^2 = \frac{\sum_{l=1}^L \sum_{i=2}^{N_l} u_{q|\tilde{q}}^{i,l}}{\sum_{j=1}^{\text{sp}(Q)} \sum_{l=1}^L \sum_{i=2}^{N_l} u_{j|\tilde{q}}^{i,l}}, \quad (18.253)$$

$$\hat{\theta}_{j|q}^3 = \frac{\sum_{l=1}^L \sum_{i=2}^{N_l} u_{j|q}^{i,l}}{\sum_{j'=1}^{\text{sp}(X)} \sum_{l=1}^L \sum_{i=2}^{N_l} u_{j'|q}^{i,l}}, \quad (18.254)$$

where  $\sum_{l=1}^L u_q^{1,l}$  is the number of occurrences of  $Q_1 = q$ ,  $\sum_{l=1}^L \sum_{i=2}^{N_l} u_{q|\tilde{q}}^{i,l}$  denotes the number of transitions from state  $\tilde{q}$  to  $q$ , and  $\sum_{l=1}^L \sum_{i=2}^{N_l} u_{j|q}^{i,l}$  is the number of observing  $X_i = j$  in state  $Q_i = q$ . Unfortunately, these counts cannot be determined directly from  $\mathcal{D}$  since we only observe the variables  $\mathbf{X}$  but not  $\mathbf{Q}$ . The variables  $\mathbf{Q}$  are hidden. Hence, the data  $\mathcal{D}$  are incomplete and parameter learning is solved by using the *EM*-algorithm [11]<sup>7</sup>. The *EM*-algorithm consists of an inference step (*E*-step) for estimating the values of the unobserved variables. Subsequently, the maximization step (*M*-step) uses this knowledge in (18.252), (18.253), and (18.254) to update the estimate. The *EM*-algorithm iterates between both steps until convergence of  $\log \mathcal{L}(\Theta; \mathcal{D})$ . In [11], it is proven that the log likelihood increases in each iteration. In (18.252) we replace  $u_q^{1,l}$  by the probability of  $Q_1 = q$  given  $\mathbf{X}^{(l)}$  obtained in the *E*-step as

$$u_q^{1,l} = P(Q_1 = q | \mathbf{X}^{(l)}), \quad (18.255)$$

$$= \frac{P(Q_1 = q, \mathbf{X}^{(l)})}{P(\mathbf{X}^{(l)})}, \quad (18.256)$$

$$= \frac{\alpha^{(l)}(Q_1 = q) \beta^{(l)}(Q_1 = q)}{\sum_{Q_i} \alpha^{(l)}(Q_i) \beta^{(l)}(Q_i)}. \quad (18.257)$$

---

<sup>7</sup>In the context of HMMs the *EM*-algorithm is also known as Baum-Welch algorithm.

Similarly, for estimating the transition probabilities

$$u_{q|\tilde{q}}^{i,l} = P(Q_i = q, Q_{i-1} = \tilde{q} | \mathbf{X}^{(l)}), \quad (18.258)$$

$$= \frac{P(Q_i = q, Q_{i-1} = \tilde{q}, \mathbf{X}^{(l)})}{P(\mathbf{X}^{(l)})}, \quad (18.259)$$

$$= \frac{P(X_1^{(l)}, \dots, X_{i-1}^{(l)}, Q_{i-1} = \tilde{q}) P(Q_i = q | Q_{i-1} = \tilde{q}) P(X_i^{(l)} | Q_i = q) P(X_{i+1}^{(l)}, \dots, X_N^{(l)} | Q_i = q)}{P(\mathbf{X}^{(l)})}, \quad (18.260)$$

$$= \frac{\alpha^{(l)}(Q_{i-1}) P(Q_i = q | Q_{i-1} = \tilde{q}) P(X_i^{(l)} | Q_i = q) \beta^{(l)}(Q_i)}{\sum_{Q_i} \alpha^{(l)}(Q_i) \beta^{(l)}(Q_i)} \quad (18.261)$$

and for the observation probabilities

$$u_{j|q}^{i,l} = P(Q_i = q | \mathbf{X}^{(l)}) \mathbb{1}_{\{X_i = j\}}, \quad (18.262)$$

$$= \frac{\alpha^{(l)}(Q_i = q) \beta^{(l)}(Q_i = q)}{\sum_{Q_i} \alpha^{(l)}(Q_i) \beta^{(l)}(Q_i)} \mathbb{1}_{\{X_i = j\}}. \quad (18.263)$$

The  $\alpha(Q_i)$  and  $\beta(Q_i)$  probabilities from the recursive forward/backward algorithm are determined in the E-step to facilitate the parameter estimation in (18.252), (18.253), and (18.254) of the M-Step. At the beginning of the EM-Algorithm, the parameters  $\Theta$  have to be initialized to reasonable values. Details are provided in [89].

We continue with the natural language processing model for the term  $P(\mathbf{w})$  in Eq. (18.215). For instance, stochastic language models [95] such as *n-grams* decompose the probability of a sequence of words as

$$P(w_1, \dots, w_N) = \prod_{i=1}^N P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=1}^N P(w_i | w_{i-(n-1)}, \dots, w_{i-1}). \quad (18.264)$$

A bigram model (2-gram), which basically is a first order Markov chain, leads to

$$P(\mathbf{w}) = \prod_{i=1}^N P(w_i | w_{i-1}). \quad (18.265)$$

The most probable word sequence  $\mathbf{w}^*$  in Eq. (18.215) is determined using a decoding algorithm such as the Viterbi algorithm.

The approach in Eq. (18.215) can also be used in statistical machine translation [96], where the acoustic model  $P(\mathbf{X}|\mathbf{w})$  is replaced by a translation model  $P(\mathbf{f}|\mathbf{w})$  which models the relation of a pair of word sequences of different languages  $\mathbf{f}$  and  $\mathbf{w}$ , where  $\mathbf{f} = (f_1, \dots, f_M)$  is a sequence with  $M$  words. The probability  $P(\mathbf{f}|\mathbf{w})$  can be interpreted as the probability that a translator produces sequence  $\mathbf{f}$  when presented sequence  $\mathbf{w}$ .

HMMs [89] for acoustic modeling of speech signals have enjoyed remarkable success over the last forty years. However, recent developments showed that there are many sophisticated extensions to HMMs represented by dynamical BNs, cf. [88] for an overview. Furthermore, in the seminal paper of [97], discriminative HMM parameter learning based on the maximum mutual information (MMI) criterion—closely related to optimizing the conditional log-likelihood objective—has been proposed, which attempts to maximize the posterior probability of the transcriptions given the speech utterances. This leads to significantly better recognition rates compared to conventional *generative* maximum likelihood learning. In speech processing, discriminative training of HMMs celebrated success over the years and in this context the extended Baum-Welch algorithm [98, 99] has been introduced.

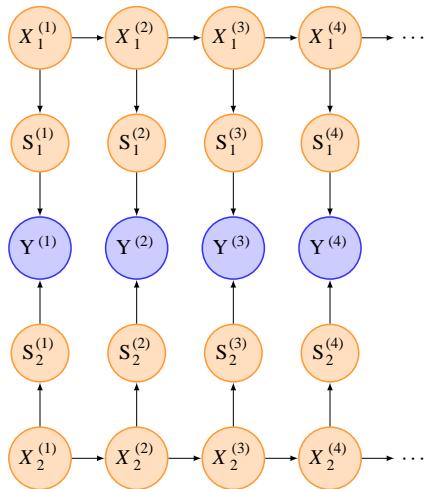
ASR for high-quality single-talker scenarios is performing reliably with good recognition performance. However, in harsh environments where the speech signal is distorted by interference with other acoustic sources, e.g., the *cocktail party problem* [100], ASR performs far from satisfactory. Recently, factorial-HMMs (FHMMs) including speaker interaction models and constraints on temporal dynamics have won the monaural speech separation and recognition challenge [101]. Remarkably, these models slightly outperformed human listeners [102] and are able to separate audio mixtures containing up to four speech signals. In the related task of multipitch tracking, Wohlmayr et al. [103] have obtained promising results using a similar FHMM model. FHMMs [63] enable to track the states of multiple Markov processes evolving in parallel over time, where the available observations are considered as a joint effect of all single Markov processes. Combining two single speakers in this way using an interaction model, we obtain the FHMM shown in Figure 18.28. The hidden state RVs denoted by  $X_k^{(t)}$  model the temporal dynamics of the pitch trajectory, where  $k$  indicates the Markov chain representing the  $k$ th speaker and  $t$  is the time frame. Realizations of observed RVs of the speech mixture spectrum at time  $t$  are collected in a  $D$ -dimensional vector  $\mathbf{Y}^{(t)} \in \mathbb{R}^D$  and  $\mathbf{S}_k^{(t)}$  models the single-speaker spectrum conditioned on state  $X_k^{(t)}$ . At each time frame, the observation  $\mathbf{Y}^{(t)}$  is considered to be produced jointly by the two single speech emissions  $\mathbf{S}_1^{(t)}$  and  $\mathbf{S}_2^{(t)}$ . This mixing process is modeled by an interaction model as for example by the MIXMAX model described in [104].

### 1.18.6.2 BNs for expert systems

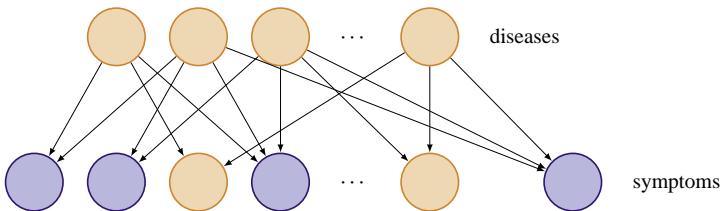
Expert and decision support systems enjoy great popularity, especially in the medical domain. They consist of two parts: a knowledge base and an inference engine. The knowledge base contains specific knowledge of some domain and the inference engine is able to process the encoded knowledge and extract information. Many of these systems are *rule-based* using logical rules. BNs have been applied in order to deal with uncertainty in these systems [54]. For instance, the Quick Medical Reference (QMR) database [105] has been developed to support medical diagnosis which consists of 600 diseases and 4000 symptoms as depicted in Figure 18.29. The aim of inference is to determine the marginal probabilities of some diseases given a set of observed symptoms. In [65], variational inference has been introduced as exact inference is infeasible for most of the practically occurring inference queries.

### 1.18.6.3 MRFs for image analysis

MRFs [84–86] have been widely used in image processing tasks over the past decades. They are employed in all stages of image analysis, i.e., for low-level vision such as image denoising,

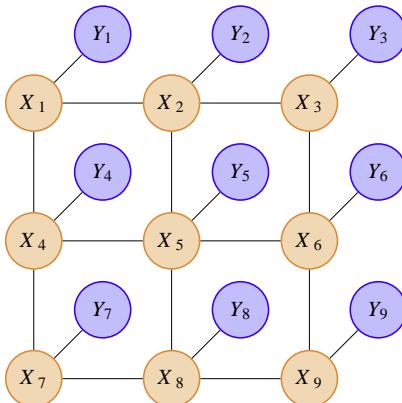
**FIGURE 18.28**

FHMM for speaker interaction. Each Markov chain models the pitch trajectory of a single speaker. At each time frame, the single speech emissions  $S_1^{(t)}$  and  $S_2^{(t)}$  jointly produce the observation  $Y^{(t)}$ .

**FIGURE 18.29**

QMR database.

segmentation, texture and optical flow modeling, and edge detection as well as for high-level tasks like face detection/recognition and pose estimation. Image processing problems can often be represented as the task of assigning labels to image pixels. For example, in the case of image segmentation, each pixel belongs to one particular label representing a segment: in the MRF in Figure 18.30, the image pixels  $\mathbf{Y}$  are observed and the underlying labels are represented as latent, i.e., unobserved, variables  $\mathbf{X}$ . The edges between the variables  $\mathbf{X}$  model contextual dependencies, e.g., using the fact that neighboring pixels have similar features. Inferring the distribution  $P_{\mathcal{M}}(\mathbf{X}|\mathbf{Y})$  is intractable for large-scale MRFs using exact methods. Approximate inference methods, such as loopy belief propagation, energy optimization, and sampling methods (MCMC) amongst others, have been deployed.

**FIGURE 18.30**

Markov random field for image analysis.

#### 1.18.6.4 FGs for decoding error-correcting codes

Transmission of information over noisy channels requires redundant information to be added to the transmitted sequence of symbols in order to facilitate the recovery of the original information from a corrupted received signal. Most error-correcting codes are either block codes or convolutional codes, including the low-density parity-check codes developed by Gallager [94].

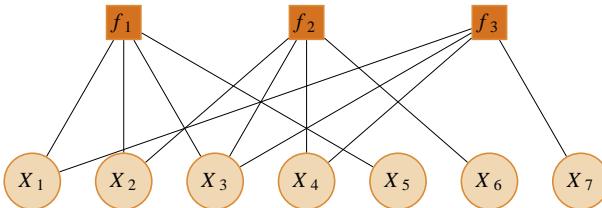
In a block code [76] the source symbol sequence of length  $K$  is converted into a symbol sequence  $\mathbf{X}$  of length  $N$  for transmission. Redundancy is added in the case of  $N$  being greater than  $K$ . Hence, a  $(N, K)$  block code is a set of  $2^K$  codewords  $\{\mathbf{x}^1, \dots, \mathbf{x}^{2^K}\}$ , where each codeword has a length of  $N$  symbols of a finite alphabet  $\mathcal{A}$ , i.e.,  $\mathbf{x}^i \in \mathcal{A}^N$  [76, 106]. Here, we assume a binary code, i.e.,  $\mathcal{A} \in \{0, 1\}$ .

In a *linear block code*, the added extra bits are a linear function of the original symbol. Those bits are known as *parity-check* bits and are redundant information. The transmitted codeword  $\mathbf{x}$  is obtained from the source sequence  $\mathbf{s}$  by calculating  $\mathbf{x} = \mathbf{G}^T \mathbf{s}$  using modulo-2 arithmetic, where  $\mathbf{G}$  is the *generator matrix* of the code. For the  $(7, 4)$  Hamming code [76] a generator matrix is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \quad (18.266)$$

and the corresponding *parity-check matrix*  $\mathbf{H}$  can be derived (details are given in [76]) as

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (18.267)$$

**FIGURE 18.31**

FG for the binary (7,4) Hamming code.

Each valid codeword  $\mathbf{x}$  must satisfy  $\mathbf{Hx} = \mathbf{0}$ . This parity-check can be expressed by the indicator function

$$I(x_1, \dots, x_7) = \mathbb{1}_{\{\mathbf{Hx}=\mathbf{0}\}}, \quad (18.268)$$

$$= \delta(x_1 \oplus x_2 \oplus x_3 \oplus x_5) \delta(x_2 \oplus x_3 \oplus x_4 \oplus x_6) \delta(x_1 \oplus x_3 \oplus x_4 \oplus x_7), \quad (18.269)$$

where  $\delta(a)$  denotes the Dirac delta function and  $\oplus$  is the modulo-2 addition. The term  $I(x_1, \dots, x_7) = 1$  if and only if  $\mathbf{x}$  is a valid codeword. Each row in  $\mathbf{H}$  corresponds to one  $\delta(\cdot)$  term. The function  $I(x_1, \dots, x_7)$  can be represented as the FG shown in Figure 18.31. It consists of the following factors:  $f_1(X_1, X_2, X_3, X_5) = \delta(x_1 \oplus x_2 \oplus x_3 \oplus x_5)$ ,  $f_2(X_2, X_3, X_4, X_6) = \delta(x_2 \oplus x_3 \oplus x_4 \oplus x_6)$ , and  $f_3(X_1, X_3, X_4, X_7) = \delta(x_1 \oplus x_3 \oplus x_4 \oplus x_7)$ .

After encoding, the codewords are transmitted over a noisy channel. One of the simplest noisy channel models is the *binary symmetric channel* where each transmitted bit is flipped (corrupted) with probability  $\epsilon$ . That is, if the input to the channel is denoted as  $X$  and the output of the channel is  $Y$ , then

$$P(Y = 0|X = 1) = \epsilon, \quad (18.270)$$

$$P(Y = 1|X = 1) = 1 - \epsilon, \quad (18.271)$$

$$P(Y = 0|X = 0) = 1 - \epsilon, \quad \text{and} \quad (18.272)$$

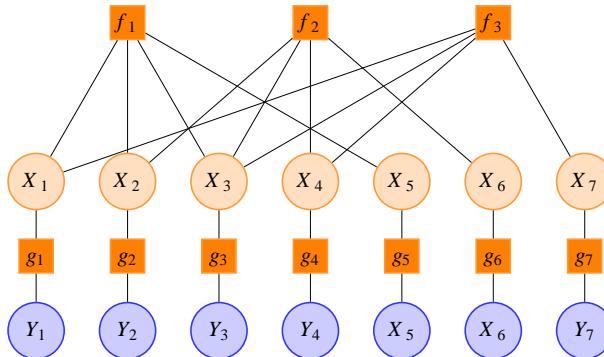
$$P(Y = 1|X = 0) = \epsilon. \quad (18.273)$$

This channel is memoryless and the channel model can be represented as  $P(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^N P(Y_i|X_i) = \prod_{i=1}^N g_i(X_i, Y_i)$ . Decoding the received symbol  $\mathbf{Y}$  requires to determine the transmitted codeword  $\mathbf{X}$  based on the observed sequence  $\mathbf{Y}$ . This is possible using the posterior probability  $P(\mathbf{X}|\mathbf{Y})$  and exploiting that

$$P(\mathbf{X}|\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{X})P(\mathbf{X})}{P(\mathbf{Y})}, \quad (18.274)$$

$$\propto P(\mathbf{Y}|\mathbf{X})I(\mathbf{X}) = \prod_{i=1}^N P(Y_i|X_i)I(\mathbf{X}), \quad (18.275)$$

where the probability  $P(\mathbf{X})$  is represented by the indicator function  $I(\mathbf{X})$  of the code, i.e., all codewords are assumed to be equally likely. The code together with the noisy channel model can be represented

**FIGURE 18.32**

FG for the binary (7,4) Hamming code and the binary symmetric channel.

by the FG shown in Figure 18.32. The probability  $P(\mathbf{X}|\mathbf{Y})$  is obtained by performing the sum-product algorithm in (18.206) and (18.207) or the max-product algorithm. If the FG contains cycles, as in this example, the decoding problem is iterative and amounts to loopy message passing, also known as turbo decoding in the coding community [70].

## 1.18.7 Implementation/code

In this section, we provide a list of recently developed code for PGMs. With respect to the features of the listed implementations and comparisons to other software, the interested reader is referred to the given references and websites.

- *Infer.NET*: Infer.NET provides a framework for large-scale Bayesian inference in PGMs and is developed at Microsoft Research Cambridge ([research.microsoft.com/en-us/um/cambridge/projects/infernet](http://research.microsoft.com/en-us/um/cambridge/projects/infernet)).
- *WEKA*: WEKA [107] is a data mining software applicable to various tasks in machine learning. Some of the implemented methods are directly related to PGMs such as discriminative parameter learning or structure learning heuristics of BNs.
- *LibDAI*: LibDAI [108] is an open source C++ library providing various exact and approximate inference methods for FGs with discrete variables.
- *HTK*: The HTK toolkit from Cambridge University ([htk.eng.cam.ac.uk](http://htk.eng.cam.ac.uk)) is a state-of-the-art tool used for time-series modeling such as speech recognition. It implements an HMM using discrete and continuous observation probabilities, e.g., mixtures of Gaussians. Furthermore, the tool supports parameter tying, discriminative HMM training and many other methods important for speech recognition.
- *GMTK*: GMTK ([ssli.ee.washington.edu/~bilmes/gmtk](http://ssli.ee.washington.edu/~bilmes/gmtk)) implements dynamic BNs for sequential data processing. It is mainly designed for speech recognition.
- *MMBN*: We recently developed a maximum margin parameter learning algorithm for BN classifiers. Details about the algorithm are given in [48]. The maximum margin parameter training method can

be downloaded at [www.spsc.tugraz.at/tools/MMBN](http://www.spsc.tugraz.at/tools/MMBN). Furthermore, some data sets and example code are provided.

### 1.18.8 Data sets

In the following, we list a selection of common data sets and repositories in machine learning. This list includes only a small subset of publicly available data. Detailed information are supplied in the given references:

- *UCI repository* [109]: One of the largest collection of data sets is the UCI repository. It includes more than 200 data sets for classification, clustering, regression, and time-series modeling. The types of features are categorical, real, and mixed. Benchmark results for most of the data are given in accompanying references.
- *MNIST data* [110]: The MNIST data set of handwritten digits contains 60000 samples for training and 10000 for testing. The digits are centered in a  $28 \times 28$  gray-level image. Benchmark classification results and the data are available at <http://yann.lecun.com/exdb/mnist/>.
- *USPS data*: This data set contains 11000 handwritten digit images collected from zip codes of mail envelopes ([www.cs.nyu.edu/~roweis/data.html](http://www.cs.nyu.edu/~roweis/data.html)). Each digit is represented as a  $16 \times 16$  gray-scale image. Other variants of this data set with different sample sizes are available. More details are given in the Appendix of [37].
- *TIMIT data* [111]: This data set is very popular in the speech processing community. It contains read speech of eight major dialects of American English sampled at 16 kHz. The corpus includes time-aligned orthographic, phonetic, and word transcriptions. This data has been used for various tasks, like time-series modeling, phone recognition and classification [112, 113].
- *Caltech101 and Caltech256*: For image analysis two data sets have been provided by Caltech containing images of objects belonging to either 101 or 256 categories. The data, performance results, and relevant papers are provided at [http://www.vision.caltech.edu/Image\\_Datasets/Caltech101](http://www.vision.caltech.edu/Image_Datasets/Caltech101).

### 1.18.9 Conclusion

PGMs turned out to be one of the best approaches for modeling uncertainty in many domains. They offer an unifying framework which allows to transfer methods among different domains. In this article, we summarized the main elements of PGMs. In particular, we reviewed three types of representations—Bayesian networks, Markov networks, and factor graphs, and we provided a gentle introduction to structure and parameter learning approaches of Bayesian networks. Furthermore, inference techniques were discussed with focus on exact methods. Approximate inference was covered briefly. The interested reader can immerse oneself following the references provided throughout the article.

Many relevant and interesting topics are not covered in detail within this review. One of these topics is approximate inference methods. In fact, tractable approximate inference has been one of the most challenging and active research fields over the past decade. Many of these advanced techniques are discussed in [1, 4, 62, 67, 75]. Another focal point of current research interests are learning

techniques for MNs. While we concentrated on structure and parameter learning of BNs under various perspectives, we completely omitted learning of MNs. Parameter learning of undirected models is essentially performed using iterative gradient-based methods. Each iteration requires to perform inference which makes parameter learning computationally expensive. Score-based structure learning suffers from similar considerations. One advantage of learning the structure of MNs is that there are no acyclicity constraints which renders structure learning in BNs hard.

---

## Glossary

Probabilistic graphical model	a probabilistic graphical model consists of a graph and a set of random variables. It represents a joint probability distribution where the graph captures the conditional independence relationships among the random variables
Bayesian network	a special instance of probabilistic graphical models consisting of a directed acyclic graph and a set of conditional probability tables associated with the nodes of the graph
Markov network	an undirected probabilistic graphical model. The joint probability is defined as the product of clique potentials associated with the graph
Factor graph	a probabilistic graphical model represented by a bipartite graph. One set of nodes represents the RVs in the graph, the other set of nodes the factors. The joint probability is defined as the product of the factors
Probabilistic inference	computation of the distribution or the most likely configuration of some variables while potentially observing some other variables using the joint probability distribution
Parameter learning	specification of the parameters of the probability distributions
Structure learning	specification of the factorization of the joint probability distribution represented by the structure of a probabilistic graphical model

---

## Acknowledgments

The authors Franz Pernkopf, Robert Peharz, and Sebastian Tschiatschek contributed equally to this work. This work was supported by the Austrian Science Fund (Project number P22488-N23) and (Project number S10610).

*Relevant Theory:* Signal Processing, Machine Learning

See this Volume, [Chapter 11](#) Parametric Estimation

See this Volume, [Chapter 19](#) Monte-Carlo methods (MCMC, Particle Filters)

See this Volume, [Chapter 20](#) Clustering

---

## References

- [1] D. Koller, N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, 2009.
- [2] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 1988.
- [3] M.I. Jordan, *Learning in Graphical Models*, MIT Press, 1999.
- [4] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- [5] T. Cover, J. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.
- [6] A. Papoulis, S.U. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 2002.
- [7] J.A. Bilmes, *Dynamic Bayesian Multinets*, in: International Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 2000.
- [8] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1996, pp. 115–123.
- [9] J. Berkson, Limitations of the application of fourfold table analysis to hospital data, *Biometrics Bull.* 2 (3) (1946) 47–53.
- [10] S.L. Lauritzen, *Graphical Models*, Oxford Science Publications, 1996.
- [11] A. Dempster, N. Laird, D. Rubin, Maximum likelihood estimation from incomplete data via the EM algorithm, *J. Roy. Stat. Soc. 30 (B)* (1977) 1–38.
- [12] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
- [13] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, *Mach. Learn. (Kluwer Academic Publishers)* 20 (1995) 197–243.
- [14] P. Spirtes, C. Glymour, R. Scheines, *Causation, Prediction, and Search*, The MIT Press, 2001.
- [15] T. Verma, J. Pearl, An algorithm for deciding if a set of observed independencies has a causal explanation, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1992.
- [16] J. Rissanen, Modeling by shortest data description, *Automatica* 14 (1978) 465–471.
- [17] W. Lam, F. Bacchus, Learning Bayesian belief networks: an approach based on the MDL principle, *Comput. Intell.* 10 (3) (1994) 269–293.
- [18] J. Suzuki, A construction of Bayesian networks from databases based on an MDL principle, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1993, pp. 266–273.
- [19] N. Friedman, D. Koller, Being Bayesian about network structure, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 2000, pp. 201–210.
- [20] W.L. Buntine, Theory refinement on Bayesian networks, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1991, pp. 52–60.
- [21] G. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, *Mach. Learn.* 9 (1992) 309–347.
- [22] D.M. Chickering, Learning Bayesian networks is NP-Complete, in: *Learning From Data: Artificial Intelligence and Statistics V*, Springer-Verlag, 1996, pp. 121–130.
- [23] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian Networks is NP-Hard, Technical Report, Technical Report No. MSR-TR-94-17, Microsoft Research, Redmond, Washington, 1994.
- [24] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, *IEEE Trans. Inform. Theory* 14 (1968) 462–467.
- [25] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian networks: search methods and experimental results, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 1995, pp. 112–128.

- [26] F. Glover, Heuristics for integer programming using surrogate constraints, *Decision Sci.* 8 (1) (1977) 156–166.
- [27] D.M. Chickering, Learning equivalence classes of Bayesian-network structures, *J. Mach. Learn. Res.* 2 (2002) 445–498.
- [28] M. Teyssier, D. Koller, Ordering-based search: A simple and effective algorithm for learning Bayesian networks, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 2005, pp. 584–590.
- [29] M. Koivisto, K. Sood, Exact Bayesian structure discovery in Bayesian networks, *J. Mach. Learn. Res.* 5 (2004) 549–573.
- [30] T. Silander, P. Myllymäki, A simple approach for finding the globally optimal Bayesian network structure, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 2006.
- [31] A.P. Singh, A.W. Moore, Finding Optimal Bayesian Networks by Dynamic Programming, Carnegie Mellon University, Technical Report, 2005.
- [32] C.P. de Campos, Z. Zeng, Q. Ji, Structure learning of Bayesian networks using constraints, in: International Conference on Machine Learning (ICML), 2009, pp. 113–120.
- [33] T. Jaakkola, D. Sontag, A. Globerson, M. Meila, Learning Bayesian network structure using LP relaxations, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, pp. 358–365.
- [34] J. Suzuki, Learning Bayesian belief networks based on the minimum description length principle: an efficient algorithm using the B&B technique, in: International Conference on Machine Learning (ICML), 1996, pp. 462–470.
- [35] J.O. Berger, *Statistical Decision Theory and Bayesian Analysis*, Springer-Verlag, 1985.
- [36] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Disc.* 2 (2) (1998) 121–167.
- [37] B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.
- [38] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
- [39] V. Vapnik, *Statistical Learning Theory*, Wiley & Sons, 1998.
- [40] F. Pernkopf, J. Bilmes, Efficient heuristics for discriminative structure learning of Bayesian network classifiers, *J. Mach. Learn. Res.* 11 (2010) 2323–2360.
- [41] J.H. Friedman, On bias, variance, 0/1-loss, and the curse of dimensionality, *Data Min. Knowl. Disc.* 1 (1997) 55–77.
- [42] R. Greiner, W. Zhou, Structural extension to logistic regression: discriminative parameter learning of belief net classifiers, in: Conference of the AAAI, 2002, pp. 167–173.
- [43] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, H. Tirri, On discriminative Bayesian network classifiers and logistic regression, *Mach. Learn.* 59 (2005) 267–296.
- [44] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, H. Tirri, When discriminative learning of Bayesian network parameters is easy, in: International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 491–496.
- [45] D. Grossman, P. Domingos, Learning Bayesian network classifiers by maximizing conditional likelihood, in: International Conference of Machine Lerning (ICML), 2004, pp. 361–368.
- [46] E.J. Keogh, M.J. Pazzani, Learning augmented Bayesian classifiers: a comparison of distribution-based and classification-based approaches, in: International Workshop on Artificial Intelligence and Statistics, 1999, pp. 225–230.
- [47] Y. Guo, D. Wilkinson, D. Schuurmans, Maximum margin Bayesian networks, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 2005.
- [48] F. Pernkopf, M. Wohlmayr, S. Tschiatschek, Maximum margin Bayesian network classifiers, *IEEE Trans. Pattern Anal. Mach. Intell.* 34 (3) (2012) 521–532.

- [49] F. Pernkopf, M. Wohlmayr, Stochastic margin-based structure learning of Bayesian network classifiers, *Pattern Recogn.* 46 (2) (2013) 464–471, Elsevier Inc., New York, NY, USA. <<http://dx.doi.org/10.1016/j.patcog.2012.08.007>>.
- [50] R. Peharz, F. Pernkopf, Exact maximum margin structure learning of Bayesian networks, in: International Conference on Machine Learning (ICML), 2012.
- [51] J.S. Yedidia, W.T. Freeman, Y. Weiss, Understanding Belief Propagation and Its Generalizations, Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, 2002.
- [52] S.M. Aji, R.J. McEliece, The generalized distributive law, *IEEE Trans. Inform. Theory* 46 (2) (2000) 325–543.
- [53] F.V. Jensen, *An Introduction to Bayesian Networks*, UCL Press Limited, 1996.
- [54] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, *Probabilistic Networks and Expert Systems*, Springer, 1999.
- [55] F.R. Kschischang, B.J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, *IEEE Trans. Inform. Theory* 47 (2) (2001) 498–519.
- [56] R. Cowell, Introduction to inference for Bayesian networks, in: M.I. Jordan (Ed.), *Learning in Graphical Models*, MIT Press, 1999.
- [57] P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, *Stat. Comput.* 2 (1992) 25–36.
- [58] C. Huang, A. Darwiche, Inference in belief networks: a procedural guide, *Int. J. Approx. Reason.* 15 (1996) 225–263.
- [59] S. Lauritzen, D. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, *J. Roy. Stat. Soc. B* 50 (1988) 157–224.
- [60] J.B. Kruskal, On the shortest spanning subtree and the traveling salesman problem, in: Proceedings of the American Mathematical Society, vol. 7, 1956, pp. 48–50.
- [61] F.V. Jensen, F. Jensen, Optimal junction trees, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1994, pp. 360–366.
- [62] M.J. Wainwright, M.I. Jordan, Graphical models, exponential families, and variational inference, *Foundations Trends Mach. Learn.* 1 (2008) 1–305.
- [63] Z. Ghahramani, M.I. Jordan, Factorial hidden Markov models, *Mach. Learn.* 29 (1997) 245–275.
- [64] T. Jaakkola, Tutorial on variational approximation methods, in: *Advanced Mean Field Methods: Theory and Practice*, MIT Press, 2000.
- [65] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul, An introduction to variational methods for graphical models, *Mach. Learn.* 37 (1999) 183–233.
- [66] J. Yedidia, W.T. Freeman, Y. Weiss, Generalized belief propagation, in: *Advances in Neural Information Processing Systems (NIPS)*, 2000, pp. 689–695.
- [67] J. Yedidia, W.T. Freeman, Y. Weiss, Constructing free-energy approximations and generalized belief propagation algorithms, *IEEE Trans. Inform. Theory* 51 (7) (2005) 2282–2312.
- [68] A.L. Yuille, CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation, *Neural Comput.* 14 (7) (2002) 1691–1722.
- [69] F.R. Kschischang, B.J. Frey, Iterative decoding of compound codes by probability propagation in graphical models, *IEEE J. Sel. Areas Commun.* 16 (1998) 219–230.
- [70] R.J. McEliece, D.J.C. MacKay, J.F. Cheng, Turbo decoding as an instance of Pearl’s ‘belief propagation’ algorithm, *IEEE J. Sel. Areas Commun.* 16 (2) (1998) 140–152.
- [71] S. Aji, G. Horn, R. McEliece, On the convergence of iterative decoding on graphs with a single cycle, in: *IEEE International Symposium on Information Theory*, 1998, pp. 276–282.
- [72] Y. Weiss, Correctness of local probability propagation in graphical models with loops, *Neural Comput.* 12 (2000) 1–41.

- [73] Y. Weiss, W.T. Freeman, Correctness of belief propagation in Gaussian graphical models of arbitrary topology, *Neural Comput.* 13 (10) (2001) 2173–2200.
- [74] J.M. Mooij, H.J. Kappen, Sufficient conditions for convergence of the sum-product algorithm, *IEEE Trans. Inform. Theory* 52 (2) (2007) 4422–4437.
- [75] T. Minka, Divergence Measures and Message Passing, Technical Report MSR-TR-2005-173, Microsoft Research Ltd., Cambridge, UK, 2005.
- [76] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.
- [77] W. Gilks, S. Richardson, D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman and Hall, 1996.
- [78] R.M. Neal, Probabilistic Inference Using Markov Chain Monte Carlo methods, Technical Report CRG-TR-93-1, University of Toronto, Department of Computer Science, 1993.
- [79] S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for on-line non-linear/non-gaussian Bayesian tracking, *IEEE Trans. Signal Process.* 50 (2) (2002) 174–188.
- [80] A. Doucet, On Sequential Monte Carlo Sampling Methods for Bayesian Filtering, Technical Report CUED/F-INFENG/TR. 310, Cambridge University, Department of Engineering, 1998.
- [81] A. Ihler, D. McAllester, Particle belief propagation, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2009, pp. 256–263.
- [82] D. Koller, U. Lerner, D. Angelov, A general algorithm for approximate inference and its application to hybrid Bayes nets, in: International Conference on Uncertainty in Artificial Intelligence (UAI), 1999, pp. 324–333.
- [83] E.B. Sudderth, A.T. Ihler, W.T. Freeman, A.S. Willsky, Nonparametric belief propagation, in: Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2003, pp. 605–612.
- [84] J.E. Besag, On the statistical analysis of dirty pictures, *J. Roy. Stat. Soc. B* 48 (1986) 259–302.
- [85] S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (6) (1984) 721–741.
- [86] S.Z. Li, *Markov Random Field Modeling in Image Analysis*, Springer-Verlag, 2009.
- [87] A.S. Willsky, Multiresolution Markov models for signal and image processing, *Proc. IEEE* 90 (8) (2002) 1396–1458.
- [88] J.A. Bilmes, C. Bartels, Graphical model architectures for speech recognition, *IEEE Signal Process. Mag.* 22 (5) (2005) 89–100.
- [89] L.R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, *Proc. IEEE* 77 (2) (1989) 257–286.
- [90] D. Barber, A.T. Cemgil, Graphical models for time-series, *IEEE Signal Proc. Mag.* 27 (6) (2010) 18–28.
- [91] N. Chater, J.B. Tenenbaum, A. Yuille, Probabilistic models of cognition: conceptual foundations, *Trends Cogn. Sci.* 10 (7) (2006) 287–291.
- [92] D. Husmeier, R. Dybowski, S. Roberts, *Probabilistic Modelling in Bioinformatics and Medical Informatics*, Springer-Verlag, 2005.
- [93] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, The MIT Press, 2006.
- [94] R.G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, 1963.
- [95] D. Jurafsky and J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, 2008.
- [96] P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, R.L. Mercer, The mathematics of statistical machine translation: parameter estimation, *Comput. Linguist.* 19 (2) (1993) 263–311.
- [97] L.R. Bahl, P.F. Brown, P.V. de Souza, R.L. Mercer, Maximum mutual information estimation of HMM parameters for speech recognition, in: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1986, pp. 49–52.
- [98] P.S. Gopalanishnan, D. Kanevsky, A. Nadas, D. Nahamoo, An inequality for rational functions with applications to some statistical estimation problems, *IEEE Trans. Inform. Theory* 37 (1) (1991) 107–113.

- [99] P.C. Woodland, D. Povey, Large scale discriminative training of hidden Markov models for speech recognition, *Comput. Speech Lang.* 16 (2002) 25–47.
- [100] E.C. Cherry, Some experiments on the recognition of speech, with one and with two ears, *J. Acoust. Soc. Am.* 25 (1953) 975–979.
- [101] M. Cooke, J.R. Hershey, S.J. Rennie, Monaural speech separation and recognition challenge, *Comput. Speech Lang.* 24 (2010) 1–15.
- [102] J.R. Hershey, S.J. Rennie, P.A. Olsen, T.T. Kristjansson, Super-human multi-talker speech recognition: a graphical modeling approach, *Comput. Speech Lang.* 24 (2010) 45–66.
- [103] M. Wohlmayr, M. Stark, F. Pernkopf, A probabilistic interaction model for multipitch tracking with factorial hidden Markov model, *IEEE Trans. Audio Speech Lang. Process.* 19 (4) (2011) 799–810.
- [104] A. Nadas, D. Nahamoo, M.A. Picheny, Speech recognition using noise-adaptive prototypes, *IEEE Trans. Acoust. Speech Signal Process.* 37 (10) (1989) 1495–1503.
- [105] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, G.E. Cooper, Probabilistic diagnosis using a reformulation of the Internist-1/QMR knowledge base. ii. evaluation of diagnostic performance, *Method. Inform. Med.* 30 (1991) 256–267.
- [106] H.-A. Loeliger, An introduction to factor graphs, *IEEE Signal Proc. Mag.* 21 (1) (2004) 28–41.
- [107] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, *SIGKDD Explor. Newsl.* 11 (2009) 10–18.
- [108] J.M. Mooij, libDAI: a free and open source C++ library for discrete approximate inference in graphical models, *J. Mach. Learn. Res.* 11 (2010) 2169–2173.
- [109] A. Frank, A. Asuncion, UCI machine learning repository, 2010. <<http://archive.ics.uci.edu/ml>>.
- [110] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE* 86 (11) (1998) 2278–2324.
- [111] J.S. Garofolo, L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, N.L. Dahlgren, Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.
- [112] A.K. Halberstadt, J. Glass, Heterogeneous measurements for phonetic classification, in: *Proceedings of EUROSPEECH*, 1997, pp. 401–404.
- [113] F. Pernkopf, T. Van Pham, J. Bilmes, Broad phonetic classification using discriminative Bayesian networks, *Speech Commun.* 51 (2) (2009) 151–166.

# A Tutorial Introduction to Monte Carlo Methods, Markov Chain Monte Carlo and Particle Filtering

# 19

**A. Taylan Cemgil***Department of Computer Engineering, Boğaziçi University 34342 Bebek, Istanbul, Turkey*

## 1.19.1 Introduction

Monte Carlo (method) (MC) is an umbrella name for an arsenal of numerical techniques for computing approximate estimates via random sampling. The estimates are very often given as the result of an intractable integral and the technique could have been named “numerical integration in high dimensions via random sampling.” The term Monte Carlo was initially coined during 1940’s by von Neumann, Ulam and Metropolis [1,2] as a “cuteness” [3] to refer to the famous casino of 1940s. However, due to the central importance of the subject and wide use of the concept, it soon became an established technical term.

Monte Carlo techniques have been further popularized in applied sciences with the wider availability of computing power, starting from the 90s, most notably in statistics, computer science, operational research and signal processing. In signal processing or operational research, MC experiments are extensively used for assessing the performance of a system or a computational method by generating a random sample of typical inputs. While the term Monte Carlo experiment is also used extensively here, the goal in such applications is “system simulation,” i.e., to understand the properties of a system under uncertainty. The Monte Carlo methods in this paper are slightly different from system simulation; we have a well defined “deterministic” computational problem with a single answer and randomization is used as a tool for approximate computation. As such, we will refer to MC in this context and the applications will be parameter estimation, prediction and model selection.

In this tutorial, we will sketch the aims, the basic techniques and the principles of Monte Carlo computation. After the illustration of the law of large numbers and central limit theorem [4], we cover basic Monte Carlo methods for sampling from elementary distributions: inversion, transform and rejection techniques [5]. We cover also Markov Chain Monte Carlo (MCMC) methods [6]; but rather than giving only the basic algorithms, we sketch the implications of some of the key results in the theory of finite Markov chains [7]. We also cover importance sampling and sequential Monte Carlo methods [8–10]. Finally we give an overview of a more advanced technique, the reversible jump method [11]. Our goal is to provide a self contained introduction with a unified notation, build up a basic appreciation of modern Monte Carlo techniques and to sharpen the readers intuition using several toy examples. Parts of this material have been used in advanced undergraduate and introductory graduate courses on Monte Carlo computation, taken primarily by students interested in techniques for data analysis with signal processing, machine learning or data mining background with some working knowledge of probability

theory and statistics. Experience has shown that even tutorial texts on statistical computation were not immediately accessible due to the notation barrier and students benefited mostly from toy examples. Inevitably, in an elementary tutorial we had to omit a lot of techniques but still tried to retain the “gist” of the subject.

## 1.19.2 The Monte Carlo principle

In an abstract setting, the main principle of a Monte Carlo technique is to generate a set of samples  $x^{(1)}, \dots, x^{(N)}$  from a target distribution  $\pi(x)$  to estimate some features of this target density  $\pi$ . Features are simply expectations of “well behaving” functions that can be approximated as averages:

$$\mathbb{E}_\pi(\varphi(x)) \approx \frac{\varphi(x^{(1)}) + \dots + \varphi(x^{(N)})}{N} \equiv \bar{E}_{\varphi,N}.$$

Provided that  $N$  is large enough, we hope that our estimate  $\bar{E}_{\varphi,N}$  converges to the true value of the expectation  $\mathbb{E}_\pi(\varphi(x))$ . More concrete examples of test functions  $\varphi(x)$  will be provided in the next section. For independent and identically distributed samples, this is guaranteed by two key mathematical results: the strong Law of Large Numbers (LLN) and the Central Limit Theorem (CLT) [4, 12]. Assuming that  $\mathbb{E}_\pi(\varphi(x)) = \mu$  and  $V_\pi(\varphi(x)) = \sigma^2$  (we have finite mean and variance), the LLN states that

$$\bar{E}_{\varphi,N} \rightarrow \mu \quad \text{a.s.}$$

Here, a.s. denotes convergence *almost surely*, meaning that  $\Pr\{\lim_{N \rightarrow \infty} |\mu - E_{\varphi,N}| = 0\} = 1$ . Whilst fluctuations are inevitable (since our approximation will be based on a random and finite sample set  $x^{(1)} \dots x^{(N)}$ ) we wish these fluctuations to be small. This is guaranteed by the CLT: for sufficiently large  $N$ , the fluctuations are approximately Gaussian distributed

$$\bar{E}_{\varphi,N} \sim \mathcal{N}\left(\bar{E}_{\varphi,N} | \mu \sigma^2 / N\right)$$

and the variance of the estimate drops with increasing  $N$ , while its mean is the desired value. This result has important practical consequences. If we can generate i.i.d. samples from the distribution of  $x$ , denoted as  $\pi(x)$ , we can estimate expectations of  $\varphi(x)$ :

1. Monte Carlo provides a “noisy” but unbiased estimate of the true value  $\mu = \mathbb{E}_\pi \varphi(x)$ .
2. The error behaves as  $O(N^{-1/2})$ .
3. The convergence rate is independent of the dimensionality of  $x$ .

The third point is particularly important. It suggests that, at least in principle, with a quite small number of samples one can compute approximate solutions for arbitrary large parameter estimation problems. All one needs is obtaining independent samples. The difficulty with this approach, however, is in generating independent samples from a target distribution  $\pi(x)$  and various Monte Carlo methods aim to provide techniques for this.

### 1.19.2.1 Illustration of the MC principle

In this section, we will illustrate the MC principle, that averages obtained from several random experiments tends to stabilize. As the running example we will focus on a problem mentioned in the famous letters between Pascal and Fermat, first writings that are considered to mark the start of the modern probability theory [13]. A French nobleman and gambler Chevalier de Méré contacted Pascal. Méré was betting that in four rolls of a single die, at least one six would turn up. Later, Méré “extended” his schema where he was betting that in 24 rolls of two dice, a pair of sixes would turn up. He was not happy with this new schema and was calling Pascal for an explanation.

Indeed, the analytical solution for this problem is elementary; we merely calculate the probability that in 4 consecutive independent throws at least a 6 comes up. A quick calculation shows indeed there is a slight advantage for Méré:

$$\begin{aligned}\Pr\{\text{Méré wins in 4 throws}\} &= 1 - \Pr\{\text{no 6 is thrown}\}^4 \\ &= 1 - (5/6)^4 = 0.5177.\end{aligned}$$

In a sense, this experiment is equivalent to playing head and tail with a carefully forged coin. The bias is small and may not be easily detected by a naive opponent. Like all Casinos, Méré exploited this subtle bias to his advantage to earn money on average in repeated trials.

However, the analytical solution for the two dice game reveals that throwing the dice only 24 times is not a good bet for Méré:

$$\Pr\{\text{Méré wins in 24 throws}\} = 1 - (35/36)^{24} = 0.4914,$$

but with 25 throws, the odds turn into his favor as

$$\Pr\{\text{Méré wins in 25 throws}\} = 1 - (35/36)^{25} = 0.5055.$$

The natural question here is if one could detect with certainty that there is indeed a systematic deviation from 0.5 by just looking at the outcomes of a sequence of games. Indeed for many problems of interest, such probability calculations will be significantly harder, if at all possible, and it is natural to ask how reliable it is to estimate such quantities using data obtained from MC experiments.

Playing the game repetitively is a MC experiment. By just recording the outcome of each game (and the total number of games played), we could in principle estimate the winning probability. To be more precise, consider the single die, 4 times throw game. Let  $x_k^{(n)}$  denote the outcome of the die at  $k$ th throw in the  $n$ th game. Formally, a game is represented by the tuple  $x \equiv (x_1, x_2, x_3, x_4)$ . We assume a fair die, as such for  $k = 1, \dots, 4$

$$x_k \sim \mathcal{U}(1, 6),$$

where  $\mathcal{U}(a, b)$  denotes a uniform distribution on integers  $x$  such that  $a \leq x \leq b$ . Since each throw in the game is independent, the target distribution  $\pi(x)$  is the uniform distribution on  $\mathcal{X} = \{1, \dots, 6\}^4$  and each game is simply a random point in  $\mathcal{X}$ .

To estimate the probability of winning, we define formally the indicator function for winning the  $n$ th game

$$\varphi(x^{(n)}) = \mathbb{I} \left\{ 0 < \sum_{k=1}^4 \mathbb{I} \left\{ x_k^{(n)} = 6 \right\} \right\}.$$

Here,  $\mathbb{I}\{x\}$  is an indicator function that is 1 if  $x$  is true and 0 if false. The test function  $\varphi$  looks awkward but it just checks if in 4 throws at least a 6 is obtained. Finally, the probability  $\theta$  that Méré wins in 4 throws can be estimated as

$$\theta = \mathbb{E}_\pi(\varphi(x)) \approx \frac{1}{N} \sum_{n=1}^N \varphi(x^{(n)}) = \bar{E}_{\varphi,N}.$$

Similarly, for the two dice, 24 throws game, formally we let  $x_{i,k}^{(n)}$  denote the outcome of the die  $i$  at  $k$ th throw in the  $n$ th game.

$$\begin{aligned} \varphi(x^{(n)}) &= \mathbb{I} \left\{ 0 < \sum_{k=1}^{24} \mathbb{I} \left\{ x_{1,k}^{(n)} = 6 \right\} \mathbb{I} \left\{ x_{2,k}^{(n)} = 6 \right\} \right\}, \\ \bar{E}_{\varphi,N} &= \frac{1}{N} \sum_{n=1}^N \varphi(x^{(n)}). \end{aligned} \quad (19.1)$$

The mean and variance of  $\varphi(x^{(n)})$  are

$$\begin{aligned} \mathbb{E}(\varphi(x^{(n)})) &= \theta \\ \text{Var}\{\varphi(x^{(n)})\} &= \theta(1 - \theta) \end{aligned}$$

respectively.<sup>1</sup> As these are finite, the strong law of large numbers applies. The law guarantees the very intuitive fact that with increasing number of games  $N$ , the estimates become increasingly more and more accurate. Technically, when  $N \rightarrow \infty$  we have convergence

$$\bar{E}_{\varphi,N} \rightarrow \theta.$$

The “speed” is given by the central limit theorem, that says that for large  $N$  the estimate is distributed

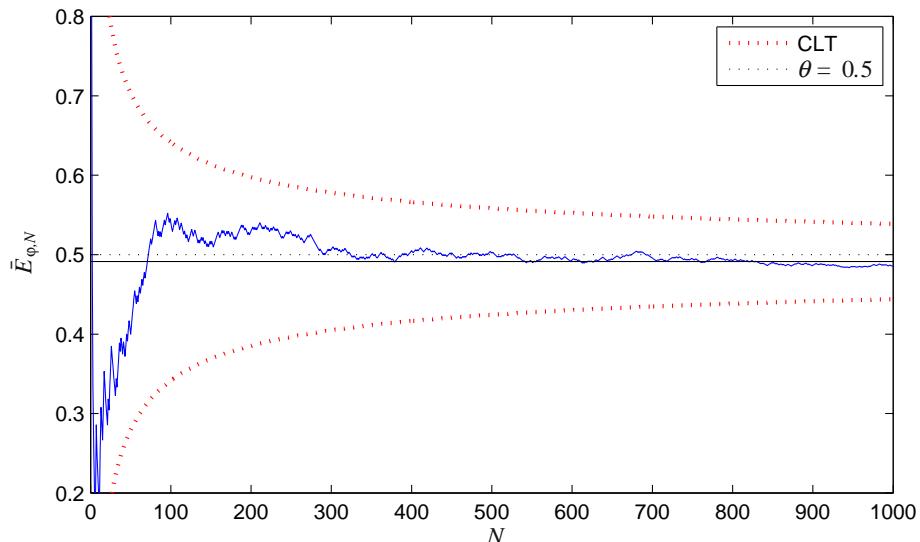
$$\bar{E}_{\varphi,N} \sim \mathcal{N}(\bar{E}_{\varphi,N} | \theta, \theta(1 - \theta)/N).$$

Provided that we can generate independent samples, the error will be  $O(N^{-1/2})$ . The most important aspect of this convergence result, that can not be overemphasized is that the error does not depend on the dimensionality of the parameter to be estimated!

To illustrate these concepts with an example consider Figure 19.1 where we simulate the estimate for a two dice, 24 throw game case for several  $N$ . The path shows the estimate  $\bar{E}_{\varphi,N}$  in Eq. (19.1) for each  $N$ . We can see that the path converges to the true value. For each  $N$ , the error bars correspond to the standard deviations given by the central limit theorem.

---

<sup>1</sup>The mean and variance of a Bernoulli random variable.



**FIGURE 19.1**

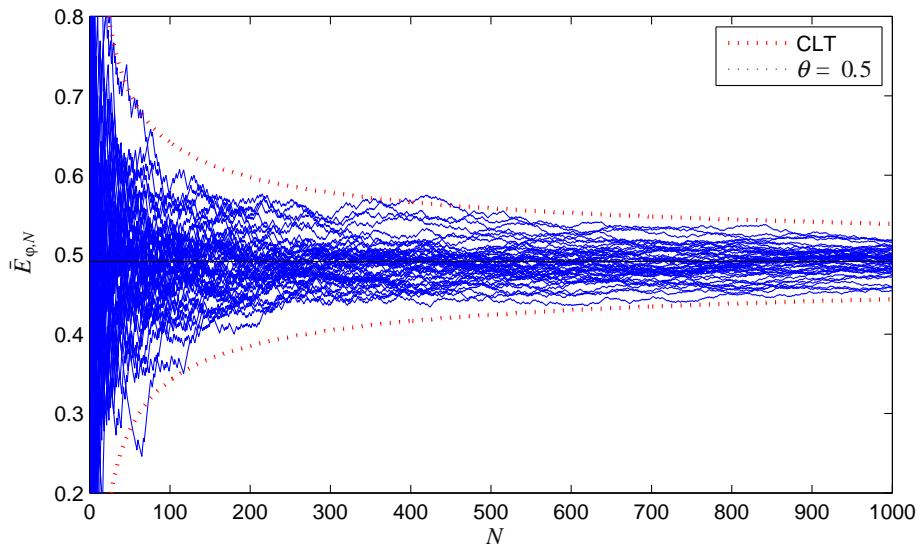
The estimate of winning probability for a pair of 6 in 24 throws game. The horizontal solid line shows the true odds  $\theta_{\text{true}}$  that is smaller than 0.5. The path shows the estimate  $\bar{E}_{\varphi, N}$  as a function of  $N$ . Law of large numbers says that all these curves eventually will converge to the true  $\theta_{\text{true}}$ , shown as a solid line.

To illustrate the implication of the central limit theorem, consider now a collection of estimates, obtained from results of  $N$  games per night with a total of  $K$  different nights. Formally, the outcome of the  $n$ th game at the  $k$ th night is denoted as  $\varphi(x^{(n,i)})$  for  $i = 1, \dots, K$  and  $n = 1, \dots, N$ . The estimate at the  $i$ th night after observing the  $N$ th game as

$$\bar{E}_{\varphi, N}^{(i)} = \frac{1}{N} \sum_{n=1}^N \varphi(x^{(n,i)}).$$

Now, we visualise all the paths together, where each path is depicted corresponds to the sequence of estimates for each independent night  $i$  (Figure 19.2). Each path corresponds to the estimate as a function of the number of games played  $N$ . We can clearly see that all the independent paths eventually stay in the “narrow tube,” as predicted by the CLT.

In the above examples, instead of experimenting with real dice, we have simulated the experiment on a computer. It turns out that it is surprisingly delicate to simulate “truly” random numbers on deterministic hardware. Instead, *pseudo-random numbers* are generated using deterministic algorithms and are briefly reviewed in the next section.

**FIGURE 19.2**

Verification of the central limit theorem. We estimate the winning probability for the 2 dice, 24 throws game. Each path corresponds to the estimate  $\bar{E}_{\varphi, N}^{(i)}$  as a function of  $N$ , where  $i = 1, \dots, K$  denotes the independent sequences of games. CLT says that at any fixed  $N$ ,  $\bar{E}_{\varphi, N}^{(i)}$  the values of these estimates is approximately normal distributed with standard deviation  $\sigma/\sqrt{N}$ . For each  $N$ , the red dotted curves designate the  $[\theta_{\text{true}} - 3\sigma/\sqrt{N}, \theta_{\text{true}} + 3\sigma/\sqrt{N}]$ . It is a common folklore in statistics to plot the  $\pm 3\sigma$  interval for a univariate Gaussian as this interval contains more than 99% of the probability mass. As given by the CLT, the paths exit the  $\pm 3\sigma/\sqrt{N}$  tube only occasionally.

### 1.19.2.2 Random number generation

“The generation of random numbers is too important to be left to chance”<sup>2</sup> and truly random numbers are impossible to generate on a deterministic computer. Published tables or other mechanical methods such as throwing dice, flipping coins, shuffling cards or turning the roulette wheels are clearly not very practical for generating the random numbers that are needed for computer simulations. Other techniques, more suited to computation exist; these rely on chaotic behavior, such as the thermal noise in Zener diodes or other analog circuits as well as the atmospheric noise (see, e.g., [www.Random.org](http://www.Random.org)) or running a hash function against a frame of a video stream. Still, the vast amount of random numbers are obtained from *pseudo-random number* generators. Apart from being very efficient, one additional advantage of these techniques is that the sequences are reproducible by setting a seed, this property is key for debugging MC code.

The most well known method for generating random numbers is based on a Linear Congruential Generator (LCG). The theory is well understood, the method is easy to implement and it runs very fast.

<sup>2</sup>Wikipedia entry on Pseudo Random Number generator.

A LCG is defined by the recurrence relation:

$$x_{n+1} \equiv (ax_n + c) \pmod{M}.$$

If the coefficients  $a$  and  $c$  are chosen carefully (e.g., relatively prime to  $M$ ),  $x$  will be roughly uniformly distributed between 0 and  $M - 1$ . Roughly uniformly means that, the sequence of numbers  $x_n$  will pass many reasonable tests for randomness. A such test suite is the so called DIEHARD tests, developed by George Marsaglia [14] that are a battery of statistical tests for measuring the quality of a random number generator.

A more recently proposed generator is the Mersenne Twister algorithm, by Matsumoto and Nishimura [15]. It has several desirable features such as a long period and being very fast. Many public domain implementations exist and it is the preferred random number generator for statistical simulations and Monte Carlo computations.

Good random number generators provide uniformly distributed numbers on an interval. Hence, provided we have a good random number generator that can generate uniformly random integers (say between 0 and  $2^{64} - 1$ ), the resulting integers can be used to generate double precision numbers almost uniformly distributed in  $[0, 1]$ . We will assume in the sequel that we have access to a good uniform random number generator that can generate real numbers on  $[0, 1]$ ; knowing that in practice such a generator generates only rational numbers (say double precision floating point numbers). We will denote this generator by  $\mathcal{U}(0, 1)$  and denote a realization, i.e., a sample drawn as  $u \sim \mathcal{U}(0, 1)$ .

### 1.19.3 Basic techniques for simulating random variables

The techniques described in this section sketch the basic techniques to generate (mostly univariate and bivariate random variables) with a given density. This is a surprisingly deep and interesting subject touching many facets of computer science. For further information, the reader is invited to look at the fantastic book by Luc Devroye, on non uniform random variate generation [16].

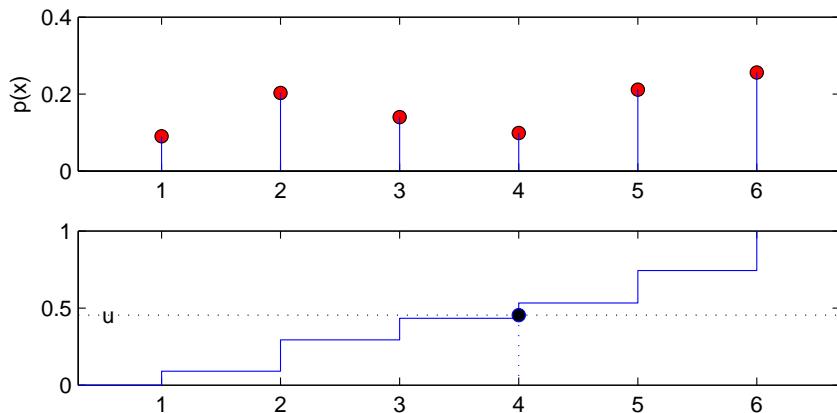
#### 1.19.3.1 Inversion

The basic technique is based on transformation; we generate a uniform random number  $u \sim \mathcal{U}(0, 1)$  and transform it to obtain  $x = g(u)$  where  $g$  is a function. This method requires calculating the cumulative density function and inverting it. Recall the definition of a Cumulative Density function (CDF)

$$F_X(x) = \int_{-\infty}^x f_X(\tau) d\tau,$$

where  $f_X$  is the probability density of the random variable  $X$ . The key observation behind the inversion method is the fact that when  $X \sim f_X$ , the quantity  $U = F_X(X)$ , when viewed as a random variable, is uniformly distributed on  $[0, 1]$ . To see this, assume that the inverse of  $F_X$  exists, and we can find  $X = F_X^{-1}(U)$ ,

$$\begin{aligned} F_X(x) &= \Pr\{X < x\}, \\ F_X(F_X^{-1}(u)) &= \Pr\{F_X^{-1}(U) < F_X^{-1}(u)\}, \\ u &= \Pr\{U < u\} \quad 0 \leq u \leq 1. \end{aligned}$$

**FIGURE 19.3**

Generation of samples from a discrete distribution. We generate  $u$  uniformly on  $[0, 1]$  and find the corresponding  $x$  via the generalised inverse  $F^{-}(u)$ . The discrete case is particularly intuitive: think of dividing a stick of length 1 into pieces of length  $\pi_i = f_X(x_i)$  and label each region with  $x_i$ . Select a point  $u$  uniformly random and return the label of the region that  $u$  falls in.

By taking the derivative, we see that the density of  $u$  is uniform. The argument works also in the opposite direction: when we first choose uniform  $U$  and pass it through the function  $X = F_X^{-1}(U)$ , the density of  $X$  will be  $f_X$ . To allow for discrete distributions or continuous densities with atoms (where single points have positive probability mass), we define the generalised inverse CDF

$$F^{-}(u) \equiv \inf\{x : F(x) \geq u\}.$$

The  $x$  generated by this method are guaranteed to be distributed by  $f_X$ . Figure 19.3 illustrates this construction. The generalized inverse allows for jumps in the cumulative density when atoms have nonzero probability (see Figure 19.4).

**Example 1.** Exponential random variables can be derived by the inversion method from uniform samples. The exponential density with *rate* parameter  $\lambda$  is

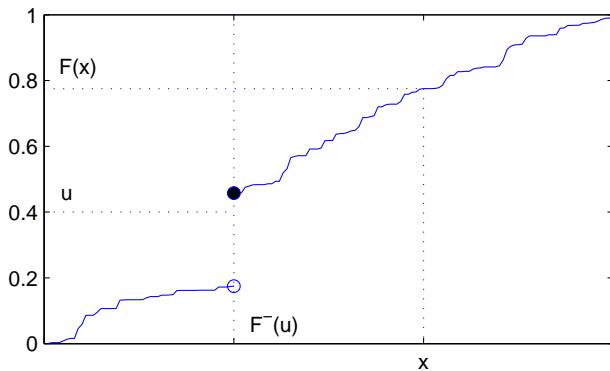
$$\mathcal{E}(x; \lambda) = \lambda \exp(-\lambda x).$$

The CDF of the exponential distribution is found as

$$F(x) = \int_0^x \lambda \exp(-\lambda \tau) d\tau = -\exp(-\lambda x) + 1.$$

The generalised inverse is obtained via

$$\begin{aligned} u &= F(x) = 1 - \exp(-\lambda x), \\ x &= -\log(1-u)/\lambda = F^{-}(u). \end{aligned}$$

**FIGURE 19.4**

Visualisation of the generalised inverse. The blue curve shows a CDF of a probability distribution with an atom, as can be seen from the discontinuity. For a given probability  $u$ ,  $F^-(u)$  is the “minimum”  $x$  such that the probability of the interval  $(\infty, x]$  is equal or exceeds  $u$ .

We can even avoid calculating  $1 - u$  by noting that both  $u$  and  $w = 1 - u$  are distributed uniformly on  $[0, 1]$ , so

$$x = F^-(w) = -\log(w)/\lambda.$$

### 1.19.3.2 Transformation

The generalised inverse method works by applying a suitable function (the generalized inverse  $F^-(u)$ ) on a random input with a known density ( $u$  with a uniform density). This method can be made more general by a technique known as the change of variables [4].

To illustrate the transformation method, we will first give an example where the mapping function will be affine (that is linear plus a constant term). Suppose we are given a random variable  $X$  with density  $f_X(x)$  and wish to find the density of  $Y$  where

$$Y = aX + b.$$

It is clear that the density of  $Y$ , denote by  $f_Y$  should be somewhat related to  $f_X$ . For example if  $X$  is uniform ( $f_X = \mathcal{U}$ ) on  $[0, 1]$  and  $a = 2$  and  $b = -1$ , one could see that  $Y$  is uniform on  $[-1, 1]$ . The general case, when  $f_X$  is any density, is perhaps slightly harder to see. To find the density of  $Y$ , we first will find the CDF  $F_Y(y)$  of  $Y$  and obtain the desired density  $f_Y(y)$  by taking the derivative w.r.t.  $y$ . In particular,

$$F_Y(y) = \Pr\{Y \leq y\} = \Pr\{aX + b \leq y\} = \begin{cases} \Pr\{X \leq (y - b)/a\}, & a > 0, \\ \Pr\{X \geq (y - b)/a\}, & a < 0. \end{cases}$$

By taking derivative we see that the density is given by

$$f_Y(y) = \frac{dF_Y}{dy} = \frac{1}{|a|} f_X((y - b)/a).$$

For the more general case, we have a more general mapping  $g$  where

$$\begin{aligned} Y &= g(X), \\ X &= g^{-1}(Y). \end{aligned}$$

The density can be found as

$$\begin{aligned} F_Y(y) &= \Pr\{Y \leq y\} = \Pr\{g(X) \leq y\} = \Pr\{X \leq g^{-1}(y)\} \\ f_Y(y) &= \frac{dF_Y}{dy} = f_X(g^{-1}(y)) \left| \frac{dg^{-1}(y)}{dy} \right| \equiv f_X(x(y)) |J(y)| \end{aligned} \quad (19.2)$$

here, the derivative of  $g^{-1}$  is denoted as the *Jacobian* and the absolute value ensures the positivity of the densities. In many textbooks it is common to drop the explicit reference to the function  $g$ ; authors often use  $y(x)$  for  $y = g(x)$  and  $x(y)$  for the inverse  $g^{-1}(y)$ . In the sequel, we will also adopt this notation. The multivariate case is analogous; here we illustrate the bivariate case, the most encountered case in derivations. The function  $g$ , that is also called a *transform*, is defined as

$$(y_1, y_2) = g(x_1, x_2).$$

If we can invert  $g$ , which is not always easy to do analytically, we find

$$\begin{aligned} x_1 &= x_1(y_1, y_2), \\ x_2 &= x_2(y_1, y_2). \end{aligned}$$

The derivative term is the *Jacobian* determinant defined as

$$J = \begin{vmatrix} \partial x_1 / \partial y_1 & \partial x_2 / \partial y_1 \\ \partial x_1 / \partial y_2 & \partial x_2 / \partial y_2 \end{vmatrix} = J(y_1, y_2).$$

Consequently, the density of the transformed variable, similar to Eq. (19.2), is given by

$$f_Y(y_1, y_2) = f_X(x_1(y_1, y_2), x_2(y_1, y_2)) |J(y_1, y_2)|.$$

**Example 2.** To illustrate the method, we describe the Box-Müller method for generating normal random variables. The Box-Müller method generates the pair  $(x_1, x_2)$

$$\begin{aligned} x_1 &\sim \mathcal{E}(x_1; 1/2) = \frac{1}{2} \exp(-x_1/2), \\ x_2 &\sim \mathcal{U}(x_2; 0, 2\pi) = \frac{1}{2\pi} \mathbb{I}\{0 \leq x_2 \leq 2\pi\}, \end{aligned}$$

where  $x_1$  is the square of the magnitude and  $x_2$  is the angle of a point in polar coordinates. The method transforms this point into cartesian coordinates

$$\begin{aligned} y_1 &= \sqrt{x_1} \cos(x_2), \\ y_2 &= \sqrt{x_1} \sin(x_2). \end{aligned}$$

It turns out, that  $y_1$  and  $y_2$  obtained via this method are independent and unit Gaussian ( $\mathcal{N}(0, 1)$ ) distributed. To show this, we find the inverse mapping

$$\begin{aligned}x_1 &= y_1^2 + y_2^2, \\x_2 &= \arctan(y_2/y_1).\end{aligned}$$

The Jacobian determinant is found as

$$J = \begin{vmatrix} \partial x_1 / \partial y_1 & \partial x_2 / \partial y_1 \\ \partial x_1 / \partial y_2 & \partial x_2 / \partial y_2 \end{vmatrix} = \begin{vmatrix} 2y_1 & \frac{1}{1+(y_2/y_1)^2} \frac{-y_2}{y_1^2} \\ 2y_2 & \frac{1}{1+(y_2/y_1)^2} \frac{1}{y_1} \end{vmatrix} = 2.$$

The resulting density is

$$\begin{aligned}f_Y(y_1, y_2) &= \mathcal{E}(x_1(y_1, y_2); 1/2)\mathcal{U}(x_2(y_1, y_2); 0, 2\pi)|J(y_1, y_2)| \\&= \frac{1}{2}\exp(-(y_1^2 + y_2^2)/2)\frac{1}{2\pi}2 = \frac{1}{\sqrt{2\pi}}\exp(-y_1^2/2)\frac{1}{\sqrt{2\pi}}\exp(-y_2^2/2) \\&= \mathcal{N}(y_1; 0, 1)\mathcal{N}(y_2; 0, 1).\end{aligned}$$

### 1.19.3.3 Rejection sampling

Rejection sampling is a technique for indirectly sampling from a target distribution  $\pi$  by sampling from a *proposal* distribution  $q$ . We reject some of the generated samples to compensate for the fact that  $q \neq \pi$ . The algorithm is as follows: We sample  $x^{(i)}$  for  $i = 1, \dots, N$  independently from  $q$ . We accept the sample  $x^{(i)}$  with acceptance probability  $a(x^{(i)})$  where the acceptance probability is

$$a(x) \equiv \frac{\pi(x)}{Mq(x)}.$$

Here,  $M$  is a positive number that guarantees  $\pi(x) \leq Mq(x)$  for all  $x$ , i.e., that  $Mq(x)$  completely covers the density  $\pi(x)$ . The approach also works when the normalization constant  $Z$  of  $\pi$  is unknown, that is we can only evaluate  $\phi$  pointwise where

$$\pi(x) = \frac{1}{Z}\phi(x).$$

In this case, we let the acceptance probability be

$$\tilde{a}(x) \equiv \frac{\phi(x)}{\tilde{M}q(x)},$$

where  $\phi(x) \leq \tilde{M}q(x)$  for all  $x$ . If a suitable  $M$  (or  $\tilde{M}$ ) can be found, the algorithm is simple to implement and requires only sampling from  $q$  and pointwise evaluation of  $\phi$ . To understand the idea behind rejection sampling, we observe that for any density function  $f(x)$ , (including obviously our target  $\pi$  and the proposal  $q$ ) we have the following identity:

$$f(x) = \int_0^{f(x)} 1 d\tau = \int \mathbb{I}\{0 \leq \tau \leq f(x)\} d\tau. \quad (19.3)$$

In other words,  $f(x)$  can be written as a *marginal density* of a distribution with density  $\mathbb{I}\{0 \leq \tau \leq f(x)\}$  on the extended space  $(x, \tau)$ . This density is simply the uniform density over the volume under the curve  $f(x)$ . The equation Eq. (19.3) says that generating samples from  $f(x)$  is equivalent to uniformly generating samples on the set  $A = \{(x, \tau) : 0 \leq \tau \leq f(x)\}$  and then simply ignoring the  $\tau$  coordinate. When  $f(x) = \phi(x)/Z$  and we don't know  $Z$

$$f(x) = \frac{1}{Z} \phi(x) = \frac{1}{Z} \int_0^{\phi(x)} 1 d\tau = \frac{1}{Z} \int \mathbb{I}\{0 \leq \tau \leq \phi(x)\} d\tau. \quad (19.4)$$

By integrating over  $x$  on both sides we get the normalization constant  $Z$  as the area of the region  $A = \{(x, \tau) : 0 \leq \tau \leq \phi(x)\}$ :

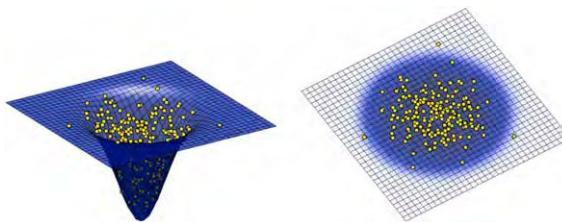
$$Z = \int \int \mathbb{I}\{0 \leq \tau \leq \phi(x)\} d\tau dx.$$

Hence

$$f(x) = \frac{\int \mathbb{I}\{0 \leq \tau \leq \phi(x)\} d\tau}{\int \int \mathbb{I}\{0 \leq \tau \leq \phi(x)\} d\tau dx}.$$

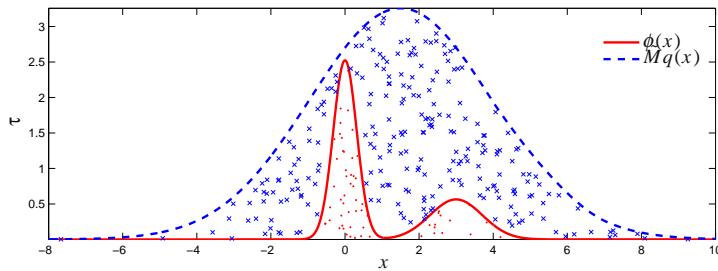
This equation suggests that  $f(x)$  is the marginal density of uniform density over the set  $A$ . A useful metaphor for understanding the concept is of thinking fishes in a lake [5], where  $x$  is a coordinate on the water surface and  $\tau$  is the depth, which is illustrated in Figure 19.5. If all fishes in the lake are distributed uniformly in the water, when viewed from above they will be marginally distributed with density  $f(x)$  (see Figure 19.6).

Note that the reasoning of Eq. (19.3) works also in the reverse direction: when we have samples  $x^{(i)}$  for  $i = 1, \dots, N$  generated from a proposal  $q(x)$ , we can select the  $\tau$  coordinate for each  $x^{(i)}$  simply by choosing  $\tau^{(i)}$  uniformly on  $[0, Mq(x^{(i)})]$ . The resulting pairs  $(x^{(i)}, \tau^{(i)})$  will be uniformly distributed on the set  $A_{Mq} = \{(x, \tau) : 0 \leq \tau \leq Mq(x)\}$ . Provided that  $\phi(x) \leq Mq(x)$ , if we now select only the pairs such that  $\tau^{(i)} < \phi(x^{(i)})$ , these accepted tuples will be distributed uniformly on the set  $A_\phi = \{(x, \tau) : 0 \leq \tau \leq \phi(x)\}$ . But, by Eq. (19.4), the marginal of this uniform density will be  $p$ .



**FIGURE 19.5**

Fishes in a lake. We view the bivariate density function  $f(x)$  as the bottom surface of a lake. Take uniformly distributed points in this volume. When viewed from above, i.e., when we ignore their depth, the points will be distributed more densely where the lake is deeper.

**FIGURE 19.6**

Rejection sampling. Sampling  $x$  from the proposal  $q(x)$  and then sampling the associated  $\tau$  uniformly on  $[0, \tilde{M}q]$  results in uniformly distributed tuples  $(x, \tau)$  (all the points under  $\tilde{M}q(x)$ ). Restricting those to only points such that  $\tau \leq \phi(x)$  provides the accepted points (red dots), the other points are rejected (blue crosses). Note that the probability  $\Pr\{\tau \leq \phi(x)\} = \phi(x)/\tilde{M}q(x)$  is the acceptance probability  $\tilde{\alpha}(x)$ .

**Algorithm 1.** Rejection Sampling

---

```

1: Construct an easy to sample density  $q(x)$  and positive number  $\tilde{M}$  such that  $\phi(x) < \tilde{M}q(x)$ 
2: for  $i = 1, 2, 3, \dots$  do
3:   Draw  $x^{(i)} \sim q(x)$ 
4:   Accept  $x^{(i)}$  with probability  $\tilde{\alpha}(x^{(i)}) = \phi(x^{(i)})/\tilde{M}q(x^{(i)})$ 
5: end for

```

---

## 1.19.4 Markov Chain Monte Carlo

In Markov Chain Monte Carlo methods [6, 17–20], instead of directly attempting to sample from  $\pi(x)$ , we sample from a Markov chain with transition density (“kernel”)  $K(x^{(n)}|x^{(n-1)})$  where  $n$  indexes the sample number.<sup>3</sup> To see how this connects to our ultimate desire to sample from a target  $\pi(x)$  consider first drawing samples from the Markov chain. Given an initial state  $x^{(1)}$ , we draw  $x^{(2)}$ , then conditioned on  $x^{(2)}$  we draw  $x^{(3)}$  and so on by iteratively drawing from  $K(x^{(n)}|x^{(n-1)})$ . The distribution of state occupancy at time  $n$ ,  $\pi_n(x)$ , is found from

$$\pi_n(x) = \int K(x|x')\pi_{n-1}(x')dx'$$

---

<sup>3</sup>Note that, the sample number is conceptually different than the “time” index in a time series model; for example, in a time series model  $x_t$  corresponds to the state at time  $t$ , whereas here  $x^n$  can correspond to a vector of all state variables—a full state trajectory. In this context, it is convenient to think of  $n$  simply as the iteration number of the sampling algorithm and  $x^{(n)}$  as a particular realisation of all random variables being sampled.

which we write more compactly as  $\pi_n = K\pi_{n-1}$ . This generates a sequence of marginal distributions  $\pi_1, \pi_2, \dots$ . Does this sequence of distributions converge and, if so, what does it converge to? A stationary distribution,  $\pi$  satisfies  $\pi = K\pi$ . A key result here is that for an ergodic chain (i.e., irreducible (chain) and aperiodic (chain)) there exists a unique distribution  $\pi$  for which  $\pi = K\pi$  and therefore a unique stationary distribution to which all the occupancy distributions converge, irrespective of their initial states [7]. For finite state Markov chains, irreducibility means that each state can be visited starting from each one (related to connectedness of the state transition diagram) and aperiodicity means that each state can be visited at any time  $n$  larger than some fixed number.

The first question is, given a Markov chain with transition kernel  $K$ , how to find the marginal distribution  $\pi_n(x)$  for large  $n$ . The second question is how to construct a transition kernel that is easy to sample from and converges to the desired target distribution. In the finite state case, a solution to the first question is provided by the eigenvalue-eigenvector decomposition of the transition matrix that will be introduced in the next section. The answer to the second question turns out to be very elegant and simple and underlies the celebrated Metropolis-Hastings Markov Chain Monte Carlo method [2,21] that will be covered afterwards.

### 1.19.4.1 Stationary distribution of a Markov Chain with finite number of states

We first focus our attention on the convergence of a Markov chain and illustrate results in the finite state case. The finite state case may seem to be too restrictive, and in a sense it indeed is. From a mathematical perspective, the theory of Markov chains for more general spaces is more delicate and it does not rely on linear algebra. Nevertheless, it turns out that the results for more general state spaces have similar flavor and understanding what is happening in the finite case provides the basic intuition about the issues involved. For technicalities on countable state spaces, see [7] or uncountable state spaces, see [22], Chapter 13.

For the finite state case, we proceed as follows. First, we note that the marginals of a Markov chain satisfy the following recursive relation

$$\pi_n(x_n) = \sum_{x_{n-1}} K(x_n|x_{n-1})\pi(x_{n-1}).$$

Now, a probability mass function on a finite state space is simply a vector of positive entries which sum up to one so in index notation we write

$$\pi_n(i) = \sum_j K_{i,j}\pi_{n-1}(j),$$

where the marginal distribution of the chain is denoted by  $\pi_n(i) \equiv \pi_n(x_n)$  and the transition model is denoted by the transition matrix  $K$  with the entries  $K_{i,j} = K(x_n = i|x_{n-1} = j)$ . Adopting the matrix notation, we write  $\pi_n = K\pi_{n-1}$ . This gives the expression for the marginal  $\pi_n$  as a function of the initial occupancy density  $\pi_0$

$$\pi_n = K^n\pi_0.$$

This is simply a (linear) fixed point iteration and the question reduces to finding a stationary point  $\pi$  that satisfies

$$\pi = \lim_{n \rightarrow \infty} K^n\pi_0 \equiv K^\infty\pi_0,$$

where  $K^\infty = \lim_{n \rightarrow \infty} K^n$ . An eigen-decomposition of the transition matrix  $K$

$$K = D\Lambda D^{-1}$$

provides insight about the limit behavior of  $K^n$  where  $D$  is the matrix of eigenvectors and  $\Lambda$  is a diagonal matrix of eigenvalues. It turns out that for transition matrices  $K$ , the maximum eigenvalue is always 1. If the magnitude of the second largest eigenvalue  $\lambda_2$  is strictly less,  $|\lambda_2| < 1$ , we have a unique stationary distribution. To see this consider

$$K^n = D\Lambda D^{-1}D\Lambda D^{-1} \cdots D\Lambda D^{-1} = D\Lambda^n D^{-1}.$$

But as  $\Lambda$  is diagonal,  $\Lambda^n$  for large  $n$  is a matrix very close to all zeros matrix with only a single 1 on the diagonal. Say without loss of generality that the first eigenvalue  $\lambda_1 = 1$  is at position  $\Lambda(1, 1)$  (otherwise construct a permutation matrix  $P$  and consider  $T = (DP^\top)(P\Lambda P^\top)(PD^{-1}) = \tilde{D}\tilde{\Lambda}\tilde{D}^{-1}$ ). Then  $K^\infty$  is the rank-one matrix given by the outer product

$$K^\infty = D(:, 1)D^{-1}(1, :).$$

Here,  $D(:, i)$  denotes the  $i$ th column of the matrix  $D$  and  $D^{-1}(i, :)$  is the  $i$ th row of  $D^{-1}$ . These are respectively the right and left eigenvectors. But what is  $D^{-1}(1, :)$ ? By definition we have

$$\begin{aligned} D^{-1}K &= \Lambda D^{-1}, \\ D^{-1}(1, :)K &= \Lambda(1, 1)D^{-1}(1, :) = D^{-1}(1, :). \end{aligned}$$

But as  $K$  is a transition matrix, all its columns sum up to one. In other words, if we would left multiply  $K$  by an all-ones vector  $v$ , we get  $vK = v$  so it is easy to see that the left eigenvector corresponding to the largest eigenvalue is proportional to the all ones vector

$$D^{-1}(1, :) = (1 \ 1 \ \dots \ 1).$$

But this implies that the columns of  $K^\infty$  are all the same as

$$K^\infty = [D(:, 1), D(:, 1), \dots, D(:, 1)].$$

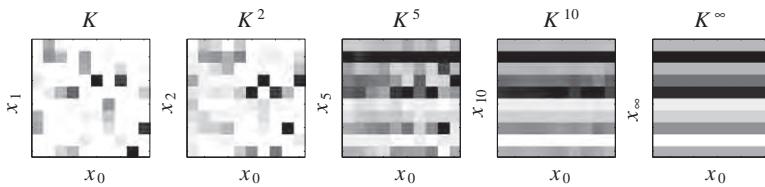
We can conclude that, regardless of the initial state, the probabilities are all the same and the stationary density is proportional to the (right) eigenvector corresponding to the eigenvalue  $\lambda_1 = 1$ . This is pictorially illustrated using a random transition matrix  $K$  in Figure 19.7. In the MC literature, we use the term transition kernel instead of transition matrix as the latter is restricted only to countable state spaces.

A natural question to ask is how fast a chain converges to stationarity. For Markov Chains, the rate of convergence is governed by the second eigenvalue. The convergence to the stationary distribution is geometric and is given by

$$\|K^n\pi_0 - \pi\|_{\text{var}} \leq C|\lambda_2|^n,$$

where  $C$  is a positive constant and the function  $\|\cdot\|_{\text{var}}$  measures the distance between two densities. It is known as the total variation norm and is given by

$$\|P - Q\|_{\text{var}} \equiv \frac{1}{2} \sum_{s \in \mathcal{X}} |P(s) - Q(s)|.$$

**FIGURE 19.7**

The powers  $K^\tau$  of a transition matrix  $K(x_n = i|x_{n-1} = j)$  with  $\tau = 1, 2, 5, 10, \infty$ . Darker colors correspond to higher transition probabilities. The identical columns of  $K^\infty$  depicts the stationary distribution, which is reached independent of any starting density. We see that for this transition matrix  $K$ , even after a few iterations  $\tau$  the columns of  $K^\tau$  are close to the stationary density, suggesting a rapid convergence. The magnitude of the second eigenvalue is  $|\lambda_2| \approx 0.64$  which verifies that our observation is indeed correct.

However, for a large transition matrix, it may be hard to show algebraically that  $|\lambda_2| < 1$ . Fortunately, there is a convergence result that states that if  $K$  is irreducible and aperiodic, then there exist  $0 < r < 1$  and  $C > 0$  s.t.

$$\|K^n\pi_0 - \pi\|_{\text{var}} \leq Cr^n,$$

where  $\pi$  is the invariant distribution [20]. This result also generalizes to more general Markov chains, such as on countable infinite or uncountable state spaces where explicit calculation of the second largest eigenvalue of the transition kernel is intractable.

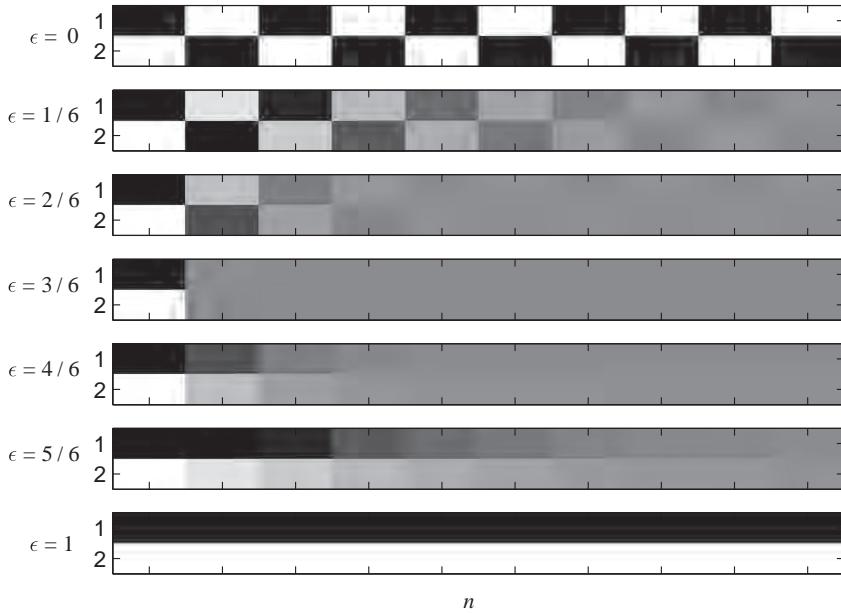
To see an example how convergence speed is affected consider a very simple chain  $x_n$  with two states, labeled as 1 and 2 with the transition matrix

$$K = \begin{pmatrix} \epsilon & 1-\epsilon \\ 1-\epsilon & \epsilon \end{pmatrix}, \quad (19.5)$$

where  $0 \leq \epsilon \leq 1$ . In other words, if at time  $t$  the chain is in state  $i$ , with probability  $\epsilon$ , the chain stays put, or jumps to the other state with probability  $1 - \epsilon$ . One natural question to ask is: where would we find the chain if we stop it at a large  $n$ ? Say we start initially from state  $i$  but in the previous section we saw that this initial choice won't matter (most of the time). It turns out that for  $0 < \epsilon < 1$  the answer is 0.5: it is equally likely to find the chain in any of the two states, regardless of the initial state but for each  $\epsilon$  the rate of convergence is different. We illustrate these concepts on Figure 19.8.

#### 1.19.4.2 Designing a Markov Chain for a given target distribution: The Metropolis-Hastings method

The discussion of the previous section suggests that, if we can design a transition kernel  $K$  such that the target density  $\pi(x)$  is its stationary distribution, at least in principle one can generate samples from the Markov chain that eventually will tend to be drawn from the target distribution. After ignoring samples obtained from an initial “burn in” period, as the chain moves towards the stationary distribution, the generated samples can be subsequently used to estimate expectations under the target distribution, as if they are independent. There are a couple of caveats: formally we require the chain to be

**FIGURE 19.8**

Convergence to the stationary distribution for the transition matrix  $K$  defined in Eq. (19.5) with  $\epsilon = 0, 1/6, 2/6, \dots, 1$  from top to bottom. Horizontal axis denotes the time  $n$  and the vertical axis is the state  $x$ . The probability  $\pi_n(x)$  is high for darker regions. For  $\epsilon = 1$  (bottom), the state transition matrix is  $K = I$  and the states are no longer connected. The chain fails to be irreducible and the chain fails to have a unique stationary distribution. For  $\epsilon = 0$  (top), the states alternate periodically. The chain is periodic and similarly fails to have a unique stationary distribution. For all other cases with  $0 < \epsilon < 1$  we have convergence to the uniform distribution, whilst with different speed. The second eigenvalues are  $\lambda_2 = 2\epsilon - 1$ ; as expected the chains converge slower when  $|\lambda_2|$  is closer to one.

ergodic—otherwise the chain might never reach the desired stationary distribution. Even if the chain is ergodic, typically we would not know in practice how close the chain is to the stationary distribution.

Here we set aside the issue of ergodicity and concentrate on designing a transition kernel  $K$  for a given target  $\pi$ . This is surprisingly simple via the approach proposed by Metropolis [1], later generalised by Hastings [21], for a rigorous treatment see [17, 23]. Note that this approach does not provide explicit formula for the transition Kernel  $K$  (which is often intractable to compute even in finite state spaces); it merely states a procedure to draw a new sample given the previous one. Suppose we are given a target density  $\pi = \phi/Z$ , where  $Z$  is a (possibly unknown) normalisation constant. The Metropolis-Hastings (MH) algorithm uses a proposal density  $q(x'|x)$ , which generates a candidate sample  $x'$  possibly dependent<sup>4</sup> on the current sample  $x$ . The algorithm is very simple, we define a chain by the

<sup>4</sup>Note that whilst one can consider proposals  $q(x')$  which are independent of  $x$ , the actual MH transition  $K(x'|x)$  nevertheless depends on  $x$ .

following rule: Accept the proposed sample with probability

$$\alpha(x \rightarrow x') = \min \left\{ 1, \frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)} \right\},$$

where  $\alpha$  is the *acceptance probability* satisfying clearly the condition  $0 \leq \alpha(x|x') \leq 1$ . If the sample is not accepted, the chain stays put at the old location  $x$ . This procedure implicitly defines the following MH transition kernel  $K$ :

$$K(x|x') = q(x|x')\alpha(x|x') + \delta(x - x')\rho(x'), \quad (19.6)$$

here,  $\delta(x)$  is the Dirac delta  $\rho$  is the *rejection probability* satisfying  $0 \leq \rho(x') \leq 1$  is given by

$$\rho(x') \equiv \int (1 - \alpha(x|x'))q(x|x')dx.$$

This simply sums up all the probability of rejection. Note that we don't directly sample from  $K$ , we merely sample from the proposal  $q$  but reject some of the samples. In practice we typically don't need to evaluate  $K$  – indeed very often this is intractable. Recall that the stationary distribution should satisfy the following condition

$$\pi(x) = \int K(x|x')\pi(x')dx'.$$

---

**Algorithm 2.** Metropolis-Hastings

---

```

1: Initialise  $x^{(1)}$  arbitrarily
2: for  $n = 2, 3, \dots$  do
3:   Propose a candidate:  $x^{new} \sim q(x^{new}|x^{(n-1)})$ 
4:   Compute acceptance probability:

$$\alpha(x^{new}|x^{(n-1)}) = \min \left\{ 1, \frac{q(x^{(n-1)}|x^{new})\pi(x^{new})}{q(x^{new}|x^{(n-1)})\pi(x^{(n-1)})} \right\}$$

5:   Sample from uniform distribution:  $a \sim \mathcal{U}(0, 1)$ 
6:   if  $a < \alpha$  then
7:     Accept candidate:  $x^{(n)} \leftarrow x^{new}$ 
8:   else
9:     Reject candidate:  $x^{(n)} \leftarrow x^{(n-1)}$ 
10:  end if
11: end for
```

---

For an arbitrary kernel  $K$ , it is hard to verify stationarity. However, if  $K$  happens to satisfy a stronger condition,<sup>5</sup> known as the *detailed balance condition*

$$K(x'|x)\pi(x) = K(x|x')\pi(x'),$$

---

<sup>5</sup>A sufficient condition; a kernel  $K$  may have  $\pi$  as the stationary distribution while not satisfying detailed balance.

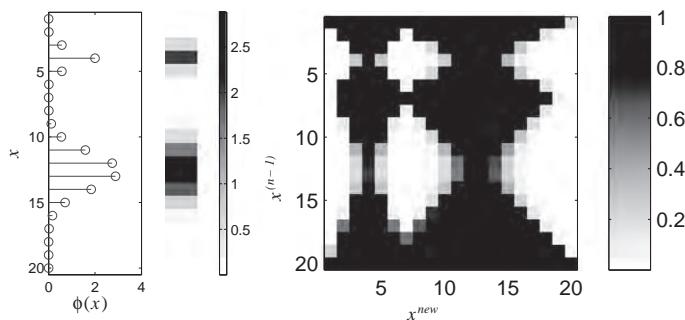
one sees directly that  $\pi$  must be the stationary distribution by integrating both sides over  $x'$ . Verification of detailed balance for the Metropolis-Hastings kernel is straightforward:

$$\begin{aligned}
 K(x|x')\pi(x') &= (q(x|x')\alpha(x|x') + \delta(x - x')\rho(x'))\pi(x') \\
 &= q(x|x')\min\left\{1, \frac{q(x'|x)\pi(x)}{q(x|x')\pi(x')}\right\}\pi(x') + \delta(x - x')\rho(x')\pi(x') \\
 &= \min\{q(x|x')\pi(x'), q(x'|x)\pi(x)\} + \delta(x - x')\rho(x')\pi(x') \\
 &= q(x'|x)\min\left\{\frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)}, 1\right\}\pi(x) + \delta(x' - x)\rho(x)\pi(x) \\
 &= K(x'|x)\pi(x).
 \end{aligned}$$

Note that to compute the acceptance probability  $\alpha$ , we only need to evaluate  $\pi$  up to a normalisation, since the normalisation constant cancels out. For a given target  $\pi$  and assumed proposal  $q(x'|x)$  we now have a procedure for sampling from the Markov chain  $K(x'|x)$  with stationary distribution  $\pi$ . Sampling from the transition (19.6) can be achieved using the procedure detailed in Algorithm 2.

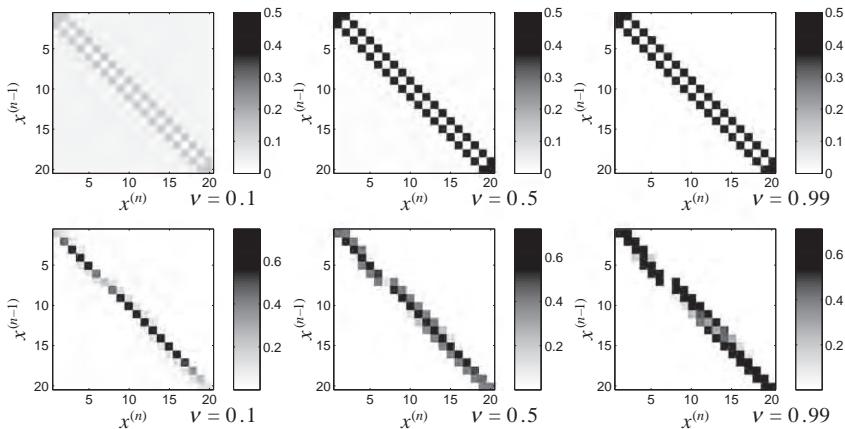
#### 1.19.4.3 Illustration

In this section, we will derive a MH algorithm for sampling from a discrete distribution  $\pi(x) = \phi(x)/Z$  where  $x \in \{1, 2, \dots, 20\}$ . The target, which has two bumps is shown on Figure 19.9. Our proposal  $q(x'|x)$  is a mixture of a random walk with a uniform density: with probability  $\nu$ , we choose a random walk. If we choose the random walk, with equal probability we let  $x' \leftarrow \max\{0, x - 1\}$  or  $x' \leftarrow \min\{20, x + 1\}$ . Alternatively, with probability  $1 - \nu$ , we choose  $x'$  uniformly on  $\{1, 2, \dots, 20\}$ . The random walk component corresponds to a “local exploration” whereas the uniform jump corresponds to a “restart.” Here, the parameter  $\nu$  is a algorithm parameter that will effect the behavior of our overall proposal and we will show how this may effect the convergence rate of the resulting MH algorithm. In Figure 19.10, we show the proposals and corresponding MH transition matrices.



**FIGURE 19.9**

Target density and the acceptance probability  $\alpha(x^{new}|x^{(n-1)})$  under a symmetric proposal  $q(x^{new}|x^{(n-1)}) = q(x^{(n-1)}|x^{new})$ .

**FIGURE 19.10**

(Top, left to right) The proposal density with  $v = 0.1, 0.5, 0.99$ . (Bottom, left to right) The corresponding transition matrices  $K$ , computed by Eq. (19.6).

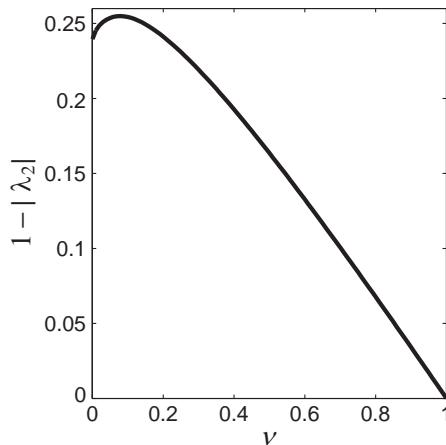
A moments thought would reveal that the local moves are not sufficient to cross the low probability region between the two bumps of the target, so setting  $v$  close to 1 might be a poor choice in terms of the convergence, also known as the *mixing time*. In this case, having the uniform proposal is certainly useful as it may help the chain to investigate the state space, hence reducing the mixing time. However, longer jumps have the risk of hitting low probability regions, increasing the rejections. The optimal choice for  $v$  seems to be somewhere in between those two extremes and we will numerically illustrate this.

In this simple example, we can numerically compute the effective MH transition matrix and compute the *spectral gap*, given as  $1 - |\lambda_2|$ ; the distance between magnitudes of two largest eigenvalues. For all  $v$ , the spectral gap is plotted in Figure 19.11. The bigger this gap, the faster is the convergence: an illustration of this fact is shown in Figure 19.12.

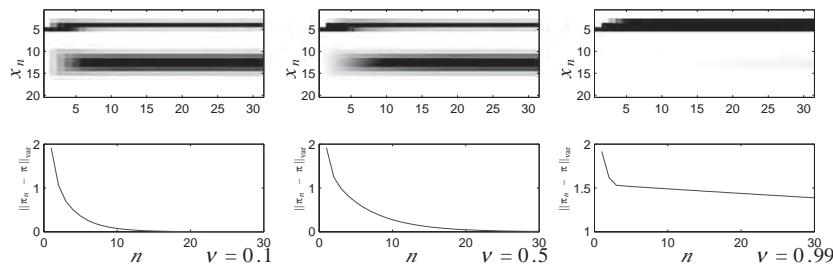
For real problems, the above computations are typically intractable but fortunately they are also not needed at all for the application of the MH algorithm. All we need is to sample from the proposal and evaluate the acceptance probability. In Figure 19.13, we show the results that would reflect a typical MH run. For different parameter settings with  $v = 0.1, 0.5, 0.99$ , we generate sample paths using the algorithm in 2. Then, following a burn in period of 100 iterations, for each state  $x$  we simply count the number of times the chain visits  $x$  to estimate  $\pi(x)$ . Formally, we construct the estimator

$$\bar{\pi}_n(x) = \frac{1}{n - 100} \sum_{\tau=101}^n [x^{(\tau)} = x]. \quad (19.7)$$

Note that, while the samples  $x^{(\tau)}$  are correlated, we calculate our estimate as if they are independent. This is still valid as after convergence all the  $x^{(\tau)}$  are effectively drawn from the same stationary distribution  $\pi(x)$ . The results are shown in Figure 19.13.

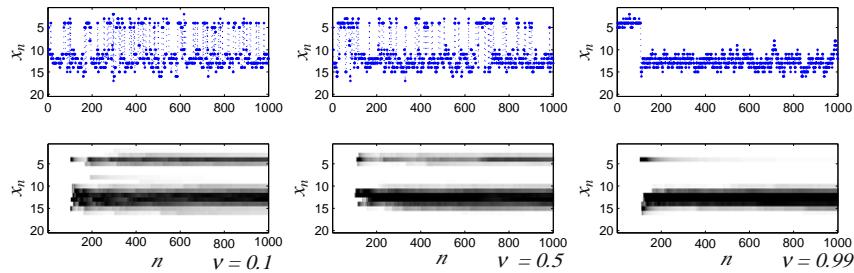
**FIGURE 19.11**

The spectral gap  $(1 - |\lambda_2|)$  computed numerically for the transition matrix  $K$  obtained using the proposals shown in Figure 19.10. The picture suggests that for fastest convergence, we need to set  $\nu \approx 0.1$ . Only using the local moves ( $\nu = 1$  case) results in poor mixing and uniform proposal seems to be a better choice for this problem. Yet, using local moves occasionally improves mixing as the gap at  $\nu = 0.1$  is slightly bigger than  $\nu = 0$ .

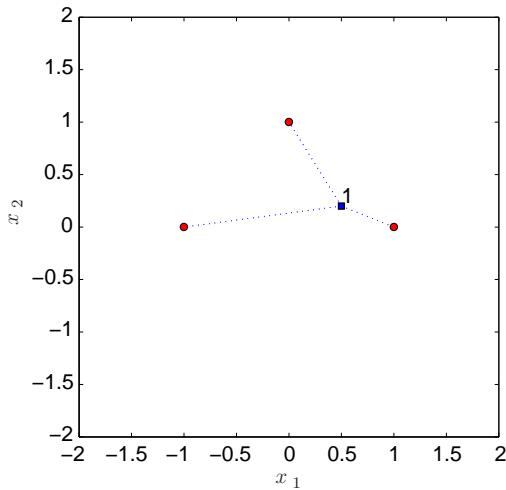
**FIGURE 19.12**

Convergence to the stationarity when using the proposal with parameters  $\nu = 0.1, 0.5, 0.99$ . (Top) The marginals  $\pi_n$ , as computed via  $K^n\pi_0$  where darker color indicates higher probability. (Bottom) The total variation distance between stationary distribution  $\pi = \pi_\infty$  and the marginal  $\pi_n$ . As predicted, the distance drops very slowly for  $\nu = 0.99$  and faster for the other cases. This example suggests that the algorithm is not very sensitive to  $\nu$  and a broad range of values are useful in practice.

**Example 3 (Object position estimation from range measurements).** In this section, we will illustrate the MH algorithm on a synthetic example of object localization. In this example, we have 3 noisy sensors at known positions  $s_j$ ,  $j = 1, \dots, 3$ , each measuring with noise the distance to an object at an unknown

**FIGURE 19.13**

Sample paths and estimates of the marginal when using the proposal with parameters  $v = 0.1, 0.5, 0.99$ . (Top) An example of a sample paths generated by the algorithm. (Bottom) Estimate of the marginal  $\bar{\pi}_\tau(x)$  (see Eq. (19.7)) using averages over a single realization after an initial burn-in period of 100 iterations.

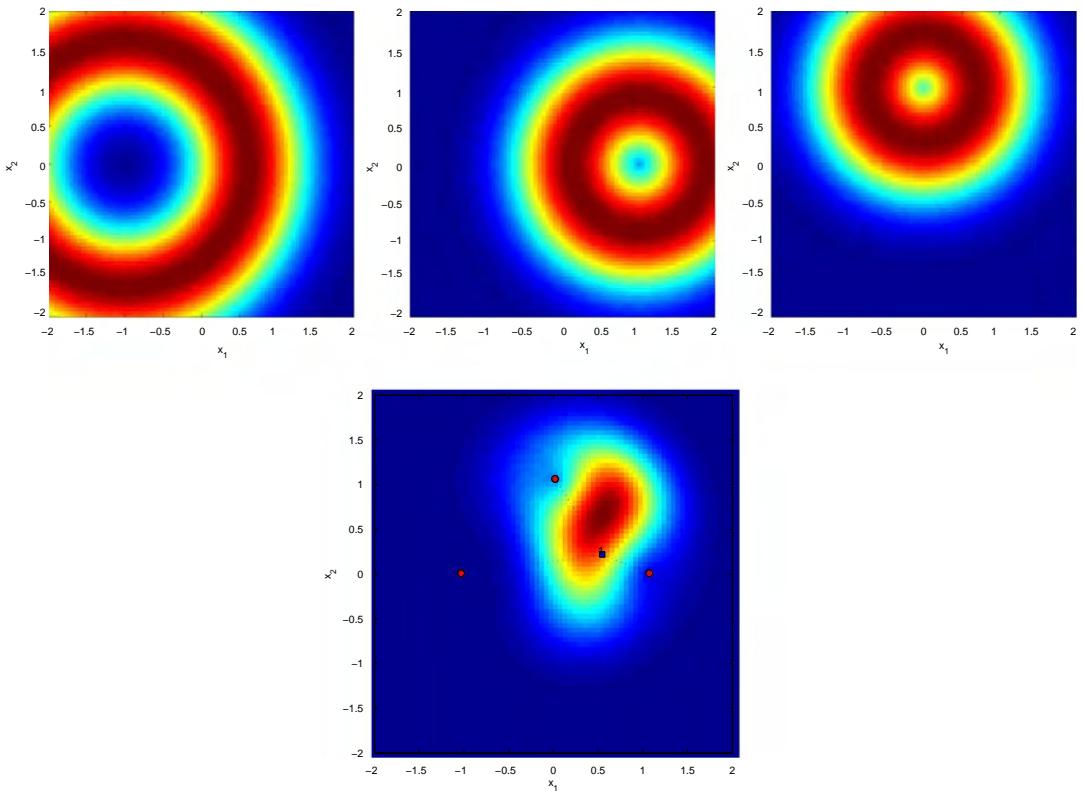
**FIGURE 19.14**

Sensor locations  $s_1, s_2$  and  $s_3$  (round red points) and the true object position (blue square). The observations are noisy versions of the true distances.

location  $x$ . The setup is shown in Figure 19.14. Each sensor observes a noisy distances with the following model

$$y_j|x, s_j \sim \mathcal{N}(y_j; \|x - s_j\|, R),$$

where  $\|x\| \equiv (\sum_k x_k^2)^{1/2}$  denotes the Euclidian distance and  $R$  is the noise variance of each sensors. We assume that we don't have any information about the position of the object so we choose a flat prior

**FIGURE 19.15**

(Top) The likelihood terms  $p(y_i|x)$ , (Bottom) The target posterior, proportional to  $\phi(x) = \prod_i p(y_i|x)$ .

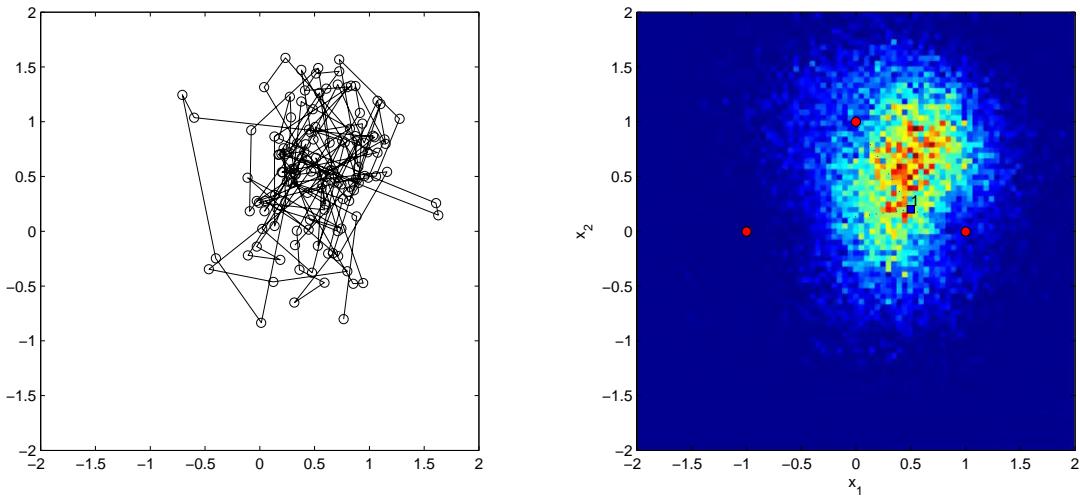
$p(x) \propto 1$ . The solution for this problem is given by the posterior

$$\pi(x) = p(x|y_1, y_2, y_3) = \frac{1}{Z} p(y_1|x)p(y_2|x)p(y_3|x)p(x) \equiv \frac{1}{Z}\phi(x).$$

The true target posterior density is depicted in Figure 19.15. As the target posterior  $\pi(x)$  is nonstandard, we will sample from it using a MH algorithm. We design a MH algorithm with a symmetric random walk proposal where  $q(x'|x) = \mathcal{N}(x, \sigma^2 I)$ . The acceptance probability is

$$\alpha(x \rightarrow x') = \min \left\{ 1, \frac{\phi(x')}{\phi(x)} \right\}.$$

Note that the symmetric proposal has been canceled out. By construction, the resulting kernel will admit  $\pi$  as the unique stationary density. In Figure 19.16, we show a sequence of samples obtained from this MH algorithm. We see that the chain visits states around the high probability region. The consecutive

**FIGURE 19.16**

(Left) A state trajectory obtained from the MH algorithm. (Right) A 2-D histogram estimate of the target density (from a longer state trajectory).

samples are dependent (as can be seen from the connecting lines) but as expected, a 2-D histogram obtained from a single chain shows a fairly accurate approximation.

#### 1.19.4.4 The Gibbs sampler

The MH schema is a very general technique for deriving MCMC algorithms. This generality stems partially from the arbitrariness of the proposal  $q$ ; in complex models the design of a reasonable proposal such that the chain mixes well can require a lot of work. It would be desirable if somehow the proposal design phase could be automated. The Gibbs sampler [20,24] is an attempt in this direction.

The Gibbs sampler is suitable for sampling a multivariate random variable  $x = (x_1, \dots, x_D)$  with intractable joint target density  $\pi(x)$ . Gibbs sampling proceeds by partitioning the set of variables  $x$  into a chosen variables  $x_d$  and the rest,  $x = (x_d, x_{-d})$ . The assumption is that the *full conditional densities*  $\pi(x_d|x_{-d})$  are tractable. One then proceeds coordinatewise by sampling from full conditionals as in Algorithm 3. The Gibbs sampler is actually a MH schema with a sequence of proposals for  $d = 1, \dots, D$

$$q_d(x|x') = p(x_d|x'_{-d})\delta(x_{-d} - x'_{-d})$$

each corresponding to a Gibbs transition kernels  $K_d$ . Moreover, it is easy to check that using this proposal results in a MH acceptance probability of 1, so that every move is accepted. This guarantees that each kernel  $K_d$  admits the same stationary distribution  $\pi$  as a stationary distribution, so we have  $\pi = K_d\pi$  for all  $d$ . Now, we see that  $\pi = K_1\pi$  and  $\pi = K_2\pi$  implies

$$\pi = K_1K_2\pi$$

---

**Algorithm 3.** Gibbs sampler

---

```

1: Initialize  $x^{(1)} = (x_1, \dots, x_D)^{(1)}$  arbitrarily
2: for  $n = 2, 3, \dots$  do
3:    $x_1^{(n)} \sim p(x_1|x_2^{(n-1)}, x_3^{(n-1)}, \dots, x_D^{(n-1)})$ 
4:    $x_2^{(n)} \sim p(x_2|x_1^{(n)}, x_3^{(n-1)}, \dots, x_D^{(n-1)})$ 
   :
5:    $x_D^{(n)} \sim p(x_D|x_1^{(n)}, x_2^{(n)}, \dots, x_{D-1}^{(n)})$ 
6: end for

```

---

and by induction

$$\pi = (K_1 K_2 \dots K_D) \pi \equiv K \pi.$$

Here, the transition kernel  $K$  is the effective Gibbs kernel after we sweep over all the variables  $x_d$  once. Unless the full conditionals are degenerate, the effective kernel  $K$  will be aperiodic and irreducible while none of the individual kernels  $K_d$  are. Note that we could visit the variables in any deterministic or order, provided each  $x_d$  is visited infinitely often in the limit. If we choose a deterministic order, the resulting sampler is known as a *deterministic scan Gibbs sampler*; if we choose a random order, not surprisingly, the sampler is called a *random scan Gibbs sampler*.

The ease of use of the Gibbs sampler has resulted the method being employed in many applications. There are very effective programs that generate a Gibbs sampler automatically from a model specification, such as BUGS. In the sequel, we will illustrate the Gibbs sampler on a change-point model for count data [25].

**Example 4 (Gibbs sampling for a change-point model).** In this model, at each time  $t$  we observe the count of an event  $y_t$ . All the counts up to an unknown time  $\tau$  come from the same distribution after which the distribution changes. We assume that the change-point  $\tau$  is uniformly distributed over  $1, \dots, T$ . The two different distributional regimes up to and after  $\tau$  are indicated by the random variables  $\lambda_i$ ,  $i = 1, 2$ , which are assumed to follow a Gamma distribution.

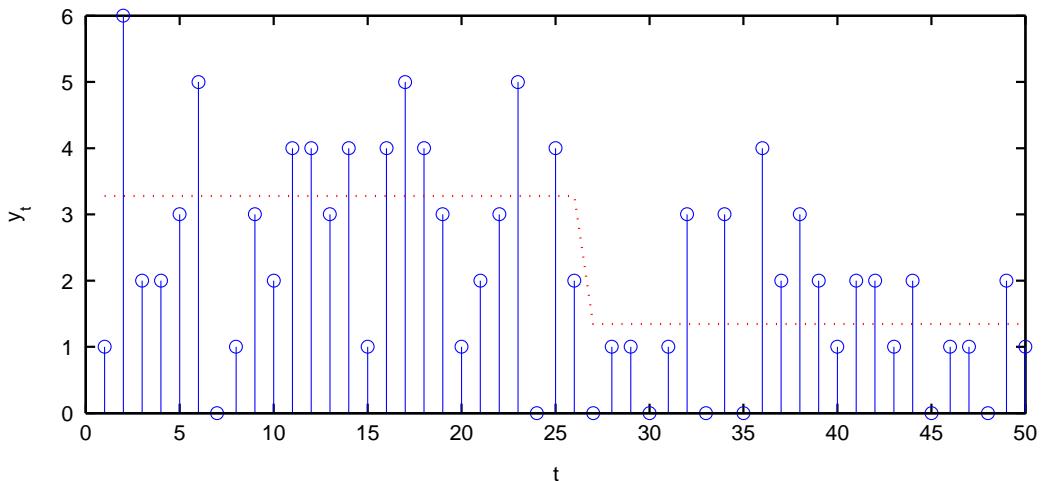
$$\mathcal{G}(\lambda_i; a, b) = \frac{1}{\Gamma(a)} b^a \lambda_i^{a-1} e^{-b\lambda_i} = e^{(a-1)\log \lambda_i - b\lambda_i - \log \Gamma(a) + a \log b}.$$

Under regime  $\lambda_i$ , the counts are assumed to be identically Poisson distributed

$$\mathcal{PO}(y_t; \lambda_i) = \frac{\lambda_i^{y_t}}{y_t!} e^{-\lambda_i} = e^{y_t \log \lambda_i - \lambda_i - \log(y_t!)}$$

This leads to the following generative model:

$$\begin{aligned} p(\tau) &= 1/T \text{ uniform,} \\ \lambda_i &\sim \mathcal{G}(\lambda_i; a, b), \quad i = 1, 2, \\ y_t &\sim \begin{cases} \mathcal{PO}(y_t; \lambda_1) & 1 \leq t \leq \tau, \\ \mathcal{PO}(y_t; \lambda_2) & \tau < t \leq T. \end{cases} \end{aligned}$$

**FIGURE 19.17**

A typical realisation from the change-point model. True intensities are shown with a dotted line. The intensity has dropped from  $\lambda_1 = 3.2$  to  $\lambda_2 = 1.2$  at  $\tau = 26$ . The time index is indicated by  $t$  and the number of counts by  $y_t$ . Generating such synthetic datasets provide intuition about the underlying models qualitative behaviour.

A typical draw from this model is shown in Figure 19.17. The inferential goal is to compute the posterior distribution of the change-point location and the intensities given the count data,  $p(\lambda_1, \lambda_2, \tau | y_{1:T})$ . In this problem the posterior is actually tractable [26] and could serve to assess the quality of the Gibbs sampling approximation.

To implement Gibbs sampling we need to compute the distribution of each variable, conditioned on the rest. These conditionals can be derived by writing the log of the full joint distribution and collecting terms that depend only on the free variable.<sup>6</sup> To derive the full conditional densities  $p(\lambda_1 | \cdot)$ ,  $p(\lambda_2 | \cdot)$  and  $p(\tau | \cdot)$ , we proceed by writing the full joint density. The log of the full joint density is given by:<sup>7</sup>

$$p(y_{1:T}, \lambda_1, \lambda_2, \tau) = \left( \prod_{t=1}^{\tau} p(y_t | \lambda_1) \right) \left( \prod_{t=\tau+1}^T p(y_t | \lambda_2) \right) p(\lambda_1) p(\lambda_2) p(\tau),$$

$$\log p(y_{1:T}, \lambda_1, \lambda_2, \tau) = \sum_{t=1}^{\tau} (+ y_t \log \lambda_1 - \lambda_1 - \log(y_t!))$$

$$+ \sum_{t=\tau+1}^T (+ y_t \log \lambda_2 - \lambda_2 - \log(y_t!))$$

<sup>6</sup>since  $p(X_d | x_{-d} = x_{-d}) = p(X_d, x_{-d} = x_{-d}) / p(x_{-d} = x_{-d}) \propto p(X_d, x_{-d} = x_{-d})$ .

<sup>7</sup>Here,  $f(x) =^+ g(x)$  means  $f(x) = g(x) + C$  where  $C$  is an irrelevant constant. This implies  $\exp(f(x)) \propto \exp(g(x))$ .

$$\begin{aligned}
& + (a - 1) \log \lambda_1 - b \lambda_1 - \log \Gamma(a) + a \log b \\
& + (a - 1) \log \lambda_2 - b \lambda_2 - \log \Gamma(a) + a \log b - \log T,
\end{aligned}$$

and collect terms that depend only on the free variable.

$$\log p(\lambda_1 | \tau, \lambda_2, y_{1:T}) = {}^+ \left( a + \sum_{t=1}^{\tau} y_t - 1 \right) \log \lambda_1 - (\tau + b) \lambda_1 = {}^+ \log \mathcal{G}(a + \sum_{t=1}^{\tau} y_t, \tau + b).$$

Calculation of  $p(\lambda_2 | \cdot)$  is similar.

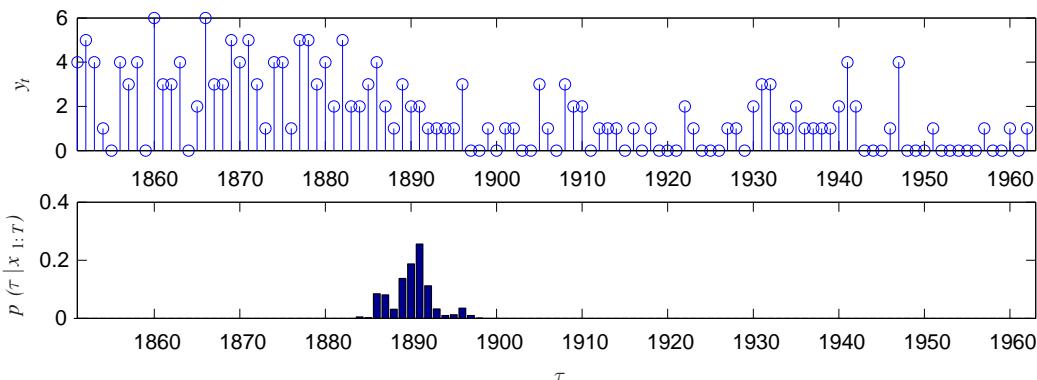
$$\begin{aligned}
\log p(\tau | \lambda_1, \lambda_2, y_{1:T}) &= \sum_{t=1}^{\tau} (+y_t \log \lambda_1 - \lambda_1 - \log(y_t!)) \\
&\quad + \sum_{t=\tau+1}^M (+y_t \log \lambda_2 - \lambda_2 - \log(y_t!)) \\
&= + \left( \sum_{t=1}^{\tau} y_t \right) \log \lambda_1 - \tau \lambda_1 + \left( \sum_{t=\tau+1}^M y_t \right) \log \lambda_2 - (M - \tau) \lambda_2.
\end{aligned}$$

We have to evaluate the above expression for  $\tau = 1, \dots, M$ . This Gibbs procedure is summarized in Algorithm 4.

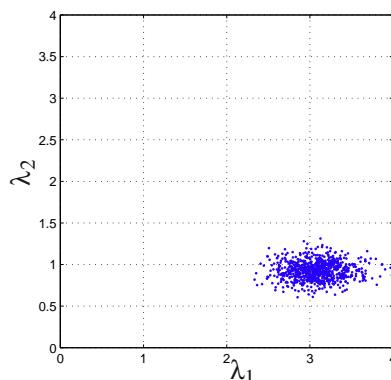
**Algorithm 4.** A Gibbs sampler for the change-point model

- 
- 1: Initialize  $\lambda_2^1, \tau^1$
  - 2: **for**  $n = 2, 3, \dots$  **do**
  - 3:    $\lambda_1^{(n)} \sim p(\lambda_1 | \lambda_2^{(n-1)}, \tau^{(n-1)}, y_{1:T}) = \mathcal{G}(a + \sum_{t=1}^{\tau} y_t, \tau + b)$
  - 4:    $\lambda_2^{(n)} \sim p(\lambda_2 | \lambda_1^{(n)}, \tau^{(n-1)}, y_{1:T}) = \mathcal{G}(a + \sum_{t=\tau+1}^M y_t, M - \tau + b)$
  - 5:    $\tau^{(n)} \sim p(\tau | \lambda_1^{(n)}, \lambda_2^{(n)}, y_{1:T})$
  - 6: **end for**
- 

We illustrate the algorithm on a coal mining disaster dataset [27], see Figure 19.18. The data consists of the number of deadly coal-mining disasters in England per year over a time span of 112 years from 1851 to 1962. It is widely agreed in the statistical literature that a change in the intensity (the expected value of the number of disasters) occurs around the year 1890, after new health and safety regulations were introduced. In Figure 19.18 we show the samples obtained from the posterior of the changepoint location. The posterior of the intensities  $\lambda_1, \lambda_2$  are shown in Figure 19.19.

**FIGURE 19.18**

Estimation of change points on the coal mining disaster dataset. (Top) The number of deadly disasters  $y_t$  each year  $t$ . (Bottom) The posterior distribution of the changepoint location  $p(\tau | y_{1:T})$ .

**FIGURE 19.19**

Samples drawn from the posterior intensities  $p(\lambda_1, \lambda_2 | y_{1:T})$  on the coal mining disaster dataset. This results clearly indicate that after the health and safety regulations, the intensity has dropped from about  $\lambda_1 = 3$  to  $\lambda_2 = 1$ .

## 1.19.5 Sequential Monte Carlo

The MCMC techniques described in the previous section are widely used in many areas of applied sciences and engineering. However, they may have some potential drawbacks in signal processing: MCMC methods are by construction batch algorithms that require availability of all data records. This can be prohibitive in some signal processing application such as tracking when the data arrives

sequentially and decisions have to be taken in real time. One could in principle design online MCMC algorithms that operate on the full data record observed so far. However, such batch algorithms take increasingly more time. In such cases, it is desirable to have alternative methods which process data sequentially and take a constant time per observation.

Sequential Monte Carlo (SMC) is an umbrella term for Monte Carlo techniques that process data in an online fashion [9]. Popular techniques such as particle filtering or sequential importance sampling (SIS) are special SMC algorithms. SMC techniques are particularly natural for probabilistic signal models where data has a natural order.<sup>8</sup> A very large class of signal models can be described as *state space models*, that have the following general form:

$$x_0 \sim p(x_0), \quad (19.8)$$

$$x_t | x_{t-1} \sim p(x_t | x_{t-1}), \quad (19.8)$$

$$y_t | x_t \sim p(y_t | x_t). \quad (19.9)$$

Here,  $x_t$  is the state and  $y_t$  are the observations and  $t = 0, \dots, T$  is a discrete time index. Note that  $t$  does not have to correspond to the wall clock, it only specifies some natural ordering of the underlying data. The Eq. (19.8) and Eq. (19.9) are known as *transition* and *observation* models. State space models, also named as hidden Markov models, are ubiquitous in signal processing and many popular models, such as linear dynamical systems or autoregressive (AR) models, can be formulated as special cases.

There are several inference problems of interest for state space models, where inference in this context is the task of using a distribution to answer questions of interest. A common inference problem is the *prediction* of an unseen future variable  $y_{T+1}$ , which requires computing  $p(y_{T+1} | y_{1:T})$  from a joint distribution. Another one is the so called *filtering* problem, estimation of the state  $x_t$  given observations so far  $y_{1:t}$ . This is achieved via calculation of the posterior quantity  $p(x_t | y_{1:t})$ . One of the challenges in statistical signal processing is to develop efficient inference routines for answering such questions.

In this context, SMC techniques [10, 28] have proved very useful. SMC methods are based on importance sampling/resampling which we review in the following sections. We first review importance sampling and resampling techniques and subsequently introduce the SMC algorithm as a sequential importance sampler.

### 1.19.5.1 Importance sampling (IS)

Consider a target distribution  $\pi(x) = \phi(x)/Z$  where the non-negative function  $\phi(x)$  is known, but the overall normalisation constant  $Z$  is assumed to be computationally intractable. The main idea of (IS) is to estimate expectations  $E_\pi(\varphi(x))$  by using weighted samples from a tractable IS distribution  $q(x)$ . Note that unlike other MC methods, our goal is not directly generating samples but estimating expectations via weighted samples. However, we can also generate unweighted independent samples, if we desire to do so by a mechanism called *resampling*.

More specifically, we can write the normalization constant as

$$Z = \int \phi(x) dx = \int \frac{\phi(x)}{q(x)} q(x) dx = \int W(x) q(x) dx = E_q(W(x)),$$

---

<sup>8</sup>SMC techniques are not necessarily limited to such systems, though, see, e.g., [20].

where  $W(x) \equiv \phi(x)/q(x)$  is called *weight function*. Therefore we have

$$\mathbb{E}_\pi(\varphi(x)) = \frac{1}{Z} \int \varphi(x) \frac{\phi(x)}{q(x)} q(x) dx = \frac{\mathbb{E}_q(\varphi(x) W(x))}{\mathbb{E}_q(W(x))}.$$

A Monte Carlo estimate of  $\mathbb{E}_\pi \varphi(x)$  is then given by

$$\hat{\mu}_N = \frac{\sum_{i=1}^N W^{(i)} \varphi(x^{(i)}) / N}{\sum_{i=1}^N W^{(i)} / N},$$

where  $W^{(i)} \equiv W(x^{(i)})$  and the ‘‘particles’’  $x^{(1)}, \dots, x^{(N)}$  are sampled from  $q(x)$ . Using normalised weights  $w^{(i)} \equiv W^{(i)} / \sum_{i'=1}^N W^{(i')}$  we can write the approximation as

$$\hat{\mu}_N = \sum_{i=1}^N w^{(i)} \varphi(x^{(i)}).$$

In other words, instead of sampling from the target density  $p(x)$ , we sample from a different tractable distribution  $q(x)$  and correct by reweighing the samples accordingly. An example is given in Figure 19.20.

#### 1.19.5.1.1 Resampling

A practical issue is that, unless the IS sampling density  $q(x)$  is close to the target density  $\pi(x)$ , the normalised weights will typically have mass in only a single component. This can be partially addressed using re-sampling as we now describe. Given a weighted particle system  $\sum_{i=1}^N w^{(i)} \delta(x - x^{(i)})$ , resampling is the term for a set of methods for generating *randomly* an unweighted particle system of the form  $\frac{1}{M} \sum_{j=1}^M \delta(x - \tilde{x}^{(j)})$  such that

$$\mathbb{E} \left( \frac{1}{M} \sum_{j=1}^M \delta(x - \tilde{x}^{(j)}) \right) = \sum_{i=1}^N w^{(i)} \delta(x - x^{(i)}),$$

where the expectation is over the draws from the resampling algorithm. In resampling, typically the total number of particles are unchanged, so we have  $N = M$ . Alternatively, one can view resampling as a randomised pruning algorithm where we discard particles with low weight and increase the number of particles with high weight. Unlike a deterministic pruning algorithm, the random nature of resampling does not introduce a systematic bias and ensures an asymptotically consistent algorithm when the number of samples  $N$  goes to infinity. In Figure 19.21, we illustrate the result of resampling from a weighted particle set. It is instructive to compare histograms obtained from weighted particle set (Figure 19.20); the resampling stage introduces additional Monte Carlo variation but the histogram is still a reasonable approximation to the underlying density. Two popular methods for resampling are multinomial resampling and systematic resampling, which we will review in the sequel. For a discussion and comparison of various resampling schemata, see [10, 29].

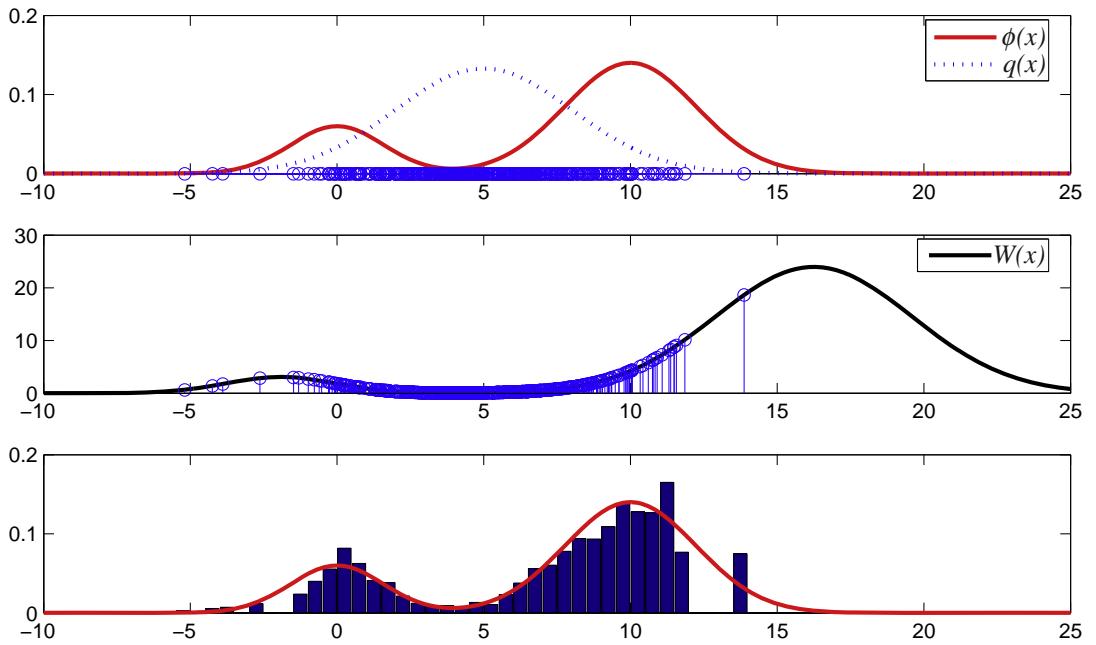
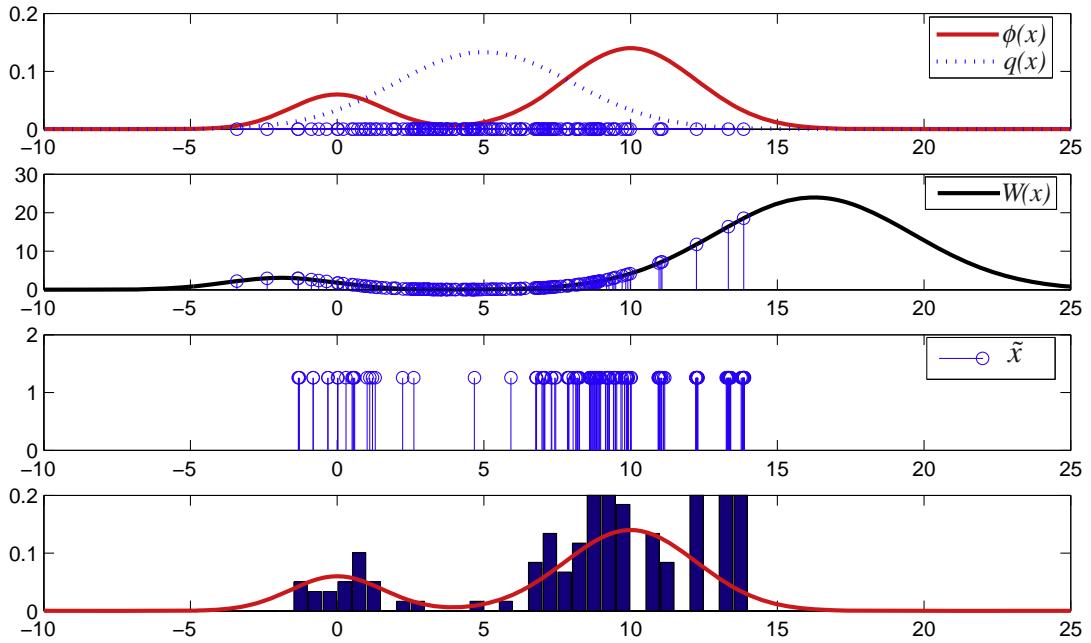
**FIGURE 19.20**

Illustration of Importance Sampling. (Top): The solid curve denotes the unnormalised target density  $\phi(x)$  and the dashed curve the tractable IS density  $q(x)$ . We assume that it is straightforward to generate samples  $x^{(i)}$  from  $q(x)$ ; these are plotted on the  $x$  axis. Middle: To account for the fact that the samples are from  $q$  and not from the target  $\pi = \phi/Z$ , we need to reweight the samples. In this example, the IS density  $q$  generated far too many samples where  $\pi$  has low mass, and too few where  $\pi$  has high mass. The samples in these regions are reweighted according to the weight function  $W(x)$ . Bottom: Binning the weighted samples from  $q$ , we obtain an approximation to  $\pi$  such that averages with respect to this approximation will be close to averages with respect to  $\pi$ .

Multinomial resampling equivalent to the inversion method when the target is a discrete distribution, as explained in Figure 19.3. Here, we view the weighted sample set  $x^{(i)}$  as a discrete distribution with categories  $i = 1, \dots, N$  where the probability of the  $i$ th category is given by the normalized weight  $w^{(i)}$ . To sample  $M$  times independently from this target, we generate  $u^{(j)} \sim \mathcal{U}(0, 1)$  for  $j = 1, \dots, M$  and we obtain an unweighted set of particles by evaluating the generalized inverse at each  $u^{(j)}$ , as explained in Figure 19.3.

Systematic resampling is quite similar to multinomial resampling, only the generation of  $u^{(j)}$  is not entirely random but “systematic.” To generate  $M$  samples we only select  $u^{(1)}$  uniformly random from the interval  $[0, 1/M]$  and set  $u^{(j)} = u^{(1)} + (j - 1)/M$ . The  $u$  are located on a uniform grid with a random initial shift. In Figure 19.22, we illustrate both methods.

**FIGURE 19.21**

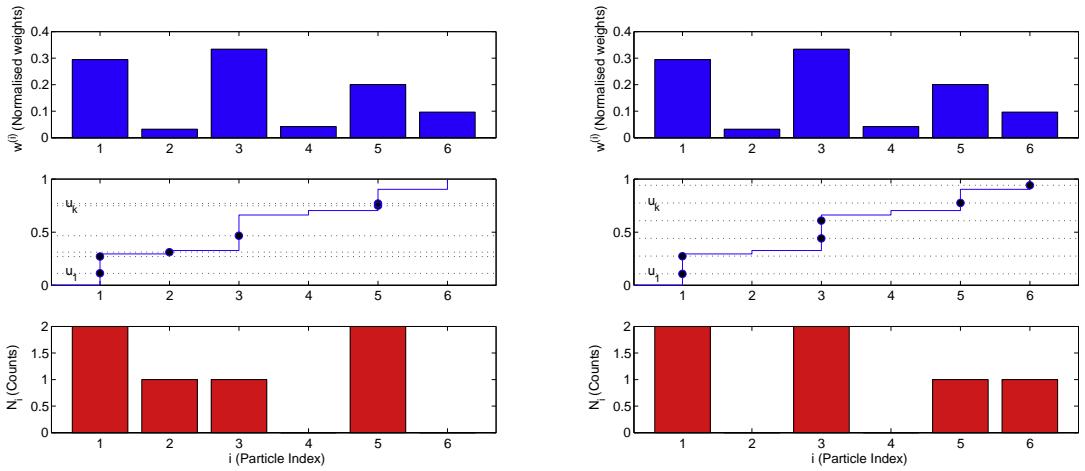
Importance Sampling with Resampling. We can generate from the weighted samples  $x$  generated by importance sampling unweighted samples  $\tilde{x}$  by resampling. Some of the particles in the original set will be replicated while others will be simply discarded.

From only a theoretical perspective, the actual details of a resampling schema are not important and the estimates converge to their true values in the infinite particle limit. One has to only ensure that the expectation under the resampled particle set is *on average* equal to the expectation under the original weighted particle set. However, with a finite number of particles, there may be notable differences.

It can be shown that systematic resampling achieves a lower variance while both methods have the same expectation and no bias. Coupled with its ease of implementation, this makes systematic resampling a popular choice in practice.

### 1.19.5.2 Sequential importance sampling

We now apply importance sampling to the state space model introduced earlier in Eq. (19.8) and Eq. (19.9). “Plain” importance sampling is actually not very effective in high dimensional problems but when executed sequentially using resampling techniques, it has become a very efficient and effective tool for inference. The resulting sequential IS methods are also known as particle filters [10]. The goal

**FIGURE 19.22**

Multinomial Resampling (left column) versus Systematic Resampling (right column). (Top) True weights, (Middle) Cummulative weight function, (Bottom) Bin counts after resampling. Note that the resulting histograms can be thought of as approximations of the original discrete distribution.

is to draw samples from the posterior

$$p(x_{1:t}|y_{1:t}) = \underbrace{p(y_{1:t}|x_{1:t})p(x_{1:t})}_{\phi(x_{1:t})} / \underbrace{p(y_{1:t})}_{Z_t},$$

where we assume that the normalisation term  $Z_t$  is intractable. An importance sampling approach uses at each time  $t$  an importance distribution  $q_t(x_{1:t})$ , from which we draw samples  $x_{1:t}^{(i)}$  with corresponding importance weights

$$W_t^{(i)} = \phi(x_{1:t}^{(i)}) / q_t(x_{1:t}^{(i)}).$$

The key idea in SMC is the sequential construction of the IS distribution  $q$  and the recursive calculation of the importance weights. Without loss of generality, we may write

$$q_t(x_{1:t}) = q_t(x_t|x_{1:t-1})q_t(x_{1:t-1}).$$

In particle filtering, one chooses a IS proposal  $q$  that only updates the current  $x_t$  and leaves previous samples unaffected. This is achieved using

$$q_t(x_{1:t}) = q_t(x_t|x_{1:t-1})q_{t-1}(x_{1:t-1}).$$

As we are free to chose the IS proposal  $q$  fairly arbitrarily, we can also “construct” the proposal on the fly conditioned on the observations seen so far:

$$q_t(x_{1:t}|y_{1:t}) = q_t(x_t|x_{1:t-1}, y_{1:t})q_{t-1}(x_{1:t-1}|y_{1:t-1}).$$

In the sequel, we won't include  $y_{1:t}$  in our notation but it should be understood that the proposal  $q$  can be constructed at the observations so far.

Due to the sequential nature of the state space model and  $q$ , the weight function  $W_t(x_{1:t})$  admits a recursive computation

$$\begin{aligned} W_t(x_{1:t}) &= \frac{\phi(x_{1:t})}{q_t(x_{1:t})} = \frac{p(y_t|x_t)p(x_t|x_{t-1})\prod_{\tau=1}^{t-1} p(y_\tau|x_\tau)p(x_\tau|x_{\tau-1})}{q_t(x_t|x_{1:t-1})\prod_{\tau=1}^{t-1} q_\tau(x_\tau|x_{1:\tau-1})} \\ &= \underbrace{\frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{1:t-1})}}_{v_t} W_{t-1}(x_{1:t-1}), \end{aligned}$$

where  $v_t$  is called the incremental weight. Particle filtering algorithms differ in their choices for  $q_t(x_t|x_{1:t-1})$ . The optimal choice (in terms of reducing the variance of weights) is the one step filtering distribution [30]

$$q_t(x_t|x_{1:t-1}) = p(x_t|x_{t-1}, y_t).$$

However, sampling from this filtering density is often difficult in practice, and simpler methods are required. The popular *bootstrap* filter uses the model transition density as the proposal

$$q_t(x_t|x_{1:t-1}) = p(x_t|x_{t-1}),$$

for which the incremental weight is  $v_t = p(y_t|x_t)$ . For the bootstrap filter, the IS distribution does not make any use of the recent observation and therefore has the tendency to lose track of the high-mass regions of the posterior. Indeed, it can be shown that the variance of the importance weights for the bootstrap filter increases in an unbounded fashion [20, 30] so that the state estimates are not reliable. In practice, therefore, after a few time steps the particle set typically loses track of the exact posterior mode. A crucial extra step to make the algorithm work is resampling which prunes branches with low weights and keeps the particle set located in high probability regions. It can be shown that although the particles become dependent due to resampling, the estimations are still consistent and converge to the true values as the number of particles increases to infinity. A generic particle filter algorithm is shown in Algorithm 5.

**Example 5 (Particle Filtering).** We will consider the following model of a sequence of Poisson random variables  $y_t$  with intensities  $\exp(l_t)$ :

$$y_t|l_t \sim p(y_t|l_t) = \mathcal{PO}(y_t; \exp(l_t)) = \exp(y_t l_t - \exp(l_t) - \log(y_t!)). \quad (19.10)$$

The latent log intensities  $l_t$  are assumed to be changing slowly with  $t$ ; this can be modeled by a random drift

$$l_t|l_{t-1} \sim p(l_t|l_{t-1}) = \mathcal{N}(l_t; l_{t-1}, R) = \exp\left(-\frac{1}{2}\frac{(l_t - l_{t-1})^2}{R} - \frac{1}{2}\log 2\pi R\right). \quad (19.11)$$

This model can be considered as an alternative to the single change point model introduced in Section 4. Rather than allowing for a single abrupt change, this model proposes slowly varying intensities where the drift is controlled by the transition variance parameter  $R$ . If  $R$  is small, consecutive log intensities  $l_t$

**Algorithm 5.** Particle Filter

---

**for**  $i = 1, \dots, N$  **do**

 Compute the IS distribution:  $q_t(x_t|x_{1:t-1}^{(i)})$  possibly using the latest observation  $y_t$ 

 Generate offsprings:  $\hat{x}_t^{(i)} \sim q_t(x_t|x_{1:t-1}^{(i)})$ 

Evaluate importance weights:

$$v_t^{(i)} = \frac{p(y_t|\hat{x}_t^{(i)})p(\hat{x}_t^{(i)}|x_{t-1}^{(i)})}{q_t(\hat{x}_t^{(i)}|x_{1:t-1}^{(i)})}, \quad W_t^{(i)} = v_t^{(i)} W_{t-1}^{(i)}$$

**end for**

$$\text{ESS} = 1 / \sum_i (\tilde{w}_{1:t}^{(i)})^2$$

**if** ESS > Threshold **then**
*No need to Resample*

 Extend particles:  $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, \hat{x}_t^{(i)}), i = 1, \dots, N$ 
**else**
*Resample*

Normalise importance weights:

$$\tilde{Z}_t \leftarrow \sum_j W_t^{(j)}, \quad \tilde{\mathbf{w}}_t \leftarrow (W_t^{(1)}, \dots, W_t^{(N)}) / \tilde{Z}_t$$

Generate Associations via a Resampling method

$$(a(1), \dots, a(N)) \leftarrow \text{Resample}(\tilde{\mathbf{w}}_t)$$

Discard or Keep particles and Reset weights:

$$x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{a(i)}, \hat{x}_t^{a(i)}), \quad W_t^{(i)} \leftarrow \tilde{Z}_t / N, \quad i = 1, \dots, N$$

**end if**


---

and  $l_{t+1}$  will be close to each other. In the limit of  $R$  going to zero, the model degenerates into a constant intensity model. In this example, we will assume that we know  $R$ .

To design a SMC algorithm, we need to decide upon a proposal mechanism and derive the expression for the incremental importance weight

$$v_t(l_t|l_{1:t-1}) = \frac{p(y_t|l_t)p(l_t|l_{t-1})}{q_t(l_t|l_{1:t-1})}.$$

For illustration purposes, we will consider here two different proposal mechanisms:

1. *Transition density as the proposal* (the Bootstrap filter):

Here, we choose as our proposal the models transition density

$$q_t(l_t|l_{1:t-1}) = p(l_t|l_{t-1}).$$

This choice is the most widely used one for constructing proposals because of its simplicity, ease of implementation and the intuitive interpretation as a “the survival of the fittest” algorithm. The resulting algorithm is known in the SMC literature as the *bootstrap filter* and in computer

vision as the *condensation algorithm*. For the bootstrap filter, the incremental weight expression becomes

$$\begin{aligned} v_t(l_t | l_{1:t-1}) &= \frac{p(y_t | l_t) p(l_t | l_{t-1})}{p(l_t | l_{t-1})} = p(y_t | l_t) \\ &= \exp(y_t l_t - \exp(l_t) - \log(y_t!)), \end{aligned}$$

## 2. The observation likelihood as the proposal

Here, we choose a density that is proportional to the likelihood term

$$q_t(l_t | l_{1:t-1}) = q_t(l_t) = p(y_t | l_t) / c_t,$$

where  $c_t = \int p(y_t | l_t) dl_t$ . Note that this proposal can only be used when the normalizer  $c_t$  is not infinite hence a density exists.<sup>9</sup> This option is perhaps less intuitive to understand but could be a lot more efficient than the bootstrap in problems where the transition model is very diffuse but the observations are very informative. Using a bootstrap filter may be inefficient in this case, as most of the proposed particles will fall into low probability regions.

$$v_t(l_t | l_{1:t-1}) = \frac{p(y_t | l_t) p(l_t | l_{t-1})}{p(y_t | l_t) / c_t} = c_t p(l_t | l_{t-1}).$$

In practice, unless we need to explicitly evaluate the marginal likelihood  $p(y_{1:t})$ , we do not need to evaluate normalizer  $c_t$  as it will cancel out when calculating the normalized weights.

To find this proposal density, we make the following observation: the Poisson density, when viewed as a function of the intensity is proportional to a Gamma density<sup>10</sup>

$$\begin{aligned} \mathcal{PO}(y; \lambda) &\propto \exp(y \log \lambda - \lambda) \\ &\propto \exp((y + 1 - 1) \log \lambda - \lambda - \log \Gamma(y + 1)) = \mathcal{G}(\lambda; y + 1, 1). \end{aligned}$$

So for a given Poisson observation, we can easily generate  $\lambda$  by a unit gamma random number generator with shape parameter  $y + 1$ . To generate the log intensities  $l$ , we sample  $\lambda$  and set  $l = \log(\lambda)$ . The density of  $l = \log \lambda$  when  $\lambda$  is gamma distributed is found via the transformation method

$$\begin{aligned} l(\lambda) &= \log(\lambda), \\ \lambda(l) &= \exp(l), \\ |J| &= \partial \lambda(l) / \partial l = \exp(l), \\ p(l) &= \mathcal{G}(\lambda(l); a, b) \exp(l). \end{aligned}$$

So the density of our proposal is

$$q_t(l_t) \propto \exp((y_t + 1)l_t - \exp(l_t) - \log \Gamma(y_t + 1)).$$

---

<sup>9</sup>Here,  $c_t$  is not to be confused with the actual normalizing constant of the model which is the marginal likelihood  $Z_t = p(y_t | y_{1:t-1})$ .

<sup>10</sup>This is due to the so called conjugacy property.

The weight expression is

$$\begin{aligned} v_t(l_t | l_{1:t-1}) &= \frac{\exp(y_t l_t - \exp(l_t) - \log \Gamma(y_t)) \exp\left(-\frac{1}{2} \frac{(l_t - l_{t-1})^2}{R}\right)}{\exp((y_t + 1) l_t - \exp(l_t) - \log \Gamma(y_t + 1))} \\ &\propto \exp\left(-\frac{1}{2} \frac{(l_t - l_{t-1})^2}{R} - l_t\right). \end{aligned}$$

Here, we have simply ignored the terms that do not depend on  $l_t$  as these do not contribute to the normalized importance weight.

As mentioned earlier, in practice the transition model is taken often as the proposal density. It should be stressed that, the choice of an efficient proposal has a profound effect on the performance and the transition model need not to be always the best choice. As the above example illustrates, there are typically always alternative proposals and in the context of an application, at least a few alternatives should be considered before opting to the transition density. As a guideline, one should look for a proposal that is close to the filtering density but that does not require extensive computation to sample. If a proposal is too costly to sample from, one may consider using a simpler but poorer proposal with more samples for a given computational cost.

## 1.19.6 Advanced Monte Carlo methods

We have reviewed some of the basic Monte Carlo techniques in previous sections. However, there are many problems that need techniques beyond standard strategies. In this section, we will review such a technique: reversible jump MCMC (RJMCMC) [11, 31]. This technique is an extension of the Metropolis-Hastings method to more general state spaces [23]. It is mostly used in settings where we wish to perform model selection, such as inferring a model where the order is unknown.

### 1.19.6.1 Reversible jump MCMC

The reversible jump MCMC is a Metropolis-Hastings method for sampling from Markov chains in state spaces of varying dimensionality. Such state spaces arise naturally in model selection problems where we have several candidate models  $\{\mathcal{M}_k\}$  with model index  $k$ , formally denoted as  $k \in \mathcal{K}$  for some finite set  $\mathcal{K}$ . With each model  $\mathcal{M}_k$  there is an associated parameter vector  $\theta_k$  of dimensionality  $N_k$ , i.e.,  $\theta_k \in \mathbb{R}^{N_k}$ . The inferential goal is to sample from the joint density  $\pi(k, \theta_k)$  of model index  $k$  and the parameters  $\theta_k$  where

$$\pi(\theta_k, k) = \frac{1}{Z_k} \phi_k(\theta_k) \tilde{p}_k, \quad (19.12)$$

$$\tilde{p}_k \equiv \frac{p_k}{\sum_{k' \in \mathcal{K}} p_{k'}}. \quad (19.13)$$

Here, for all models  $k \in \mathcal{K}$ ,  $Z_k = \int d\theta_k \phi_k(\theta_k)$  are unknown normalizing constants and  $p_k$  are prior weights, usually taken as flat with  $p_k = 1$ . These quantities have direct counterparts in a Bayesian

model selection setting: we have a probability model that couples observations  $x$  with parameters as  $p(x, \theta_k, k) = p(x, \theta_k|k)p(k)$ . In this context, the target density is the posterior

$$\pi(\theta_k, k) = p(\theta_k, k|x) = p(\theta_k|k, x)p(k|x)$$

and the joint distribution of parameters and data for each model  $\mathcal{M}_k$  is  $\phi_k(\theta_k) = p(x, \theta_k|k)$  and  $Z_k = \int d\theta_k \phi_k(\theta_k) = p(x|k)$ . To find the model order after observing the data  $x$  we would need the posterior marginal

$$p(k|x) = \frac{p(k, x)}{\sum_{k' \in \mathcal{K}} p(k', x)} = \frac{\tilde{p}_k Z_k}{\sum_{k \in \mathcal{K}} \tilde{p}_k Z_k} = \frac{p_k Z_k}{\sum_{k \in \mathcal{K}} p_k Z_k},$$

where the equalities follow by noting that  $p(k, x) = \int p(\theta_k, k, x)d\theta_k = \tilde{p}_k Z_k$  and  $p(x) = \sum_k \tilde{p}_k Z_k$ . With these results, the posterior can be written as

$$\pi(\theta_k, k) = \frac{1}{Z_k} \phi_k(\theta_k) \frac{p_k Z_k}{\sum_{k \in \mathcal{K}} p_k Z_k} = \phi_k(\theta_k) \frac{p_k}{\sum_{k \in \mathcal{K}} p_k Z_k}.$$

One possible approach for calculating the posterior of models order  $p(k|x)$  would require estimation of individual normalizing constants  $Z_k$ . The RJ-MCMC approach circumvents this need by setting up a chain to sample from  $(k, \theta_k)$  pairs. The difficulty is that now the target density  $\pi$  is defined on a nonstandard state space  $\mathcal{E}$  where

$$\begin{aligned} \mathcal{E} &= \bigcup_{k \in \mathcal{K}} \mathcal{E}_k, \\ \mathcal{E}_k &\equiv \{\{k\} \times \mathbb{R}^{N_k}\}, \end{aligned}$$

and we need to set up a so called *transdimensional* Markov chain that can jump across spaces of different dimensions  $\mathcal{E}_k$  and  $\mathcal{E}_{k'}$  as  $(k, \theta_k) \rightarrow (k', \theta_{k'})$ . This chain is “nonstandard” in the sense that its dimension varies over time but is otherwise completely natural for the class of model selection problems.

There is a technical caveat: The “ordinary” Metropolis-Hastings algorithm requires that we design a proposal

$$q(k', \theta_{k'}|k, \theta_k) = q(\theta_{k'}|k', k, \theta_k)q(k'|k, \theta_k)$$

and evaluate the acceptance probability as

$$\alpha((k, \theta_k) \rightarrow (k', \theta_{k'})) = \min \left\{ 1, \frac{q(k, \theta_k|k', \theta_{k'})\pi(k', \theta_{k'})}{q(k', \theta_{k'}|k, \theta_k)\pi(k, \theta_k)} \right\}.$$

We need to construct the proposals carefully, as as a proper lower dimensional space embedded into a higher dimensional space may have measure zero with respect to the probability measure defined on the larger space.<sup>11</sup> The remedy comes from the transformation method introduced earlier. By introducing auxiliary variables, we design the proposal as a “reversible” transformation between both spaces.

---

<sup>11</sup>The probability of jumping to an isolated point (a proper low dimensional space) under a proposal with an absolutely continuous density is zero. As an example, think of the probability of hitting an interval  $(-\epsilon, \epsilon)$  when  $\epsilon \rightarrow 0$ .

Suppose we wish to jump from a lower dimensional space  $\mathcal{E}_k$  to a higher dimensional space  $\mathcal{E}_{k'}$ . We define  $\bar{N}_k$  such that  $N_{k'} = N_k + \bar{N}_k$  and draw the  $\bar{N}_k$  dimensional vector  $u_k$  from a proposal  $q_k(u_k)$  and match the dimensions of both spaces via a transformation defined as

$$\theta_{k'} = g_{k \rightarrow k'}(\theta_k, u_k).$$

The mapping  $g_{k \rightarrow k'}$  is fairly general but must be invertible such that  $g_{k \rightarrow k'}^{-1} = g_{k' \rightarrow k}$  exists. As an alternative notation for  $g$  or its inverse  $g^{-1}$ , we omit the letter  $g$  and write  $\theta_{k'}(\theta_k, u_k)$  and  $(\theta_k(\theta_{k'}), u_k(\theta_{k'}))$  respectively. More naturally, it is also possible to match dimensions on a higher dimensional space where  $N_{k'} + \bar{N}_k = N_k + \bar{N}_k$  where  $(\theta_{k'}, u_{k'}) = g(\theta_k, u_k)$  but we won't discuss this option further here.

In order to design a MH algorithm, we will construct our proposal in two stages as in

$$q(k', \theta_{k'}|k, \theta_k) = q(k'|k, \theta_k)q(\theta_{k'}|k', k, \theta_k).$$

The first factor in this proposal is the probability of choosing a jump from  $k$  to  $k'$  when the chain is in  $\theta_k$ . Often, this term is taken to be independent of  $\theta_k$ , so  $q(k'|k) = q(k'|k, \theta_k)$ . We will denote the jump probabilities  $q(k'|k)$  as  $q_{k \rightarrow k'}$ . The second factor, which we will denote as  $f_{k'}(\theta_{k'}) \equiv q(\theta_{k'}|k', k, \theta_k)$ , will be defined via the variable transformation  $g_{k \rightarrow k'}$ . Noting that  $\theta_{k'} = g_{k \rightarrow k'}(\theta_k, u_k)$ , for some density  $f_k(\theta_k)$ , the density of  $\theta_{k'}$  can be written as is

$$\begin{aligned} f_{k'}(\theta_{k'}) &= f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'})) \left| \frac{\partial g_{k \rightarrow k'}^{-1}(\theta_{k'})}{\partial \theta_{k'}} \right| \\ &= f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'})) \left| \frac{\partial g_{k \rightarrow k'}(\theta_k, u_k)}{\partial (\theta_k, u_k)} \right|^{-1}. \end{aligned}$$

To simplify the notation, in the sequel we will denote the Jacobian terms as

$$J_{k \rightarrow k'} \equiv \frac{\partial g_{k \rightarrow k'}(\theta_k, u_k)}{\partial (\theta_k, u_k)} \quad J_{k' \rightarrow k} \equiv \frac{\partial g_{k \rightarrow k'}^{-1}(\theta_{k'})}{\partial \theta_{k'}} = J_{k \rightarrow k'}^{-1}.$$

Now, interpret  $f_k(\theta_k)$  as a reverse jump proposal  $q(\theta_k|k, k', \theta_{k'})$ . The ratio of the proposals to compute the MH-acceptance probability can be written for this case as

$$\begin{aligned} \frac{q(k, \theta_k|k', \theta_{k'})}{q(k', \theta_{k'}|k, \theta_k)} &= \frac{f_k(\theta_k)q_{k' \rightarrow k}}{f_{k'}(\theta_{k'})q_{k \rightarrow k'}} = \frac{f_k(\theta_k)q_{k' \rightarrow k}}{f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'}))q_{k \rightarrow k'}} \left| \frac{\partial g_{k \rightarrow k'}(\theta_k, u_k)}{\partial (\theta_k, u_k)} \right| \\ &= \frac{q_{k' \rightarrow k}}{q_k(u_k(\theta_{k'}))q_{k \rightarrow k'}} |J_{k \rightarrow k'}|. \end{aligned}$$

The ratio of the targets is

$$\frac{\pi(k', \theta_{k'})}{\pi(k, \theta_k)} = \frac{\phi_{k'}(\theta_{k'}) \frac{p_{k'}}{\sum_{\kappa \in \mathcal{K}} p_{\kappa} Z_{\kappa}}}{\phi_k(\theta_k) \frac{p_k}{\sum_{\kappa \in \mathcal{K}} p_{\kappa} Z_{\kappa}}} = \frac{\phi_{k'}(\theta_{k'}) p_{k'}}{\phi_k(\theta_k) p_k}.$$

Consequently, we arrive at the rather complicated looking formula for the RJ-MCMC acceptance probability

$$\alpha((k, \theta_k) \rightarrow (k', \theta_{k'})) = \min \left\{ 1, \frac{\phi_{k'}(\theta_{k'}) p_{k'}}{\phi_k(\theta_k) p_k} \frac{q_{k' \rightarrow k}}{q_k(u_k) q_{k \rightarrow k'}} |J_{k \rightarrow k'}| \right\} \quad (19.14)$$

$$\equiv \min\{1, r_{k \rightarrow k'}\} \quad (19.15)$$

$$\equiv \alpha_{k \rightarrow k'}. \quad (19.16)$$

The  $r_{k \rightarrow k'}$  argument min function is known as the MH-ratio. The acceptance probability of the jump in the reverse direction is

$$\begin{aligned} \alpha_{k' \rightarrow k} &= \min \left\{ 1, \frac{\phi_k(\theta_k) p_k}{\phi_{k'}(\theta_{k'}) p_{k'}} \frac{q_k(u_k) q_{k \rightarrow k'}}{q_{k' \rightarrow k}} |J_{k' \rightarrow k}| \right\} \\ &= \min\{1, r_{k' \rightarrow k}\} = \min \left\{ 1, r_{k \rightarrow k'}^{-1} \right\} \end{aligned}$$

Note that for a pair of reversible jumps  $(k \rightarrow k')$  and  $(k' \rightarrow k)$ , the corresponding MH-ratios are simply the inverses of each other  $r_{k' \rightarrow k} = r_{k \rightarrow k'}^{-1}$ .

### 1.19.6.2 RJ-MCMC algorithm

The RJ-MCMC algorithm is actually a special Metropolis Hastings method with a special proposal mechanism for transdimensional moves. To construct the algorithm, for each pair  $k, k' \in \mathcal{K}$  we need to decide on the jump probabilities  $q_{k \rightarrow k'}$  and  $q_{k' \rightarrow k}$ . Note that these are just parameters of the sampling algorithm and not the model, but a careful choice is important for good mixing. In practice, often only jumps between “adjacent”  $k$  and  $k'$  are used as it is relatively simpler to design proposals with reasonable acceptance probability. Here, a popular choice is only allowing moves with  $k' = k+1, k' = k-1, k' = k$ , where these are called *birth, death and update* moves.

If a pair of moves have nonzero probability,  $q_{k \rightarrow k'} q_{k' \rightarrow k} > 0$ , we have to have to design the following:

- For jumps to a larger space  $N_{k'} > N_k$  we design a suitable invertible transformation  $g_{k \rightarrow k'}$  and a proposals for  $q_k(u_k)$  to sample a  $\bar{N}_k$  dimensional random vector to match the dimensions such that  $N_{k'} = N_k + \bar{N}_k$ . Note that our choice completely fixes the moves in the reverse direction as well.
- Calculate the Jacobian determinants  $|J_{k \rightarrow k'}| = |J_{k' \rightarrow k}|^{-1}$ .
- Calculate the acceptance probabilities. Remember that if

$$\alpha_{k' \rightarrow k} = \min \{1, r_{k' \rightarrow k}\} \Leftrightarrow \alpha_{k \rightarrow k'} = \min \{1, r_{k \rightarrow k'}\} = \min \left\{ 1, \frac{1}{r_{k' \rightarrow k}} \right\}$$

A summary of the RJ-MCMC is shown as Algorithm 6

**Example 6 (Reversible Jump - MCMC).** As a illustrative toy example, we define a target density on the union of two spaces  $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$  where

$$\mathcal{E}_1 = \{\{1\} \times r\} \quad \mathcal{E}_2 = \{\{2\} \times (x, y)\}$$

**Algorithm 6.** Reversible Jump Markov Chain Monte Carlo

---

```

1: Initialise  $(k^{(1)}, \theta_{(1)}^{(1)})$  arbitrarily
2: for  $\tau = 2, 3, \dots$  do
3:   Let  $(k, \theta_k) = (k^{(\tau-1)}, \theta_{k^{(\tau-1)}}^{(\tau-1)})$ 
4:   Propose a move type:  $k^{\text{new}} \sim q_{k \rightarrow k'}$ 
5:   if  $k = k^{\text{new}}$ , an ordinary move in the same space then
6:      $\theta^{\text{new}} \sim q_k(\theta_k | \theta_k^{(\tau-1)})$ 
7:   end if
8:   if  $k < k^{\text{new}}$ , a transdimensional move to a larger space then
9:     Propose  $u \sim q_k(u_k)$ 
10:     $\theta^{\text{new}} \leftarrow g_{k \rightarrow k^{\text{new}}}(\theta_k, u)$ 
11:    Compute the acceptance probability
        
$$\alpha((k, \theta_k) \rightarrow (k^{\text{new}}, \theta^{\text{new}})) = \min \left\{ 1, \frac{\phi_{k^{\text{new}}}(\theta_{k^{\text{new}}}) p_{k^{\text{new}}}}{\phi_k(\theta_k) p_k} \frac{q_{k^{\text{new}} \rightarrow k}(u) q_{k \rightarrow k^{\text{new}}}}{q_k(u) q_{k \rightarrow k^{\text{new}}}} |J_{k \rightarrow k^{\text{new}}}| \right\}$$

12:   end if
13:   if  $k > k^{\text{new}}$ , a transdimensional move to a smaller space then
14:      $(\theta^{\text{new}}, u) \leftarrow g_{k \rightarrow k^{\text{new}}}(\theta_k)$ 
15:     Compute the acceptance probability
        
$$\alpha((k, \theta_k) \rightarrow (k^{\text{new}}, \theta^{\text{new}})) = \min \left\{ 1, \frac{\phi_{k^{\text{new}}}(\theta_{k^{\text{new}}}) p_{k^{\text{new}}}}{\phi_k(\theta_k) p_k} \frac{q_{k^{\text{new}}}(\theta^{\text{new}}) q_{k \rightarrow k^{\text{new}}}}{q_{k^{\text{new}}}(\theta_k) q_{k \rightarrow k^{\text{new}}}} |J_{k \rightarrow k^{\text{new}}}| \right\}$$

16:   end if
17:   Sample from uniform distribution:  $a \sim \mathcal{U}(0, 1)$ 
18:   if  $a < \alpha((k, \theta_k) \rightarrow (k^{\text{new}}, \theta^{\text{new}}))$  then
19:     Accept candidate:  $(k^{(\tau)}, \theta_{(\tau)}^{(\tau)}) \leftarrow (k^{\text{new}}, \theta^{\text{new}})$ 
20:   else
21:     Reject candidate and stay put:  $(k^{(\tau)}, \theta_{k^{(\tau)}}^{(\tau)}) \leftarrow (k^{(\tau-1)}, \theta_{k^{(\tau-1)}}^{(\tau-1)})$ 
22:   end if
23: end for

```

---

with parameters corresponding to each space as

$$\theta_1 \equiv r \quad \theta_2 \equiv (x, y)$$

We choose, for  $k = 1$ , a target proportional to a truncated exponential distribution. For  $k = 2$ , our target is a two dimensional Gaussian density truncated on an elliptical region

$$\begin{aligned} \phi_1(r) &= e^{-\lambda r} [0 \leq r \leq 1] \\ \phi_2(x, y) &= \exp \left( -\frac{(x - \mu_x)^2 + (y - \mu_y)^2}{2\sigma^2} \right) \left[ \left( \frac{x}{a} \right)^2 + \left( \frac{y}{b} \right)^2 \leq 1 \right] \end{aligned}$$

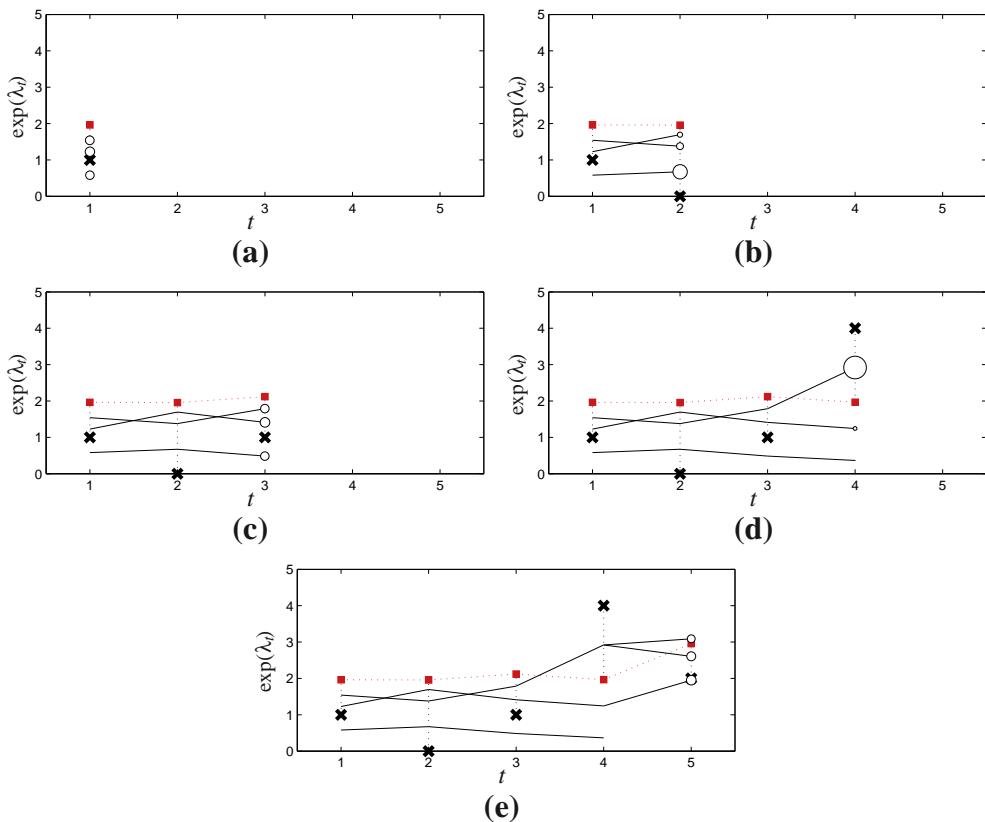
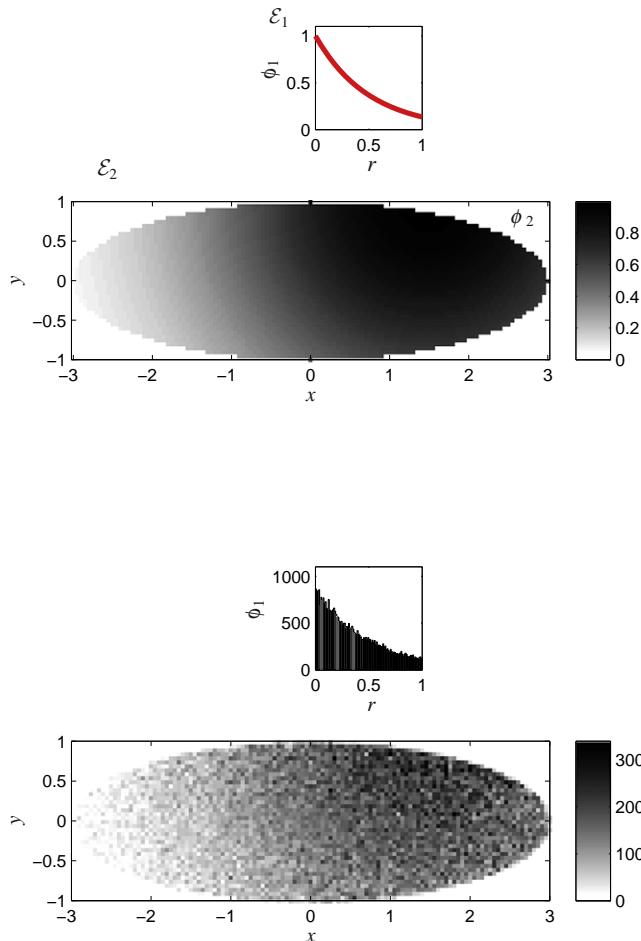
**FIGURE 19.23**

Illustration of the Bootstrap particle filter for the model in Eq. (19.10) and Eq. (19.11) using  $N = 3$  particles. The observations  $y_t$  are shown as black crosses. The squares show the true state of the process  $\exp(\lambda_t)$  used for generating the observations  $y_t$ . Remember that  $\langle y_t \rangle = \exp(\lambda_t)$  due to the Poisson assumption so it is natural to plot the  $y_t$  and  $\exp(\lambda_t)$  on the same plot. At each time  $t$ , the particles are shown as individual trajectories of  $\lambda_{1:t}$  with the circle size proportional to the normalized weight at time  $t$ . The edges denote the ancestral links. While the term “particle” suggests that each particle is only a point in  $\lambda_t$ , it is actually more natural to view a particle as an entire trajectory of  $\lambda_{1:t}$ . At each time  $t$ , each particle selects a new position  $l_{t+1}$  and the new weight  $W_{t+1}(l_{1:t+1})$  is computed recursively. If the effective sample size drops below a chosen threshold, we compute the normalized weights  $\tilde{w}^{(1:N)}$  and select each particle via resampling (See at time  $t = 4$ ) and reset the weights to  $1/N$ .

the normalization constants  $Z_1 = \int_0^1 e^{-\lambda r} dr$  and  $Z_2 = \int \int \phi_2(x, y) dx dy$  are assumed to be unknown. As shown earlier, these cancel out in the acceptance probability. The target density is

$$\pi(k, \theta_k) = \frac{1}{p_1 Z_1 + p_2 Z_2} ([k = 1] \phi_1(r) p_1 + [k = 2] \phi_2(x, y) p_2) \quad (19.17)$$

**FIGURE 19.24**

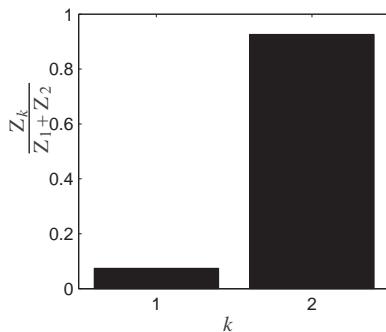
(Top) The target density  $\pi \propto [k = 1]\phi_1(r)p_1 + [k = 2]\phi_2(x, y)p_2$ , with parameters  $\lambda = 1, a = 3, b = 1, \mu_x = 1.5, \mu_y = 1, \sigma^2 = 4, p_1 = p_2 = 1$ . (Bottom) Histograms obtained from samples of the RJ-MCMC algorithm.

This target density is illustrated on Figures 19.23, 19.24, 19.25.

We will derive now a RJ-MCMC method to sample from the target density defined in Eq. (19.17).

### 1.19.6.2.1 Acceptance probabilities

To design a RJ-MCMC algorithm, we should define an “ordinary” MCMC algorithm to sample within each space  $\mathcal{E}_k$  for all  $k \in \mathcal{K}$  and transdimensional moves for each pair  $k, k'$  such that  $k \neq k'$  using the general formula for the RJ-MCMC acceptance probability  $\alpha$  as given in 19.14.

**FIGURE 19.25**

Histogram estimate of the marginal  $\pi(k)$ . By simply counting and normalizing the number of steps the chain spends in each space, we can estimate the marginal  $\pi(k)$ , with probabilities  $p_1 Z_1 / (p_1 Z_1 + p_2 Z_2)$  and  $p_2 Z_2 / (p_1 Z_1 + p_2 Z_2)$ . Such quantities are useful for model comparison.

$\mathcal{E}_1 \rightarrow \mathcal{E}_1$ .

We use a Metropolis algorithm with an independent uniform proposal such that

$$r' \sim \mathcal{U}(0, 1),$$

giving the acceptance probability as

$$\alpha_{1 \rightarrow 1} = \min \left\{ 1, \frac{\phi_1(r')}{\phi_1(r)} \right\}, \quad (19.18)$$

$\mathcal{E}_2 \rightarrow \mathcal{E}_2$

We choose a symmetric random walk proposal

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \epsilon, \quad (19.19)$$

where  $\epsilon \sim \mathcal{N}(0, vI)$ . Since the proposal is symmetric, the Metropolis acceptance probability is

$$\alpha_{2 \rightarrow 2} = \min \left\{ 1, \frac{\phi_2(x', y')}{\phi_2(x, y)} \right\} \quad (19.20)$$

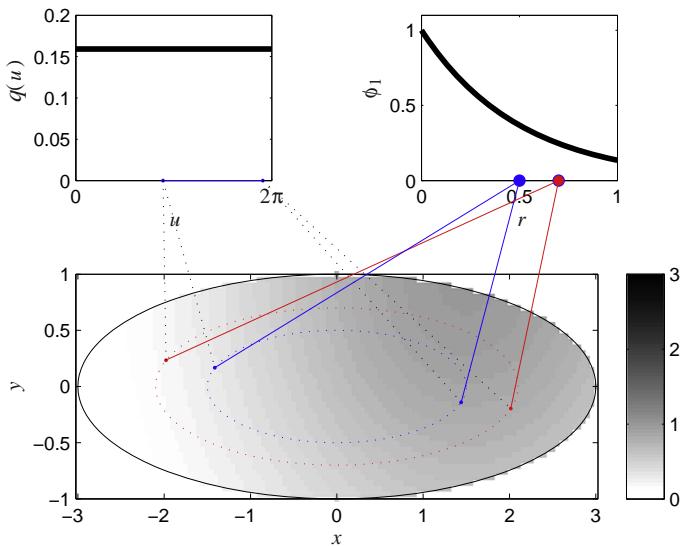
$\mathcal{E}_1 \rightarrow \mathcal{E}_2$

In order to make a transition from the one dimensional space  $\mathcal{E}_1$  to the two dimensional space  $\mathcal{E}_2$ , we define the auxiliary variable  $u_1$  such that

$$u_1 \sim q_1(u_1) = U(0, 2\pi) = \frac{1}{2\pi} \quad (19.21)$$

and define the following one-to-one transformation  $(x, y) = g(r, u)$  as

$$x = ar \cos(u) \quad y = br \sin(u).$$

**FIGURE 19.26**

The mapping  $(x, y) = g(r, u)$ . For trans-dimensional jumps between the spaces  $\mathcal{E}_1$  (top, right) and  $\mathcal{E}_2$  (bottom), to match the dimensions, we draw an angle uniformly  $u \sim q(u) = \mathcal{U}(0, 2\pi)$  and construct a one-to-one mapping  $g$ . A proposal  $f_1$  on  $\mathcal{E}_1$  translates to a proposal  $f_2$  on  $\mathcal{E}_2$  as  $f_2(x, y) = f_1(r(x, y))q(u(x, y))|J_{2 \rightarrow 1}|$ . The acceptance probability does not depend on  $f_1$  or  $f_2$  as these cancel out; only the term  $q(u(x, y))|J_{2 \rightarrow 1}|$  remains.

Each  $r$  is mapped uniformly into a point on the ellipse  $x^2/a + y^2/b = r^2$ . This mapping is depicted in Figures 19.26, 19.27. To compute the acceptance probability, we need the Jacobian. The partial derivatives of the transformation are obtained as follows

$$\begin{aligned} \frac{\partial x}{\partial r} &= a \cos(u) & \frac{\partial x}{\partial u} &= -ar \sin(u) \\ \frac{\partial y}{\partial r} &= b \sin(u) & \frac{\partial y}{\partial u} &= br \cos(u) \end{aligned} \quad (19.22)$$

and the Jacobian term can then be determined as

$$\begin{aligned} |J_{1 \rightarrow 2}| &= \left| \frac{\partial x}{\partial r} \frac{\partial y}{\partial u} - \frac{\partial x}{\partial u} \frac{\partial y}{\partial r} \right| \\ &= |a \cos(u)br \cos(u) + ar \sin(u)b \sin(u)| = \left| abr(\cos^2(u) + \sin^2(u)) \right| = abr \end{aligned}$$

Finally, substituting the results the acceptance probability  $\alpha_{1 \rightarrow 2}$  is obtained as

$$\alpha_{1 \rightarrow 2} = \min \left\{ 1, \frac{\phi_2(x, y)p_2}{\phi_1(r)p_1} \frac{q_{2 \rightarrow 1}}{q_{1 \rightarrow 2}} \frac{1}{(1/2\pi)} |J_{1 \rightarrow 2}| \right\} \quad (19.23)$$

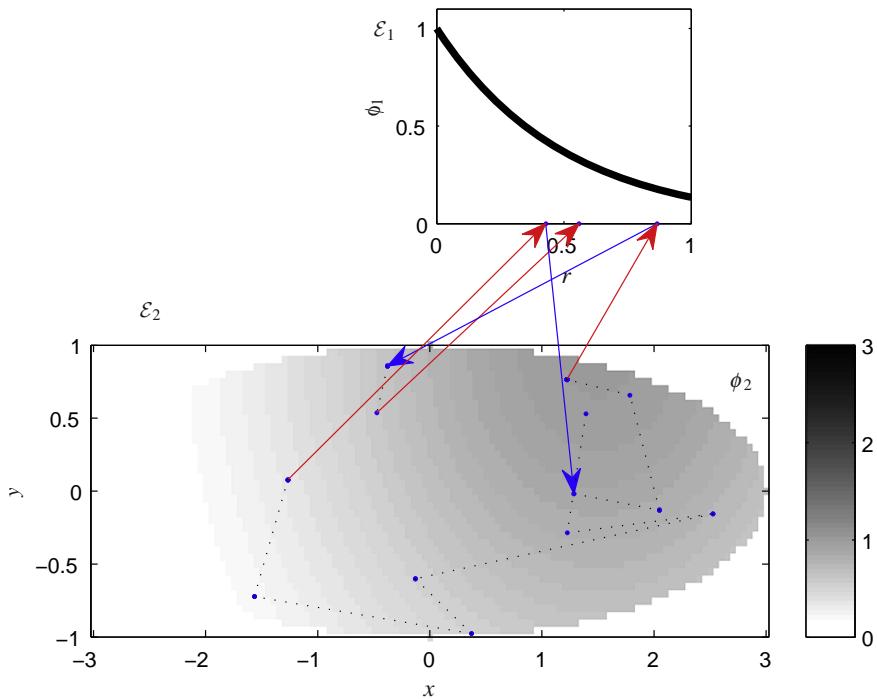
**FIGURE 19.27**

Illustration of the RJ-MCMC algorithm on the toy problem. The dotted lines corresponds to accepted moves within the same space, and arrows correspond to transdimensional moves. The algorithm proceeds as follows: Suppose at step  $\tau$ , the chain is in space  $\mathcal{E}_{k(\tau)}$ . We first propose the next index  $k'$  with probability  $q_{k(\tau) \rightarrow k'}$ . If  $k' = k(\tau)$ , we use an ordinary MH move. Otherwise, we draw a new  $u \sim p(u)$  and propose a new point  $\theta_{k'}(\theta_{k(\tau)}^{(\tau)}, u) \in \mathcal{E}_{k'}$  and accept it with probability  $\alpha_{k(\tau) \rightarrow k}'$ . If the move is accepted we set  $(k^{(\tau+1)}, \theta_{k^{(\tau+1)}}^{(\tau+1)}) \leftarrow (k', \theta_{k'})$ . Otherwise, the chain stays at the same point point is equal to the old point  $(k^{(\tau+1)}, \theta_{k^{(\tau+1)}}^{(\tau+1)}) \leftarrow (k^{(\tau)}, \theta_{k(\tau)}^{(\tau)})$ .

$$\mathcal{E}_2 \rightarrow \mathcal{E}_1$$

We find the inverse transformation  $g^{-1}$ ,

$$r = \sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2}} \quad u = \arccos\left(\frac{xb}{\sqrt{x^2b^2 + y^2a^2}}\right)$$

The Jacobian  $J_{2 \rightarrow 1}$  is obtained by calculating the partial derivatives

$$\begin{aligned} \frac{\partial r}{\partial x} &= \frac{xb}{a\sqrt{x^2b^2 + y^2a^2}} & \frac{\partial r}{\partial y} &= \frac{ya}{b\sqrt{x^2b^2 + y^2a^2}} \\ \frac{\partial u}{\partial x} &= -\frac{yab}{x^2b^2 + y^2a^2} \frac{y}{|y|} & \frac{\partial u}{\partial y} &= \frac{xab}{x^2b^2 + y^2a^2} \frac{y}{|y|} \end{aligned}$$

and the determinant is

$$|J_{2 \rightarrow 1}| = \frac{1}{\sqrt{x^2 b^2 + y^2 a^2}}$$

Note that we could have avoided this calculation by directly using the Jacobian of the inverse as  $|J_{2 \rightarrow 1}| = |J_{1 \rightarrow 2}|^{-1} = 1/(abr)$ , giving the same result. The acceptance probability  $\alpha_{2 \rightarrow 1}$  is found as follows:

$$\alpha_{2 \rightarrow 1} = \min \left\{ 1, \frac{\phi_1(r)p_1}{\phi_2(x, y)p_2} \frac{q_{1 \rightarrow 2}(1/2\pi)}{q_{2 \rightarrow 1}} |J_{2 \rightarrow 1}| \right\} \quad (19.24)$$

## 1.19.7 Open issues and problems

In this tutorial, we have sketched in a self contained manner the basic techniques and the principles of Monte Carlo computation, along with several examples. Monte Carlo computation is a very broad topic and with the available computing power, researchers are tackling increasingly more complex models and problems. The need for extracting structure from large datasets forces the researchers to devise increasingly more complicated models that ask for more efficient inference methodologies. In some research communities, most notably machine learning, the MC methods are considered not very scalable to very large data sets due to their heavy computational requirements. It is an open issue in research for developing MC methods for such datasets. The research frontier, at the time of this writing, is swiftly moving. Researchers deal with techniques such as adaptive methods that learn suitable proposals[32], combining sequential Monte Carlo with MCMC [33], likelihood free inference [6] or exploiting parallel computation [34] for scalability.

## 1.19.8 Further reading

To investigate topics more deeper, we suggest a few key references. For basic probability theory, Grimmett and Stirzaker [4] is a very good reference. More rigorous material, needed for a deeper understanding of advanced subjects is covered in Rosenthal [12] and for the theory of Markov chains Norris is a key reference [7]. A very gentle introduction to basic Monte Carlo methods from a machine learning perspective is in MacKay's book [5]. A good tutorial introduction to Markov chain Monte Carlo methods can be found in [6,18]. There are also excellent tutorials on sequential Monte Carlo methods, most notably by Doucet et. al. [10,30] or Liu et. al. [8,20].

## Glossary

acceptance probability	17
aperiodic (chain)	14
Box-Müller method	14
CDF Cumulative Density Function	8
central limit theorem	4

CLT Central Limit Theorem	2
detailed balance	19
deterministic sca	25
ergodic chain	14
full conditional density	23
generalised inverse	8
Gibbs sampler	23
inversion method	8
irreducible (chain)	14
IS importance sampling	29
Jacobian	10
law of large numbers	4
LCG Linear Congruential Generator	7
Linear Congruential Generator a method for generating pseudorandom numbers based on a recursive formula	47
LLN Law of Large Numbers	2
Markov Chain Monte Carlo a Monte Carlo method that depends on simulating from a ergodic Markov chain to generate estimates of a stationary distribution	14
MCMC Markov Chain Monte Carlo	1
Mersenne Twister a Linear Congruential Generator with a provably very long period	7
MH Metropolis-Hastings	17
Monte Carlo (method) a family of numerical techniques for computing approximate estimates via random sampling	1
Multinomial resampling	31
particle filtering	27
pseudo-random numbers	7
random scan	25
rejection probability	18
resampling	29,30
RJ-MCMC reversible jump MCMC	36,38,40
RJ-MCMC acceptance probability	40
SIS sequential importance sampling	27
SMC sequential Monte Carlo	27,29
spectral gap	20
Systematic resampling	31
total variation norm	16
transformation method	10
transition kernel	16
transition matrix	15

---

## References

- [1] N. Metropolis, S. Ulam, The Monte Carlo method, *Journal of the Am. Statist. Assoc.* 44 (247) (1949) 335–341.
- [2] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys.* 21 (1953) 1087–1091.
- [3] C. Geyer, Handbook of Markov Chain Monte Carlo, chapter Introduction to Markov Chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.
- [4] G.R. Grimmett, D.R. Stirzaker, *Probability and Random Processes*, Oxford University Press, 2001.
- [5] D.J.C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.
- [6] S. Brooks, A. Gelman, G. Jones, X. Meng (eds.), *Handbook of Markov Chain Monte Carlo* (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.
- [7] J.R. Norris, *Markov Chains*, Cambridge University Press, 1997.
- [8] J.S. Liu, R. Chen, Sequential Monte Carlo methods for dynamic systems, *J. Am. Statist. Assoc.* 93 (1998) 1032–1044.
- [9] A. Doucet, N. de Freitas, N.J. Gordon (eds.), *Sequential Monte Carlo Methods in Practice*, Springer Verlag, 2001.
- [10] A. Doucet, A.M. Johansen, Handbook of Nonlinear Filtering, chapter A Tutorial on Particle Filtering and Smoothing: Fifteen years Later, Oxford University Press, 2010.
- [11] P.J. Green, Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika* 82 (4) (1995) 711–732.
- [12] Jeffrey S. Rosenthal, A first look at rigorous probability theory, World Scientific, second ed., 2006.
- [13] Charles M. Grinstead, Laurie J. Snell, American Mathematical Society, fourth ed., July 2006.
- [14] The marsaglia random number cd-rom with the diehard battery of tests of randomness.
- [15] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator, *ACM Trans. Model Comput. Simul.* 8 (1) (1998) 3–30.
- [16] Luc Devroye, *Non-Uniform Random Variate Generation*, 1986.
- [17] Luke Tierney, Markov chains for exploring posterior distributions, *Ann. Statist.* 22 (1994) 1701–1762.
- [18] W.R. Gilks, S. Richardson, D.J. Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice*, CRC Press, London, 1996.
- [19] G.O. Roberts, J.S. Rosenthal, Markov Chain Monte Carlo: Some practical implications of theoretical results, *Can. J. Statist.* 26 (1998) 5–31.
- [20] J.S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, 2004.
- [21] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, *Biometrika* 57 (1970) 97–109.
- [22] O. Cappé, E. Moulines, T. Rydén, *Inference in Hidden Markov Models*, Springer-Verlag, New York, 2005.
- [23] Luke Tierney, A note on metropolis-hastings kernels for general state spaces, *Ann. Appl. Probab.* 8 (1998) 1–9.
- [24] S. Geman, D. Geman. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, in: M.A. Fischler, O. Firschein (Eds.), *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, Kaufmann, Los Altos, CA, 1987, pp. 564–584.
- [25] A.M. Johansen, L. Evers, N. Whiteley, Monte Carlo Methods, Online Resource, 2008 (Lecture Notes).
- [26] P. Fearnhead, Exact and Efficient Bayesian inference for multiple changepoint problems, Technical report, Dept. Math. Stat., Lancaster University, 2003.
- [27] Jarrett, A note on the intervals between coal mining disasters, *Biometrika* 66 (1979) 191–193.

- [28] A. Doucet, N. de Freitas, N.J. Gordon (eds.), Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, 2001.
- [29] O. Cappe, R. Douc, E. Moulines, Comparison of resampling schemes for particle filtering, in: 4th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagreb, Croatia, September 2005.
- [30] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, *Statist. Comput.* 10 (3) (2000) 197–208.
- [31] Y. Fan, S. Sisson, Handbook of Markov Chain Monte Carlo, Chapter Reversible jump Markov chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.
- [32] C. Andrieu, J. Thoms. A tutorial on adaptive mcmc, *Statist. Comput.* 18 (4) (2008) 343–373.
- [33] Pierre Del Moral, Arnaud Doucet, Ajay Jasra. Sequential Monte Carlo samplers, *J. Roy. Statist. Soc. Ser. B Stat. Methodol.* 68 (3) (2006) 411–436.
- [34] Anthony Lee, Christopher Yau, Michael B Giles, Arnaud Doucet, Christopher C. Holmes, On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods, May 2009, pp. 4–6.

# Clustering

# 20

**Dao Lam and Donald C. Wunsch**

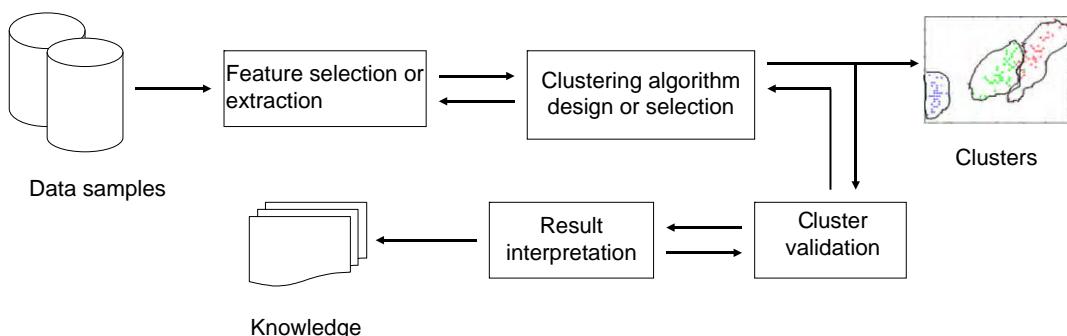
*Missouri University of Science and Technology, Applied Computational Intelligence Lab,  
Department of Electrical and Computer Engineering, Rolla, MO, USA*

## 1.20.1 Introduction

Every day, society generates large amounts of data, which can be subjected to analysis and management. One vital way to handle these data is to classify or group them into a set of categories or clusters. In order to learn a new object or understand a new phenomenon, people seek features to describe it; additionally, they compare it with other known objects or phenomena based on similarity or dissimilarity, generalized as proximity, according to some certain standards or rules. In unsupervised classification, also called clustering or exploratory data analysis, no labeled data are available. The goal of clustering is to separate a finite, unlabeled data set into a finite, discrete set of “natural,” hidden data structures, rather than to accurately characterize unobserved samples generated from the same probability distribution [1,2]. As noted by Backer and Jain, “in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create “intersecting” clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups” [3].

Clustering algorithms partition data into clusters (groups, subsets, or categories), but these clusters have no universally agreed-upon definition [4]. A cluster can be “a set of entities which are alike, and entities from different clusters are not alike” or “an aggregate of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it” [4]. A cluster also can be defined as a continuous region of a space containing a relatively high density of points, separated from other such regions by regions containing a relatively low density of points.

Furthermore, a cluster can be defined based on the method used to solve the clustering problem. In hierarchical clustering, a cluster is an entity of connectivity. In partition-based clustering, each cluster is represented by a single prototype vector. In model-based clustering, each cluster is modeled by a statistical distribution. In density estimation clustering, each cluster is defined as a connected dense region. Subspace clustering views clusters as a subspace. If graph theory techniques are used for clustering, each cluster typically is called a clique, a subset of nodes fully connected among themselves. From a membership point of view, clustering can be classified as hard or fuzzy. In hard clustering, each object belongs to one and only one cluster, while in fuzzy clustering, each object has some degree of membership in each cluster.

**FIGURE 20.1**

Clustering procedure (From [5]). Many if not most papers consider mainly the algorithm design or selection, but all the steps shown are important.

In describing a cluster, most researchers consider internal homogeneity and external separation [6–8], i.e., patterns in the same cluster should be similar to each other, while patterns in different clusters should not. Similarities and dissimilarities both should have the potential to be examined in a clear and meaningful way.

Figure 20.1 depicts the four basic steps of the cluster analysis procedure.

Clustering has been applied in a wide variety of fields, ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer sciences (web mining, spatial database analysis, textual document collection, image segmentation) and life and medical sciences (genetics, biology, microbiology, paleontology, psychiatry, clinic, pathology), to earth sciences (geography, geology, remote sensing), social sciences (sociology, psychology, archeology, education), and economics (marketing, business) [4,9]. Accordingly, clustering also is known as numerical taxonomy, learning without a teacher (or unsupervised learning), typological analysis and partitioning. While the diversity of these names reflects the important position of clustering in scientific research, the differing terminologies and goals also results in confusion. Clustering algorithms developed to solve a particular problem in a specialized field usually make assumptions in favor of the application of interest. These biases inevitably affect the algorithm's performance in other problems that do not satisfy the same premises.

Clustering has a long history, dating back to Aristotle [8]. General references on clustering techniques include textbooks by Duda et al. [10], Hartigan [9], Everitt et al. [4], Jain and Dubes [6], Späth [11], Duran and Odell [12], Gordon [7], and Anderberg [13]. Important survey papers on clustering techniques also exist in the literature. Starting from a statistical pattern recognition perspective, Jain, Murty and Flynn reviewed clustering algorithms and other important issues related to cluster analysis [14], while Hansen and Jaumard described clustering problems under a mathematical programming scheme [8]. Kolatch and He investigated applications of clustering algorithms for spatial database systems [15] and information retrieval [16], respectively. Berkhin [17] further expanded the topic to the field of data mining. Murtagh [18] reported advances in hierarchical clustering

**Table 20.1** Notations Used in this Chapter

K,c	Number of clusters
$X = \{x_1, \dots, x_j, \dots, x_N\}$	A set of $N$ input patterns
$x = (x_1, x_2, \dots, x_d)^T \in \mathbb{R}^d$	A pattern belonging to a $d$ -dimensional pattern space
$\{C_1, \dots, C_K\}$	A $K$ -partition of input patterns
$D(\cdot, \cdot)$	Distance function
$S(\cdot, \cdot)$	Similarity function
$J(\cdot)$	Criterion function
$M = (m_1, \dots, m_K)$	Prototype matrix of clusters
$\Gamma = \{\gamma_{ij}\}$	Partition matrix
$\Theta = (\theta_1, \dots, \theta_K)$	Model parameter matrix

algorithms, and Baraldi surveyed several fuzzy and neural network clustering methods [19]. Additional survey papers include those by Baraldi and Schenato [20], Bock [21], Dubes [22], Fasulo [23], and Jain et al. [24]. In addition to these review papers, comparative research on clustering algorithms is also significant [25] presented empirical results for five typical clustering algorithms. Wei et al., [26] compared fast algorithms for large databases. Scheunders [27] compared several clustering techniques for color image quantization, emphasizing computation time and the possibility of obtaining global optima. Applications and evaluations of different clustering algorithms for the analysis of gene expression data from DNA microarray experiments were described by Shamir and Sharan, [28] and Tibshirani et al. [29]. An experimental evaluation of document clustering techniques based on hierarchical and  $K$ -means clustering algorithms was summarized by Steinbach et al. [30].

High-dimensional clustering has become very significant as data continue to become more complex. Surveys regarding biclustering [31,32] and tutorials on subspace clustering [33] review a variety of clustering techniques for such high-dimensional data. On the other hand, researchers such as Andrew Ng [34] have suggested achieving improved performance in classification using unsupervised methods. This idea combines both supervised and unsupervised learning to accomplish recognition tasks. The purpose of this chapter is to extend the review offered by Xu and Wunsch [35] in order to provide a description of some influential and important clustering algorithms rooted in statistics, computer science, and machine learning, with an emphasis on signal and image processing. Much of this chapter's material has been taken or adapted from [5,35–37].

## 1.20.2 Clustering algorithms

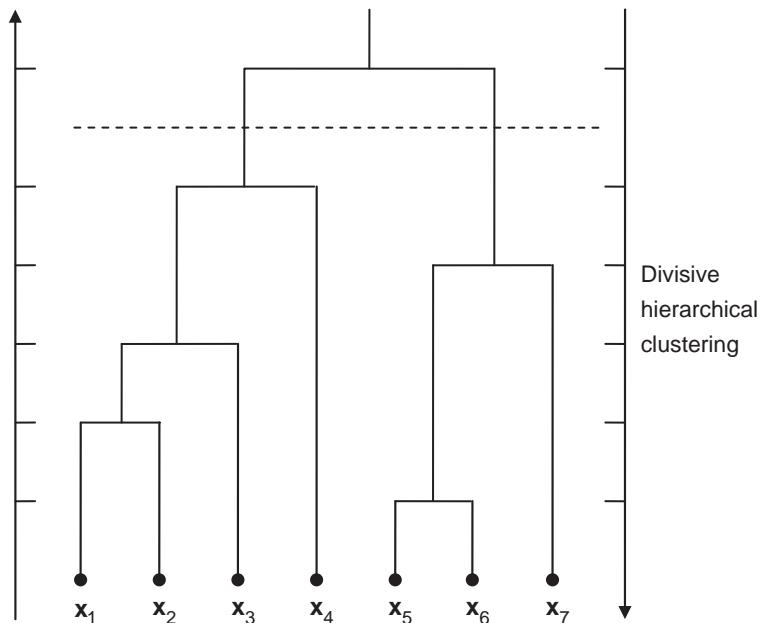
Several taxonomies of clustering algorithms [4,6,8,14,15,17] have been utilized, but the most widely accepted clustering techniques are hierarchical clustering and partitional clustering based on the properties of the clusters they generate [14]. Hierarchical clustering gathers data objects with a sequence of partitions, either from singleton clusters to a cluster including all individuals or vice versa. Partitional

clustering directly divides data objects into some pre-specified (either known or estimated) number of clusters without the hierarchical structure.

We begin with a discussion of hierarchical clustering followed by classical partitional clustering algorithms. Then, recent advanced techniques for subspace clustering and biclustering will be discussed. Finally, we will review unsupervised feature learning in the last part of the section. Table 20.1 lists notations used in this chapter.

### 1.20.2.1 Hierarchical clustering

As described by Xu and Wunsch [35], hierarchical clustering (HC) algorithms organize data into a hierarchical structure according to the proximity matrix. The results of HC usually are depicted by a binary tree or dendrogram, as depicted in Figure 20.2. The root node of the dendrogram represents the whole data set, and each leaf node represents a data object. The intermediate nodes thus describe the extent to which the objects are proximal to each other, and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or one object and one cluster. The best clustering results can be obtained by cutting the dendrogram at different levels. This representation provides very informative descriptions and a visualization of the potential data clustering structures, especially when real hierarchical relationships exist in the data, such as in the data from evolutionary



**FIGURE 20.2**

An example of a dendrogram in hierarchical clustering. The dotted line corresponds to choosing a number of clusters based on where the tree is cut. Note that the line does not necessarily have to be a straight, horizontal cut.

research on different species of organisms. HC algorithms are classified primarily as agglomerative methods or divisive methods. Agglomerative clustering starts with  $N$  clusters, each of which includes exactly one object. A series of merge operations then combines all objects into the same group. Divisive clustering proceeds in an opposite way. The entire data set initially belongs to a single cluster, and then a procedure successively divides it until all clusters are singletons. For a cluster with  $N$  objects, there are  $2^{N-1} - 1$  possible two-subset divisions, which is very computationally expensive [38]. Therefore, divisive clustering is not commonly used in practice. We focus on agglomerative clustering in the following discussion.

The variety of agglomerative clustering algorithms originally came about because of the different definitions for the distance between two clusters. The general formula for distance was proposed by Lance and Williams [39] as:

$$\begin{aligned} D(C_l, (C_i, C_j)) = & \alpha_i D(C_l, C_i) + \alpha_j D(C_l, C_j) + \\ & \beta D(C_i, C_j) + \gamma |D(C_l, C_i) - D(C_l, C_j)|, \end{aligned} \quad (20.1)$$

where  $D()$  is the distance function, and  $\alpha_i$ ,  $\alpha_j$ ,  $\beta$ , and  $\gamma$  are coefficients whose values depend on the scheme used.

The formula describes the distance between a cluster  $l$  and a new cluster formed by the merging of the two clusters  $i$  and  $j$ . Note that when  $\alpha_i = \alpha_j = 1/2$ ,  $\beta = 0$ , and  $\gamma = -1/2$ , the formula becomes

$$D(C_l, (C_i, C_j)) = \min(D(C_l, C_i), D(C_l, C_j)), \quad (20.2)$$

which corresponds to the single linkage method. When  $\alpha_i = \alpha_j = \gamma = 1/2$  and  $\beta = 0$ , the formula is

$$D(C_l, (C_i, C_j)) = \max(D(C_l, C_i), D(C_l, C_j)), \quad (20.3)$$

which corresponds to the complete linkage method.

Several more complicated agglomerative clustering algorithms, including group average linkage, median linkage, centroid linkage, and Ward's method, can also be constructed by selecting appropriate coefficients in the formula. A detailed table describing the coefficient values for different algorithms is offered in [6, 18]. Single-linkage, complete-linkage and average-linkage methods, all of which also are called graph methods, consider all points of a pair of clusters when calculating their inter-cluster distance. The other methods are called geometric methods because they use geometric centers to represent clusters, and they determine the distances between clusters based on these centers. The important features and properties of these methods were summarized by Everitt [4]. More inter-cluster distance measures, especially the mean-based ones, were introduced by Yager [40], who further discussed their potential for controlling the hierarchical clustering process.

Recently, to handle the limits of HC clustering and large-scale data sets, many new HC techniques have been proposed. Guha, Rastogi and Shim [41] developed an HC algorithm called CURE to explore more sophisticated cluster shapes. The crucial feature of CURE lies in its use of a set of well-scattered points to represent each cluster, which makes it possible to find rich cluster shapes other than hyperspheres while avoiding both the chaining effect [4] of the minimum linkage method and the tendency to favor clusters with similar centroid sizes. These representative points are further gathered towards the cluster centroid according to an adjustable parameter in order to weaken the effects of outliers.

CURE utilizes a random sample (and partition) strategy to reduce computational complexity [42] also proposed another agglomerative HC algorithm, ROCK, to group data with qualitative attributes. They used a novel measure, a “link,” to describe the relationship between a pair of objects and their common neighbors. Like CURE, ROCK uses a random sample strategy to handle large data sets.

Noticing the inability of centroid-based HC to identify arbitrary cluster shapes, relative hierarchical clustering (RHC) was developed, which considers both the internal distance (distance between a pair of clusters that may be merged to yield a new cluster) and the external distance (distance from the two clusters to the rest). It uses the ratio of these distances to decide proximities [43]. Leung et al. [44] demonstrated an interesting hierarchical clustering technique based on scale-space theory. They interpreted clustering using a blurring process, in which each data point is regarded as a light point in an image, and a cluster is represented as a blob. [45] extended agglomerative HC to handle both numeric and nominal data. The proposed algorithm, called SBAC (Similarity-Based Agglomerative Clustering), employs a mixed data measurement scheme that pays extra attention to less common matches of feature values. Parallel techniques for HC are discussed by Olson [46] and Dahlhaus [47].

### 1.20.2.2 Partitional clustering

As opposed to hierarchical clustering, partitional clustering assigns data into  $K$  clusters without any hierarchical structure by optimizing a criterion function, one of which is the sum of squared error criterion. Suppose we want to organize a set of objects  $\mathbf{x}_j \in \Re^d$ ,  $j = 1, \dots, N$ , into  $K$  subsets  $C = \{C_1, \dots, C_K\}$ . The squared error criterion then is defined as:

$$\mathbf{J}(\mathbf{U}, \mathbf{M}) = \sum_{i=1}^K \sum_{j=1}^N u_{ij} \|\mathbf{x}_j - \mathbf{m}_i\|^2, \quad (20.4)$$

where

$\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_k]$  is the cluster prototype (means) matrix.

$\mathbf{U} = \{u_{ij}\}$  is a partition matrix,  $u_{ij} \in [0, 1]$  and  $\sum_{i=1}^K u_{ij} = 1$ .  $u_{ij}$  is the membership of data  $x_j$  in cluster  $C_i$ . In hard clustering,  $u_{ij} = 0$  or  $1$ , as is the case with well-known k-means. In fuzzy clustering,  $u_{ij}$  can have any value from  $0$  to  $1$ , as is the case with Fuzzy C-means (FCM).

### 1.20.2.3 K-means and Fuzzy C-means algorithm

The K-means algorithm is one of the best-known clustering algorithms [48]. The basic steps of K-means are as follows:

- S1.** Initialize a  $K$ -partition randomly or based on some prior knowledge, and calculate the cluster prototype matrix  $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_K]$ ;
- S2.** Assign each object in the data set to the nearest cluster  $C_w$ ;
- S3.** Recalculate the cluster prototype matrix based on the current partition:

$$\mathbf{m}_i = \frac{1}{N_i} \sum_{x_j \in C_i} \mathbf{x}_j. \quad (20.5)$$

**S4.** Repeat steps 2–3 until there is no change for each cluster.

Note that the update in step S3 requires all the data points. One way to enhance the K-means to online learning is to adjust the cluster means each time a data point is presented:

$$\mathbf{m}^{\text{new}} = \mathbf{m}^{\text{old}} + \eta(\mathbf{x} - \mathbf{m}^{\text{old}}), \quad (20.6)$$

where  $\eta$  is the learning rate.

In the signal processing community, an algorithm similar to k-means, known as the Linde-Buzo-Gray (LBG) algorithm, was suggested for vector quantization (VQ) [49] in order to signal compression. In this context, prototype vectors are called code words, which constitute a code book. VQ aims to represent the data with a reduced number of elements while minimizing information loss.

As stated above regarding fuzzy clustering, the object can hold a certain degree of membership in all clusters. This is particularly useful when the boundary between the clusters is ambiguous. The procedure of fuzzy C-means [50] is similar to that of k-means, with the exception of the following updating rules:

- S1.** Select appropriate values for  $m$ ,  $c$ , and a small positive number  $\varepsilon$ . Initialize the prototype matrix  $\mathbf{M}$  randomly. Set step variable  $t = 0$ ;
- S2.** Calculate (at  $t = 0$ ) or update (at  $t > 0$ ) the membership matrix  $\mathbf{U}$  using:

$$u_{ij}^{(t+1)} = 1 / \left( \sum_{l=1}^c (D(x_j, m_l)/D(x_j, m_l))^{2/(1-m)} \right), \quad \text{for } i = 1, \dots, c \text{ and } j = 1, \dots, N; \quad (20.7)$$

- S3.** Update the prototype matrix  $\mathbf{M}$  using:

$$\mathbf{m}_i^{(t+1)} = \left( \sum_{j=1}^N (u_{ij}^{(t+1)})^m \mathbf{x}_j \right) / \left( \sum_{j=1}^N (u_{ij}^{(t+1)})^m \right), \quad \text{for } i = 1, \dots, c \quad (20.8)$$

- S4.** Repeat steps 2–3 until  $\|\mathbf{M}^{(t+1)} - \mathbf{M}^{(t)}\| < \varepsilon$ .

FCM, like k-means, suffers from initial partition dependence, as well as noise and outliers. Yager and Filev [51] proposed the mountain method to estimate the cluster centers as an initial partition. Gath and Geva [52] addressed the initialization problem by dynamically adding cluster prototypes, which are located in the space that is not represented well by the previously generated centers. Changing the proximity distance can improve the performance of FCM in relation to outliers [53]. In another approach for mitigating the effect of noise and outliers, Keller interpreted memberships as “the compatibility of the points with the class prototype” [54] rather than as the degree of membership. This relaxes to and  $u_{ij}$  to  $u_{ij} > 0$  results in a possibilistic C-means clustering algorithm.

#### 1.20.2.4 Mixture density-based clustering

From a probabilistic perspective, as described by Xu and Wunsch [35], data objects are assumed to be generated according to several probability distributions. Data points in different clusters are generated

by different probability distributions and can be derived from different types of density functions (e.g., multivariate Gaussian or  $t$ -distribution), or from the same families but with different parameters. If the distributions are known, finding the clusters of a given data set is equivalent to estimating the parameters of several underlying models. Suppose the prior probability (also known as the mixing probability)  $P(C_i)$  for cluster  $C_i$ ,  $i = 1, \dots, K$  (here,  $K$  is assumed to be known; methods for estimating  $K$  are discussed in section II-M) and the conditional probability density  $p(\mathbf{x}|C_i, \boldsymbol{\theta}_i)$  (also known as the component density), where  $\boldsymbol{\theta}_i$  is the unknown parameter vector, are known. Then, the mixture probability density for the entire data set is expressed as:

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{i=1}^K p(\mathbf{x}|C_i, \boldsymbol{\theta}_i)P(C_i), \quad (20.9)$$

where  $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$ , and  $\sum_{i=1}^K P(C_i) = 1$ . As long as the parameter vector  $\boldsymbol{\theta}$  is decided, the posterior probability for assigning a data point to a cluster can be calculated easily with Bayes's theorem. Here, the mixtures can be constructed with any type of component, but more commonly, a multivariate Gaussian density is used due to its complete theory and analytical tractability [4, 36]. Parameter  $\boldsymbol{\theta}$  can be estimated using maximum likelihood estimation (ML).

Unfortunately, because the solutions of the likelihood equations cannot be obtained analytically in most circumstances [55], iteratively optimal approaches are required to approximate the ML estimates. Among these methods, the expectation-maximization (EM) algorithm is the most popular [56]. EM regards the data set as incomplete and divides each data point  $\mathbf{x}_j$  into two parts,  $\mathbf{x}_j = \{\mathbf{x}_j^g, \mathbf{x}_j^m\}$ , in which  $\mathbf{x}_j^g$  represents the observable features and  $\mathbf{x}_j^m = (x_{j1}^m, \dots, x_{jK}^m)$  is the missing data, where  $x_{ji}^m$  chooses a value of 1 or 0 according to whether or not  $\mathbf{x}_j$  belongs to the component  $i$ . Thus, the complete data log-likelihood is:

$$l(\boldsymbol{\theta}) = \sum_{j=1}^N \sum_{i=1}^K \mathbf{x}_{ji}^m \log[P(C_i)p(\mathbf{x}_j^g|\boldsymbol{\theta}_i)]. \quad (20.10)$$

The standard EM algorithm generates a series of parameter estimates  $\{\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^T\}$ , where  $T$  represents reaching the convergence criterion, as accomplished through the following steps:

1. Initialize  $\boldsymbol{\theta}^0$  and set  $t = 0$ ;
2. E-step: Compute the expectation of the complete data log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = E[\log p(\mathbf{x}^g, \mathbf{x}^m|\boldsymbol{\theta})|\mathbf{x}^g, \boldsymbol{\theta}^t]. \quad (20.11)$$

3. M-step: Select a new parameter estimate that maximizes the  $Q$ -function:

$$\boldsymbol{\theta}^{t+1} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t). \quad (20.12)$$

4. Increase  $t = t + 1$ ; Repeat steps 2 and 3 until the convergence condition is satisfied.

The major disadvantages of the EM algorithm are its sensitivity to the selection of initial parameters, the effect of a singular covariance matrix, the possibility of convergence to a local optimum, and the slow convergence rate [56]. Variants of EM that address these problems were discussed by McLachlan and Krishnan [56].

A valuable theoretical note pertains to the relationship between the EM algorithm and the K-means algorithm. Celeux and Govaert [57] proved that a classification EM (CEM) algorithm under a spherical Gaussian mixture is equivalent to the K-means algorithm.

MCLUST is a software implementation of a multivariate Gaussian distribution developed by Fraley and Raftery [58]. The initial guess is computed from agglomerative hierarchical clustering, along with the maximum number of clusters. Each component is parameterized by virtue of eigenvalue decomposition, represented as  $\Sigma = \lambda \boldsymbol{\Pi} \mathbf{A} \boldsymbol{\Pi}^T$ , where  $\lambda$  is a scalar,  $\boldsymbol{\Pi}$  is the orthogonal matrix of eigenvectors, and  $\mathbf{A}$  is the diagonal matrix based on the eigenvalues of  $\Sigma$ . The optimal clustering result is achieved by checking the Bayesian Information Criterion (BIC) value.

Gaussian Mixture Density Decomposition (GMDD) also is based on multivariate Gaussian densities and is designed as a recursive algorithm that sequentially estimates each component [59]. GMDD views data points that are not generated from a distribution as noise and the original data as a contaminated model. It utilizes an enhanced model-fitting estimator to iteratively construct each component from the contaminated model. AutoClass considers more families of probability distributions (e.g., Poisson and Bernoulli) for different data types [60]. AutoClass uses a Bayesian approach to find the optimal partition of the given data based on the prior probabilities.

### 1.20.2.5 Neural network-based clustering

Popular neural network-based clustering algorithms include Self-Organizing Feature Maps (SOFM) and Adaptive Resonance Theory (ART), both of which were reviewed by Xu and Wunsch [5, 35].

The objective of SOFM is to represent high-dimensional input patterns with prototype vectors that can be visualized in a usually two-dimensional lattice structure [61, 62]. Each unit in the lattice is called a neuron, and adjacent neurons are connected to each other, offering a clear map depicting how this neuron network fits itself to the input space. Input patterns are fully connected to all neurons via adaptable weights, and during the training process, neighboring input patterns are projected into the lattice, corresponding to adjacent neurons. In this sense, some authors prefer to think of SOFM as a method by which to display latent data structures in a visual way rather than via a clustering approach.

While SOFM enjoy the merits of input space density approximation and independence regarding the order of input patterns, a number of user-dependent parameters become problematic when applied in real practice. Like the *K*-means algorithm, SOFM must predefine the size of the lattice, i.e., the number of clusters, which is unknown in most circumstances. Additionally, trained SOFM may suffer from input space density misrepresentation [63], where areas of low and high pattern density may be over-represented and under-represented, respectively. Kohonen [62] reviewed a variety of SOFM variants that help to eliminate the drawbacks of basic SOFM and broaden its applications. SOFM also can be integrated with other clustering approaches (e.g., *K*-means algorithm or HC) to provide more effective, faster clustering. Su and Chang [64] and Vesanto and Alhoniemi [65] illustrated two such hybrid systems. The basic procedures of SOFM are then summarized using an algorithm, as follows:

1. Determine the topology of the SOFM. Initialize the weight vectors  $\mathbf{w}_j(0)$  for  $j = 1, \dots, K$ , randomly;
2. Present an input pattern  $\mathbf{x}$  to the network. Choose the winning node  $J$  that has the minimum Euclidean distance to  $\mathbf{x}$ ,

$$J = \arg_j \min (\|\mathbf{x} - \mathbf{w}_j\|). \quad (20.13)$$

3. Calculate the current learning rate and size of the neighborhood;
4. Update the weight vectors of all the neurons in the neighborhood of  $J$ ,

$$\mathbf{w}_j(t + 1) = \mathbf{w}_j(t) + h_{Jj}(t)(\mathbf{x} - \mathbf{w}_j(t)), \quad (20.14)$$

where  $h_{Jj}(t)$  is the neighborhood function, defined, for example, as

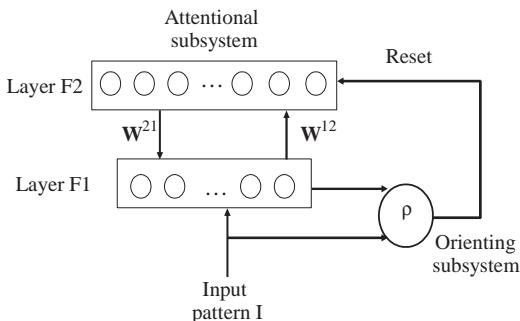
$$h_{Jj}(t) = \eta(t) \exp \left( \frac{-\|\mathbf{r}_J - \mathbf{r}_j\|^2}{2\sigma^2(t)} \right), \quad (20.15)$$

where  $\mathbf{r}_J$  and  $\mathbf{r}_j$  represent the positions of the corresponding neurons on the lattice, and  $\sigma(t)$  is the monotonically decreasing Gaussian kernel width function.

5. Repeat steps 2–4 until the change of the neuron’s position is below a pre-specified small positive number.

Adaptive resonance theory (ART) was developed by Carpenter and Grossberg [66] as a solution to the plasticity and stability dilemma. ART can learn arbitrary input patterns in a stable, fast and self-organizing way, thus overcoming the effect of learning instability that plagues many other competitive networks. ART is not, as is popularly imagined, a neural network architecture. It is a learning theory suggesting that resonance in neural circuits can trigger fast learning. As such, it subsumes a large family of current and future neural network architectures with many variants. ART1, the first member, only deals with binary input patterns [67], although it can be extended to arbitrary input patterns by a variety of coding mechanisms. ART2 extends the applications to analog input patterns [66], and ART3 introduces a new mechanism originating from elaborate biological processes to achieve a more efficient parallel search in hierarchical structures [66]. By incorporating two ART modules, which receive input patterns ( $\text{ART}_a$ ) and corresponding labels ( $\text{ART}_b$ ), respectively, with an inter-ART module, the resulting ARTMAP system can be used for supervised classifications [68]. The match-tracking strategy ensures consistency in category prediction between two ART modules by dynamically adjusting the vigilance parameter of  $\text{ART}_a$ .

As seen in Figure 20.3, ART1 consists of 2-layer nodes, the feature representation field F1 and category (cluster) representation F2. The neurons in layer F1 are activated by input patterns, while the neurons in layer F2 store cluster information. The neurons in layer F2 already being used as representations of input patterns are said to be committed; otherwise, they are uncommitted. Two layers are connected by adaptive weights  $W^{12}$  and  $W^{21}$ . The parameter  $\rho$  determines when the pattern and the expectation match, at which point weight adaption occurs.

**FIGURE 20.3**

ART1 architecture. Two layers are included in the attentional subsystem, connected via bottom-up and top-down adaptive weights. Their interactions are controlled by the orienting subsystem through a vigilance parameter. An attractive feature of this approach is that the number of clusters does not need to be specified in advance.

The ART1 algorithm can be described as follows:

1. Initialize weight matrices  $\mathbf{W}^{12}$  and  $\mathbf{W}^{21}$  as,  $\mathbf{W}^{12} = \xi / (\xi + d - 1)$ , where  $d$  is the dimensionality of the binary input  $\mathbf{x}$ ,  $\xi$  is a parameter that is larger than 1 and  $\mathbf{W}_{ji}^{21} = 1$ ;
2. For a new pattern  $\mathbf{x}$ , calculate the input from layer  $F_1$  to layer  $F_2$  as:  $T_j = \sum_{i=1}^d \mathbf{W}_{ij}^{12} x_i$ ;
3. Activate layer  $F_2$  by choosing node  $J$  with the winner-takes-all rule  $T_J = \max_j \{T_j\}$ ;  
Compare the expectation from layer  $F_2$  with the input pattern. If  $\rho \leq \frac{|\mathbf{x} \cap \mathbf{W}_J^{21}|}{|\mathbf{x}|}$ , where  $\cap$  represents the logic AND operation, go to step 5a; otherwise, go to step 5b.
4.
  - a. Update the corresponding weights for the active node as  $\mathbf{W}_J^{21} = \mathbf{x} \cap \mathbf{W}_J^{21}$  and  $\mathbf{W}_J^{12} = \frac{\xi \mathbf{W}_J^{12}}{\xi - 1 + |\mathbf{W}_J^{12}|}$ ;
  - b. Send a reset signal to disable the current active node using the orienting subsystem, and return to step 3;
5. Present another input pattern, and return to step 2 until all patterns are processed.

Note the relationship between the ART network and other clustering algorithms described in traditional and statistical language. Moore [69] used several clustering algorithms to explain the clustering behaviors of ART1 and therefore induced and proved a number of important ART1 properties, notably, its equivalence to varying  $K$ -means clustering. She also showed how to adapt these algorithms under the ART1 framework.

Fuzzy ART (FA) benefits from incorporating fuzzy set theory with ART. FA operates in a way similar to ART1 and uses the fuzzy set operators to replace the binary operators, allowing it to work

for all real data sets. FA exhibits many desirable characteristics, such as fast and stable learning and atypical pattern detection. Huang and Heileman [70] investigated and revealed more FA properties, classified as template, access, reset and the number of learning epochs. To overcome these shortcomings, Williamson [71] described Gaussian ART (GA), in which each cluster is modeled with a Gaussian distribution and represented geometrically as a hyperellipsoid. GA does not inherit the offline fast learning property of FA. Hypersphere ART (HA) [72] for hyperspherical clusters and ellipsoid ART (EA) ([73] for hyperellipsoidal clusters are proposed to offer a more efficient representation of clusters while maintaining the important properties of FA.

### 1.20.2.6 Spectral clustering

Spectral clustering has become popular because it is easy to implement yet outperforms traditional algorithms, such as k-means.

The name *spectral clustering* comes from the technique of using the spectrum (eigenvalues) of the similarity matrix to perform dimension reduction before clustering with traditional clustering methods. The similarity matrix is built from the dataset.

Given a dataset  $\mathbf{x}_1, \dots, \mathbf{x}_n$  the similarity matrix can be built as follows. Each vertex  $v_i$  represents a data point  $x_i$ . Two vertices are connected by an edge weighted by  $w_{ij}$ , which is the similarity between the two data points, usually given in the clustering problem. If no prior information about the similarity exists, a Gaussian similarity function can be used instead:  $w_{ij, i \neq j} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ , where  $\sigma$  controls the width of the neighborhood. The degree of vertex  $v_i$  is defined as  $d_i = \sum_{j=1}^n w_{ij}$ .

The adjacent matrix is given by  $\mathbf{W} = (w_{ij})_{i,j=1 \dots n'}$  and the degree matrix is defined as the diagonal matrix  $\mathbf{D}$  with the degrees  $d_1, \dots, d_n$  on the diagonal. From those, a Laplacian matrix can be built:

Unnormalized Laplacian matrix:  $\mathbf{L} = \mathbf{D} - \mathbf{W}$ .

Normalized Laplacian matrix [74]:  $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1} \mathbf{L}$ .

Normalized Laplacian matrix [75]:  $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ .

Each Laplacian matrix has a respective corresponding spectral clustering algorithm. As these algorithms are similar, we only list here the normalized spectral clustering algorithm by Ng et al. [74]; for other algorithms, see [76]:

- S1.** Compute the normalized Laplacian  $\mathbf{L}_{\text{sym}}$ .
- S2.** Compute the first  $k$  eigenvectors  $\mathbf{u}_1, \dots, \mathbf{u}_k$  of  $\mathbf{L}_{\text{sym}}$ .
- S3.** Let  $\mathbf{U} \in R^{n \times k}$  be the matrix containing the vectors  $\mathbf{u}_1, \dots, \mathbf{u}_k$  as columns.
- S4.** From the matrix  $\mathbf{T}$ , form  $\mathbf{U}$  by normalizing the rows to norm 1.
- S5.** For  $i=1, \dots, n$ , let  $\mathbf{y}_i \in R^k$  be the vector corresponding to the  $i$ th row of  $\mathbf{T}$ .
- S6.** Cluster the points  $\mathbf{y}_i$  with  $k$ -means into clusters  $C_1, \dots, C_k$ .

A special spectral clustering technique is to change the representation of the abstract data points  $\mathbf{x}_i$  into points  $\mathbf{y} \in R^k$ . Laplacian properties make this change of representation useful so that clusters can be made trivially using k-means.

Concerning computational complexity, spectral clustering is  $O(n^3)$  because of the computation of eigenvectors. One can reduce this complexity substantially by making the similarity matrix sparse and applying a special routine for the sparse matrix to compute only the first  $k$  smallest eigenvectors.

### 1.20.2.7 Subspace clustering and biclustering

As technology continues to develop, more and more increasingly complex (especially in dimension) data are being generated, and the assumption that the number of elements is larger than the number of features no longer holds true. This voids many approaches and demands new algorithms to process high-dimensional data.

In this section, we will first clarify the difference between biclustering (BC) and subspace clustering (SC) by explaining their applications and intra-cluster relationships. Then, we will review a variety of popular algorithms for both clustering techniques.

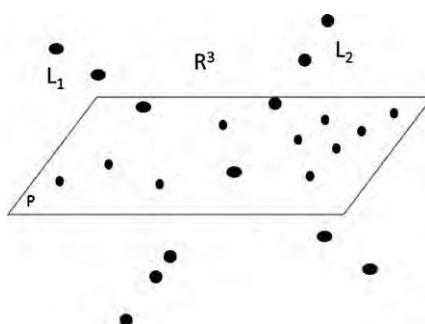
The inputs for clustering are a set of features  $F$  (called the set of genes  $V$  in BC, or the set of dimensions  $D$  in SC, row-wise) and a set of samples  $S$  (called the set of conditions  $U$  in BC, or the set of vectors  $V$  in SC, column-wise).  $F$  coupled with  $S$  forms an input matrix for the clustering problem, matrix  $E$ , whose size is  $|F| \times |S|$ , where  $|\cdot|$  denotes the cardinal of the set.

A bicluster  $(U', V')$  is defined by a subset of genes  $V'$  in  $V$  and a subset of conditions  $U'$  in  $U$ . Different clustering algorithms use different criteria to qualify a bicluster solution. In order to define a subspace cluster, let  $\{x_j \in R^d\}$  be a set of points drawn from the union of  $n$  subspaces. A subspace cluster then is defined as:

$$S_i = \{x : x = \mu_i + U_i y\}_{i=1}^n, \quad (20.16)$$

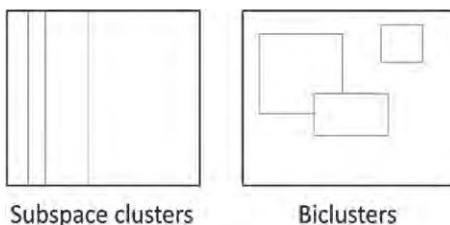
where  $\mu_i$  is a point in subspace  $S_i$ ,  $U \in R^{D \times d_i}$  is a basis for subspace  $S_i$ , and  $y \in R^{d_i}$  is a low-dimensional representation of the subspace. The goal of subspace clustering is to determine the number of subspaces  $n$ , their dimensions, and their bases and then to segment the points according to the subspaces (see Figure 20.4a).

The following section addresses the difference between biclustering and subspace clustering in partition structures, intra-cluster relationships, and application datasets.



**FIGURE 20.4a**

Three subspaces—two lines and a plane.

**FIGURE 20.4b**

Subspace clusters and biclusters.

### **1.20.2.7.1 Comparing subspace clustering and biclustering**

#### **1.20.2.7.1.1 Partition structure**

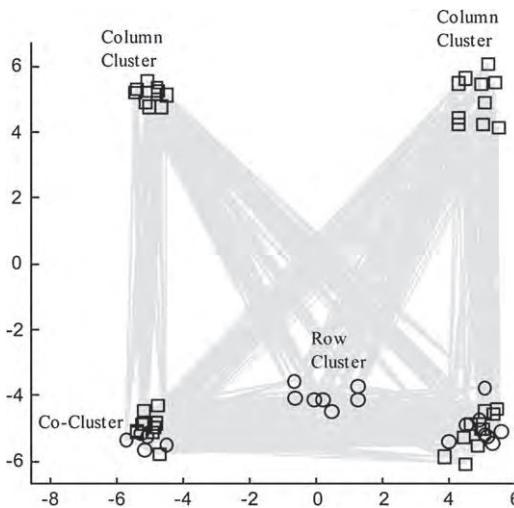
SC is a 1-dimensional matrix clustering that groups the elements in  $V$  into groups that form a subspace in the  $D$ -dimension. A permutation is performed on matrix  $E$  after clustering produces consecutive samples that belong to the same group (see Figure 20.4b). On the other hand, BC provides a new way to look at the data structure. It is a 2-dimensional clustering, also called co-clustering, in which a bicluster of  $E$  is a submatrix of  $E$  formed by a subset of  $F$  and a subset of  $S$ . Performing a permutation on matrix  $E$  after biclustering reveals that the biclusters form small rectangles inside the big rectangle  $E$ . Therefore, biclustering and subspace clustering produce very different results.

Biclustering, first used by Cheng and Church [77] in the bioinformatics community, addresses this problem by performing clustering simultaneously on both the row (gene) and column (sample) dimensions instead of clustering these two dimensions separately [31, 78]. In essence, biclustering can be regarded as a combination of clustering and automatic feature selection if one dimension (e.g., column) is treated as data objects and the other dimension (e.g., row) as description features. This task becomes particularly challenging without ground truth.

Note that the feature selection in biclustering is different from the feature selection usually considered in subspace clustering in that biclustering selects different subsets of features for different clusters of data objects, while standard feature selection chooses a subset of features from the candidate pool for all data objects. As such, biclustering can identify local relationships between subsets of genes and subsets of samples or conditions. Biclustering indicates gene groups that display similar patterns across a set of conditions, which is important for gene functional annotation and co-regulated gene identification [79] or in terms of gene groups that are related to certain cancer types (for cancer classification discovery and diagnosis) [78]. In contrast, subspace clustering focuses on uncovering global relationships between genes and samples or conditions. In fields other than bioinformatics, biclustering is also known as co-clustering or block clustering, among other names.

#### **1.20.2.7.1.2 Intra-cluster relationship**

One way in which BC and SC differ is in the data relationships between elements in the same cluster. In BC, data in the same clusters are homogeneous, i.e., the bicluster in BC must have some kind of homogeneity across the rows or columns of the sub-matrix, meaning that the rows or columns must be the same or have coherent additive or multiplicative values [31]. On the contrary, data in the same SC cluster can be very different, but as long as the samples lie in the same subspace, they are grouped into the same cluster.

**FIGURE 20.5**

Normal cluster (row cluster, column cluster) vs. bicluster (co-cluster) (From [48]).

Figure 20.5 illustrates the relationships between elements in the same bicluster. Matrix  $\mathbf{E}$  is formed from the relationship between points in  $R^2$ , which includes 20 points for the row objects (circles) and 40 points for the column objects (squares). Matrix  $\mathbf{E}$  has the entry  $E_{i,j} = \text{Euclidian distance between the } i\text{th point among the row objects and the } j\text{th point among the column objects}$ . Points are chosen so that they form three clusters for row objects and four clusters for column objects. The most remarkable observation in this arrangement is that the points in row and column objects form two co-clusters (the portion of the figure that circles and squares cohabit). The length of a line connecting a circle to a square is the value of the entry in matrix  $\mathbf{E}$ . Therefore, the subset of rows and columns where points cohabit in matrix  $\mathbf{E}$  displays homogeneity in that the entries in this sub-matrix are much smaller than those in the other sub-matrix.

### 1.20.2.7.1.3 Dataset

Biclustering can be applied comfortably to relational datasets, those in which some relationship exists between the rows and columns. In other words, datasets with relational characteristics are very good for biclustering. Below are some examples of biclustering data.

- *Gene expression:* Each sample comprises a gene, and each gene is profiled differently in each sample. See [31] for more information about gene expression.
- *Document classification:* Each row is a document, and each column is a word in a document. An entry in matrix  $\mathbf{E}$  is the occurrence of a word in a document.
- *Netflix database:* Rows are the number of movies, and columns are the number of people who rent movies from Netflix. A matrix  $\mathbf{E}$  entry measures a user's satisfaction with a specific movie.

Unlike BC, SC is suitable for space datasets, meaning that if the positions of all the samples in the dataset can be plotted, they all will form subspaces. The problem is that those subspaces are not easy to

visualize and typically are not independent. The most applicable form of subspace clustering is motion segmentation, which can be found in [80].

### 1.20.2.7.2 Subspace clustering algorithms

#### 1.20.2.7.2.1 Matrix factorization algorithms

Algebraic clustering algorithms are based on linear algebra matrix factorization. As a pure math theory development, the algorithms require independent subspaces and noise-free data.

Let  $\mathbf{X}$  be the matrix representing the dataset and  $[\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n] = \mathbf{X}\Gamma$  be the sorted  $\mathbf{X}$  according to the  $n$  subspaces, where subspace  $\mathbf{S}_i \in R^{D \times N_i}$  is the matrix containing  $N_i$  points  $\Gamma \in R^{n \times n}$  and is an unknown permutation matrix.  $\mathbf{S}_i$  can be factorized as  $\mathbf{s}_i = \mathbf{U}_i \mathbf{Y}_i$ , where  $\mathbf{U}_i \in R^{D \times d_i}$  is a basis of subspace  $i$  and  $\mathbf{Y} \in R^{d_i \times N_i}$  is the low-dimensional representation of  $\mathbf{S}_i$ . Substitute  $\mathbf{S}_i$  into  $\mathbf{X}\Gamma$  as follows:

$$\mathbf{X}\Gamma = [\mathbf{U}_1, \mathbf{U}_2, \dots, \mathbf{U}_n] \begin{bmatrix} \mathbf{Y}_1 & & \\ & \mathbf{Y}_2 & \\ & & \dots \\ & & & \mathbf{Y}_n \end{bmatrix} = \mathbf{U}\mathbf{Y}, \quad (20.17)$$

where  $\mathbf{U} \in R^{D \times r}$  and  $\mathbf{Y} \in R^{r \times N}$  with  $r = \text{rank}(X) = \sum_{i=1}^n d_i$ .

Algebraic clustering next involves trying to find matrix  $\Gamma$  such that  $\mathbf{X}\Gamma$  can be factorized to  $\mathbf{U}$  and  $\mathbf{Y}$ . This can be achieved using the linear algebra manipulation over  $\mathbf{X}$  [81,82].

#### 1.20.2.7.2.2 Iterative methods

Iterative algorithms are based on a two-step iteration. Given an initial clustering, we can fit each cluster into a subspace using PCA, and then assign a data point to each cluster based on its distance from the subspace. Iteration between the two steps occurs until convergence is achieved. These algorithms work in a way similar to k-means clustering. Algorithms such as the k-means projective algorithm [83] are based on this idea. In general, a  $K$ -subspace iterative method works by letting  $w_{ij}$  be the membership of data point  $j$  in cluster  $j$ ;  $w_{ij} = 1$  if  $j$  belongs to subspace  $i$ , and 0 otherwise. Minimizing the square distance allows us to find,  $\{\mu_i\}$ ,  $\{\mathbf{U}_i\}$ , and  $\{\mathbf{y}_j\}$ , as well as membership  $w_{ij}$ :

$$\min_{\{\mu_i\}, \{\mathbf{U}_i\}, \{\mathbf{y}_i\}, \{w_{ij}\}} \sum_{j=1}^N \sum_{i=1}^n w_{ij} \|\mathbf{x}_j - \mu_i - \mathbf{U}_i \mathbf{y}_j\|^2 \quad (20.18)$$

$$\text{s.t } w_{ij} \in \{0, 1\} \text{ and } \sum_{i=1}^n w_{ij} = 1.$$

We begin solving Eq. 20.18 by initializing  $\{\mu_i\}$ ,  $\{\mathbf{U}_i\}$  and  $\{\mathbf{y}_j\}$ , which allows us to find the optimal  $w_{ij}$  using:

$$w_{ij} = \begin{cases} 1 & \text{if } i = \text{argmin} \|\mathbf{x}_j - \mu_k - \mathbf{U}_k \mathbf{y}_j\| \\ 0 & \text{otherwise} \end{cases}. \quad (20.19)$$

Once  $w_{ij}$  is known,  $\mu_i$ ,  $\mathbf{U}_i$ , and  $\mathbf{y}_j$  can be recomputed using PCA for each cluster. This simple algorithm guarantees convergence but requires a good initialization and cannot handle outliers.

### 1.20.2.7.2.3 Statistical method

We start by building a generative model for the data. As in [84], data in a single subspace  $S$  can be modeled as a Gaussian:

$$\mathbf{x} = \boldsymbol{\mu} + \mathbf{U}\mathbf{y} + \mathbf{e}, \quad (20.20)$$

where  $\mathbf{y}$  is Gaussian with a mean of zero and a unit variance of  $\sim \mathcal{N}(0, \mathbf{I})$  and  $\mathbf{e}$  is also Gaussian with a mean of zero and a unit variance of  $\sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ .

$\mathbf{x}$  is then normally distributed  $\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{U}\mathbf{U}^T + \sigma^2 \mathbf{I})$ .

Mixture probabilistic PCA extends the single Gaussian by regarding each subspace as Gaussian and the dataset as a mixture of Gaussians, which therefore can be clustered using an EM algorithm. Let  $p(x)$  be the Gaussian mixture of the dataset:

$$p(\mathbf{x}) = \sum_{i=1}^n \pi_i \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \mathbf{U}_i \mathbf{U}_i^T + \sigma_i^2 \mathbf{I}), \quad (20.21)$$

where  $\pi_i$  is the weighted mixture.

Now we can iterate between steps E and M to compute the Gaussian parameters  $(\boldsymbol{\mu}_i, \mathbf{U}_i, \sigma_i, \pi_i)$  according to [80].

In step E:

$$R_{ij} = \frac{p(\mathbf{x}_j | i) \pi_i}{p(\mathbf{x}_j)}. \quad (20.22)$$

In step M:

$$\pi_i = \frac{1}{N} \sum_{j=1}^N R_{ij}, \quad (20.23)$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^N R_{ij} \mathbf{x}_j}{\sum_{j=1}^N R_{ij}}. \quad (20.24)$$

$\mathbf{U}_i$  and noise variance  $\sigma_i^2$  are determined from the SVD of the local responsibility weighted covariance matrix:

$$\mathbf{s}_i = \frac{1}{\pi_i N} \sum_{j=1}^N R_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T. \quad (20.25)$$

These two steps are repeated until convergence occurs.

### 1.20.2.7.2.4 RANSAC method

RANSAC [85] is a powerful statistical method for fitting a model given a dataset that is corrupted by noise and outliers. This technique is used widely in computer vision when solving the fundamental matrix in a stereo correspondence problem. The basic idea of RANSAC is to pick  $d$  points from the dataset, enough to estimate and fit model, and then compute the residue of each point to this model,

choosing only points whose residue is lower than a certain threshold as inliers. These steps then are repeated with other randomly selected  $d$  points until enough samples are drawn. RANSAC can be applied in subspace clustering [86] by finding and situating one subspace as the model and then marking certain points in that subspace as inliers and the rest as outliers. After the first subspace is discovered, the inliers are removed from the dataset, and another RANSAC is repeated to discover the next subspace. This procedure is repeated until all the subspaces are recovered. PCA is used to compute the basis of the subspace for each set of inliers.

#### 1.20.2.7.2.5 Sparse subspace clustering

Sparse representation and compressive sensing are emerging areas in signal and image processing [87]. Due to the development of several efficient L1-norm solvers, sparse representation has been applied in many research areas, especially in clustering. The idea behind subspace clustering using compressed sensing [88] is that a point  $\mathbf{x}_j$  will have sparse representation in the sensing matrix  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_{j-1}, \mathbf{x}_{j+1}, \dots, \mathbf{x}_N]$ , which means that the nonzero coefficients in sparse representation correspond to the points that lie in the same subspace as  $\mathbf{y}_j$ . After applying sparse representation to all the points, we build a matrix of sparse representation coefficients,  $C_{N \times N}$ . Subspace segmentation can be achieved by applying K-means to a subset of eigenvectors of the Laplacian of  $C$ .

Let  $w_{ij}$  be the linear combination of  $\mathbf{x}_j$ . If the data are noise-free,  $w_{ij}$  can be found using l1-optimization:

$$\begin{aligned} & \min \sum_{i \neq j} |w_{ij}| \\ & \text{s.t. } \mathbf{x}_j = \sum_{i \neq j} w_{ij} \mathbf{x}_i. \end{aligned} \quad (20.26)$$

If the data contain noise, a relaxation is added into the optimization:

$$\min_{w_{ij}} \sum_{i \neq j} |w_{ij}| + \mu \|\mathbf{x}_j - \sum_{i \neq j} w_{ij} \mathbf{x}_i\|^2, \quad (20.27)$$

where  $\mu$  is a positive relaxation coefficient.

In the case of outliers, sparse vector outliers are added to the sparse representation of

$$\mathbf{x}_j = \sum_{k \neq j} w_{kj} \mathbf{x}_k + \mathbf{e}_j. \quad (20.28)$$

The l1-norm optimization becomes:

$$\min_{w_{ij}, \mathbf{e}_j} \sum_{i \neq j} |w_{ij}| + \mu \left\| \mathbf{x}_j - \sum_{i \neq j} w_{ij} \mathbf{x}_i - \mathbf{e}_j \right\|^2. \quad (20.29)$$

The sparse representation of each data point then is used as the featured vector for spectral clustering [76].

In [89], Vidal extended the solution to disjointed subspaces and derived theoretical bounds relating the principal angles between the subspace and the distribution of the data points across all of the subspaces.

### 1.20.2.7.3 Biclustering algorithms

Let  $\mathbf{I} \subset \mathbf{G}$  and  $\mathbf{J} \subset \mathbf{S}$  be subsets of the rows and columns for the gene expression data matrix  $\mathbf{E}$ ;  $\mathbf{E}_{IJ} = (\mathbf{I}, \mathbf{J})$  is then the submatrix with rows  $\mathbf{I}$  and columns  $\mathbf{J}$ . A bicluster corresponds to such a submatrix that exhibits certain homogeneity. The goal of a biclustering algorithm, then, is to identify a set of biclusters with pairs of row and column subsets. The complexity of the biclustering problem has been shown to be NP-complete [31], which leads to many heuristics falling into the five major categories summarized in Table 20.2, according to Madeira and Oliveira.

The first biclustering research was presented by Cheng and Church [77]. The mean squared residue is used to measure the coherence of the rows and columns in a bicluster, which is defined as:

$$H(\mathbf{I}, \mathbf{J}) = \frac{1}{|\mathbf{I}||\mathbf{J}|} \sum_{i \in \mathbf{I}, j \in \mathbf{J}} (e_{ij} - e_{iJ} - e_{IJ} + e_{IJ})^2, \quad (20.30)$$

**Table 20.2** Biclustering Algorithms

Algorithm category	Description	Examples
Combination of iterative row and column clustering	Also known as two-way clustering. Uses existing clustering algorithms for the row and column dimensions separately. Combines the one-way results to produce biclusters.	Interrelated two-way clustering (ITWC) [90]; Double conjugated clustering (DCC) [91]; Coupled two-way clustering (CTWC) [92]; Fuzzy adaptive subspace iteration-based two-way clustering (FASIC) [93]
Divide and conquer	Divides the original problem into a set of smaller subproblems. Solves the subproblems and combines the solutions to obtain the final solution to the original problem.	Block clustering and its variants [94, 95]
Greedy iterative search	Iteratively inserts or removes rows or columns from the biclusters, guided with some criterion function.	$\delta$ -biclusters [77]; Flexible overlapped biclustering (FLOC) [96]; xMOTIF [95]; Order preserving submatrices (OPSM) [97]
Exhaustive bicluster enumeration	Performs exhaustive enumeration but with some restrictions on the size of biclusters.	Statistical algorithmic method for bicluster analysis (SAMBA) [98]; pclusters [99]
Distribution parameter identification	Considers underlying statistical models for biclusters. Estimates model parameters that best fit the data.	Probabilistic relational models [99]; Plaid models [100]

where  $e_{iJ}$  is the mean of the  $i$ th row,  $e_{Ij}$  is the mean of the  $j$ th column, and  $e_{IJ}$  is the mean of the bicluster. A submatrix  $\mathbf{E}_{IJ}$  is called a  $\delta$ -bicluster if  $\mathbf{H}(\mathbf{I}, \mathbf{J}) \leq \delta$  for some  $\delta \geq 0$ . Possible ways to find the largest  $\delta$ -biclusters, which should also have relatively high row variance, include the evolutionary algorithm [101], multi-objective particle swarm optimization [102], and other greedy iterative search approaches.

Another popular biclustering algorithm is the interrelated two-way clustering (ITWC) algorithm. ITWC generates biclusters by combining clustering results from each dimension of the data matrix in an iterative way. Within each iteration, a set of gene clusters first is created by standard clustering algorithms, e.g.,  $K$ -means or SOFM, followed by the independent clustering of the sample dimension based on each gene cluster. The clustering results from the previous steps then are combined, and heterogeneous groups, which are pairs of columns that do not share gene features used for clustering, are identified. Finally, the genes are sorted in descending order of the cosine distance, and only the first one-third of sorted genes is kept so that a reduced gene set is obtained for each heterogeneous group. The above steps then are repeated using the reduced gene set until some stopping conditions are satisfied. Other two-way clustering algorithms include the coupled two-way clustering (CTWC) algorithm [92], which uses hierarchical clustering with the Euclidean distance, the double conjugated clustering (DCC) algorithm [91], which uses SOFM with the dot product for similarity, and the fuzzy adaptive subspace iteration-based two-way clustering (FASIC) algorithm, which is integrated with the concept of fuzzy membership.

Xu and Wunsch [37] developed a new two-way clustering technique called BARTMAP, the structure of which consists of two Fuzzy ART modules, ARTa and ARTb, which communicate through the inter-ART module. The inputs of the ARTa module are samples, and the inputs of ARTb are genes. The goal of BARTMAP is to integrate the clustering results on the dimensions of columns and rows of the data matrix to create biclusters that capture the local relationships between genes and samples. More concretely, the first step of BARTMAP is to create a set of  $K_g$  gene clusters  $\mathbf{G}_i, i = 1, \dots, K_g$ , for  $N$  genes by using the ARTb module. Then, upon the presentation of a new sample in ARTa, the candidate sample cluster that is eligible to represent this sample is decided based on the winner take all rule. If this candidate cluster corresponds to an uncommitted neuron, a new one-element cluster is created. However, if a committed neuron is picked, the weights of this winning neuron are only updated if it displays behaviors similar to those of other members in the cluster formed in the ARTb module, as measured by the Pearson correlation coefficient.

Lazzeroni and Owen [100] developed a plaid model, treating the matrix as the sum of additive layers. This model was extended further to combine external grouping information and to generate biclusters of profiles of repeated measures [103]. Gu and Liu [104] proposed a Bayesian biclustering model with a Gibbs sampling procedure for statistical inference. McAllister et al. [105] considered modeling biclustering as either a network flow problem or a traveling salesman problem. A systematic comparison of five greedy search-based biclustering algorithms, together with a reference method (Bimax), was presented by Prelić et al. [106].

CTWC uses hierarchical clustering with the Euclidean distance, while DCC relies on self-organizing maps with the dot product for similarity [91]. FASIC adopts the concept of fuzzy set theory and applies the fuzzy-adaptive subspace iteration algorithm to generate gene clusters under a progressive clustering framework. Those clusters then are scored to generate  $p$ -values indicating the significance of the differential expression [93]. Tchagang et al. [107] demonstrated the effectiveness of biclustering

for detecting and diagnosing ovarian cancer. Mitra et al. [108] showed a biclustering application for microRNA expression profile-based cancer classification of multiple cancer types, including melanoma, ovarian, renal, colon, and leukemia.

### 1.20.2.8 Deep learning clustering

All of the clustering algorithms presented so far have not required any explicit feedback, i.e., they all engage in unsupervised learning. In contrast to this is supervised learning, in which the program learns to map functions from input to output by training data with labels. However, in practice, the available data will include some labeled samples and a large collection of unlabeled samples. Algorithms that work with such a mix of data are said to engage in semi-supervised learning; one of the most powerful semi-supervised clustering techniques is deep learning.

As human-computer interaction (HCI) continues to develop, more and more images and data will be created. A large and rapidly-growing demand exists for the ability to search, organize and recognize media content. The most popular methods are based on hand-crafted features and therefore are very time and resource consuming [109, 110]. On the other hand, researchers such as Andrew Ng [34] have suggested the novel idea of achieving best performance in classification using unsupervised methods. This idea combines both supervised and unsupervised learning to accomplish recognition tasks.

The central idea behind unsupervised feature learning is the sparse coding neural network [111]. Sparse coding retains only the characteristic structure of the image at the input of the network. In short, the goal of sparse coding is as follows:

*Input:* given images  $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m \in R^{n \times n}$ .

*Output:* dictionary bases  $\phi_1, \phi_2, \dots, \phi_k \in R^{n \times n}$ .

So that in every image  $\mathbf{X} = \sum_{j=1}^n a_j \phi_j$ , most  $a_j$ 's are 0.

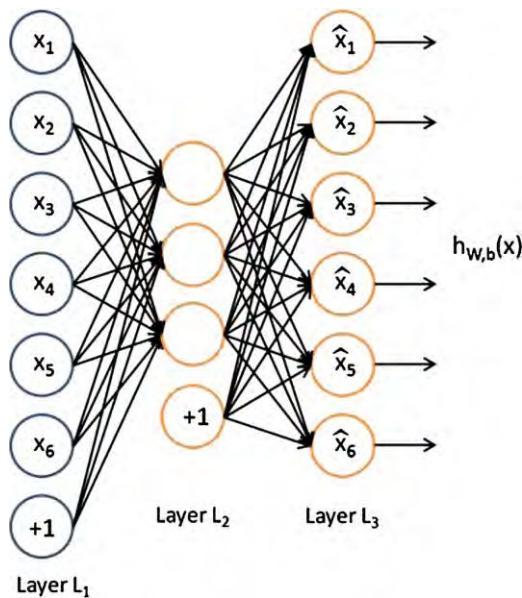
#### 1.20.2.8.1 Sparse auto-coding network

Sparse auto-coding uses a 3-layer neural network with input, hidden and output layers. The terminology *auto sparse encoder* implies something about this network's structure. The value at the output layer is exactly the same as at the input layer. The input and output layers have the same number of neurons, which differs from the hidden layer, in which the interesting structure of the data is retained (see Figure 20.6).

When training the sparse auto-coding network, a sparse constraint is imposed on the hidden layer in order to force the network to reflect the structure of the input. We denote the average activation of hidden unit  $j$  (over the entire training set) as  $\rho_j$  and the number of neurons in the hidden layer as  $s$ . We will enforce  $\rho_j = \rho$ , where  $\rho$  is a sparsity parameter, by adding the penalty term  $\sum_{j=1}^s KL(\rho \parallel \hat{\rho}_j)$  in the objective function of backpropagation [112]:

$$J_{\text{sparse}}(\mathbf{W}, \mathbf{b}) = J(\mathbf{W}, \mathbf{b}) + \beta \sum_{j=1}^s KL(\rho \parallel \hat{\rho}_j), \quad (20.31)$$

where  $\mathbf{W}$  represents the weight of the connections in the network,  $J(\mathbf{W}, \mathbf{b})$  is the sum of the square-error cost function of training the neural network, and KL refers to Kullback-Leiber [113], which measures

**FIGURE 20.6**

Example structure of a sparse auto-coding network.

the difference in the distribution of  $\rho$  and  $\rho_j$ .

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}. \quad (20.32)$$

To clarify the result of training the sparse auto-encoder network, we investigate the hidden layer:

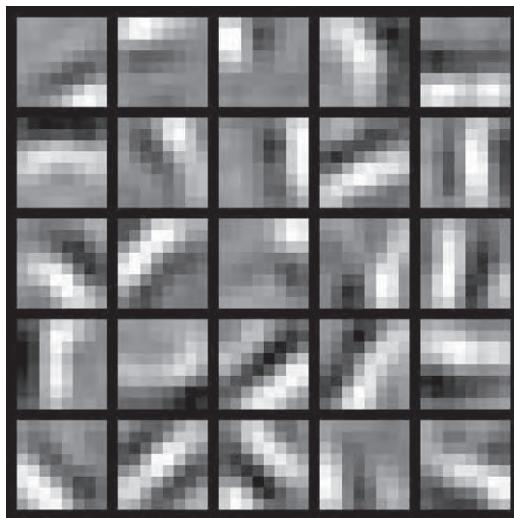
$$a_i = f \left( \sum_{j=1}^n W_{ij} x_j \right), \quad (20.33)$$

where  $a_i$  is the output of neuron  $i$  in the hidden layer, and  $w_{ij}$  is the weight of the connection from input  $x_j$  to neuron  $i$ , by which we can determine the maximum value of  $a_i$ . To avoid arriving at a trivial solution, we force  $\|\mathbf{x}\| \leq 1$  (by normalization).  $a_i$  obtains the maximum when:

$$x_{j \max} = \frac{W_{ij}}{\sum_{j=1}^n W_{ij}^2}, \quad (20.34)$$

which indicates that the image with the structure  $W_{ij}$  will be fired maximally on  $a_i$ . Due to the properties of the backpropagation algorithm, the weights connecting the hidden layer to the output layer are updated separately from those connecting the input layer to the hidden layer; those weights are not considered here because they do not have a specific meaning in sparse representation.

The result of training 200 images in a  $5 \times 5$  grid is shown in Figure 20.7.

**FIGURE 20.7**

Each cell represents features detected in the training data and stored in the hidden unit. The features detected here are edges at different positions and orientations (From Andrew Ng; Sparse autoencoder. CS 294A Machine Learning lecture notes).

#### 1.20.2.8.2 *Independent subspace analysis (ISA) network*

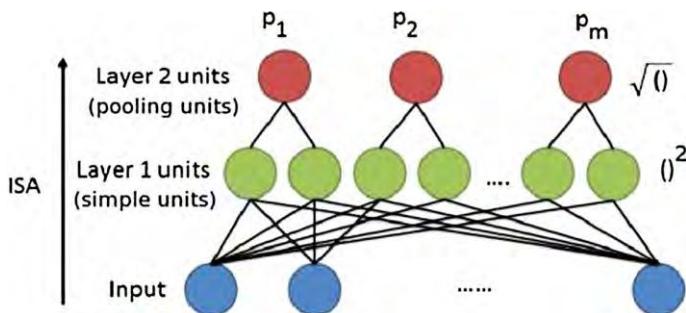
Another neural network that can be used to auto-learn features is the 3-layer independent subspace analysis network (ISA) [34]. The hidden layer has square nonlinearity, and the second layer has square root nonlinearity. The hidden layer is connected to the output layer via a few links of adjacent neighbors and represents the subspace structure of the neural network. The structure of an ISA network is illustrated in Figure 20.8a.

The output of the ISA network is:

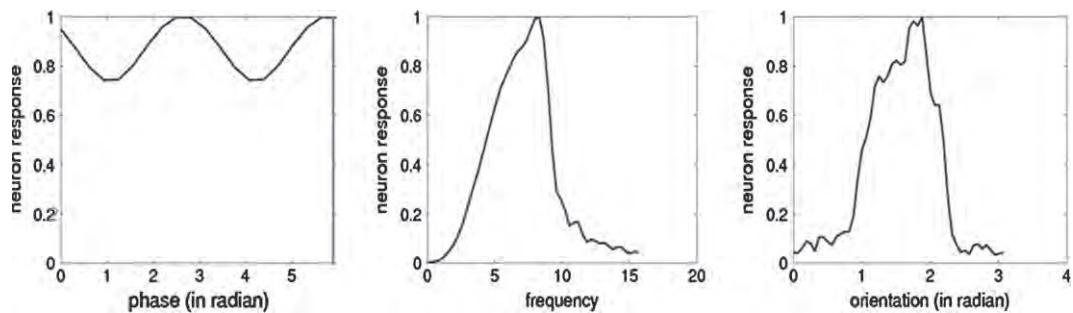
$$p_i = (x; \mathbf{W}, \mathbf{V}) = \sqrt{\sum_{k=1}^m \mathbf{V}_{ik} \left( \sum_{j=1}^n \mathbf{W}_{kj} x_j \right)^2}. \quad (20.35)$$

Similar to the sparse auto-encoding network, the ISA network learns the structure of the image by finding the sparse representation in the hidden layer by solving:

$$\begin{aligned} \min_{\mathbf{W}} \sum_{t=1}^T \sum_{i=1}^m p_i(x^t; \mathbf{W}, \mathbf{V}), \\ \text{subject } \mathbf{W}\mathbf{W}^T = \mathbf{I}, \end{aligned} \quad (20.36)$$

**FIGURE 20.8a**

ISA structure (from [34]).

**FIGURE 20.8b**

ISA response with image changes. These plots show that the ISA network is robust to translation (phase) and selective to frequency and rotation changes.

where  $\{\mathbf{x}^t\}$  are whitened input examples,  $\mathbf{W} \in R^{k \times n}$  represents the weights connecting the input layer to the hidden layer, and  $\mathbf{V} \in R^{m \times k}$  represents the weights connecting the hidden layer to the output layer.  $\mathbf{V}$  often is fixed in order to force the sparse representation of  $\mathbf{x}$ , and the orthogonal constraint ensures that feature learning is diverse (see Figure 20.8b).

The features learned by the ISA are invariant to translation, frequency and rotation and therefore are suitable for recognition tasks [114]. Many experiments have confirmed that ISA performs much better than sparse coding. However, to further extract features from large images or videos, those networks that auto-learn features can be stacked to form hierarchical networks.

After training the network, any image can be fed into it, and the output at both layers can be used as features for clustering. To process videos, features are extracted for each frame of the video and then quantized into visual words [115] using K-means clustering. A video is represented as a frequency histogram over those visual words by assigning features to the closest vocabulary word.

Based on unsupervised feature learning, several clustering applications have been developed, including speech processing. Lee et al. [116], action recognition [34], and character recognition [117].

### 1.20.3 Clustering validation

As Figure 20.1 depicted, the indispensable step in clustering is validation. Clustering validation is used to objectively and quantitatively evaluate the obtained clustering structure. There are three validation index categories: external indices, internal indices and relative indices [7].

Given a data set  $X$  and a clustering structure  $C$  obtained from a certain clustering algorithm on  $X$ , external criteria were used to compare  $C$  to a pre-specified structure containing a priori information about the data structure  $X$ . In contrast, internal criteria were used to evaluate the clustering structure exclusively from  $X$ . Relative criteria were used to compare  $C$  with other clustering structures from different clustering algorithms or from the same clustering algorithm with different parameters on  $X$ . The primary interest regarding relative criteria is how to estimate the optimal number of clusters. Xu and Wunsch [5] discussed those criteria in detail.

Another problem closely related to cluster validation is the assessment of clustering tendencies, in which the algorithm tries to determine whether clusters are present one step prior to actual clustering. Most assessment techniques utilize data visualization to present a view of the data structure. Yang et al. [96] proposed an interactive hierarchical display by which to look at the data at various levels. One of the most popular approaches comes from [118], in which the data are reordered according to their similarity, resulting in an intensity image display. This method has produced several derivatives, such as iVAT and reVAT [119, 120].

---

### 1.20.4 Applications

#### 1.20.4.1 Image segmentation

One of the broadest applications of clustering in image processing is image segmentation. The simplest approach to segmenting an image is using hierarchical clustering [121]. K-means can also be applied for segmentation. In many cases in which segments are not connected and can be very widely scattered, the feature vector in k-means will include pixel coordinates instead of just pixel colors. Segmentation also can be regarded as a graph cut problem in which spectral clustering has been developed, yielding very good results [74, 75].

#### 1.20.4.2 MRI images

Magnetic resonance imaging (MRI) provides a visualization of the internal structures of objects and living organisms. MRI images have better contrast than computerized tomography; therefore, most medical image segmentation research uses MRI images. Segmenting an MRI image is a key task in many medical applications, such as surgical planning and abnormality detection. MRI segmentation aims to partition an input image into significant anatomical areas, each of which is uniform according to certain image properties.

MRI segmentation can be formulated as a clustering problem in which a set of feature vectors obtained through transformation image measurements and pixel positions is grouped into a number of structures [14]. There are significantly fewer clusters than intensity levels in the original image, so this clustering process can be regarded as removing redundant information from the MRI.

Balaifar [122] used FCM for MRI segmentation. As FCM only considers image intensity, its results are not acceptable when working with noisy images. Many algorithms have been introduced to make FCM robust against noise [123, 124].

Another approach to image segmentation is to model each of the segments as a Gaussian distribution and apply mixture density-based clustering to the MRI images [125].

#### 1.20.4.3 Vision-guided robot

One of the main problems in robotics is enabling the robot to guide itself autonomously through the environment. A vision-guided robot equipped with a camera can attain that goal by clustering the images captured from the environment. Martinez and Vitria [126] found a successful solution, developing a normal mixture model-based (NMM) algorithm to cluster the images of the surrounding environment. This algorithm views images as column vectors whose dimensions are the intensities of the image pixels; this is the first clustering method projected to lower the dimensional space, which it accomplishes using principal component analysis. EM clustering then is applied to learn all the models in the mixtures in the PCA space. Each model corresponds to a different area of the environment. In order to improve the results, they employed a Genetic Algorithm that uses a population of mixture models rather than a single mixture. The resulting algorithm has a higher successful recognition rate than PCA or Fisher discriminant analysis.

#### 1.20.4.4 Motion clustering

One application of subspace clustering is motion clustering, in which rigid objects in a given video are segmented based on their motions in the scene. Features in the same object will have the same motion pattern with 3 degrees of freedom in space. Under an affine projection model, this motion was positioned in an affine subspace of 3 dimensions at most, i.e., in a linear 4-dimensional subspace. More specifically, suppose over  $F$  frames of the video, we successfully extract and track a set of  $N$  feature points  $\{x_{fi} \in R^2\}_{i=1\dots N, f=1\dots F}$ . The set of  $\{x_{fi} \in R^2\}, f = 1\dots F$  forms the trajectory; by stacking them, we obtain a data point  $y_i$  in  $2F$ -dimension. As noted previously, data point lies in a subspace of, at most, 3 dimensions. The problem of motion segmentation then becomes the problem of clustering those data points  $y_i$  in  $R^{2F}$ , which is the problem of subspace clustering. Vidal [127] contributed a significant amount of research exploring the use of different subspace clustering techniques on this application. Experiments were tested on the Hopkins 155 dataset, showing that sparse subspace clustering outperforms other subspace clustering algorithms.

---

### 1.20.5 Open issues and problems

Most clustering algorithms are available through software packages or are easy to implement. While offering substantial merits, they also have many disadvantages that make them inappropriate for some

signal and image processing needs. For example, the complexity of agglomerative hierarchical clustering is  $O(N^2)$ , which is not suitable for large-scale data clustering.

The typical clustering challenges are summarized as follows:

*Scalability:* Scalability in both time and memory is the most important challenge faced by clustering algorithms because of the ever-increasing amount of data from all applications. For example, some applications require the clustering of billions of images or documents available on the Internet. It is impractical for a clustering algorithm to run for several months to complete one run, especially if there is an extraordinary need for memory. Therefore, linear or near-linear complexity is very desirable.

*Dimensionality:* The algorithm should be able to handle data with many features, even more than the number of objects in the data set. Identifying relevant features and capturing the intrinsic dimension is important for describing the real data structure.

*Robustness:* Noise and outliers are always present in the data, so the clustering algorithm must be able to detect and remove them.

*Cluster number:* A classical clustering problem is to correctly identify the numbers of clusters, as many current algorithms require the user to input this parameter. This is difficult because a lack of prior knowledge or an incorrect estimation of the true number of clusters will prevent the clustering process from uncovering the real clustering structures.

*Parameter reliance:* Many algorithms, not just limited to clustering algorithms, require users to specify 3 or more parameters, which can decrease a user's confidence in utilizing the algorithm unless he knows the problem well enough to select these values within a sensible range. Thus, a good clustering algorithm should depend on only one parameter.

*Cluster shape:* Clustering algorithms should have the ability to detect all possible cluster shapes, rather than being confined to some particular shape, such as hyperspheres or hyper-rectangles.

*Order insensitivity:* One of the most important properties of a clustering algorithm, especially when being applied to large-scale data, is its online or incremental ability. However, the result of clustering may depend on the order of the presentation of input patterns. Decreasing the effects of the order of input patterns is a challenging problem.

*Good visualization:* Good visualization will help users understand the resulting clusters, allowing them to interpret and extract useful information to solve their problems.

---

## 1.20.6 Conclusion

This chapter has offered a review of emerging clustering algorithms with broad applications in computer science, machine learning, and statistics. While some of the clustering techniques, such as hierarchical, mixture, and fuzzy clustering, are suitable for typical data, biclustering and subspace clustering are very effective for high-dimensional datasets. To automatically learn features from a dataset, unsupervised feature learning clustering techniques may offer superior performance to hand-crafted feature selection.

---

## Glossary

ART (Adaptive Resonance Theory)	an unsupervised learning model in which learning occurs as a result of interaction between top-down observer expectations and bottom-up sensory information through a reset module, resulting in a theory that learning is mediated by resonance in a neural circuit
Affinity matrix	in the context of spectral clustering, an $N \times N$ matrix measures the affinity between data points. The measuring function is usually a Gaussian function
BIC (Bayesian Information Criterion)	a criterion for model selection when fitting data to a model, used to mitigate the problem of over fitting
Biclustering	a clustering approach that clusters the matrix of a data set in both columns and rows. Many clustering algorithms tessellate the data space. Biclustering selects data that is common to two subspaces, and can be considered as an unsupervised version of heteroassociative learning
Cluster	a group of objects that share some common property
Clustering	the process of dividing objects into clusters
Cluster validation	a post-processing step in clustering that objectively and qualitatively evaluates the resulting clusters
Competitive learning neuron network	a network in which the output nodes compete for the right to respond to a subset of the input data
Dendogram	a binary tree used to illustrate the arrangement of the clusters produced by hierarchical clustering
Dissimilarity	a proximity measurement of the difference between two data points
Eigenvalues, eigenvectors	in a square matrix, a non-zero vector when multiplied by the matrix yields a vector that lies parallel to the original
Expectation Maximization (EM)	an iterative method that finds the maximum likelihood of the estimate of parameters in statistical models, where the model has some latent variables. An iteration performs the Expectation step, which creates a function evaluating the expectation of the log-likelihood of the current parameters, and a Maximization step, which computes the parameters, thus maximizing the function in the Expectation step
Fuzzy clustering	a type of clustering in which the data point can belong to many different clusters denoted by its membership
Fuzzy c-means (FCM)	a partition-based clustering method in which a data point can belong to two or more clusters. The procedure is very similar to k-means clustering

Gaussian mixture	a probabilistic model in which a whole probability distribution is represented by the summation of several Gaussian distributions
Hard clustering	a type of clustering in which the data point can only belong to one cluster
Hierarchical clustering	a type of clustering that organizes data into hierarchical structures based on a proximity matrix
Image segmentation	the process of assigning a label to every pixel in an image such that pixels with the same label have the same visual characteristics
Independent component analysis	a method of finding a linear representation of non-Gaussian data so that the components are statistically independent
Kernel method	an approach in which a kernel function is used to map the data into a higher dimensional feature space and which allows a variety of clustering methods. The kernel functions allow computation through an inner product rather than on the coordinates of the data, thus resulting in a lower computational requirement
K-means	a classical clustering method of partitioning data into k clusters in which each data point belongs to the nearest mean
K-medoids	a process similar to k-means except that the mean is replaced by a data point as the center. K-medoids works with an arbitrary distance and is not limited to $L_2$
L1-norm	the sum of the absolute value of the components in a vector. The L1-norm is used widely in minimizing linear underestimated equations because of its property of sparse solutions
Large-scale data clustering	a special clustering problem in which the amount of data is extremely large (Big Data)
Learning rate	a parameter controlling the adjustments made during the training process. If the learning rate is large, the algorithm learns quickly; otherwise, the result in slow learning
Mahalanobis distance	a distance measure that takes into account the covariance among the dimensions
Maximum Likelihood Estimation (MLE)	A method of estimating the parameters of a statistical model in which the parameters are chosen to maximize the joint likelihood function
Online learning	a learning algorithm in which learning occurs sequentially as the data are introduced
Outlier	in statistics, a sample that lies far away from the rest of the data
Partitional clustering	as opposed to hierarchical clustering, this is the process of partitioning data into k clusters, which often is based on some optimized criterion function
Possibilistic c-means	a generalization of FCM, in which the membership of a data point in one cluster has nothing to do with other cluster. Membership is interpreted here as the degree of compatibility with the cluster prototype

Principle component analysis	a procedure that uses orthogonal transformation to extract a set of values of linearly uncorrelated variables called principle components
Prototype	a representation of a cluster, usually the means of data points in that cluster
Rand index	an external criterion for cluster validation
Self-organizing feature map	a neuron network trained using unsupervised learning to produce a low-dimensional, discrete representation of the training samples
Semi-supervised clustering	a type of clustering in which data are both labeled and unlabeled
Spectral clustering	a clustering technique making use of eigenvectors to perform dimension reduction before clustering in fewer dimensions
Subspace clustering	a type of clustering in which each cluster is a subspace
Unsupervised learning	a learning algorithm in which the data are not labeled
Vector quantization	a variant of k-means used in communications problems such as optimizing analog to digital conversion.
Vigilance parameter	a parameter that controls adaptation in ART

*Relevant Theory:* Signal Processing, Machine Learning

See this Volume, [Chapter 15](#) Neuron Networks

See this Volume, [Chapter 21](#) Unsupervised Learning and Latent Variable Models

See this Volume, [Chapter 22](#) Semi-supervised Learning

---

## References

- [1] A. Baraldi, E. Alpaydin, Constructive feedforward ART clustering networks – Part I and II, *IEEE Trans. Neural Networks* 13 (2002) 645–677.
- [2] V. Cherkassky, F. Mulier, *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, Inc., 1998.
- [3] E. Backer, A. Jain, A clustering performance measure based on fuzzy set decomposition, *IEEE Trans. Pattern Anal. Mach. Intell.* 3 (1981) 66–75.
- [4] B. Everitt, S. Landau, M. Leese, *Cluster Analysis*, Arnold, 2001.
- [5] R. Xu, D. Wunsch, II, *Wiley-IEEE Press, Clustering*, 2009.
- [6] A. Jain, R. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, 1988.
- [7] A. Gordon, *Classification*, second ed., Chapman and Hall/CRC Press, 1999.
- [8] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, *Math. Program.* 79 (1997) 191–215.
- [9] J. Hartigan, *Clustering Algorithms*, John Wiley & Sons, Inc., 1975.
- [10] R. Duda, P. Hart, D. Stork, *Pattern Classification*, second ed., John Wiley & Sons, Inc., 2001.
- [11] H. Späth, *Cluster Analysis Algorithms*, Ellis Horwood Ltd, 1980.
- [12] B. Duran, P. Odell, *Cluster Analysis: A Survey*, Springer-Verlag, 1974.
- [13] M. Anderberg, *Cluster Analysis for Applications*, Academic Press, 1973.
- [14] A. Jain, M. Murty, P. Flynn, Data clustering: a review, *ACM Comput. Sur.* 31 (1999) 264–323.
- [15] E. Kolatch, Clustering algorithms for spatial databases: a survey, 2001, Available from: <http://citeseer.nj.nec.com/436843.html>.
- [16] Q. He, A review of clustering algorithms as applied to IR, University of Illinois, Urbana-Champaign, 1999.

- [17] P. Berkhin, Survey of clustering data mining techniques, 2001, Available from: <<http://www.acrue.com/products/rpclusterreview.pdf>>, cached at <<http://citeseer.nj.nec.com/berkhin02survey.html>>.
- [18] F. Murtagh, A survey of recent advances in hierarchical clustering algorithms, *Comput. J.* 26 (1983) 354–359.
- [19] A. Baraldi, P. Blonda, A survey of fuzzy clustering algorithms for pattern recognition – Part I and II, *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 29 (1999) 778–801.
- [20] A. Baraldi, L. Schenato, Soft-to-Hard Model Transition in Clustering: A Review, International Computer Science Institute, Berkeley, CA, 1999, pp. TR-99-010.
- [21] H. Bock, Probabilistic models in cluster analysis, *Computational Statistics & Data Analysis* 23 (1996) 5–28.
- [22] R. Dubes, *Cluster Analysis and Related Issue*, World Science Publishing Company, 1993.
- [23] D. Fasulo, An analysis of recent work on clustering algorithms, University of Washington, Department of Computer Science & Engineering, 1999.
- [24] A. Jain, R. Duin, J. Mao, Statistical pattern recognition: a review, *IEEE Trans. Pattern Anal. Mach. Intel.* 22 (2000) 4–37.
- [25] A. Rauber, J. Paralic, E. Pampalk, Empirical evaluation of clustering algorithms, *J. Inform. Organ. Sci.* 24 (2000) 195–209.
- [26] C. Wei, Y. Lee, C. Hsu, Empirical comparison of fast clustering algorithms for large data sets, in: Proceedings of 33rd Hawaii International Conference on System Sciences, Maui, Hawaii, 2000.
- [27] P. Scheunders, A comparison of clustering algorithms applied to color image quantization, *Pattern Recognition Letter* 18 (1997) 1379–1384.
- [28] R. Shamir, R. Sharan, Algorithmic approaches to clustering gene expression data, in: T. S. T. Jiang, Y. Xu, M. Zhang (eds.), *Current Topics in Computational Molecular Biology*, MIT Press, 2002, pp. 269–300.
- [29] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, P. Brown, Clustering methods for the analysis of DNA microarray data, Technical Report, Department of Statistics, Stanford University, 1999.
- [30] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, in: KDD Workshop on Text Mining, 2000.
- [31] S.C. Madeira, A.L. Oliveira, Biclustering algorithms for biological data analysis: a survey, *IEEE trans. comput. biol. bioinf.* 1 (2004) 24–45.
- [32] A. Tanay, R. Sharan, R. Shamir, Biclustering Algorithms: A Survey, in: E. b. A. S. Chapman (ed.), *Handbook of Computational Molecular Biology*, CRC Computer and Information Science Series, 2005.
- [33] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, *SIGKDD Explor. Newsl.* 6 (2004) 90–105.
- [34] Q. V. Le, W. Y. Zou, S. Y. Yeung, A. Y. Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR, 2011, pp. 3361–3368.
- [35] R. Xu, D. Wunsch, Survey of clustering algorithms II, *IEEE Trans. Neural Networks* 16 (2005) 645–678.
- [36] R. Xu, D.C. Wunsch, Clustering algorithms in biomedical research: a review, *IEEE Rev. Biomed. Eng.* 3 (2010) 120–154.
- [37] R. Xu, D. Wunsch, BARTMAP: A viable structure for biclustering, *Neural Netw.* 24 (2011) 709–716.
- [38] H. Sharp Jr, Cardinality of finite topologies, *J. Combinatorial Theory* 5 (1968) 82–86.
- [39] G. Lance, W. Williams, A general theory of classification sorting strategies: 1, Hierarchical systems, *Comput. J.* 9 (1967) 373–380.
- [40] R. Yager, Intelligent control of the hierarchical agglomerative clustering process, *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 30 (2000) 835–845.
- [41] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of ACM SIGMOD International Conference on Management of Data, 1998, pp. 73–84.

- [42] S. Guha, R. Rastogi, K. Shim, ROCK: a robust clustering algorithm for categorical attributes, *Inform. Syst.* 25 (2000) 345–366.
- [43] R. Mollineda, E. Vidal, A relative approach to hierarchical clustering, in: E. M. Torres, A. Sanfeliu (ed.), *Pattern Recognition and Applications, Frontiers in Artificial Intelligence and Applications*. vol. 56, IOS Press, Amsterdam, 2000, pp. 19–28.
- [44] Y. Leung, J. Zhang, Z. Xu, Clustering by scale-space filtering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000) 1396–1410.
- [45] C. Li, G. Biswas, Unsupervised learning with mixed numeric and nominal data, *IEEE Trans. Knowl. Data Eng.* 14 (2002) 673–690.
- [46] C. Olson, Parallel algorithms for hierarchical clustering, *Parallel Comput.* 21 (1995) 1313–1325.
- [47] E. Dahlhaus, Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition, *J. Algorithms* 36 (2000) 205–240.
- [48] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.
- [49] A. Gersho, R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.
- [50] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
- [51] R. Yager, D. Filev, Approximate clustering via the mountain method, *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 24 (1994) 1279–1284.
- [52] I. Gath, A. Geva, Unsupervised optimal fuzzy clustering, *IEEE Trans. Pattern Anal. Mach. Intel.* 11 (1989) 773–781.
- [53] R. Hathaway, J. Bezdek, Y. Hu, Generalized fuzzy c-Means clustering strategies using  $L_p$  norm distances, *IEEE Trans. Fuzzy Syst.* 8 (2000) 576–582.
- [54] R. Krishnapuram, J. Keller, A possibilistic approach to clustering, *IEEE Trans. Fuzzy Syst.* 1 (1993) 98–110.
- [55] M. Figueiredo, A. Jain, Unsupervised learning of finite mixture models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (2002) 381–396.
- [56] G. McLachlan, T. Krishnan, *The EM Algorithm and Extensions*, Wiley, New York, 1997.
- [57] G. Celeux, G. Govaert, A classification EM algorithm for clustering and two stochastic versions, *Comput. Stat. Data Anal.* 14 (1992) 315–332.
- [58] C. Fraley, A. Raftery, Model-based clustering, discriminant analysis, and density estimation, *J. Am. Stat. Assoc.* 97 (2002) 611–631.
- [59] X. Zhuang, Y. Huang, K. Palaniappan, Y. Zhao, Gaussian mixture density modeling, decomposition, and applications, *IEEE Trans. Image Process.* 5 (1996) 1293–1302.
- [60] P. Cheeseman, J. Stutz, Bayesian classification (AutoClass): theory and results, in: G. P.-S. U. Fayyad, P. Smyth, R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 153–180.
- [61] T. Kohonen, The self-organizing map, in: *Proceedings of the IEEE*, 1990, pp. 1464–1480.
- [62] T. Kohonen, *Self-Organizing Maps*, Springer, 2001.
- [63] S. Haykin, *Neural networks: A Comprehensive Foundation*, second ed., Prentice Hall, 1999.
- [64] M. Su, H. Chang, Fast self-organizing feature map algorithm, *IEEE Trans. Neural Networks* 11(3) (2000) 721–733.
- [65] J. Vesanto, E. Alhoniemi, Clustering of the self-organizing map, *IEEE Trans. Neural Networks*, 11 (3) (2000) 586–600.
- [66] G. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns, *Applied Optics*. (1987) 26.
- [67] G. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing* 37 (1987).
- [68] G. Carpenter, S. Grossberg, J. Reynolds, ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network, *IEEE Trans. Neural Networks* 4 (1991) 169–181.

- [69] B. Moore, ART1 and pattern clustering, in: Proceedings of the 1988 Connectionist Models Summer School, 1989.
- [70] M.G.J. Huang, G. Heileman, Fuzzy ART properties, *IEEE Trans. Neural Networks* 8 (2) (1995) 203–213.
- [71] J. Williamson, Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps, *IEEE Trans. Neural Networks* 9 (5) (1996) 881–897.
- [72] G. Anagnostopoulos, M. Georgopoulos, Hypersphere ART and ARTMAP for unsupervised and supervised incremental learning, in: Proceedings of the IEEE-INNS-ENNS Intl. Joint Conf. on Neural Networks (IJCNN'00), 2000.
- [73] G. Anagnostopoulos, M. Georgopoulos, Ellipsoid ART and ARTMAP for incremental unsupervised and supervised learning, in: presented at the Proceedings of the IEEE-INNS-ENNS Intl. Joint Conf. on Neural Networks, 2001.
- [74] A. Ng, M. Jordan, Y. Weiss, On Spectral Clustering: analysis and an algorithm, in: Advances in Neural Information Processing Systems, 2001, pp. 849–856.
- [75] S. Jianbo, Normalized Cuts and Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000) 888–905.
- [76] U. Luxburg, A tutorial on spectral clustering, *Stat. Compu.* 17 (2007) 395–416.
- [77] Y. Cheng, G. Church, Bioclustering of expression data, in: Presented at the Proceedings of Eighth International Conference on Intelligent Systems for Molecular Biology, 2000.
- [78] S. Busyggin, O. Prokopyev, P. Pardalos, Bioclustering in data mining, *Comput. Oper. Res.* 35 (2008) 2964–2987.
- [79] S. Madeira, M. Teixeira, I. Sá-Correia, A. Oliveira, Identification of regulatory modules in time series gene expression data using a linear time bioclustering algorithm, *IEEE/ACM Trans. Comput. Biol. Bioinf.* 7 (2010) 153–165.
- [80] R. Vidal, A tutorial on subspace clustering, *IEEE Signal Process.* 2010.
- [81] J. Costeira, T. Kanade, A multibody factorization method for independently moving objects., *Int. J. Comput. Vision* 29 (1998) 159–179.
- [82] C.W. Gear, Multibody grouping from motion images, *Int. J. Comput. Vision* 29 (1998) 133–150.
- [83] P.S. Bradley, O.L. Mangasarian, k-plane clustering, *J. of Global Optim.* 16 (2000) 23–32.
- [84] M. Tipping, C. Bishop, Probabilistic principal component analysis, *J. Roy. Stat. Soc. Ser. B* 61 (1999) 611–622.
- [85] M. Fischler, R. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM* 24 (1981) 381–395.
- [86] J. Yan, M. Pollefeys, Articulated motion segmentation using RANSAC with priors, in: Presented at the Workshop on Dynamical Vision, 2005.
- [87] L. Donoho, Compressed Sensing, *IEEE Trans. Information Theory* 52 (2006) 1289–1306.
- [88] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: Presented at the IEEE Conference on Computer Vision and, Pattern Recognition, 2009.
- [89] E. Elhamifar, R. Vidal, Sparse Subspace Clustering, Algorithm, Theory, and Applications, arXiv:1203.1005v1, 2012.
- [90] C. Tang, A. Zhang, Interrelated two-way clustering and its application on gene expression data, *Int. J. Artif. Intell. Tools* 14 (2005) 577–597.
- [91] G. J. S. Busyggin, E. Kramer, Double conjugated clustering applied to leukemia microarray data, in: SIAM Data Mining Workshop on Clustering High Dimensional Data and Its Applications, 2002.
- [92] G. Getz, H. Gal, I. Kela, D. Notterman, E. Domany, Coupled two-way clustering analysis of breast cancer and colon cancer gene expression data, *Bioinformatics* 1 (2003) 1079–1089.
- [93] J. Shaik and M. Yeasin, Fuzzy-adaptive-subspace-iteration-based two way clustering of microarray data, *IEEE/ACM Trans. Comput. Biol. Bioinf.* 6 (2009) 244–259.
- [94] J. Hartigan, Direct clustering of a data matrix, *Journal of American Statistical Association* 67 (1972) 123–129.

- [95] D. Duffy, A. Quiroz, A permutation based algorithm for block clustering, *Journal of Classification* 8 (1991) 65–91.
- [96] J. Yang, W. Wang, H. Wang, P. Yu, Enhanced biclustering on expression data, in: *Proceedings of the Third IEEE Conference on Bioinformatics and Bioengineering*, 2003.
- [97] J. Yang, M.O. Ward, E.A. Rundensteiner, Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate data sets, *Comput. Graphics* 27 (2003) 265–283.
- [98] G. Babu, M. Murty, A near-optimal initial seed value selection in K-Means algorithm using a genetic algorithm, *Pattern Recognition Letters* 14 (1993).
- [99] J. G. K. Beyer, R. Ramakrishnan, U. Shaft, When is nearest neighbor meaningful, in: *Proceedings of 7th International Conference on Database Theory*, 1999, pp. 217–235.
- [100] L. Lazzeroni, A. Owen, Plaid models for gene expression data, *Statistica Sinica*, 6 (2002).
- [101] F. Divina, J. Aguilar-Ruiz, Biclustering of expression data with evolutionary computation, *IEEE Trans. Knowl. Data Eng* 18 (2006) 590–602.
- [102] Z. L. J. Liu, X. Hu, Y. Chen, Biclustering of microarray data with MOSPO based on crowding distance, *BMC Bioinformatics*, 10 (Suppl. 4) (2009) S9.
- [103] H. Turner, T. Bailey, W. Krzanowski, C. Hemingway, Biclustering models for structured microarray data, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2005, pp. 316–329.
- [104] Gu, J. Liu, Bayesian biclustering of gene expression data, *BMC Genomics* 9 (2007).
- [105] P. DiMaggio Jr., S. McAllister, C. Floudas, X. Feng, J. Rabinowitz, H. Rabitz, Biclustering via optimal re-ordering of data matrices in systems biology: rigorous methods and comparative studies, *BMC Bioinf.* 9 (1) (2008) 458.
- [106] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, A systematic comparison and evaluation of biclustering methods for gene expression data, *Bioinformatics* 22 (2006) 1122–1129.
- [107] A. Tchagang, A. Tewfik, M. DeRycke, K. Skubitz, A. Skubitz, Early detection of ovarian cancer using group biomarkers, *Mol. Cancer Ther.* 7 (1) (2008) 27–37.
- [108] R. Mitra, S. Bandyopadhyay, U. Maulik, M. Zhang, SFSS Class: an integrated approach for miRNA based tumor classification, *BMC Bioinformatics* 11 (Suppl 1) (2010) S22.
- [109] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR*, 2005, pp. 886–893.
- [110] D. G. Lowe, Object recognition from local scale-invariant features, in: *The Proc. Seventh IEEE Int Computer Vision Conf.*, 1999, pp. 1150–1157.
- [111] B. Olshausen, D. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, *Nature* 381 (1996) 607–609.
- [112] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Harvard University, 1974.
- [113] S. Kullback, R. A. Leibler, On Information and Sufficiency, *Annals of Mathematical Statistics*, 1951, 79–86.
- [114] I. Goodfellow, Q. Le, A. Saxe, H. Lee, A. Ng, Measuring invariances in deep networks, in: *NIPS*, 2010.
- [115] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, C. Schmid, Evaluation of local spatio-temporal features for action recognition, in: *BMVC*, 2010.
- [116] H. Lee, Y. Largman, P. Pham, A. Y. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, in: *Advances in Neural Information Processing Systems* 22, 2009, pp. 1096–1104.
- [117] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, A. Y. Ng, Text detection and character recognition in scene images with unsupervised feature learning, in: *Proceedings of the 11th International Conference on Document Analysis and Recognition*, 2011.

- [118] J.C. Bezdek, R.J. Hathaway, J.M. Huband, Visual assessment of clustering tendency for rectangular dissimilarity matrices, *IEEE Trans. Fuzzy Syst.* 15 (2007) 890–903.
- [119] T.C. Havens, J.C. Bezdek, An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm, *IEEE Trans. Knowl. Data Eng.* 24 (2012) 813–822.
- [120] J. M. Huband, J. C. Bezdek, R. J. Hathaway, Revised visual assessment of (cluster) tendency (reVAT), in: *Fuzzy Information, IEEE Annual Meeting of the Processing NAFIPS '04*, vol. 1, 2004, pp. 101–104.
- [121] D. Forsyth, J. Ponce, *Computer vision: a modern approach*: Prentice Hall, 2002.
- [122] M. Balafar, Medical image segmentation using fuzzy C-mean (FCM), Bayesian method and user interaction, in: *International Conference on Wavelet Analysis and, Pattern Recognition*, 2008, pp. 68–73.
- [123] D. Q. Zhang, S. C. Chen, A novel kernelized fuzzy c-means algorithm with application in medical image segmentation, *Artif. Intel. Med.* vol. 32 (2004) 37–50.
- [124] M. Balafar, MRI segmentation of medical images using FCM with initialized class centers via genetic algorithm, in: *International Symposium on Information Technology*, 2008, pp. 1–4.
- [125] A. Diplaros, N. Vlassis, T. Gevers, A spatially constrained generative model and an EM algorithm for image segmentation., *IEEE Trans. Neural Netw.* 18 (2007) 798–808.
- [126] A. Martinez, J. Vitria, Clustering in image space for place recognition and visual annotations for human-robot interaction, *IEEE Trans. syst. Man Cyber. B* 31 (2001) 669–682.
- [127] R. Vidal, Subspace Clustering, *IEEE Signal Process. Mag.* 28 (2011) 52–68.

# Unsupervised Learning Algorithms and Latent Variable Models: PCA/SVD, CCA/PLS, ICA, NMF, etc. 21

Andrzej Cichocki

*Laboratory for Advanced Brain Signal Processing, RIKEN, Brain Science Institute,  
Wako-shi, Saitama 3510198, Japan*

## 1.21.1 Introduction and of problems statement

Matrix factorizations and their extensions to tensor (multiway arrays) factorizations and decompositions have become prominent techniques for Latent Variable Models (LVM), linear and Multiway or Multilinear Blind Source Separation (MBSS), and (multiway) Generalized Component Analysis (GCA) (especially, multilinear Independent Component Analysis (ICA), Nonnegative Matrix and Tensor Factorizations (NMF/NTF), Smooth Component Analysis (SmoCA), and Sparse Component Analysis (SCA)). Moreover, matrix and tensor decompositions have many other applications beyond blind source separation, especially for feature extraction, classification, dimensionality reduction and multiway clustering. The recent trends in MBSS and GCA are to consider problems in the framework of matrix and tensor factorizations or more general multi-dimensional data for signal decomposition within probabilistic generative models and exploit *a priori* knowledge about the true nature, diversities, morphology or structure of latent (hidden) variables or sources such as spatio-temporal-spectral decorrelation, statistical independence, nonnegativity, sparseness, smoothness or lowest possible complexity. The goal of MBSS can be considered as estimation of true physical sources and parameters of a mixing system, while the objective of GCA is to find a reduced (or hierarchical and structured) component representation for the observed (sensor) multi-dimensional data, that can be interpreted as physically or physiologically meaningful coding or blind signal decomposition. The key issue is to find such a transformation (or coding) which has true physical meaning and interpretation. In this chapter we briefly review some efficient unsupervised learning algorithms for linear and multilinear blind source separation, blind source extraction and blind signals decomposition, using various criteria, constraints and assumptions. Moreover, we briefly overview emerging models and approaches for constrained matrix/tensor decompositions with applications to group and linked multilinear BSS/ICA, feature extraction, multiway Canonical Correlation Analysis (CCA), and Partial Least Squares (PLS) regression problems.

Although the basic models for matrix and tensor (i.e., multiway array) decompositions and factorizations, such as Tucker and Canonical Polyadic (CP) decomposition models, were proposed long a time ago [1–7], they have recently emerged as promising tools for the exploratory analysis of large-scale and multi-dimensional data in diverse applications, especially, for dimensionality reduction, multilinear independent component analysis, feature extraction, classification, prediction, and multiway clustering [5,8–12]. By virtue of their multiway nature, tensors constitute powerful tools for the analysis and fusion

of large-scale, multi-modal massive data, together with a mathematical backbone for the discovery of underlying hidden complex data structures [8,13].

The problem of separating or extracting source signals from a sensor array, without knowing the transmission channel characteristics and the sources, is addressed by a number of related BSS or GCA methods such as ICA and its extensions: Topographic ICA, Multilinear ICA, Kernel ICA, Tree-dependent Component Analysis, Multi-resolution Subband Decomposition ICA [13–18], Non-negative Matrix Factorization (NMF) [19,20], Sparse Component Analysis (SCA) [21–25], and Multi-channel Morphological Component Analysis (MCA) [26,27].

The mixing and filtering processes of the unknown input sources  $x_j(t)$  ( $j = 1, 2, \dots, J$ ) may correspond to different mathematical or physical models, depending on the specific applications [17,28]. Most linear BSS models can be expressed algebraically in their simplest forms as some specific form of a constrained matrix factorization: Given observation (often called sensor or input) data matrix  $\mathbf{Y} = [y_{it}] = [\mathbf{y}(1), \dots, \mathbf{y}(T)] \in \mathbb{R}^{I \times T}$ , we perform the matrix factorization (see Figure 21.1a):

$$\boxed{\mathbf{Y} = \mathbf{AX} + \mathbf{E} = \mathbf{AB}^T + \mathbf{E},} \quad (21.1)$$

where  $\mathbf{A} \in \mathbb{R}^{I \times J}$  represents the unknown mixing matrix,  $\mathbf{E} \in \mathbb{R}^{I \times T}$  is an unknown matrix representing errors or noises,  $\mathbf{X} = \mathbf{B}^T = [\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(T)] \in \mathbb{R}^{J \times T}$  contains the corresponding latent (hidden) components that give the contribution of each basis vector,  $T$  is the number of available samples,  $I$  is the number of observations and  $J$  is the number of sources or components.

In general the number of source signals  $J$  is unknown, and it can be larger, equal or smaller than the number of observations. The above model can be written in an equivalent scalar (element-wise) form (see Figure 21.1b):

$$y_{it} = \sum_{j=1}^J a_{ij} x_{jt} + e_{it} \quad \text{or} \quad y_i(t) = \sum_{j=1}^J a_{ij} x_j(t) + e_i(t) \quad (t = 1, 2, \dots, T). \quad (21.2)$$

Usually, the latent components represent unknown source signals with specific statistical properties or temporal structures and matrices have clear statistical properties and meanings. For example, the rows of the matrix  $\mathbf{X}$  that represent unknown sources or components should be statistically independent for ICA, sparse for SCA [21–24,29], nonnegative for NMF [8], or have other specific and additional morphological properties such as smoothness, continuity, or orthogonality [14,16,27].

In many signal processing applications the data matrix  $\mathbf{Y} = [\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)] \in \mathbb{R}^{I \times T}$  is represented by vectors  $\mathbf{y}(t) \in \mathbb{R}^I$  ( $t = 1, 2, \dots, T$ ) of multiple measurements or recordings for a set of discrete time points  $t$ . As mentioned above, the compact aggregated matrix Eq. (21.1) can be written in a vector form as a system of linear equations (see Figure 21.1c), that is,

$$\mathbf{y}(t) = \mathbf{Ax}(t) + \mathbf{e}(t) \quad (t = 1, 2, \dots, T), \quad (21.3)$$

where  $\mathbf{y}(t) = [y_1(t), y_2(t), \dots, y_I(t)]^T$  is a vector of the observed signals at the discrete time point  $t$  whereas  $\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_J(t)]^T$  is a vector of unknown sources at the same time point.

In order to estimate sources, sometimes we try first to identify the mixing matrix  $\mathbf{A}$  or its (pseudo)-inverse (unmixing) matrix  $\mathbf{W}$  and then estimate the sources. Usually, the inverse (unmixing) system

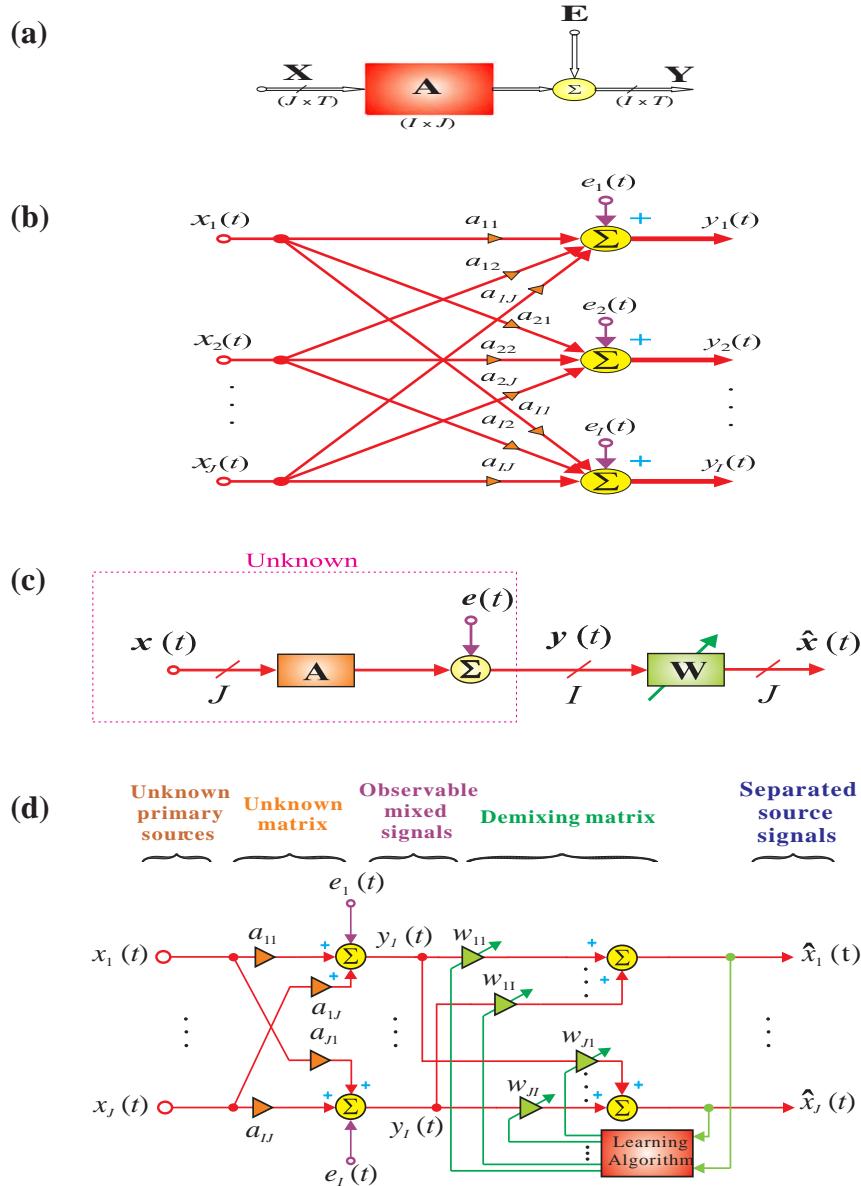


FIGURE 21.1

Basic linear (instantaneous) BSS model: (a) Block diagram expressed as a matrix factorization:  $\mathbf{Y} = \mathbf{AX} + \mathbf{E}$ , (b) its detailed model, (c) BSS via separating (demixing) matrix  $\mathbf{W} = \mathbf{A}^\dagger$ , and (d) its detailed model.

should be adaptive in such a way that it has some tracking capability in a nonstationary environment (see Figure 21.1d). We assume here that only the sensor vectors  $\mathbf{y}(t)$  are available and we need to estimate the parameters of the unmixing system online. This enables us to perform indirect identification of the mixing matrix  $\mathbf{A}$  (for  $I \geq J$ ) by estimating the separating matrix  $\mathbf{W} = \hat{\mathbf{A}}^\dagger$ , where the symbol  $(\cdot)^\dagger$  denotes the Moore–Penrose pseudo-inverse, and simultaneously estimate the sources. In other words, for  $I \geq J$  the original sources can be estimated by the linear transformation

$$\hat{\mathbf{x}}(t) = \mathbf{W}\mathbf{y}(t) \quad (t = 1, 2, \dots, T). \quad (21.4)$$

However, instead of estimating the source signals by projecting the observed signals by using the demixing matrix  $\mathbf{W}$ , it is often more convenient to identify an unknown mixing matrix  $\mathbf{A}$  (e.g., when the demixing system does not exist, especially when the system is underdetermined, i.e., the number of observations is lower than the number of source signals:  $I < J$ ) and then estimate the source signals by exploiting some *a priori* information about the source signals by applying a suitable optimization procedure.

There appears to be something magical about the BSS since we are estimating the original source signals without knowing the parameters of the mixing and/or filtering processes. It is difficult to imagine that one can estimate this at all. In fact, without some *a priori* knowledge it is not possible to *uniquely* estimate the original source signals. However, one can usually estimate them up to certain indeterminacies. In mathematical terms these indeterminacies and ambiguities can be expressed as arbitrary scaling and permutations of the estimated source signals. Let  $\Psi$  denote a specific BSS algorithm, then it holds that

$$\hat{\mathbf{X}} = \Psi(\mathbf{Y}) = \Psi(\mathbf{AX}) = \mathbf{PDX}, \quad (21.5)$$

where  $\mathbf{P}$  and  $\mathbf{D}$  are the permutation matrix and a diagonal scaling matrix (with nonzero entries), respectively. As a result, the original source signals represented by the rows of  $\mathbf{X}$  are recovered as  $\hat{\mathbf{X}}$  with the unavoidable ambiguities of scaling and permutations. If the BSS via constrained matrix factorization leads to the estimation of true components (sources) with arbitrary scaling and permutations, we call such factorization essentially unique. These indeterminacies preserve, however, the waveforms of the original sources. Although these indeterminacies seem to be rather severe limitations, in a great number of applications these limitations are not crucial, since the most relevant information about the source signals is contained in the temporal waveforms or time–frequency patterns of the source signals and usually not in their amplitudes or the order in which they are arranged in the system output. For some models, however, there is no guarantee that the estimated or extracted signals will have exactly the same waveforms as the source signals, and then the requirements must be sometimes further relaxed to the extent that the extracted waveforms are distorted (i.e., time delayed, filtered, or convolved) versions of the primary source signals.

In some applications the mixing matrix  $\mathbf{A}$  is ill-conditioned. In such cases some special models and algorithms should be applied. Although some decompositions or matrix factorizations provide an exact reconstruction of the data (i.e.,  $\mathbf{Y} = \mathbf{AX}$ ), we will consider here constrained factorizations which are approximative in nature. In fact, many problems in signal and image processing can be solved in terms of matrix factorizations. However, different cost functions and imposed constraints may lead to different types of matrix factorizations.

The BSS problems are closely related to the concept of linear inverse problems or more generally, to solving a large ill-conditioned system of linear equations (overdetermined or underdetermined), where

it is required to estimate the vectors  $\mathbf{x}(t)$  (also in some cases to identify the matrix  $\mathbf{A}$ ) from noisy data [14,30,31]. Physical systems are often contaminated by noise, thus, our task is generally to find an optimal and robust solution in a noisy environment. Wide classes of extrapolation, reconstruction, estimation, approximation, interpolation, and inverse problems can be converted into minimum norm problems of solving underdetermined systems of linear Eqs. (21.1) for  $J > I$  [14,31]. Generally speaking, in signal processing applications, an overdetermined ( $I > J$ ) system of linear Eqs. (21.1) describes filtering, enhancement, deconvolution and identification problems, while the underdetermined case describes inverse and extrapolation problems [14,30]. Although many different BSS criteria and algorithms are available, most of them exploit various diversities or constraints imposed for estimated components and/or factor matrices such as orthogonality, mutual independence, nonnegativity, sparsity, smoothness, predictability or lowest complexity. By diversities we usually mean different morphological characteristics or features of the signals.

More sophisticated or advanced approaches for GCA use combinations or the integration of various diversities, in order to separate or extract sources with various constraints, morphology, structures, or statistical properties, and to reduce the influence of noise and undesirable interferences [14,32].

All the above-mentioned BSS models are usually based on a wide class of unsupervised or semi-supervised learning algorithms. Unsupervised learning algorithms try to discover a structure underlying a data set, extract the meaningful features, and find useful representations of the given data. Since data can always be interpreted in many different ways, some knowledge is needed to determine which features or properties best represent true latent (hidden) components. For example, PCA finds a low-dimensional representation of the data that captures most of its variance. On the other hand, SCA tries to explain the data as a mixture of sparse components (usually, in the time–frequency domain), and NMF seeks to explain the data by parts-based localized additive representations (with nonnegativity constraints).

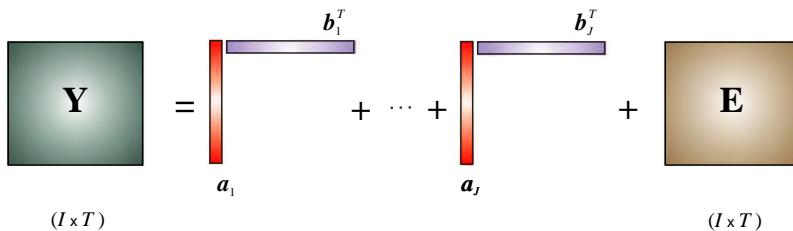
Most linear BSS or GCA models can be represented as constrained bilinear matrix factorization problems, with suitable constraints imposed on the component matrix:

$$\begin{aligned} \mathbf{Y} &= \mathbf{AB}^T + \mathbf{E} = \sum_{j=1}^J \mathbf{a}_j \mathbf{b}_j^T + \mathbf{E} \\ &= \sum_{j=1}^J \mathbf{a}_j \circ \mathbf{b}_j + \mathbf{E}, \end{aligned} \tag{21.6}$$

where  $\mathbf{Y} \in \mathbb{R}^{I \times T}$  is a known data matrix (representing observations or measurements),  $\mathbf{E} \in \mathbb{R}^{I \times T}$  represents errors or noise,  $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J] \in \mathbb{R}^{I \times J}$  is the unknown full column rank mixing (basis) matrix with  $J$  basis vectors  $\mathbf{a}_j \in \mathbb{R}^I$ , and  $\mathbf{B} = \mathbf{X}^T = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_J] \in \mathbb{R}^{T \times J}$  represent the unknown components, latent variables or sources  $\mathbf{b}_j \in \mathbb{R}^T$  (see Figure 21.2).

Note that we have symmetry of the factorization: For Eq. (21.6) we could write  $\mathbf{Y}^T \cong \mathbf{BA}^T$ , so the meaning of “sources” and “mixture” are often somewhat arbitrary.

The unknown components usually have clear statistical properties and meaning. For example, the columns of the matrix  $\mathbf{B}$  should be as statistically mutually independent as possible, or sparse, piecewise smooth, and/or nonnegative, etc., according to the physical meaning of the components, and to research

**FIGURE 21.2**

Bilinear BSS model. The data matrix  $\mathbf{Y} \in \mathbb{R}^{I \times T}$  is approximately represented by a sum or linear combination of rank-1 matrices  $\mathbf{Y}^{(j)} = \mathbf{a}_j \circ \mathbf{b}_j = \mathbf{a}_j \mathbf{b}_j^T \in \mathbb{R}^{I \times T}$ . The objective is to estimate all vectors  $\mathbf{a}_j$  and  $\mathbf{b}_j$  and the optimal index (rank)  $J$ . In general such decomposition is not unique, and we need to impose various constraints (e.g., nonnegativity, orthogonality, independence, or sparseness) to extract unique components with desired diversities or properties.

the field from which the data are collected. The BSS methods estimate these components maximizing the *a priori* information by imposing proper constraints on the components, then leading to various categories of BSS, such as ICA, SCA, SmoCA, NMF, etc. [8, 14, 33, 34].

Very often multiple subject, multiple task data sets can be represented by a set of data matrices  $\mathbf{Y}_n$ . It is therefore necessary to perform simultaneous constrained matrix factorizations:

$$\mathbf{Y}_n \approx \mathbf{A}_n \mathbf{B}_n^T \quad (n = 1, 2, \dots, N), \quad (21.7)$$

subject to various additional constraints (e.g.,  $\mathbf{A}_n = \mathbf{A}$  for all  $n$ , and their columns are mutually independent and/or sparse). This problem is related to various models of group ICA, with suitable pre-processing, dimensionality reduction and post-processing procedures [35–37]. This form of factorization is typical for EEG/MEG related data for multi-subjects, multi-tasks, while the factorization for the transposed of  $\mathbf{Y}$  is typical for fMRI data [35–38].

The multilinear BSS concept introduced in Section 1.21.5.4 is more general and flexible than the group ICA, since various constraints can be imposed on the component matrices for different modes (i.e., not only mutual independence but also nonnegativity, sparseness, smoothness or orthogonality). There are neither theoretical nor experimental reasons why statistical independence (used by ICA) should be the only one right concept to be used to extract latent components [33]. In real world scenarios, latent (hidden) components have various complex properties and features. In other words, true unknown components are seldom all statistically independent. Therefore, if we apply only one criterion like ICA, we may fail to extract all desired components with correct interpretations. Rather we need to apply the fusion of several strategies by employing several suitably chosen criteria and their associated learning algorithms to extract all desired components for specific modes [8, 33]. For these reasons we often consider the multilinear BSS methods, which exploit multiple criteria or diversities for the component matrices for various modes, rather than only statistical independence (see Section 1.21.5).

## 1.21.2 PCA/SVD and related problems

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are well established tools with applications virtually in all areas of science, machine learning, signal processing, engineering,

genetics and neural computation, to name just a few, where large data sets are encountered [39, 40]. The basic aim of PCA is to find a few linear combinations of variables, called principal components, which point in orthogonal directions while explaining as much of the variance of the data as possible.

PCA is perhaps one of the oldest and best-known techniques from multivariate analysis and data mining. It was introduced by Pearson, who used it in a biological context and further developed by Hotelling in research work related to psychometry. PCA was also developed independently by Karhunen in the context of probability theory and was subsequently generalized by Loève (see [14, 39, 41] and references therein). The purpose of PCA is to derive a relatively small number of uncorrelated linear combinations (principal components) of a set of random zero-mean variables  $\mathbf{y}(t)$ , while retaining as much of the information from the original variables as possible.

Following are several of the objectives of PCA/SVD:

1. dimensionality reduction;
2. determination of linear combinations of variables;
3. feature selection: the choice of the most useful variables;
4. visualization of multi-dimensional data;
5. identification of underlying variables;
6. identification of groups of objects or of outliers.

Often the principal components (PCs) (i.e., directions on which the input data has the largest variance) are usually regarded as important or significant, while those components with the smallest variance called minor components (MCs) are usually regarded as unimportant or associated with noise. However, in some applications, the MCs are of the same importance as the PCs, for example, in the context of curve or surface fitting or total least squares (TLS) problems.

Generally speaking, PCA is often related and motivated by the following two problems:

1. Given random vectors  $\mathbf{y}(t) \in \mathbb{R}^I$  ( $t = 1, 2, \dots, T$ ), with finite second order moments and zero mean, find the reduced  $J$ -dimensional ( $J < I$ ) linear subspace that minimizes the expected distance of  $\mathbf{y}(t)$  from the subspace. This problem arises in the area of data compression, where the task is to represent all the data with a reduced number of parameters while assuring minimum distortion due to the projection.
2. Given random vectors  $\mathbf{y}(t) \in \mathbb{R}^I$ , find the  $J$ -dimensional linear subspace that captures most of the variance of the data. This problem is related to feature extraction, where the objective is to reduce the dimension of the data while retaining most of its information content.

It turns out that both problems have the same optimal solution (in the sense of least-squares error), which is based on the second order statistics (SOS), in particular, on the eigen structure of the data covariance matrix.

**Remark.** If the signals have standard deviation (SD) one, the covariance and correlation matrices are identical. The correlation matrix is the covariance matrix of the mean centered and scaled vectors. Since, we consider here zero mean signals with SD one, both matrices are equivalent.

PCA is designed for a general random variable regardless whether it is of zero mean or not. But at the first step we center the data, otherwise the first principal component will converge to its mean value.

### 1.21.2.1 Standard PCA

Without loss of generality let us consider a data matrix  $\mathbf{Y} \in \mathbb{R}^{T \times I}$  with  $T$  samples and  $I$  variables and with the columns centered (zero mean) and scaled. Equivalently, it is sometimes convenient to consider the data matrix  $\mathbf{Y}' = [\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)] \in \mathbb{R}^{I \times T}$ . The standard PCA can be written in terms of the scaled sample covariance matrix  $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^T\} \in \mathbb{R}^{I \times I}$  as follows: Find orthonormal vectors by solving the following optimization problem

$$\max_{\mathbf{v}_i} \{\mathbf{v}_i \mathbf{R}_{yy} \mathbf{v}_i^T\}, \quad \text{s.t. } \mathbf{v}_i^T \mathbf{v}_i = 1, \quad \mathbf{v}_i^T \mathbf{v}_k = 0 \quad \forall i > k \quad (i = 1, 2, \dots, I). \quad (21.8)$$

PCA can be converted to the eigenvalue problem of the covariance matrix of  $\mathbf{y}(t)$  ( $t = 1, 2, \dots, T$ ), and it is essentially equivalent to the Karhunen-Loëve transform used in image and signal processing. In other words, PCA is a technique for the computation of eigenvectors and eigenvalues for the scaled estimated covariance matrix

$$\mathbf{R}_{yy} = E\{\mathbf{y}(t)\mathbf{y}^T(t)\} = \mathbf{V}\Lambda\mathbf{V}^T = \sum_{i=1}^I \lambda_i \mathbf{v}_i \mathbf{v}_i^T, \quad (21.9)$$

where  $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_I\}$  is a diagonal matrix containing the  $I$  eigenvalues (ordered in decreasing values) and  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_I] \in \mathbb{R}^{I \times I}$  is the corresponding orthogonal or unitary matrix consisting of the unit length eigenvectors referred to as principal eigenvectors. It should be noted that the covariance matrix is symmetric positive definite, that is why the eigenvalues are nonnegative. We assume that for noisy data all eigenvalues are positive and we put them in descending order.

The basic problem we try to solve is the standard eigenvalue problem which can be formulated using the equations

$$\mathbf{R}_{yy} \mathbf{v}_i = \lambda_i \mathbf{v}_i \quad (i = 1, 2, \dots, I), \quad (21.10)$$

where  $\mathbf{v}_i$  are the orthonormal eigenvectors,  $\lambda_i$  are the corresponding eigenvalues and  $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^T\}$  is the covariance matrix of zero-mean signals  $\mathbf{y}(t)$  and  $E$  is the expectation operator. Note that (21.10) can be written in matrix form as  $\mathbf{V}^T \mathbf{R}_{yy} \mathbf{V} = \Lambda$ , where  $\Lambda$  is the diagonal matrix of eigenvalues (ranked in descending order) of the estimated covariance matrix  $\mathbf{R}_{yy}$ .

The Karhunen-Loëve-transform determines a linear transformation of an input vector  $\mathbf{y}$  as

$$\tilde{\mathbf{y}}_P(t) = \mathbf{V}_S^T \mathbf{y}(t), \quad (21.11)$$

where  $\mathbf{y}(t) = [y_1(t), y_2(t), \dots, y_I(t)]^T$  is the zero-mean input vector,  $\tilde{\mathbf{y}}_P = [\tilde{y}_1(t), \tilde{y}_2(t), \dots, \tilde{y}_J(t)]^T$  is the output vector called the vector of principal components (PCs), and  $\mathbf{V}_S = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J] \in \mathbb{R}^{I \times J}$ , with  $J < I$ , is the set of signal subspace eigenvectors, with the orthonormal vectors  $\mathbf{v}_j = [v_{j1}, v_{j2}, \dots, v_{jI}]^T$ , (i.e.,  $(\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij})$  for  $j \leq i$ , where  $\delta_{ij}$  is the Kronecker delta that equals 1 for  $i = j$ , and zero otherwise. The vectors  $\mathbf{v}_j$  ( $j = 1, 2, \dots, J$ ) are eigenvectors of the covariance matrix, while the variances of the PCs  $\tilde{y}_j(t) = \mathbf{v}_j^T \mathbf{y}(t)$  are the corresponding principal eigenvalues, i.e.,  $\lambda_j = E\{\tilde{y}_j^2\}$ .

On the other hand, the  $(I - J)$  minor components are given by

$$\tilde{\mathbf{y}}_M(t) = \mathbf{V}_N^T \mathbf{y}(t), \quad (21.12)$$

where  $\mathbf{V}_N = [\mathbf{v}_{J+1}, \mathbf{v}_{J+2}, \dots, \mathbf{v}_I]$  consists of the  $(I - J)$  eigenvectors associated with the smallest eigenvalues.

### 1.21.2.2 Determining the number of principal components

A quite important problem arising in many application areas is the determination of the dimensions of the signal and noise subspaces. In other words, a central issue in PCA is choosing the number of principal components to be retained [42–44]. To solve this problem, we usually exploit a fundamental property of PCA: It projects the sensor data  $\mathbf{y}(t) \in \mathbb{R}^I$  from the original  $I$ -dimensional space onto a  $J$ -dimensional output subspace  $\tilde{\mathbf{y}}(t) \in \mathbb{R}^J$  (typically, with  $J \ll I$ ), thus performing a dimensionality reduction which retains most of the intrinsic information in the input data vectors. In other words, the principal components  $\tilde{\mathbf{y}}_j(t) = \mathbf{v}_j^T \mathbf{y}(t)$  are estimated in such a way that, for  $J \ll I$ , although the dimensionality of data is strongly reduced, the most relevant information is retained in the sense that the input data can be approximately reconstructed from the output data (signals) by using the transformation  $\hat{\mathbf{y}} = \mathbf{V}_S \tilde{\mathbf{y}}$ , that minimizes a suitable cost function. A commonly used criterion is the minimization of the mean squared error  $\|\mathbf{y} - \mathbf{V}_S \mathbf{V}_S^T \mathbf{y}\|_2^2$  subject to orthogonality constraint  $\mathbf{V}_S^T \mathbf{V}_S$ .

Under the assumption that the power of the signals is larger than the power of the noise, the PCA enables us to divide observed (measured) sensor signals:  $\mathbf{y}(t) = \mathbf{y}_s(t) + \mathbf{e}(t)$  into two subspaces: the *signal subspace* corresponding to principal components associated with the largest eigenvalues called the principal eigenvalues:  $\lambda_1, \lambda_2, \dots, \lambda_J$ , ( $I > J$ ) and associated eigenvectors  $\mathbf{V}_S = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J]$  called the principal eigenvectors and the *noise subspace* corresponding to the minor components associated with the eigenvalues  $\lambda_{J+1}, \dots, \lambda_I$ . The subspace spanned by the  $J$  first eigenvectors  $\mathbf{v}_j$  can be considered as an approximation of the noiseless signal subspace. One important advantage of this approach is that it enables not only a reduction in the noise level, but also allows us to estimate the number of sources on the basis of distribution of the eigenvalues. However, a problem arising from this approach is how to correctly set or estimate the threshold which divides eigenvalues into the two subspaces, especially when the noise is large (i.e., the SNR is low). The scaled covariance matrix of the observed data can be written as

$$\begin{aligned} \mathbf{R}_{yy} &= [\mathbf{V}_S, \mathbf{V}_N] \begin{bmatrix} \mathbf{\Lambda}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_N \end{bmatrix} [\mathbf{V}_S, \mathbf{V}_N]^T \\ &= \mathbf{V}_S \mathbf{\Lambda}_S \mathbf{V}_S^T + \mathbf{V}_N \mathbf{\Lambda}_N \mathbf{V}_N^T, \end{aligned} \quad (21.13)$$

where  $\mathbf{V}_S \mathbf{\Lambda}_S \mathbf{V}_S^T$  is a rank- $J$  matrix,  $\mathbf{V}_S \in \mathbb{R}^{I \times J}$  contains the eigenvectors associated with  $J$  principal (signal + noise subspace) eigenvalues of  $\mathbf{\Lambda}_S = \text{diag}\{\lambda_1 \geq \lambda_2 \dots \geq \lambda_J\}$  in a descending order. Similarly, the matrix  $\mathbf{V}_N \in \mathbb{R}^{I \times (I-J)}$  contains the  $(I - J)$  (noise subspace) eigenvectors that correspond to the noise eigenvalues  $\mathbf{\Lambda}_N = \text{diag}\{\lambda_{J+1}, \dots, \lambda_I\} = \sigma_e^2 \mathbf{I}_{I-J}$ . This means that, theoretically, the  $(I - J)$  smallest eigenvalues of  $\mathbf{R}_{yy}$  are equal to  $\sigma_e^2$ , so we can theoretically (in the ideal case) determine the dimension of the signal subspace from the multiplicity of the smallest eigenvalues under the assumption that the variance of the noise is relatively low and we have a perfect estimate of the covariance matrix. However, in practice, we estimate the sample covariance matrix from a limited number of samples and the (estimated) smallest eigenvalues are usually different, so the determination of the dimension of the signal subspace is not an easy task. Usually we set the threshold between the signal and noise eigenvalues by using some heuristic procedure or a rule of thumb. A simple ad hoc rule is to plot the eigenvalues in decreasing order and search for an elbow, where the signal eigenvalues are on the left side and the noise eigenvalues are on the right side. Another simple technique is to compute the cumulative percentage of the total variation explained by the PCs and retain the number of PCs that represent, say

95% of the total variation. Such techniques often work well in practice, but their disadvantage is that they need a subjective decision from the user [44].

Alternatively, can use one of the several well-known information theoretic criteria, namely, Akaike's Information Criterion (AIC), the Minimum Description Length (MDL) criterion and Bayesian Information Criterion (BIC) [43,45,46]. To do this, we compute the probability distribution of the data for each possible dimension (see [46]).

Unfortunately, AIC and MDL criteria provide only rough estimates (of the number of sources) that are rather very sensitive to variations in the SNR and the number of available data samples.

Many sophisticated methods have been introduced such as a Bayesian model selection method, which is referred to as the Laplace method. It is based on computing the evidence for the data and it requires integrating out all the model parameters. The BIC method can be thought of as an approximation to the Laplace criterion. The calculation involves an integral over the Stiefel manifold which is approximated by the Laplace method [43]. One problem with the AIC, MDL and BIC criteria mentioned above is that they have been derived by assuming that the data vectors  $\mathbf{y}(t)$  have a Gaussian distribution [46]. This is done for mathematical tractability, by making it possible to derive closed form expressions. The Gaussianity assumption does not usually hold exactly in the BSS and other signal processing applications. For setting the threshold between the signal and noise eigenvalues, one might even suppose that the AIC, MDL and BIC criteria cannot be used for the BSS, particularly ICA problems, because in those cases we assume that the source signals  $x_j(t)$  are non-Gaussian. However, it should be noted that the components of the data vectors  $\mathbf{y}(t) = \mathbf{Ax}(t) + \mathbf{e}(t)$  are mixtures of the sources, and therefore they often have distributions that are not too far from a Gaussian distribution.

An alternative simple heuristic method [42,47] computes the GAP (smoothness) index between consecutive eigenvalues defined as

$$\text{GAP}(p) = \frac{\text{var}[\{\tilde{\lambda}_i\}_{i=p+1}^{I-1}]}{\text{var}[\{\tilde{\lambda}_i\}_{i=p}^{I-1}]} = \frac{\hat{\sigma}_{p+1}^2}{\hat{\sigma}_p^2} \quad (p = 1, 2, \dots, I-2), \quad (21.14)$$

where  $\tilde{\lambda}_i = \lambda_i - \lambda_{i+1}$  and  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_I > 0$  are the eigenvalues of the covariance matrix for the noisy data and the sample variances are computed as follows:

$$\hat{\sigma}_p^2 = \text{var}[\{\tilde{\lambda}_i\}_{i=p}^{I-1}] = \frac{1}{I-p} \sum_{i=p}^{I-1} \left( \tilde{\lambda}_i - \frac{1}{I-p} \sum_{i=p}^{I-1} \tilde{\lambda}_i \right)^2. \quad (21.15)$$

The number of components (for each mode) are selected using the following criterion:

$$\hat{J} = \arg \min_{p=1,2,\dots,I-3} \{\text{GAP}(p)\}. \quad (21.16)$$

Recently, Ulfarsson and Solo [44] proposed an alternative more sophisticated method called SURE (Stein's Unbiased Risk Estimator) which allows to estimate the number of PC components.

Extensive numerical experiments indicate that the Laplace method usually outperforms the BIC method while the GAP and SURE methods can often achieve significantly better performance than the Laplace method.

### 1.21.2.3 Whitening

Some adaptive algorithms for blind separation require prewhitening (also called sphering or normalized spatial decorrelation) of mixed (sensor) signals. A random, zero-mean vector  $\tilde{\mathbf{y}}$  is said to be white if its covariance matrix is an identity matrix, i.e.,  $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\} = \mathbf{I}_J$  or  $E\{\tilde{y}_i\tilde{y}_j\} = \delta_{ij}$ , where  $\delta_{ij}$  is the Kronecker delta. In whitening, the sensor vectors  $\mathbf{y}(t)$  are pre-processed using the following transformation:

$$\tilde{\mathbf{y}}(t) = \mathbf{Q}\mathbf{y}(t). \quad (21.17)$$

Here  $\tilde{\mathbf{y}}(t)$  denotes the whitened vector, and  $\mathbf{Q}$  is an  $J \times I$  whitening matrix. For  $J < I$ , where  $J$  is known in advance,  $\mathbf{Q}$  simultaneously reduces the dimension of the data vectors from  $I$  to  $J$ . In whitening, the matrix  $\mathbf{Q}$  is chosen such that the scaled covariance matrix  $E\{\tilde{\mathbf{y}}(t)\tilde{\mathbf{y}}(t)^T\}$  becomes the unit matrix  $\mathbf{I}_J$ . Thus, the components of the whitened vectors  $\tilde{\mathbf{y}}(t)$  are mutually uncorrelated and have unit variance, i.e.,

$$\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\} = E\{\mathbf{Q}\mathbf{y}\mathbf{y}^T\mathbf{Q}^T\} = \mathbf{Q}\mathbf{R}_{\mathbf{y}\mathbf{y}}\mathbf{Q}^T = \mathbf{I}_J. \quad (21.18)$$

Generally, the sensor signals are mutually correlated, i.e., the covariance matrix  $\mathbf{R}_{\mathbf{y}\mathbf{y}} = E\{\mathbf{y}\mathbf{y}^T\}$  is a full (not diagonal) matrix. It should be noted that the matrix  $\mathbf{Q} \in \mathbb{R}^{J \times I}$  is not unique, since multiplying it by an arbitrary orthogonal matrix from the left still preserves property (21.18). Since the covariance matrix of sensor signals  $\mathbf{y}(t)$  is symmetric positive semi-definite, it can be decomposed as follows:

$$\mathbf{R}_{\mathbf{y}\mathbf{y}} = \mathbf{V}\Lambda\mathbf{V}^T = \mathbf{V}\Lambda^{1/2}\Lambda^{1/2}\mathbf{V}^T, \quad (21.19)$$

where  $\mathbf{V}$  is an orthogonal matrix and  $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_J\}$  is a diagonal matrix with positive eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_J > 0$ . Hence, under the condition that the covariance matrix is positive definite, the required decorrelation matrix  $\mathbf{Q}$  (also called a whitening matrix or Mahalanobis transform) can be computed as follows:

$$\mathbf{Q} = \Lambda^{-1/2}\mathbf{V}^T = \text{diag}\left\{\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \dots, \frac{1}{\sqrt{\lambda_J}}\right\}\mathbf{V}^T \quad (21.20)$$

or

$$\mathbf{Q} = \mathbf{U}\Lambda^{-1/2}\mathbf{V}^T, \quad (21.21)$$

where  $\mathbf{U}$  is an arbitrary orthogonal matrix. This can be easily verified by substituting (21.20) or (21.21) into (21.18):

$$\mathbf{R}_{\mathbf{y}\mathbf{y}} = E\{\mathbf{y}\mathbf{y}^T\} = \Lambda^{-1/2}\mathbf{V}^T\mathbf{V}\Lambda\mathbf{V}^T\mathbf{V}\Lambda^{-1/2} = \mathbf{I}_J, \quad (21.22)$$

or

$$\mathbf{R}_{\mathbf{y}\mathbf{y}} = \mathbf{U}\Lambda^{-1/2}\mathbf{V}^T\mathbf{V}\Lambda\mathbf{V}^T\mathbf{V}\Lambda^{-1/2}\mathbf{U}^T = \mathbf{I}_J. \quad (21.23)$$

Alternatively, we can apply the Cholesky decomposition

$$\mathbf{R}_{\mathbf{y}\mathbf{y}} = \mathbf{L}\mathbf{L}^T, \quad (21.24)$$

where  $\mathbf{L}$  is a lower triangular matrix. The whitening (decorrelation) matrix in this case is

$$\mathbf{Q} = \mathbf{U}\mathbf{L}^{-1}, \quad (21.25)$$

where  $\mathbf{U}$  is an arbitrary orthogonal matrix, since

$$\mathbf{R}_{yy} = E\{yy^T\} = \mathbf{Q}\mathbf{R}_{yy}\mathbf{Q}^T = \mathbf{U}\mathbf{L}^{-1}\mathbf{L}\mathbf{L}^T(\mathbf{L}^{-1})^T\mathbf{U}^T = \mathbf{I}_J. \quad (21.26)$$

In the special case when  $y(t) = \mathbf{e}(t)$  is colored Gaussian noise with  $\mathbf{R}_{ee} = E\{\mathbf{e}\mathbf{e}^T\} \neq \sigma_e^2 \mathbf{I}_J$ , the whitening transform converts it into a white noise (i.i.d.) process. If the covariance matrix is positive semi-definite, we can take only the positive eigenvalues and the associated eigenvectors.

It should be noted that after pre-whitening of sensor signals  $y(t)$ , in the model  $\mathbf{Y} \cong \mathbf{AX}$ , the new mixing matrix defined as  $\mathbf{A}_* = \mathbf{QA}$  is orthogonal (i.e.,  $\mathbf{A}_*\mathbf{A}_*^T = \mathbf{I}_J$ ), if source signals are uncorrelated (i.e.,  $\mathbf{R}_{xx} = E\{\mathbf{xx}^T\} = \mathbf{I}_J$ ), since we can write  $\mathbf{R}_{\tilde{y}\tilde{y}} = E\{\tilde{y}\tilde{y}^T\} = \mathbf{A}_*E\{\mathbf{xx}^T\}\mathbf{A}_*^T = \mathbf{I}_J$ .

#### 1.21.2.4 SVD

The Singular Value Decomposition (SVD) is a tool of both great practical and theoretical importance in signal processing and data analysis [40]. The SVD of a data matrix  $\mathbf{Y} \in \mathbb{R}^{T \times I}$  of rank- $J$  with  $J \leq I$ , assuming without loss of generality that  $T > I$ , leads to the following matrix factorization

$$\boxed{\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T = \sum_{j=1}^J \sigma_j \mathbf{u}_j \mathbf{v}_j^T}, \quad (21.27)$$

where the matrix  $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_T] \in \mathbb{R}^{T \times T}$  contains the  $T$  left singular vectors,  $\Sigma \in \mathbb{R}_+^{T \times I}$  has nonnegative elements on the main diagonal representing the singular values  $\sigma_j$  and the matrix  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_I] \in \mathbb{R}^{I \times I}$  represents the  $I$  right singular vectors called the loading factors. The nonnegative quantities  $\sigma_j$ , sorted as  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_J > \sigma_{J+1} = \sigma_{J+2} = \dots = \sigma_I = 0$  can be shown to be the square roots of the eigenvalues of the symmetric matrix  $\mathbf{Y}^T \mathbf{y} \in \mathbb{R}^{I \times I}$ . The term  $\mathbf{u}_j \mathbf{v}_j^T$  is a  $T \times I$  rank-1 matrix often called the  $j$ th eigenimage of  $\mathbf{Y}$ . Orthogonality of the SVD expansion ensures that the left and right singular vectors are orthogonal, i.e.,  $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$  and  $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$ , with  $\delta_{ij}$  the Kronecker delta (or equivalently  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$  and  $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ ).

In many applications, it is more practical to work with the truncated form of the SVD, where only the first  $P \leq J$ , (where  $J$  is the rank of  $\mathbf{Y}$ , with  $J < I$ ) singular values are used, so that

$$\mathbf{Y} \cong \mathbf{U}_P \Sigma_P \mathbf{V}_P^T = \sum_{j=1}^P \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \quad (21.28)$$

where  $\mathbf{U}_P = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_P] \in \mathbb{R}^{T \times P}$ ,  $\Sigma_P = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_P\}$  and  $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_P] \in \mathbb{R}^{I \times P}$ . This is no longer an exact decomposition of the data matrix  $\mathbf{Y}$ , but according to the **Eckart-Young** theorem it is the best rank- $P$  approximation in the least-squares sense and it is still unique (neglecting signs of vectors ambiguity) if the singular values are distinct.

For noiseless data, we can use the following decomposition:

$$\mathbf{Y} = [\mathbf{U}_S, \mathbf{U}_N] \begin{bmatrix} \Sigma_S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\mathbf{V}_S, \mathbf{V}_N]^T, \quad (21.29)$$

where  $\mathbf{U}_S = [\mathbf{u}_1, \dots, \mathbf{u}_J] \in \mathbb{R}^{T \times J}$ ,  $\Sigma_S = \text{diag}\{\sigma_1, \dots, \sigma_J\}$  and  $\mathbf{U}_N = [\mathbf{u}_{J+1}, \dots, \mathbf{u}_I]$ . The set of matrices  $\{\mathbf{U}_S, \Sigma_S, \mathbf{V}_S\}$  represents this signal subspace and the set of matrices  $\{\mathbf{U}_N, \Sigma_N, \mathbf{V}_N\}$

represents the null subspace or, in practice for noisy data, the noise subspace. The  $J$  columns of  $\mathbf{U}$ , corresponding to these non-zero singular values that span the column space of  $\mathbf{Y}$ , are called the left singular vectors. Similarly, the  $J$  columns of  $\mathbf{V}$  are called the right singular vectors and they span the row space of  $\mathbf{Y}$ . Using these terms the SVD of  $\mathbf{Y}$  can be written in a more compact form as:

$$\mathbf{Y} = \mathbf{U}_{\mathcal{S}} \boldsymbol{\Sigma}_{\mathcal{S}} \mathbf{V}_{\mathcal{S}}^T = \sum_{j=1}^J \sigma_j \mathbf{u}_j \mathbf{v}_j^T. \quad (21.30)$$

Perturbation theory for the SVD is partially based on the link between the SVD and the PCA and symmetric Eigenvalue Decomposition (EVD). It is evident, that from the SVD of the matrix  $\mathbf{Y} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T$  with rank- $J \leq I \leq T$ , we have

$$\boxed{\mathbf{Y} \mathbf{Y}^T = \mathbf{U} \boldsymbol{\Sigma}_1^2 \mathbf{U}^T}, \quad (21.31)$$

$$\boxed{\mathbf{Y}^T \mathbf{Y} = \mathbf{V} \boldsymbol{\Sigma}_2^2 \mathbf{V}^T}, \quad (21.32)$$

where  $\boldsymbol{\Sigma}_1 = \text{diag}\{\sigma_1, \dots, \sigma_I\}$  and  $\boldsymbol{\Sigma}_2 = \text{diag}\{\sigma_1, \dots, \sigma_I\}$ . This means that the singular values of  $\mathbf{Y}$  are the positive square roots of the eigenvalues of  $\mathbf{Y}^T \mathbf{Y}$  and the eigenvectors  $\mathbf{U}$  of  $\mathbf{Y} \mathbf{Y}^T$  are the left singular vectors of  $\mathbf{Y}$ . Note that if  $I < T$ , the matrix  $\mathbf{Y} \mathbf{Y}^T$  will contain at least  $T - I$  additional eigenvalues that are not included as singular values of  $\mathbf{Y}$ .

An estimate  $\hat{\mathbf{R}}_{yy}$  of the covariance matrix corresponding to a set of (zero-mean) observed vectors  $\mathbf{y}(t) \in \mathbb{R}^I$  may be computed as  $\hat{\mathbf{R}}_{yy} = (1/T) \sum_{t=1}^T \mathbf{y}(t) \mathbf{y}^T(t)$ . An alternative equivalent way of computing  $\hat{\mathbf{R}}_{yy}$  is to form a data matrix  $\mathbf{y}^T = [\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)] \in \mathbb{R}^{I \times T}$  and represent the estimated covariance matrix by  $\hat{\mathbf{R}}_{yy} = \frac{1}{T} \mathbf{y} \mathbf{y}^T$ . Hence, the eigenvectors of the sample covariance matrix  $\hat{\mathbf{R}}_{yy}$  are the left singular vectors  $\mathbf{U}$  of  $\mathbf{y}$  and the singular values  $\sigma_j$  of  $\mathbf{y}$  are the positive square roots of the eigenvalues of  $\hat{\mathbf{R}}_{yy}$ .

From this discussion it follows that all the algorithms for PCA and EVD can be applied (after some simple tricks) to the SVD of an arbitrary matrix  $\mathbf{Y}$  without any need to directly compute or estimate the covariance matrix. The opposite is also true; the PCA or EVD of the covariance matrix  $\hat{\mathbf{R}}_{yy}$  can be performed via the SVD numerical algorithms. However, for large data matrices  $\mathbf{Y}$ , the SVD algorithms usually become more costly, than the relatively efficient and fast PCA adaptive algorithms. Several reliable and efficient numerical algorithms for the SVD do however, exist [48].

The approximation of the matrix  $\mathbf{Y}$  by a rank-1 matrix  $\sigma \mathbf{u} \mathbf{v}^T$  of two unknown vectors  $\mathbf{u} = [u_1, u_2, \dots, u_T]^T \in \mathbb{R}^T$  and  $\mathbf{v} = [v_1, v_2, \dots, v_I]^T \in \mathbb{R}^I$ , normalized to unit length, with an unknown scaling constant term  $\sigma$ , can be presented as follows:

$$\mathbf{Y} = \sigma \mathbf{u} \mathbf{v}^T + \mathbf{E}, \quad (21.33)$$

where  $\mathbf{E} \in \mathbb{R}^{T \times I}$  is the matrix of the residual errors  $e_{it}$ . In order to compute the unknown vectors we minimize the squared Euclidean error as [49]

$$D_1(\mathbf{Y} \| \sigma \mathbf{u} \mathbf{v}^T) = \|\mathbf{E}\|_F^2 = \sum_{it} e_{it}^2 = \sum_{i=1}^I \sum_{t=1}^T (y_{ti} - \sigma u_t v_i)^2. \quad (21.34)$$

The necessary conditions for the minimization of (21.34) are obtained by equating the gradients to zero:

$$\frac{\partial D_1}{\partial u_t} = -2\sigma \sum_{i=1}^I (y_{ti} - \sigma u_t v_i) v_i = 0, \quad (21.35)$$

$$\frac{\partial D_1}{\partial v_i} = -2\sigma \sum_{t=1}^T (y_{ti} - \sigma u_t v_i) u_t = 0. \quad (21.36)$$

These equations can be expressed as follows:

$$\sum_{i=1}^I y_{ti} v_i = \sigma u_t \sum_{i=1}^I v_i^2, \quad \sum_{t=1}^T y_{ti} u_t = \sigma v_i \sum_{t=1}^T u_t^2. \quad (21.37)$$

It is interesting to note that (taking into account that  $\sigma u_t = \sum_{i=1}^I y_{ti} v_i / \sum_{i=1}^I v_i^2$  (see Eq. (21.37)) the cost function (21.34) can be expressed as [49]

$$\begin{aligned} D_1 &= \|\mathbf{E}\|_F^2 = \sum_{i=1}^I \sum_{t=1}^T (y_{ti} - \sigma u_t v_i)^2 \\ &= \sum_{i=1}^I \sum_{t=1}^T y_{ti}^2 - 2 \sum_{t=1}^T (\sigma u_t) \sum_{i=1}^I (y_{ti} v_i) + \sum_{t=1}^T (\sigma u_t)^2 \sum_{i=1}^I v_i^2 \\ &= \sum_{i=1}^I \sum_{t=1}^T y_{ti}^2 - \frac{\sum_{i=1}^I (\sum_{t=1}^T y_{ti} v_i)^2}{\sum_{i=1}^I v_i^2}. \end{aligned} \quad (21.38)$$

In matrix notation the cost function can be written as

$$\|\mathbf{E}\|_F^2 = \|\mathbf{Y}\|_F^2 - \frac{\mathbf{v}^T \mathbf{Y}^T \mathbf{Y} \mathbf{v}}{\|\mathbf{v}\|_2^2} = \|\mathbf{Y}\|_F^2 - \sigma^2, \quad (21.39)$$

where the second term is called the Rayleigh quotient. The maximum value of the Rayleigh quotient is exactly equal to the maximum eigenvalue  $\lambda_1 = \sigma_1^2$ .

Taking into account that the vectors are normalized to unit length, that is,  $\mathbf{u}^T \mathbf{u} = \sum_{t=1}^T u_t^2 = 1$  and  $\mathbf{v}^T \mathbf{v} = \sum_{i=1}^I v_i^2 = 1$ , we can write the above equations in a compact matrix form as

$$\mathbf{Y} \mathbf{v} = \sigma \mathbf{u}, \quad \mathbf{Y}^T \mathbf{u} = \sigma \mathbf{v} \quad (21.40)$$

or equivalently (by substituting one of Eq. (21.40) into another)

$$\mathbf{Y}^T \mathbf{Y} \mathbf{v} = \sigma^2 \mathbf{v}, \quad \mathbf{Y} \mathbf{Y}^T \mathbf{u} = \sigma^2 \mathbf{u}, \quad (21.41)$$

which are classical eigenvalue problems which estimate the maximum eigenvalue  $\lambda_1 = \sigma_1^2 = \sigma_{\max}^2$  with the corresponding eigenvectors  $\mathbf{u}_1 = \mathbf{u}$  and  $\mathbf{v}_1 = \mathbf{v}$ . The solutions of these problems give the best first rank-1 approximation of Eq. (21.27).

In order to estimate the next singular values and the corresponding singular vectors, we may apply a deflation approach, that is,

$$\mathbf{Y}_j = \mathbf{Y}_{j-1} - \sigma_j \mathbf{u}_j \mathbf{v}_j^T \quad (j = 1, 2, \dots, J), \quad (21.42)$$

where  $\mathbf{Y}_0 = \mathbf{Y}$ . Solving the same optimization problem (21.34) for the residual matrix  $\mathbf{Y}_j$  yields the set of consecutive singular values and corresponding singular vectors. Repeating the reduction of the matrix yields the next set of solution until the deflation matrix  $\mathbf{Y}_{J+1}$  becomes zero. In the general case, we have:

$$\mathbf{Y}\mathbf{v}_j = \sigma_j \mathbf{u}_j, \quad \mathbf{Y}^T \mathbf{u}_j = \sigma_j \mathbf{v}_j \quad (j = 1, 2, \dots, J). \quad (21.43)$$

Using the property of orthogonality of the eigenvectors and the equality  $\mathbf{u}^T \mathbf{Y} \mathbf{v} = \mathbf{v}^T \mathbf{Y}^T \mathbf{u} = \sigma$ , we can estimate the precision of the matrix approximation with the first  $P \leq J$  pairs of singular vectors [49]:

$$\begin{aligned} \|\mathbf{Y} - \sum_{j=1}^P \sigma_j \mathbf{u}_j \mathbf{v}_j^T\|_F^2 &= \sum_{t=1}^T \sum_{i=1}^I (y_{ti} - \sum_{j=1}^J \sigma_j u_{tj} v_{ij})^2 \\ &= \|\mathbf{Y}\|_F^2 - \sum_{j=1}^P \sigma_j^2, \end{aligned} \quad (21.44)$$

and the residual error reduces exactly to zero when the number of singular values is equal to the matrix rank, that is, for  $P = J$ . Thus, we can write for the rank- $J$  matrix:

$$\|\mathbf{Y}\|_F^2 = \sum_{j=1}^J \sigma_j^2. \quad (21.45)$$

Note that the SVD can be formulated as the following constrained optimization problem:

$$\min_{\mathbf{U}, \mathbf{V}, \Sigma} \{\|\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T\|_F^2\}, \quad \text{s.t. } \mathbf{U}^T \mathbf{U} = \mathbf{I}, \quad \mathbf{V}^T \mathbf{V} = \mathbf{I}, \quad \text{diag}(\Sigma) \geq 0. \quad (21.46)$$

The cost function can be expressed as follows:

$$\begin{aligned} \|\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T\|_F^2 &= \text{tr}(\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T)^T \text{tr}(\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T) \\ &= \|\mathbf{Y}\|_F^2 - 2 \sum_{j=1}^J \sigma_j \mathbf{u}_j^T \mathbf{Y} \mathbf{v}_j + \sum_{j=1}^J \sigma_j^2. \end{aligned} \quad (21.47)$$

Hence, for  $J = 1$  (with  $\mathbf{u}_1 = \mathbf{u}$  and  $\mathbf{v}_1 = \mathbf{v}$ ) the problem (21.46) (called the best rank-1 approximation):

$$\min_{\mathbf{u}, \mathbf{v}} \{\|\mathbf{Y} - \sigma \mathbf{u} \mathbf{v}^T\|_F^2\}, \quad \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1, \quad \sigma > 0, \quad (21.48)$$

can be reformulated as an equivalent maximization problem (called the maximization variance approach):

$$\max_{\mathbf{u}, \mathbf{v}} \{\mathbf{u}^T \mathbf{Y} \mathbf{v}\}, \quad \text{s.t. } \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1, \quad (21.49)$$

with a maximal singular value  $\sigma = \mathbf{u}^T \mathbf{Y} \mathbf{v}$ .

### 1.21.2.5 Very large-scale PCA/SVD

Suppose that  $\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}^T$  is the SVD decomposition of an  $T \times I$ -dimensional matrix, where  $T$  is very large and  $I$  is moderate. Then matrices  $\Sigma$  and  $\mathbf{V}$  can be obtained from EDV/PCA decomposition of the  $I \times I$ -dimensional matrix  $\mathbf{Y}^T\mathbf{Y} = \mathbf{V}\Sigma^2\mathbf{V}^T$ . The orthogonal matrix  $\mathbf{U}$  can be then obtained from  $\mathbf{U} = \mathbf{Y}\mathbf{V}\Sigma^{-1}$ .

**Remark.** It should be noted that for moderate large-scale problems with  $I \gg T$  and typically  $I < 10^4$  we can more efficiently solve the SVD problem by noting that the PCA/SVD of a semi-positive definite matrix (of much smaller dimension)

$$\mathbf{R}_{\mathbf{YY}'} = \mathbf{YY}^T = \mathbf{U}\Lambda\mathbf{U}^T \in \mathbb{R}^{T \times T}, \quad (21.50)$$

with  $\mathbf{U} \in \mathbb{R}^{T \times J}$ , has the same eigenvalues as the matrix  $\mathbf{R}_{\mathbf{YY}'} = \mathbf{Y}^T\mathbf{Y}$ , since we can write  $\mathbf{R}_{\mathbf{YY}'} = \mathbf{YY}^T = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma\mathbf{U}^T = \mathbf{U}\Sigma^2\mathbf{U}^T$  and  $\mathbf{R}_{\mathbf{YY}'} = \mathbf{Y}^T\mathbf{Y} = \mathbf{V}\Sigma\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{V}\Sigma^2\mathbf{V}^T$  and  $\Sigma = \text{diag}\{\sigma_1, \dots, \sigma_J\}$ .

Moreover, if the vector  $\mathbf{v}$  is an eigenvector of the matrix  $\mathbf{R}_{\mathbf{YY}'} = \mathbf{Y}^T\mathbf{Y}$ , then  $\mathbf{Y}\mathbf{v}$  is an eigenvector of  $\mathbf{R}_{\mathbf{YY}'}$ , since we can write

$$\mathbf{Y}^T\mathbf{Y}\mathbf{v} = \lambda\mathbf{v}, \quad (\mathbf{YY}^T)(\mathbf{Y}\mathbf{v}) = \lambda(\mathbf{Y}\mathbf{v}). \quad (21.51)$$

In many practical applications dimension  $I \gg T$  could be so large that the data matrix  $\mathbf{Y} \in \mathbb{R}^{T \times I}$  can not be loaded into the memory of moderate class of computers. The simple solution is to partition of data matrix  $\mathbf{Y}$  into, say,  $Q$  slices as  $\mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_Q]$ , where the size of the  $q$ th slice  $\mathbf{Y}_q \in \mathbb{R}^{T \times (I/Q)}$  and can be optimized depending on the available computer memory [50, 51]. The scaled covariance matrix (cross matrix product) can be then calculated as

$$\mathbf{YY}^T = \frac{1}{Q} \sum_{q=1}^Q \mathbf{Y}_q \mathbf{Y}_q^T = \mathbf{U}\Sigma^2\mathbf{U}^T \in \mathbb{R}^{T \times T}, \quad (21.52)$$

since  $\mathbf{Y}_q = \mathbf{U}\Sigma\mathbf{V}_q^T$ , where  $\mathbf{V}_q \in \mathbb{R}^{(I/Q) \times J}$  are orthogonal matrices. Instead of computing the whole right singular matrix  $\mathbf{V} = \mathbf{Y}^T\mathbf{U}\Sigma^{-1} \in \mathbb{R}^{I \times J}$ , we can perform calculations on the slices of the partitioned data matrix  $\mathbf{Y}$  as  $\mathbf{V}_q^T = \Sigma^{-1}\mathbf{U}^T\mathbf{Y}_q$  for  $q = 1, 2, \dots, Q$ . The concatenated slices  $[\mathbf{V}_1^T, \mathbf{V}_2^T, \dots, \mathbf{V}_Q^T]^T$  form the matrix of the right singular vectors  $\mathbf{V}^T$ . Such approach do not require loading the entire data matrix  $\mathbf{Y}$  at once in the computer memory but instead use a sequential access to dataset. Of course, this method will work only if the dimension  $T$  is moderate, typically less than 10,000 and the dimension  $I$  can be theoretically arbitrary large.

### 1.21.2.6 Basic algorithms for PCA/SVD

There are three basic or fundamental methods to compute PCA/SVD:

- The minimal reconstruction error approach:

$$\mathbf{v}^* = \underset{\mathbf{v}^T\mathbf{v}=1}{\operatorname{argmin}} \left\{ \sum_{t=1}^T [y(t) - (\mathbf{v}^T \mathbf{y}(t))\mathbf{v}]^2 \right\}. \quad (21.53)$$

- The best rank-1 approximation approach:

$$(\mathbf{u}^*, \mathbf{v}^*, \sigma^*) = \underset{\mathbf{u}, \mathbf{v}, \sigma}{\operatorname{argmin}} \left\{ \|\mathbf{Y} - \sigma \mathbf{u} \mathbf{v}^T\|_F^2 \right\} \quad \text{s.t. } \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1, \quad \sigma > 0. \quad (21.54)$$

- The maximal variance approach:

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}} \{ \mathbf{v}^T \mathbf{Y}^T \mathbf{Y} \mathbf{v} \} \quad \text{s.t. } \|\mathbf{v}\|_2 = 1. \quad (21.55)$$

or in the more general form (for  $j = 1, 2, \dots, J$ ):

$$\begin{aligned} (\mathbf{u}_j^*, \mathbf{v}_j^*) &= \underset{\mathbf{u}_j, \mathbf{v}_j}{\operatorname{argmax}} \{ \mathbf{u}_j^T \mathbf{Y} \mathbf{v}_j \}, \\ \text{s.t. } \|\mathbf{u}_j\|_2 &= \|\mathbf{v}_j\|_2 = 1 \quad \mathbf{u}_j^T \mathbf{u}_i = 0, \quad \mathbf{v}_j^T \mathbf{v}_i = 0, \quad \forall i < j. \end{aligned} \quad (21.56)$$

All the above approaches are connected or linked and almost equivalent. Often we impose additional constraints such as sparseness and/or nonnegativity (see Section 1.21.2.6.3).

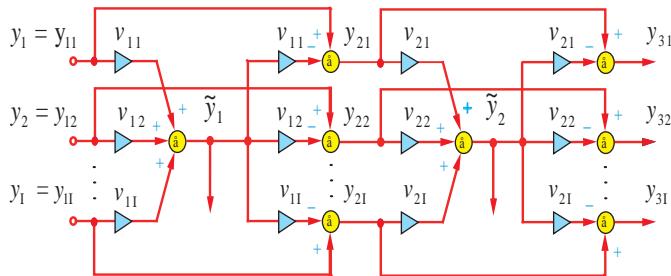
### 1.21.2.6.1 Simple iterative algorithm using minimal reconstruction error approach

One of the simplest and intuitively understandable approaches to the derivation of adaptive algorithms for the extraction of true principal components without need to compute explicitly the covariance matrix is based on self-association (also called self-supervising or the replicator principle) [30, 52]. According to this approach, we first compress the data vector  $\mathbf{y}(t)$  to one variable  $\tilde{y}_1(t) = \mathbf{v}_1^T \mathbf{y}(t)$  and next we attempt to reconstruct the original data from  $\tilde{y}_1(t)$  by using the transformation  $\hat{\mathbf{y}}(t) = \mathbf{v}_1 \tilde{y}_1(t)$ . Let us assume, that we wish to extract principal components (PCs) sequentially by employing the self-supervising principle (replicator) and a deflation procedure [30].

Let us consider a simple processing unit (see Figure 21.3)

$$\tilde{y}_1(t) = \mathbf{v}_1^T \mathbf{y}(t) = \sum_{i=1}^I v_{1i} y_i(t), \quad (21.57)$$

which extracts the first principal component, with  $\lambda_1 = E\{y_1^2(t)\}$ . Strictly speaking, the factor  $\tilde{y}_1(t)$  is called the first principal component of  $\mathbf{y}(t)$ , if the variance of  $\tilde{y}_1(t)$  is maximally large under the constraint that the principal vector  $\mathbf{v}_1$  has unit length.

**FIGURE 21.3**

The sequential extraction of principal components and the deflation procedure.

The vector  $\mathbf{v}_1 = [v_{11}, v_{12}, \dots, v_{1I}]^T$  should be determined in such a way that the reconstruction vector  $\hat{\mathbf{y}}(t) = \mathbf{v}_1 \tilde{\mathbf{y}}_1(t)$  will reproduce (reconstruct) the training vectors  $\mathbf{y}(t)$  as correctly as possible, according to the following cost function  $J(\mathbf{v}_1) = E\{\|\mathbf{e}_1(t)\|_2^2\} \approx \sum_{t=1}^T \|\mathbf{e}_1(t)\|_2^2$ , where  $\mathbf{e}_1(t) = \mathbf{y}(t) - \mathbf{v}_1 \tilde{\mathbf{y}}_1(t)$ .

In general, the loss (cost) function is expressed as

$$D_i(\mathbf{v}_i) = \sum_{t=1}^T \|\mathbf{e}_i(t)\|_2^2, \quad (21.58)$$

where

$$\mathbf{e}_j = \mathbf{y}_{j+1} = \mathbf{y}_j - \mathbf{v}_j \tilde{\mathbf{y}}_j, \quad \tilde{\mathbf{y}}_j = \mathbf{v}_j^T \mathbf{y}_j, \quad \mathbf{y}_1(t) = \mathbf{y}(t).$$

In order to increase the convergence speed, we can minimize the cost function (21.58) by employing the cascade recursive least-squares (CRLS) approach for optimal updating of the learning rate  $\eta_i$  [52,53]:

$$\mathbf{y}_1(t) = \mathbf{y}(t), \quad (21.59)$$

$$\tilde{\mathbf{y}}_j(t) = \mathbf{v}_j^T(t) \mathbf{y}_j(t) \quad (j = 1, 2, \dots, J), \quad (21.60)$$

$$\mathbf{v}_j(t+1) = \mathbf{v}_j(t) + \frac{\tilde{\mathbf{y}}_j(t)}{\eta_j(t)} [\mathbf{y}_j(t) - \tilde{\mathbf{y}}_j(t) \mathbf{v}_j(t)], \quad (21.61)$$

$$\eta_j(t+1) = \eta_j(t) + |\tilde{\mathbf{y}}_j(t)|^2, \quad \eta_j(0) = \frac{1}{T} \sum_{i=1}^I \sum_{t=1}^T y_{ji}^2(t), \quad (21.62)$$

$$\mathbf{y}_{j+1}(t) = \mathbf{y}_j(t) - \tilde{\mathbf{y}}_j(t) \mathbf{v}_{j*}, \quad (21.63)$$

$$\mathbf{v}_{j+1}(0) = \mathbf{v}_j - [\mathbf{v}_{1*}, \dots, \mathbf{v}_{j*}] [\mathbf{v}_1, \dots, \mathbf{v}_{j*}]^T \mathbf{v}_j, \quad (21.64)$$

where  $\mathbf{v}_{i*}$  denotes the vector  $\mathbf{v}_j(t)$  after achieving convergence. The above algorithm is fast and accurate for extracting sequentially an arbitrary number of eigenvectors in PCA.

### 1.21.2.6.2 Power methods for standard PCA

Many iterative algorithms for PCA exploit the Rayleigh quotient (RQ) of the specific covariance matrix as the cost function and power methods [54, 55]. The Rayleigh quotient  $R(\mathbf{v})$  is defined for  $\mathbf{v} \neq \mathbf{0}$ , as

$$R(\mathbf{v}) = R(\mathbf{v}, \mathbf{R}_{yy}) = \frac{\mathbf{v}^T \mathbf{R}_{yy} \mathbf{v}}{\mathbf{v}^T \mathbf{v}}, \quad (21.65)$$

It is worth to note that

$$\lambda_{\max} = \max R(\mathbf{v}, \mathbf{R}_{yy}), \quad \lambda_{\min} = \min R(\mathbf{v}, \mathbf{R}_{yy}), \quad (21.66)$$

where  $\lambda_{\max}$  and  $\lambda_{\min}$  denote the largest and the smallest eigenvalues of the covariance matrix  $\mathbf{R}_{yy}$ , respectively.

More generally, the critical points and critical values of  $R(\mathbf{v}, \mathbf{R}_{yy})$  are the eigenvectors and eigenvalues of  $\mathbf{R}_{yy}$ . Assuming that the eigenvalues of the covariance matrix can be ordered as  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_J$ , and that the principal eigenvector  $\mathbf{v}_1$  has unit length, i.e.,  $\mathbf{v}_1^T \mathbf{v}_1 = 1$ , we can estimate it using the following iterations

$$\mathbf{v}_1(k+1) = \frac{\mathbf{R}_{yy} \mathbf{v}_1(k)}{\mathbf{v}_1^T(k) \mathbf{R}_{yy} \mathbf{v}_1(k)}. \quad (21.67)$$

Taking into account that  $\tilde{\mathbf{y}}_1^{(k)}(t) = \mathbf{v}_1^T(k) \mathbf{y}(t)$  and  $\widehat{\mathbf{R}}_{yy} = \langle \mathbf{y}(t) \mathbf{y}^T(t) \rangle$ , we can then use the following simplified formula

$$\mathbf{v}_1(k+1) = \frac{\sum_{t=1}^T \tilde{\mathbf{y}}_1^{(k)}(t) \mathbf{y}(t)}{\sum_{t=1}^T [\tilde{\mathbf{y}}_1^{(k)}(t)]^2} \quad (21.68)$$

or more generally, for a number of higher PCs, we may use the deflation approach as

$$\mathbf{v}_j(k+1) = \frac{\sum_{t=1}^T \tilde{\mathbf{y}}_j^{(k)}(t) \mathbf{y}_j(t)}{\sum_{t=1}^T [\tilde{\mathbf{y}}_j^{(k)}(t)]^2} \quad (j = 1, 2, \dots, J), \quad (21.69)$$

where  $\tilde{\mathbf{y}}_j^{(t)}(t) = \mathbf{v}_j^T(k) \mathbf{y}_j(t)$ . After the convergence of the vector  $\mathbf{v}_j(k)$  to  $\mathbf{v}_{j*}$ , we perform the deflation as:  $\mathbf{y}_{j+1} = \mathbf{y}_j - \mathbf{v}_{j*} \tilde{\mathbf{y}}_j$ ,  $\mathbf{y}_1 = \mathbf{y}$ .

The above fast PCA algorithm can be derived also in a slightly modified form by minimizing the cost function:

$$\begin{aligned} D(\mathbf{v}, \mathbf{y}^{(k)}) &= \sum_{t=1}^T \|\mathbf{y}(t) - \tilde{\mathbf{y}}^{(k)} \mathbf{v}\|_2^2 \\ &= \sum_{t=1}^T \|\mathbf{y}(t)\|_2^2 + \|\mathbf{v}\|_2^2 \sum_{t=1}^T [\tilde{\mathbf{y}}^{(k)}(t)]^2 - 2\mathbf{v}^T \sum_{t=1}^T \tilde{\mathbf{y}}^{(k)}(t) \mathbf{y}(t), \end{aligned} \quad (21.70)$$

subject to the constraint  $\|\mathbf{v}\|_2 = 1$ . The cost function achieves equilibrium when the gradient of  $D$  is zero, i.e.,

$$\mathbf{v}_* = \frac{\sum_{t=1}^T \tilde{\mathbf{y}}_*(t) \mathbf{y}(t)}{\sum_{t=1}^T \tilde{\mathbf{y}}_*^2(t)}. \quad (21.71)$$

This suggests the following iteration formula [55,56]:

$$\mathbf{v}(k+1) = \frac{\sum_{t=1}^T \tilde{y}^{(k)}(t) \mathbf{y}(t)}{\|\sum_{t=1}^T \tilde{y}^{(k)}(t) \mathbf{y}(t)\|_2}. \quad (21.72)$$

**Remark.** It should be noted that the convergence rate of the power algorithm depends on a ratio  $\lambda_2/\lambda_1$ , where  $\lambda_2$  is the second largest eigenvalue of  $\mathbf{R}_{yy}$ . This ratio is generally smaller than one, allowing adequate convergence of the algorithm. However, if the eigenvalue  $\lambda_1 = \lambda_{\max}$  has one or more other eigenvalues of  $\mathbf{R}_{yy}$  close by, in other words, when  $\lambda_1$  belongs to a cluster of eigenvalues then the ratio can be very close to one, causing very slow convergence, and as a consequence, the estimated eigenvector  $\mathbf{v}$  may be inaccurate for noisy data or collinear components. For multiple eigenvalues, the power method will fail to converge.

### 1.21.2.6.3 Extensions of standard PCA

#### 1.21.2.6.3.1 Sparse PCA

The importance of the standard PCA is mainly due to the following three important properties:

1. The principal components sequentially capture the maximum variability (variance) of the data matrix  $\mathbf{Y}$ , thus guaranteeing minimal information loss in the sense of mean squared errors.
2. The principal components are uncorrelated, i.e.,  $E\{\tilde{y}_i \tilde{y}_j\} = \delta_{ij} \lambda_j$ .
3. The PCs are hierarchically organized with the respect to decreasing values of their variances (eigenvalues of the covariance matrix).

The standard PCA/SVD has also several disadvantages, especially for large-scale and noisy problems. A particular disadvantage that the standard principal components  $\tilde{y}_j(t) = \mathbf{v}_j^T \mathbf{y}(t) = \sum_{i=1}^I v_{ij} y_i(t)$  are usually a linear combination of all the variables  $y_j(t)$ . In other words, all weights  $v_{ij}$  (referred as loadings) are not zero. This means that the principal vectors  $\mathbf{v}_j$  are dense (not sparse) what makes often the physical interpretation of the principal components difficult in many applications. For example, in many applications (from biology to image understanding) the coordinate axes have a physical interpretations (each axis might correspond to a specific feature), but only if the components are sparsely represented, i.e., by a very few non zero loadings (coordinates). Recently several modifications of PCA have been proposed which impose some sparseness for the principal (basis) vectors and corresponding components are called sparse principal components [54,57–62]. The main idea in SPCA is to force the basis vectors to be sparse, however the sparsity profile should be adjustable or well controlled via some parameters in order to discover specific features in the observed data. In other words, our objective is to estimate sparse principal components, i.e., the sets of sparse vectors  $\mathbf{v}_j$  spanning a low-dimensional space that represent most of the variance present in the data  $\mathbf{Y}$ .

Many methods have been proposed for estimating sparse principal components, based on either the maximum-variance property of the principal components, or on the regression/reconstruction error property. The most important approaches which extract the principal eigenvectors sequentially can be summarized as the following constrained or penalized optimization problems [54,57,59–63]:

1.  $l_1$ -norm constrained SPCA or the SCoTLASS procedure [59] uses the maximal variance characterization for the principal components. The first sparse principal component solves the optimization problem:

$$\max_{\mathbf{v}} \{\mathbf{v}^T \mathbf{Y}^T \mathbf{Y} \mathbf{v}\} \quad \text{s.t. } \|\mathbf{v}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_1 \leq c_1 \quad (21.73)$$

and subsequent components solve the same problem but with the additional constraint that they must be orthogonal to the previous components. When the parameter  $c_1$  is sufficiently large, then the optimization problem (21.73) simply yields the first standard principal component of  $\mathbf{Y}$ , and when  $c_1$  is small, then the solution is sparse.

2.  $l_1$ -norm penalized SPCA [64]

$$\max_{\mathbf{v}} \{\mathbf{v}^T \mathbf{Y}^T \mathbf{Y} \mathbf{v} - \alpha \|\mathbf{v}\|_1\} \quad \text{s.t. } \|\mathbf{v}\|_2^2 \leq 1, \quad (21.74)$$

where the penalty parameter  $\alpha \geq 0$  controls the level of sparseness.

3.  $l_0$ -norm penalized SPCA [57]

$$\max_{\mathbf{v}} \{\mathbf{v}^T \mathbf{Y}^T \mathbf{Y} \mathbf{v} - \alpha \|\mathbf{v}\|_0\} \quad \text{s.t. } \|\mathbf{v}\|_2^2 \leq 1, \quad (21.75)$$

4. SPCA using Penalized Matrix decomposition (PMD) [61, 64]

$$\max_{\mathbf{u}, \mathbf{v}} \{\mathbf{u}^T \mathbf{Y} \mathbf{v}\}, \quad \text{s.t. } \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad P_1(\mathbf{u}) \leq c_1, \quad P_2(\mathbf{v}) \leq c_2, \quad (21.76)$$

where positive parameters  $c_1, c_2$  control sparsity level and the convex penalty functions  $P_1(\mathbf{u})$  and  $P_2(\mathbf{v})$  can take a variety of different forms. Useful examples are:

$$\begin{aligned} P(\mathbf{v}) &= \|\mathbf{v}\|_1 = \sum_{i=1}^I |v_i| \quad \text{LASSO,} \\ P(\mathbf{v}) &= \|\mathbf{v}\|_0 = \sum_{i=1}^I |\text{sign}(v_i)| \quad \text{CARDINALITY PENALTY,} \\ P(\mathbf{v}) &= \sum_{i=1}^I |v_i| + \lambda \sum_{i=2}^I |v_i - v_{i-1}| \quad \text{FUSED LASSO.} \end{aligned}$$

5. SPCA via regularized SVD (sPCA-rSVD) [54, 65]

$$\max_{\mathbf{u}, \mathbf{v}} \{\mathbf{u}^T \mathbf{Y} \mathbf{v} - \alpha P(\mathbf{v})\} \quad \text{s.t. } \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1, \quad (21.77)$$

6. Two-way functional PCA/SVD [66]

$$\max_{\mathbf{u}, \mathbf{v}} \{\mathbf{u}^T \mathbf{Y} \mathbf{v} - \frac{\alpha}{2} P_1(\mathbf{u}) P_2(\mathbf{v})\}, \quad \text{s.t. } \|\mathbf{u}\|_2^2 \leq 1, \quad \|\mathbf{v}\|_2^2 \leq 1. \quad (21.78)$$

## 7. Sparse SVD [67]

$$\max_{\mathbf{u}, \mathbf{v}} \left\{ \mathbf{u}^T \mathbf{Y} \mathbf{v} - \frac{1}{2} \mathbf{u}^T \mathbf{u} \mathbf{v}^T \mathbf{v} - \frac{\alpha_1}{2} P_1(\mathbf{u}) - \frac{\alpha_2}{2} P_2(\mathbf{v}) \right\}. \quad (21.79)$$

## 8. Generalized SPCA [68]

$$\max_{\mathbf{u}, \mathbf{v}} \left\{ \mathbf{u}^T \mathbf{Q} \mathbf{Y} \mathbf{R} \mathbf{v} - \frac{\alpha_1}{2} P_1(\mathbf{u}) - \frac{\alpha_2}{2} P_2(\mathbf{v}) \right\}, \quad \text{s.t. } \mathbf{u}^T \mathbf{Q} \mathbf{u} \leq 1, \mathbf{v}^T \mathbf{R} \mathbf{v} \leq 1, \quad (21.80)$$

where  $\mathbf{Q} \in \mathbb{R}^{T \times T}$  and  $\mathbf{R} \in \mathbb{R}^{I \times I}$  are symmetric positive definite matrices.

## 9. Generalized nonnegative SPCA [63]

$$\max_{\mathbf{u}, \mathbf{v}} \{ \mathbf{u}^T \mathbf{Y} \mathbf{R} \mathbf{v} - \alpha \|\mathbf{v}\|_1 \}, \quad \text{s.t. } \mathbf{u}^T \mathbf{u} \leq 1, \mathbf{v}^T \mathbf{R} \mathbf{v} \leq 1, \mathbf{v} \geq 0. \quad (21.81)$$

## 10. Robust SPCA [69]

$$\max_{\mathbf{v}} \{ Var(\mathbf{v}^T \mathbf{y}(1), \mathbf{v}^T \mathbf{y}(2), \dots, \mathbf{v}^T \mathbf{y}(T)) - \alpha \|\mathbf{v}\|_1 \}, \quad \text{s.t. } \mathbf{v}^T \mathbf{v} = 1, \quad (21.82)$$

where  $\mathbf{y}(t) \in \mathbb{R}^I$  ( $t = 1, 2, \dots, T$ ) are multivariate observations collected in the rows of the data matrix  $\mathbf{Y} \in \mathbb{R}^{T \times I}$  and  $Var$  is a robust variance measure.

There exist some connections between the above listed optimizations problems, and in fact they often deliver almost identical or at least similar solutions. All the optimization problems addressed here are rather difficult, generally non-convex maximization problems, and we can not make a claim with respect their global optimality. Even if the goal of obtaining a local minimizer is, in general, unattainable we must be content with convergence to a stationary point. However, using so formulated optimization problems, the common and interesting feature of the resulting algorithms is, that they provide a closed-form iterative updates (which is a generalized power method):

$$\mathbf{v} = \frac{\mathcal{S}(\mathbf{Y}^T \mathbf{Y} \mathbf{v})}{\|\mathcal{S}(\mathbf{Y}^T \mathbf{Y} \mathbf{v})\|_2}, \quad (21.83)$$

or

$$\mathbf{u} = \frac{\mathcal{S}(\mathbf{Y} \mathbf{v})}{\|\mathcal{S}(\mathbf{Y} \mathbf{v})\|_2}, \quad \mathbf{v} = \frac{\mathcal{S}(\mathbf{Y}^T \mathbf{u})}{\|\mathcal{S}(\mathbf{Y}^T \mathbf{u})\|_2}, \quad (21.84)$$

where  $\mathcal{S}$  is a simple operator (shrinkage) which can be written in explicit form and can be efficiently computed [54, 60].

For example, by applying the optimization problem (21.77) we obtain the sparse SVD algorithm. Sparse SVD can be performed by solving the following optimization problem:

$$\max_{\mathbf{u}, \mathbf{v}} \{ \mathbf{u}^T \mathbf{Y} \mathbf{v} - \alpha_1 \|\mathbf{u}\|_1 - \alpha_2 \|\mathbf{v}\|_1 \} \quad (21.85)$$

$$\text{s.t. } \mathbf{u}^T \mathbf{u} \leq 1, \mathbf{v}^T \mathbf{v} \leq 1, \quad (21.86)$$

which leads to the following update formulas:

$$\mathbf{u} = \frac{\mathcal{S}_1(\mathbf{Y}\mathbf{v}, \alpha_1)}{\|\mathcal{S}_1(\mathbf{Y}\mathbf{v}, \alpha_1)\|_2}, \quad \mathbf{v} = \frac{\mathcal{S}_1(\mathbf{Y}^T\mathbf{u}, \alpha_2)}{\|\mathcal{S}_1(\mathbf{Y}^T\mathbf{u}, \alpha_2)\|_2}, \quad (21.87)$$

where the non-negative parameters  $\alpha_1\alpha_2$  control sparsity level and the operator  $\mathcal{S}_1$  means the shrinkage function defined as

$$\mathcal{S}_1(\mathbf{Y}\mathbf{v}, \alpha) = \text{sign}(\mathbf{Y}\mathbf{v})[|\mathbf{Y}\mathbf{v}| - \alpha]_+. \quad (21.88)$$

In fact, when  $\mathcal{S}$  is the identity operator, the above update schemes are nothing else but power method for the standard PCA/SVD (see Algorithm 1).

---

**Algorithm 1.** Power Method for Sparse PCA/SVD

---

**Require:** Data matrix  $\mathbf{Y}^{(1)} = \mathbf{Y}$ ,

sparsity coefficients  $\alpha_n \geq 0$ , ( $n = 1, 2$ ), initial vectors  $\mathbf{u}$  and  $\mathbf{v}$

1: For  $j = 1, 2, \dots, J$ :

    Repeat until convergence of  $\mathbf{u}_j$  and  $\mathbf{v}_j$ ;

$$2: \quad \mathbf{u}_j = \frac{\mathcal{S}_1(\mathbf{Y}^{(j)}\mathbf{v}_j, \alpha_1)}{\|\mathcal{S}_1(\mathbf{Y}^{(j)}\mathbf{v}_j, \alpha_1)\|_2};$$

$$3: \quad \mathbf{v}_j = \frac{\mathcal{S}_1(\mathbf{Y}^{(j)T}\mathbf{u}_j, \alpha_2)}{\|\mathcal{S}_1(\mathbf{Y}^{(j)T}\mathbf{u}_j, \alpha_2)\|_2};$$

$$4: \quad \sigma_j = \mathbf{u}_j^T \mathbf{Y}^{(j)} \mathbf{v}_j;$$

$$5: \quad \mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} - \sigma_j \mathbf{u}_j \mathbf{v}_j^T;$$

6: **return**  $\mathbf{U}^* = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_J]$ ,  $\mathbf{V}^* = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J]$ ,  $\Sigma^* = \text{diag}\{\sigma_1, \dots, \sigma_J\}$ .

---

In contrast to standard PCA, SPCA enables to reveal often multi-scale hierarchical structures from the data [62]. For example, for EEG data the SPCA generates spatially localized, narrow bandpass functions as basis vectors, thereby achieving a joint space and frequency representation, what is impossible to obtain by using standard PCA. The algorithm for sparse PCA/SVD can be easily extended to sparse CCA/PLS (see Section 1.21.2.6.6).

#### 1.21.2.6.4 Deflation techniques for SPCA

In sparse PCA the eigenvectors are not necessary mutually orthogonal. In fact, we usually alleviate or even completely ignore the orthogonality constraints. Due to this reason the standard Hotelling deflation procedure:

$$\mathbf{R}_{yy}^{(j+1)} \leftarrow \mathbf{R}_{yy}^{(j)} - \lambda_j \mathbf{v}_j \mathbf{v}_j^T \quad (j = 1, 2, \dots, J), \quad (21.89)$$

does not guarantee the positive semi-definiteness of the covariance matrix.

Due to this reason we usually use the Schur compliment deflation [70]:

$$\mathbf{R}_{yy}^{(j+1)} \leftarrow \mathbf{R}_{yy}^{(j)} - \mathbf{R}_{yy}^{(j)} \mathbf{v}_j \mathbf{v}_j^T \mathbf{R}_{yy}^{(j)} / \lambda_j \quad (21.90)$$

or alternatively the projection deflation

$$\mathbf{R}_{yy}^{(j+1)} \leftarrow (\mathbf{I} - \mathbf{v}_j \mathbf{v}_j^T) \mathbf{R}_{yy}^{(j)} (\mathbf{I} - \mathbf{v}_j \mathbf{v}_j^T). \quad (21.91)$$

Empirical studies by Mackey [70] indicate that the Schur deflation and the projection deflation outperform the standard Hotelling's deflation consistently for SPCA, and usually the projection deflation is even better than (or at least comparable with) the Schur deflation.

---

**Algorithm 2.** Deflation Algorithm for Sparse PCA [70]

---

**Require:** Covariance matrix  $\mathbf{R}_{yy}^{(0)} = \mathbf{R}_{yy}$  and  $J$ ,  
 1: (1)  $\mathbf{Q}_0 \leftarrow \mathbf{I}$ ,  
 2: For  $j = 1, 2, \dots, J$ :  
 3:    $\mathbf{v}_j = \operatorname{argmax}\{\mathbf{v}^T \mathbf{R}^{(j-1)} \mathbf{v}\} \quad \text{s.t. } \|\mathbf{v}\|_1 \leq 1$ ;  
 4:    $\mathbf{q}_j = \mathbf{Q}_{j-1} \mathbf{v}_j$ ;  
 5:    $\mathbf{R}^{(j)} \leftarrow (\mathbf{I} - \mathbf{q}_j \mathbf{q}_j^T) \mathbf{R}_{yy}^{(j-1)} (\mathbf{I} - \mathbf{q}_j \mathbf{q}_j^T)$ ;  
 6:    $\mathbf{Q}^{(j)} = \mathbf{Q}^{(j-1)} (\mathbf{I} - \mathbf{q}_j \mathbf{q}_j^T)$ ;  
 7:    $\mathbf{v}^{(j)} = \mathbf{v}^{(j)}/\|\mathbf{v}^{(j)}\|_2$ ;  
 8: **return**  $\mathbf{V}^* = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J]$ .

---

### 1.21.2.6.5 Kernel PCA

Standard PCA applies linear transformation for dimensionality reduction and may not be effective for nonlinear data. In kernel PCA we apply implicitly nonlinear transformation to potentially very high dimensional feature space  $\mathcal{F}$ .

Let consider a data matrix  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{I \times N}$  with vectors  $\mathbf{x}_n \in \mathbb{R}^I$  (typically, with  $N < I$ ), which are nonlinearly projected feature subspace

$$\phi : \mathbf{x}_n \in \mathbb{R}^I \rightarrow \phi(\mathbf{x}_n) \in \mathcal{F}. \quad (21.92)$$

The covariance matrix in the feature space can be expressed as

$$\hat{\mathbf{C}} = \frac{1}{N} \Phi(\mathbf{X}) \Phi^T(\mathbf{X}) = \frac{1}{N} \sum_{n=1}^N \phi(\mathbf{x}_n) \phi^T(\mathbf{x}_n), \quad (21.93)$$

where  $\Phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_N)]$  (we assume without loss of generality that  $\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$ ). However, in practice, we almost never compute the nonlinear mapping  $\phi(\mathbf{x})$  explicitly, since the feature space can be very high dimensional in the kernel method, but rather exploit so called "kernel trick", i.e., employ the nonlinear mapping implicitly [71].

Let  $\mathbf{v}_j$  be eigenvector of the covariance matrix  $\hat{\mathbf{R}}_{\mathbf{xx}} = 1/N \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T$ , then the vector  $\mathbf{v}_j$  belongs to linear space spanned by the data points  $\mathbf{x}_n$  ( $n = 1, 2, \dots, N$ ) since  $\hat{\mathbf{R}}_{\mathbf{xx}} \mathbf{v}_j = \lambda_j \mathbf{v}_j$  implies that  $\mathbf{v}_j = 1/(\lambda_j N) \sum_{n=1}^N \mathbf{x}_n (\mathbf{x}_n^T \mathbf{v}) = \sum_{n=1}^N \alpha_{jn} \mathbf{x}_n$ . So, the standard PCA can be formulated in dual form as  $\mathbf{X}^T \mathbf{X} \boldsymbol{\alpha}_j = N \lambda_j \boldsymbol{\alpha}_j$ , where  $\mathbf{K} = \mathbf{X}^T \mathbf{X} \in \mathbb{R}^{N \times N}$  is referred to as linear kernel matrix with entries  $k_{nm} = \mathbf{x}_n^T \mathbf{x}_m$  expressed by inner products.

Similarly, eigenvector  $\mathbf{v}_j$  of the covariance matrix  $\hat{\mathbf{C}}$  defined by Eq. (21.93) can be written as  $\mathbf{v}_j = \sum_{n=1}^N \alpha_{jn} \phi(\mathbf{x}_n)$ . Taking into account that

$$\hat{\mathbf{C}} \mathbf{v}_j = \lambda_j \mathbf{v}_j \quad (j = 1, 2, \dots, J) \quad (21.94)$$

we can write

$$\left( \sum_{n=1}^N \phi(\mathbf{x}_n) \phi^T(\mathbf{x}_n) \right) \left( \sum_{m=1}^N \alpha_m \phi(\mathbf{x}_m) \right) = \lambda_j \left( \sum_{m=1}^N \alpha_{jm} \phi(\mathbf{x}_m) \right). \quad (21.95)$$

By multiplying the both side of the above equation by  $\phi^T(\mathbf{x}_n)$ , we obtain

$$\sum_{n=1}^N \sum_{m=1}^N \phi^T(\mathbf{x}_n) \phi(\mathbf{x}_n) \phi^T(\mathbf{x}_n) \phi(\mathbf{x}_m) \alpha_{jm} = \lambda_j \sum_{m=1}^N \alpha_{jm} \phi^T(\mathbf{x}_n) \phi(\mathbf{x}_m), \quad (21.96)$$

which can be written in a compact matrix form as  $\mathbf{K}^2 \boldsymbol{\alpha}_j = \lambda_j \mathbf{K} \boldsymbol{\alpha}_j$  or equivalently

$$\mathbf{K} \boldsymbol{\alpha}_j = \lambda_j \boldsymbol{\alpha}_j, \quad (21.97)$$

where  $\mathbf{K}$  is a kernel positive definite symmetric matrix with entries defined as  $k_{nm} = \phi^T(\mathbf{x}_n) \phi(\mathbf{x}_m)$ . We can estimate the kernel PCA features for new sample as

$$y_j(\mathbf{x}) = \phi^T(\mathbf{x}) \mathbf{v}_j = \sum_{n=1}^N \alpha_{jn} k(\mathbf{x}, \mathbf{x}_n) \quad (21.98)$$

Summarizing, the Kernel PCA algorithm can be formulated as follows:

1. Step 1: Select kernel function (e.g., Gaussian kernel:  $k_{nm}(\mathbf{x}_n, \mathbf{x}_m) = \exp(-\|\mathbf{x}_n - \mathbf{x}_m\|^2/2\sigma^2)$  or polynomial kernel:  $k_{nm}(\mathbf{x}_n, \mathbf{x}_m) = (\mathbf{x}_n^T \mathbf{x}_m + c)^d$ ) and compute the kernel matrix with entries  $k_{nm} = \phi^T(\mathbf{x}_n) \phi(\mathbf{x}_m)$  ( $n, m = 1, 2, \dots, N$ ) for training set.
2. Step 2: Compute eigenvalues and eigenvectors pairs of  $\mathbf{K}$  as  $\{\lambda_j, \boldsymbol{\alpha}_j\}$  ( $j = 1, 2, \dots, J$ ).
3. Step 3: Normalize the eigenvectors  $\boldsymbol{\alpha}_j \leftarrow \boldsymbol{\alpha}_j / \lambda_j$ .
4. Step 4: To project a test feature  $\phi(\mathbf{x})$  onto  $\mathbf{v}_j$  we need to compute  $y_j(\mathbf{x}) = \phi^T(\mathbf{x}) \mathbf{v}_j = \phi^T(\mathbf{x}) (\sum_{n=1}^N \alpha_{jn} \phi(\mathbf{x}_n)) = \sum_{n=1}^N \alpha_{jn} k(\mathbf{x}, \mathbf{x}_n)$ . So, explicit nonlinear mapping  $\phi(\cdot)$  is not required here.

### 1.21.2.6.6 Sparse CCA

The Canonical Correlation Analysis (CCA), introduced by Hotelling [72], can be considered as a generalization or extension of PCA (see [73, 74]). The standard CCA is a classical method for determining the relationship between two sets of variables. Given two zero-mean (i.e., centered) data sets  $\mathbf{X} \in \mathbb{R}^{I \times N}$  and  $\mathbf{Y} \in \mathbb{R}^{I \times M}$  on the same set of  $I$  observations, CCA seeks linear combinations of the variables in  $\mathbf{X}$  and the variables in  $\mathbf{Y}$  that are maximally correlated with each other. Formally, the classical CCA computes two projection vectors  $\mathbf{w}_x^{(1)} = \mathbf{w}_x \in \mathbb{R}^N$  and  $\mathbf{w}_y^{(1)} = \mathbf{w}_y \in \mathbb{R}^M$  such that the correlation coefficient

$$\rho = \frac{\mathbf{w}_x^T \mathbf{X}^T \mathbf{Y} \mathbf{w}_y}{\sqrt{(\mathbf{w}_x^T \mathbf{X}^T \mathbf{X} \mathbf{w}_x)(\mathbf{w}_y^T \mathbf{Y}^T \mathbf{Y} \mathbf{w}_y)}} \quad (21.99)$$

is maximized.

In a similar way, we can formulate a kernel CCA by replacing inner product matrices by kernel matrices:

$$\rho = \max_{\alpha_x, \alpha_y} \frac{\alpha_x^T \mathbf{K}_x^T \mathbf{K}_y \alpha_y}{\sqrt{(\alpha_x^T \mathbf{K}_x^T \mathbf{K}_x \alpha_x)(\alpha_y^T \mathbf{K}_y^T \mathbf{K}_y \alpha_y)}}, \quad (21.100)$$

where  $\mathbf{K}_x \in \mathbb{R}^{I \times I}$  and  $\mathbf{K}_y \in \mathbb{R}^{I \times I}$  are suitably designed kernel matrices. The above optimization problem can be solved by generalized eigenvalue decomposition.

Since  $\rho$  is invariant to the scaling of the vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$ , the standard CCA can be formulated equivalently as the following constrained optimization problem:

$$\max_{\mathbf{w}_x, \mathbf{w}_y} \{(\mathbf{w}_x^T \mathbf{X}^T \mathbf{Y} \mathbf{w}_y)^2\} \quad \text{s.t. } \mathbf{w}_x^T \mathbf{X}^T \mathbf{X} \mathbf{w}_x = \mathbf{w}_y^T \mathbf{Y}^T \mathbf{Y} \mathbf{w}_y = 1. \quad (21.101)$$

We will refer to  $\mathbf{t}_1 = \mathbf{X} \mathbf{w}_x$  and  $\mathbf{u}_1 = \mathbf{Y} \mathbf{w}_y$  as the canonical variables.

For sparse CCA we usually assume that the columns of  $\mathbf{X}$  and  $\mathbf{Y}$  have been standardized to have mean zero and standard deviation one.

In order to obtain sparse CCA we must impose suitable sparsity constraints on the canonical vectors, for example by applying the PMD approach [61, 64]:

$$\max_{\mathbf{w}_x, \mathbf{w}_y} \{(\mathbf{w}_x^T \mathbf{X}^T \mathbf{Y} \mathbf{w}_y)^2\} \quad \text{s.t. } \|\mathbf{w}_x\|_2^2 \leq 1, \|\mathbf{w}_y\|_2^2 \leq 1, P_1(\mathbf{w}_x) \leq c_1, P_2(\mathbf{w}_y) \leq c_2, \quad (21.102)$$

where  $P_1$  and  $P_2$  are convex penalty functions and positive parameters  $c_1, c_2$  control sparsity level (see Section 1.21.2.6.3). Since  $P_1$  and  $P_2$  are generally chosen to yield sparse canonical vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$ , we call this criterion the sparse CCA.

Note that since the data are standardized, cross product matrices  $\mathbf{X}^T \mathbf{X}$  and  $\mathbf{Y}^T \mathbf{Y}$  are approximated by identity matrices, and consequently the constraints  $\mathbf{w}_x^T \mathbf{X}^T \mathbf{X} \mathbf{w}_x \leq 1$  and  $\mathbf{w}_y^T \mathbf{Y}^T \mathbf{Y} \mathbf{w}_y \leq 1$  may be replaced by  $\|\mathbf{w}_x\|_2^2 \leq 1$  and  $\|\mathbf{w}_y\|_2^2 \leq 1$ , respectively [64].

The algorithms for the sparse CCA will be similar to sparse SVD/PCA (see Algorithm 3) in which the data matrix  $\mathbf{Y}$  in the SPCA should be replaced by  $\mathbf{Z} = \mathbf{X}^T \mathbf{Y}$  in the SCCA and the vectors  $\mathbf{u}$  and  $\mathbf{v}$  by the vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$ , respectively.

In order to compute multiple canonical vectors for CCA we need to apply a suitable deflation in a similar way as for SPCA. For example, we can compute the vectors  $\mathbf{w}_x^{(j)}$  and  $\mathbf{w}_y^{(j)}$  for  $j = 1, 2, \dots, J$  by applying criterion (21.102) to the data sets:

$$\mathbf{Z}^{(1)} = \mathbf{X}^T \mathbf{Y}, \quad \mathbf{Z}^{(j+1)} = \mathbf{Z}^{(j)} - (\mathbf{w}_x^{(j)T} \mathbf{Z}^{(j)} \mathbf{w}_y^{(j)}) \mathbf{w}_x^{(j)} \mathbf{w}_y^{(j)T}. \quad (21.103)$$

Note that the deflation and generalized power method do not impose automatically orthogonality constraints vectors  $\mathbf{w}_x^{(j)}$  and  $\mathbf{w}_y^{(j)}$ . In order to enforce orthogonality for the subsequently computed on the vectors  $\mathbf{w}_x^{(j+1)}$ , we can apply the Gram-Schmidt orthogonalization method [68]:

$$\mathbf{w}_x^{(j+1)} = \frac{(\mathbf{I} - \mathbf{W}_x^{(j)} \mathbf{W}_x^{(j)T})(\mathbf{X}^T \mathbf{Y} \mathbf{w}_y^{(j)})}{\|(\mathbf{I} - \mathbf{W}_x^{(j)} \mathbf{W}_x^{(j)T})(\mathbf{X}^T \mathbf{Y} \mathbf{w}_y^{(j)})\|_2} \quad (j = 1, 2, \dots), \quad (21.104)$$

where  $\mathbf{W}_x^{(j)} = [\mathbf{w}_x^{(1)}, \mathbf{w}_x^{(2)}, \dots, \mathbf{w}_x^{(j)}] \in \mathbb{R}^{N \times j}$ . Similar formulas can be written for  $\mathbf{w}_y^{(j+1)}$ .

### 1.21.2.6.7 Sparse partial least squares

The Partial Least Squares (PLS) methods, originally developed in chemometrics and econometrics by Herman Wold and his coworkers (see [75] and references therein) are particularly suitable for the analysis of relationships among multi-modal brain data (e.g., EEG, MEG, ECoG (electrocorticogram), fMRI) or relationships between measures of brain activity and behavior data [76,77]. The standard PLS approaches have been recently summarized in [75,78] and their suitability to model relationships between brain activity and behavior (experimental design) has been highlighted in [76].

There are two related basic PLS methods: PLS correlation (PLSC), which analyzes correlations or (associations) between two or more sets of data (e.g., two modalities brain data or brain and behavior data) and PLS regression (PLSR) methods, which attempt to predict one set of data from another independent set of data that constitutes the predictors (e.g., predict experimental behavior data from brain data such as multichannel ECoG or scalp EEG from ECoG by performing simultaneous recordings for epileptic patients).

In order to predict the response variables represented by the matrix  $\mathbf{Y}$  from the independent variables  $\mathbf{X}$ , the standard PLSR techniques find a set of common orthogonal latent variables (also called latent vectors, score vectors or components) by projecting both  $\mathbf{X}$  and  $\mathbf{Y}$  onto a subspace, which ensures a maximal covariance between the latent variables of  $\mathbf{X}$  and  $\mathbf{Y}$ . In other words, the prediction is achieved by simultaneous approximative decompositions training data sets:  $\mathbf{X} \in \mathbb{R}^{I \times N}$  and  $\mathbf{Y} \in \mathbb{R}^{I \times M}$  into factor matrices (components) ( $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_J] \in \mathbb{R}^{I \times J}$ ,  $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_J] \in \mathbb{R}^{N \times J}$  and  $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_J] \in \mathbb{R}^{I \times J}$ ,  $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_J] \in \mathbb{R}^{M \times J}$ ), respectively:

$$\boxed{\mathbf{X} = \mathbf{TP}^T + \mathbf{E} = \sum_{j=1}^J \mathbf{t}_j \mathbf{p}_j^T + \mathbf{E},} \quad (21.105)$$

$$\boxed{\mathbf{Y} = \mathbf{UQ}^T + \mathbf{F} = \sum_{j=1}^J \mathbf{u}_j \mathbf{q}_j^T + \mathbf{F},} \quad (21.106)$$

with the constraint that these components explain as much as possible of the covariance between  $\mathbf{X}$  and  $\mathbf{Y}$  [64,77]. In other words, the key concept of basic PLS algorithms is to maximize covariance between the vectors  $\mathbf{t}_j$  and  $\mathbf{u}_j$  expressed in a general form as:

$$\boxed{\max\{g(\mathbf{t}_j^T \mathbf{u}_j)\} \quad \forall j,} \quad (21.107)$$

subject to suitable constraints, where the function  $g(\cdot)$  may take different forms: the identity, the absolute value or the square function, and we define the vectors as  $\mathbf{t}_j = \mathbf{X}\mathbf{w}_x^{(j)}$  and  $\mathbf{u}_j = \mathbf{Y}\mathbf{w}_y^{(j)}$ .

More generally, the objective of the standard PLS is to find in the first step directions defined by  $\mathbf{W}_x = [\mathbf{w}_x^{(1)}, \mathbf{w}_x^{(2)}, \dots, \mathbf{w}_x^{(J)}] \in \mathbb{R}^{N \times J}$  and  $\mathbf{W}_y = [\mathbf{w}_y^{(1)}, \mathbf{w}_y^{(2)}, \dots, \mathbf{w}_y^{(J)}] \in \mathbb{R}^{M \times J}$  that maximize the set of cost functions (see Figure 21.4):

$$\max_{\mathbf{w}_x^{(j)}, \mathbf{w}_y^{(j)}} \{g(\mathbf{w}_x^{(j)T} \mathbf{X}^T \mathbf{Y} \mathbf{w}_y^{(j)})\} \quad \text{s.t. } \|\mathbf{w}_x^{(j)}\|_2 = 1, \quad \|\mathbf{w}_y^{(j)}\|_2 = 1, \quad (21.108)$$

$$\mathbf{w}_x^{(j)T} \mathbf{X}^T \mathbf{X} \mathbf{w}_x^{(k)} = 0, \quad \mathbf{w}_y^{(j)T} \mathbf{Y}^T \mathbf{Y} \mathbf{w}_y^{(k)} = 0 \quad \forall k < j \quad (j = 1, 2, \dots, J).$$

$$\mathbf{X} = \mathbf{T} \mathbf{P}^T + \mathbf{E} = \sum_{j=1}^J \mathbf{t}_j \mathbf{p}_j^T + \mathbf{E}$$

$$\mathbf{Y} = \mathbf{U} \mathbf{Q}^T + \mathbf{F} = \sum_{j=1}^J \mathbf{u}_j \mathbf{q}_j^T + \mathbf{F}$$

**FIGURE 21.4**

Illustration of a basic PLS model. In this model  $\mathbf{X}$  is a matrix of predictors,  $\mathbf{Y}$  is the response matrix,  $\mathbf{P}$  and  $\mathbf{Q}$  are loading matrices for the predictor and response with corresponding  $\mathbf{T}$  and  $\mathbf{U} \approx \mathbf{T}\mathbf{D}$  component matrices, and matrices  $\mathbf{E}$  and  $\mathbf{F}$  are corresponding error terms. Our objective is to perform approximative decompositions of the matrices  $\mathbf{X}$  and  $\mathbf{Y}$  and by imposing additional orthogonality and optional sparsity constraints.

In other words, the basic approach to the standard PLS is to find these direction vectors  $\mathbf{w}_x^{(j)}$  and  $\mathbf{w}_y^{(j)}$  from successive optimizations problems (see also Eq. (21.102)). The latent components can be defined as  $\mathbf{T} = \mathbf{XW}_x$  and  $\mathbf{U} = \mathbf{YW}_y$ , or equivalently  $\mathbf{t}_j = \mathbf{Xw}_x^{(j)}$  and  $\mathbf{u}_j = \mathbf{Yw}_y^{(j)}$  for  $j = 1, 2, \dots, J$ .

The main difference between PLS and CCA is that the CCA maximize the correlation while the PLS maximize the covariance. Moreover, in the PLS the primary objective is to estimate matrices  $\mathbf{T}$  and  $\mathbf{U}$  while in the CCA we estimate only projection matrices  $\mathbf{W}_x$  and  $\mathbf{W}_y$ . Anyway, the PLS and CCA are quite similar for the standardized data (that is after the columns of  $\mathbf{X}$  and  $\mathbf{Y}$  have been standardized to have mean zero and standard deviation one) the both method are equivalent.

The PLS decomposition allows us to find relationships between the components which can be used to predict the values of the dependent variables  $\mathbf{Y}_{\text{new}}$  for new situations, if the values of the corresponding independent variables  $\mathbf{X}_{\text{new}}$  are available:

$$\mathbf{T}_{\text{new}} = \mathbf{X}_{\text{new}}[\mathbf{P}^T]^\dagger, \quad \mathbf{Y}_{\text{new}} \approx \mathbf{T}_{\text{new}} \mathbf{D} \mathbf{Q}^T. \quad (21.109)$$

The predictive relationship between the two sets of data is driven by a linear regression model involving  $J$  pairs of latent variables:  $\mathbf{u}_j = d_j \mathbf{t}_j + \tilde{\mathbf{e}}_j$ , (or in a matrix form  $\mathbf{U} = \mathbf{T}\mathbf{D} + \tilde{\mathbf{E}}$ ), where  $d_j$  is a scaling regression coefficient and  $\tilde{\mathbf{e}}_j$  represents residual errors [79]. Using this basic inner regression model the standard PLS model can be expressed as

$$\mathbf{Y} = \sum_{j=1}^J d_j \mathbf{t}_j \mathbf{q}_j^T + \mathbf{E} = \mathbf{XB} + \mathbf{E} \quad (21.110)$$

where the regression coefficients matrix has been defined as  $\mathbf{B} = \sum_{j=1}^J d_j \mathbf{w}_x^{(j)} \mathbf{q}_j^T$ .

**Algorithm 3.** Power Method for Sparse CCA/PLS

---

**Require:** Normalized (standardized) data matrices  $\mathbf{X}^{(1)} = \mathbf{X}$ ,  $\mathbf{Y}^{(1)} = \mathbf{Y}$ , sparsity coefficients  $\alpha_n \geq 0$  ( $n = 1, 2$ ), initial vectors  $\mathbf{w}_x^{(1)}$  and  $\mathbf{w}_y^{(1)}$  and  $\mathbf{Z}^{(1)} = \mathbf{X}^{(1)T} \mathbf{Y}^{(1)}$ ,

1: For  $j = 1, 2, \dots, J$ :

- (a) Repeat until convergence of  $\mathbf{w}_x^{(j)}$  and  $\mathbf{w}_y^{(j)}$ ;
- 2:      $\mathbf{w}_x^{(j)} = \frac{\mathcal{S}_1(\mathbf{Z}^{(j)} \mathbf{w}_y^{(j)}, \alpha_1)}{\|\mathcal{S}_1(\mathbf{Z}^{(j)} \mathbf{w}_y^{(j)}, \alpha_1)\|_2};$
- 3:      $\mathbf{w}_y^{(j)} = \frac{\mathcal{S}_1(\mathbf{Z}^{(j)T} \mathbf{w}_x^{(j)}, \alpha_2)}{\|\mathcal{S}_1(\mathbf{Z}^{(j)T} \mathbf{w}_x^{(j)}, \alpha_2)\|_2};$
- 4: (b) Apply optional orthogonalization procedure according to Eq. (21.104)
- 5: (c)  $\sigma_j = |\mathbf{w}_x^{(j)T} \mathbf{Z}^{(j)} \mathbf{w}_y^{(j)}|$ ;
- 6: (d)  $\mathbf{Z}^{(j+1)} = \mathbf{Z}^{(j)} - \sigma_j \mathbf{w}_x^{(j)} \mathbf{w}_y^{(j)T};$
- 7: **return**  $\mathbf{W}_x^* = [\mathbf{w}_x^{(1)}, \mathbf{w}_x^{(2)}, \dots, \mathbf{w}_x^{(J)}]$ ,  $\mathbf{W}_y^* = [\mathbf{w}_y^{(1)}, \mathbf{w}_y^{(2)}, \dots, \mathbf{w}_y^{(J)}]$ ,  
 $\Sigma^* = \text{diag}\{\sigma_1, \dots, \sigma_J\};$
- 8: **return** Optional  $\mathbf{T} = \mathbf{X} \mathbf{W}_x$ ,  $\mathbf{U} = \mathbf{Y} \mathbf{W}_y$

---

Assuming that the vectors component matrices are estimated sequentially (using deflation approach), the loading factors and regression coefficient can be calculated as:

$$\mathbf{q}_j = \frac{\mathbf{Y}^T \mathbf{u}_j}{\mathbf{u}_j^T \mathbf{u}_j}, \quad d_j = \frac{\mathbf{t}_j^T \mathbf{u}_j}{\mathbf{t}_j^T \mathbf{t}_j} \quad (j = 1, 2, \dots, J). \quad (21.111)$$

Often, we impose additional (optional) sparseness and/or orthogonality constraints (imposed on components matrices  $\mathbf{T}$  and  $\mathbf{U}$  or the loading matrices), in order to achieve better predictive power. For high dimensional data sets, it is often appropriate to assume that independent and dependent data has unit covariance matrices, i.e.,  $\mathbf{X}^T \mathbf{X} \approx \mathbf{I}_N$  and  $\mathbf{Y}^T \mathbf{Y} \approx \mathbf{I}_M$ . In the case of the sparse PLS, we can formulate as the following optimization problem

$$\begin{aligned} & \max_{\mathbf{w}_x^{(1)}, \mathbf{w}_y^{(1)}} \left\{ g(\mathbf{w}_x^{(1)T} \mathbf{X}^T \mathbf{Y} \mathbf{w}_y^{(1)}) - \alpha_1 \|\mathbf{w}_x^{(1)}\|_1 - \alpha_2 \|\mathbf{w}_y^{(1)}\|_1 \right\} \\ & \text{s.t. } \|\mathbf{w}_x^{(1)}\|_2^2 \leq 1, \quad \|\mathbf{w}_y^{(1)}\|_2^2 \leq 1. \end{aligned} \quad (21.112)$$

The PLS algorithms have been proven to be particularly powerful for strongly collinear data and for underdetermined independent data (i.e., more variables than observations) [77]. The widely reported sensitivity to noise of the PLS regression is attributed to redundant (irrelevant) latent variables, their selection remains an open problem. Other problems include the difficulty of interpreting the loadings corresponding to the latent variables, and the fact that the number of latent variables cannot exceed the minimum number of predictors and observations.

Sparse PCA, CCA, and PLS deliver components (or latent vectors) that can be considered as a compromise between maximizing the variance, correlation or covariance, respectively and improving the interpretability of the data, especially for large-scale data with possibly fewer observations than

**Algorithm 4.** Alternative Algorithm for Sparse PLS

---

**Require:** Normalized (standardized) data matrices  $\mathbf{X}^{(1)} = \mathbf{X}$ ,  $\mathbf{Y}^{(1)} = \mathbf{Y}$ , sparsity coefficients  $\alpha_n \geq 0$ , ( $n = 1, 2$ );  
 1: For  $j = 1, 2, \dots, J$ :  
   Set (a)  $\mathbf{Z}^{(j)} = \mathbf{X}^{(j)T} \mathbf{Y}^{(j)}$  and use the SVD to compute the first pair of singular vectors:  
   the left singular vector  $\mathbf{w}_x^{(j)}$  and the right singular vector  $\mathbf{w}_y^{(j)}$ ;  
   (b) Repeat until the convergence of sparse  $\mathbf{w}_x^{(j)}$  and  $\mathbf{w}_y^{(j)}$ :  
     2:    $\mathbf{w}_x^{(j)} \leftarrow \mathcal{S}_1(\mathbf{Z}^{(j)} \mathbf{w}_y^{(j)}, \alpha_1)$ , normalize new vector  $\mathbf{w}_x^{(j)}$ ;  
     3:    $\mathbf{w}_y^{(j)} \leftarrow \mathcal{S}_1(\mathbf{Z}^{(j)T} \mathbf{w}_x^{(j)}, \alpha_2)$ , normalize new vector  $\mathbf{w}_y^{(j)}$ ;  
     4: (b) Apply optional orthogonalization procedure according to Eq. (21.104)  
     5: (c)  $\mathbf{t}_j = \frac{\mathbf{X}^{(j)} \mathbf{w}_x^{(j)}}{\mathbf{w}_x^{(j)T} \mathbf{w}_x^{(j)}}$ ,  $\mathbf{u}_j = \frac{\mathbf{Y}^{(j)} \mathbf{w}_y^{(j)}}{\mathbf{w}_y^{(j)T} \mathbf{w}_y^{(j)}}$ ;  
     6: (d)  $\mathbf{p}_j = \frac{\mathbf{X}^{(j)T} \mathbf{t}_j}{\mathbf{t}_j^T \mathbf{t}_j}$ ,  $\mathbf{q}_j = \frac{\mathbf{Y}^{(j)T} \mathbf{u}_j}{\mathbf{u}_j^T \mathbf{u}_j}$ ;  
     7: (e)  $\mathbf{X}^{(j+1)} = \mathbf{X}^{(j)} - \mathbf{t}_j \mathbf{p}_j^T$ ;  
     8:    $\mathbf{Y}^{(j+1)} = \mathbf{Y}^{(j)} - \mathbf{u}_j \mathbf{q}_j^T$ ;  
   9: **return**  $\mathbf{W}_x, \mathbf{W}_y, \mathbf{T}, \mathbf{P}, \mathbf{U}, \mathbf{Q}$ ;

---

variables. The extraction of components and latent variables sequentially, one by one, by applying deflation techniques, allows us to stop the algorithms after achieving the desired number of components and/or achieving an error measure below a specified threshold.

### 1.21.3 ICA and related problems

There are at least three basic approaches for ICA, coming from different models of source signals [13, 14, 17]. The first simple model considers weakly stationary Gaussian processes and exploits directly the second order statistics (SOS). These signals are separable if their spectra are distinct; therefore, the separation is based on the spectral diversity of the sources. The second approach takes the nonstationarity of the signals into account by modeling them as independently distributed Gaussian variables whose variance is changing in time. For the third basic approach one assumes that source signals are represented by sequence of identically and independently distributed random variables. The condition of separability of such source signals requires that at most only one signal is Gaussian, so the approach is said to be based on the non-Gaussianity and it exploits higher order statistics (HOS). Strictly speaking the first two approaches do not explicitly or implicitly assume statistical independence but rather generalized decorrelation via simultaneous joint diagonalization of time-delayed covariance matrices.

Spatial decorrelation (or prewhitening) is often considered a necessary (but not sufficient) condition for stronger stochastic independence criteria. After prewhitening ICA tasks usually become somewhat easier and well-posed (or less ill-conditioned), because the subsequent separating (unmixing) system is described by an orthogonal matrix for real-valued signals and by a unitary matrix for complex-valued signals and weights. Furthermore, the spatio-temporal and time-delayed decorrelation can be used to identify the mixing matrix and to perform blind source separation of colored sources under certain weak

conditions [13, 14, 17, 18, 80]. Temporal, spatial and spatio-temporal decorrelations play important roles in EEG/MEG data analysis. These techniques are based only on second-order statistics (SOS). They are the basis for modern subspace methods of spectrum analysis and array processing and are often used in a preprocessing stage in order to improve the convergence properties of adaptive systems, to eliminate redundancy or to reduce noise. Spatio-temporal data has both spatial (i.e., location) and temporal (i.e., time related) components.

One possible extension of the ICA is the so-called Independent Subspace Analysis (ISA). ISA can be applied to situations where not all source signals can be separated from each other by linear transformations. The goal here is to decompose the linear space spanned by rows of  $\mathbf{Y}$  to a direct (orthogonal) sum of linear subspaces, such that elements of each subspace are statistically independent of the others. The ISA problem can be approached by applying an ICA algorithm, which aims to separate each component from the others as much as possible. In the postprocessing step we usually perform clustering of the obtained components according to their residual mutual dependence.

### 1.21.3.1 The AMUSE algorithm and its properties

AMUSE (Algorithm for Multiple Unknown Signal Extraction) is probably the simplest BSS algorithm which belongs to the group of Second-Order Statistics Spatio-Temporal Decorrelation (SOS-STD) algorithms [14, 81]. It provides similar decompositions for noiseless data to the well known and popular SOBI and TDSEP algorithms [82, 83]. This class of algorithms are sometimes classified or referred to as ICA algorithms. However, these algorithms do not exploit implicitly or explicitly the statistical independence. Moreover, in contrast to standard higher order statistics ICA algorithms, they are able to estimate colored Gaussian distributed sources and their performance in estimation of original sources is usually better if the sources have temporal structures.

The AMUSE algorithm has some similarity with the standard PCA. The main difference is that the AMUSE employs PCA two times in two separate steps: In the first step, standard PCA is applied for the whitening (sphering) of the data and in the second step PCA is applied to the time delayed covariance matrix of the pre-whitened data. Mathematically the AMUSE algorithm consists of the following two stage procedure: In the first step we apply standard or robust prewhitening as a linear transformation  $\hat{\mathbf{y}}(t) = \mathbf{Q}\mathbf{y}(t)$ , where  $\mathbf{Q} = \mathbf{R}_{yy}^{-1/2} = (\mathbf{V}\Lambda\mathbf{V}^T)^{-1/2} = \mathbf{V}(\Lambda)^{-1/2}\mathbf{V}^T$  of the standard covariance matrix  $\mathbf{R}_{yy} = E\{\mathbf{y}(t)\mathbf{y}^T(t)\}$ , and  $\mathbf{y}(t)$  is a vector of observed data for time point  $t$ . Next, (for the pre-whitened data) the SVD is applied for the time-delayed covariance matrix  $\mathbf{R}_{\hat{\mathbf{y}}}(p) = E\{\hat{\mathbf{y}}(t)\hat{\mathbf{y}}^T(t-p)\} = \mathbf{U}_1\boldsymbol{\Sigma}_1\mathbf{V}_1^T$  (typically, with  $p = 1$ ), where  $\boldsymbol{\Sigma}_1$  is a diagonal matrix with decreasing singular values and  $\mathbf{U}_1, \mathbf{V}_1$  are orthogonal matrices containing the of left and right singular vectors. Then, an unmixing (separating) matrix is estimated as  $\mathbf{W} = \mathbf{U}_1^T\mathbf{Q}$ .

The main advantages of the AMUSE algorithm over the other BSS/ICA algorithms is its simplicity and that it allows the automatic ordering of the components, due to application of SVD. In fact, the components are ordered according to the decreasing values of the singular values of the time-delayed covariance matrix. In other words, the AMUSE algorithm exploits the simple principle that the estimated components tends to be less complex or more precisely that they have better linear predictability than any mixture of those sources. It should be emphasized that all components estimated by AMUSE are uniquely defined and consistently ranked. The consistent ranking is due to the fact that these singular values are always ordered in decreasing values. For real-world data probability that two singular values achieve

exactly the same value is very small, so ordering is consistent and unique [14]. The one disadvantage of the AMUSE algorithm it is that is very sensitive to additive noise since the algorithm exploits only one time delayed covariance matrix.

### 1.21.3.2 The SOBI algorithm and its extensions

The second-order blind identification (SOBI) algorithm is a classical blind source separation (BSS) algorithm for wide-sense stationary (WSS) processes with distinct spectra [82]. This algorithm has proven to be very useful in biomedical applications and it has became more popular than the other algorithms.

There is a common practice in ICA/BSS research to exploit the “average eigen-structure” of a large set of data matrices formed as functions of the available data (typically, covariance or cumulant matrices for different time delays). In other words, the objective is to extract reliable information (e.g., estimation of sources and/or of the mixing matrix) from the eigen-structure of a possibly large set of data matrices [14, 82, 84, 85]. However, since in practice we only have a finite number of samples of signals corrupted by noise, the data matrices do not exactly share the same eigen-structure. Furthermore, it should be noted that determining the eigen-structure on the basis of one or even two data matrices usually leads to poor or unsatisfactory results, because such matrices, based usually on an arbitrary choice, may have degenerate eigenvalues leading to loss of the information contained in other data matrices. Therefore, from a statistical point of view, in order to provide robustness and accuracy, it is necessary to consider the average eigen-structure by taking into account simultaneously a possibly large set of data matrices [14, 85].

The average eigen-structure can be easily implemented via the linear combination of several time-delayed covariance matrices and by applying the standard EVD or SVD. An alternative approach to EVD/SVD is to apply the approximate joint diagonalization procedure (JAD). The objective of this procedure is to find the orthogonal matrix  $\mathbf{U}$  which diagonalizes a set of matrices [82, 84, 85]:

$$\mathbf{R}_{\tilde{\mathbf{y}}}(p_r) = \mathbf{U} \mathbf{D}_r \mathbf{U}^T + \boldsymbol{\varepsilon}_r \quad (r = 1, 2, \dots, R), \quad (21.113)$$

where  $\mathbf{R}_{\tilde{\mathbf{y}}}(p_r) = E\{\tilde{\mathbf{y}}(t)\tilde{\mathbf{y}}^T(t - p_r)\} \in \mathbb{R}^{J \times J}$  are data matrices (for example, time-delayed covariance matrices), the  $\mathbf{D}_r$  are diagonal and real, and  $\boldsymbol{\varepsilon}_r$  represent additive errors (as small as possible). If  $R > 2$  for arbitrary matrices  $\mathbf{R}_x(p_r)$ , the problem becomes overdetermined and generally we can not find an exact diagonalizing matrix  $\mathbf{U}$  with  $\boldsymbol{\varepsilon}_r = \mathbf{0}, \forall r$ . An important advantage of the Joint Approximate Diagonalization (JAD) is that several numerically efficient algorithms exist for its computation, including Jacobi techniques (one sided and two sided), Alternating Least Squares (ALS), PARAFAC (Parallel Factor Analysis) and subspace fitting techniques employing the efficient Gauss–Newton optimization [85]. This idea has been implemented as a robust SOBI algorithm which can be briefly outlined as follows [14, 86]:

1. Perform robust orthogonalization  $\tilde{\mathbf{y}}(t) = \mathbf{Q}\mathbf{y}(t)$ , similar as to the AMUSE algorithm.
2. Estimate the set of covariance matrices:

$$\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}}(p_r) = (1/T) \sum_{t=1}^T \tilde{\mathbf{y}}(t)\tilde{\mathbf{y}}^T(t - p_r) = \mathbf{Q}\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}}(p_r)\mathbf{Q}^T \quad (21.114)$$

for a preselected set of time lags  $(p_1, p_2, \dots, p_R)$  or bandpass filters  $B_r$ .

3. Perform JAD:  $\mathbf{R}_{\hat{\mathbf{y}}}(p_r) = \mathbf{U}\mathbf{D}_r\mathbf{U}^T$ ,  $\forall r$ , i.e., estimate the orthogonal matrix  $\mathbf{U}$  using one of the available numerical algorithms.
4. Estimate the source signals as  $\hat{\mathbf{x}}(t) = \mathbf{U}^T \mathbf{Q} \mathbf{y}(t)$ , and then the mixing matrix as  $\hat{\mathbf{A}} = \mathbf{Q}^+ \mathbf{U}$ .

The main advantage of the SOBI algorithm is its robustness with respect to additive noise if number of covariance matrices is sufficiently large (typically,  $R > 100$ ).

Several extensions and modifications of the SOBI algorithm have been proposed, e.g., Robust SOBI [14], thinICA [87], and WASOBI [88,89] (the weights-adjusted variant of SOBI), which is asymptotically optimal (in separating Gaussian parametric processes). The WASOBI for the separation of independent stationary sources is modeled as Auto-Regressive (AR) random processes. The matrices are computed for the case of Gaussian sources, for which the resulting separation would be asymptotically optimal approaching the corresponding Cramer-Rao bound (CRB) for the best possible separation [88,90].

### 1.21.3.3 ICA based on higher order statistics (HOS)

The ICA of a random vector  $\mathbf{y}(t) \in \mathbb{R}^I$  can be performed by finding an  $J \times I$ , (with  $I \geq J$ ), full rank separating (transformation) matrix  $\mathbf{W}$ , such that the output signal vector  $\hat{\mathbf{x}}(t) = [\hat{x}_1(t), \hat{x}_2(t), \dots, \hat{x}_J(t)]^T$  (components) estimated by

$$\hat{\mathbf{x}}(t) = \mathbf{W}\mathbf{y}(t) = \mathbf{A}^\dagger \mathbf{y}, \quad (21.115)$$

are as independent as possible as evaluated by an information-theoretic cost function such as the minimum of the Kullback-Leibler divergence [14,17,91,92].

The statistical independence of random variables is a more general concept than decorrelation. Roughly speaking, we say that the random variables  $x_i$  and  $x_j$  are statistically independent if knowledge of the value of  $x_i$  provides no information about the values of  $x_j$ . Mathematically, the independence of  $x_i$  and  $x_j$  can be expressed by the relationship

$$p(x_i, x_j) = p(x_i)p(x_j), \quad (21.116)$$

where  $p(x)$  denotes the probability density function (pdf) of the random variable  $x$ . In other words, signals are independent if their joint pdf can be factorized.

If independent signals are zero-mean, then the generalized covariance matrix of  $f(x_i)$  and  $g(x_j)$ , where  $f(x)$  and  $g(x)$  are different, odd nonlinear activation functions (e.g.,  $f(x) = \tanh(x)$  and  $g(x) = x$  for super-Gaussian sources) is a non-singular diagonal matrix:

$$\mathbf{R}_{fg} = E\{\mathbf{f}(\hat{\mathbf{x}})\mathbf{g}^T(\hat{\mathbf{x}})\} = \begin{bmatrix} E\{f(\hat{x}_1)g(\hat{x}_1)\} & & 0 \\ & \ddots & \\ 0 & & E\{f(\hat{x}_J)g(\hat{x}_J)\} \end{bmatrix}, \quad (21.117)$$

i.e., the covariances  $E\{f(\hat{x}_i)g(\hat{x}_j)\}$  are all zero. It should be noted that for odd  $f(\hat{x})$  and  $g(\hat{x})$ , if the probability density function of each zero-mean source signal is even, then the terms of the form  $E\{f(\hat{x}_i)\}E\{g(\hat{x}_i)\}$  equal zero. The true general condition for the statistical independence of signals is the vanishing of all high-order cross-cumulants [93–95]. The diagonalization principle can be expressed as

$$\mathbf{R}_{fg}^{-1} = \mathbf{\Lambda}^{-1}, \quad (21.118)$$

where  $\Lambda$  is any diagonal positive definite matrix (typically,  $\Lambda = \mathbf{I}$ ). By pre-multiplying the above equation by the matrices  $\mathbf{W}$  and  $\Lambda$ , we obtain:

$$\Lambda \mathbf{R}_{fg}^{-1} \mathbf{W} = \mathbf{W}, \quad (21.119)$$

which suggests the following simple iterative multiplicative learning algorithm [96]

$$\tilde{\mathbf{W}}(k+1) = \Lambda \mathbf{R}_{fg}^{-1} \mathbf{W}(k), \quad (21.120)$$

$$\mathbf{W}(t+1) = \tilde{\mathbf{W}}(t+1) [\tilde{\mathbf{W}}^T(t+1) \tilde{\mathbf{W}}(t+1)]^{-1/2}, \quad (21.121)$$

where the last equation represents the symmetric orthogonalization that keeps the algorithm stable. The above algorithm is very simple, but it needs the prewhitening of the sensor data and it is sensitive to noise [14, 97].

Assuming prior knowledge of the source distributions  $p_j(x_j)$ , we can estimate  $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_J]^T$  by using maximum likelihood (ML):

$$L = T \log |\det \mathbf{W}| + \sum_{t=1}^T \sum_{j=1}^J \log p_j(\mathbf{w}_j^T \mathbf{y}(t)). \quad (21.122)$$

Using the gradient descent of the likelihood we obtain the infomax update rule

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \eta \frac{\partial L}{\partial \mathbf{W}} = \mathbf{W}(k) + \eta ([\mathbf{W}^T(k)]^{-1} - \frac{1}{T} \sum_{t=1}^T \mathbf{f}(\hat{\mathbf{x}}(t)) \mathbf{y}^T(t)), \quad (21.123)$$

where  $\hat{\mathbf{x}}(t) = \mathbf{W}(k) \mathbf{y}(t)$  and  $\mathbf{f}(\hat{\mathbf{x}}) = [f_1(\hat{x}_1), f_2(\hat{x}_2), \dots, f_J(\hat{x}_J)]^T$  is an entry-wise nonlinear score function defined by:

$$f_j(x_j) = -\frac{p'_j(x_j)}{p_j(x_j)} = -\frac{d \log(p_j(x_j))}{d(x_j)}. \quad (21.124)$$

Using the natural gradient descent to increase the likelihood we get [98]:

$$\begin{aligned} \Delta \mathbf{W} &= \mathbf{W}(k+1) - \mathbf{W}(k) = -\eta \frac{\partial L}{\partial \mathbf{W}} \mathbf{W}^T \mathbf{W} \\ &= \eta [\mathbf{I} - \langle \mathbf{f}(\hat{\mathbf{x}}) \hat{\mathbf{x}}^T \rangle] \mathbf{W}(k). \end{aligned} \quad (21.125)$$

Alternatively, for signals corrupted by additive Gaussian noise, we can use higher order matrix cumulants. As an illustrative example, let us consider the following cost function which is a measure of independence [99]:

$$J(\mathbf{W}, \hat{\mathbf{x}}) = -\frac{1}{2} \log |\det (\mathbf{W} \mathbf{W}^T)| - \frac{1}{1+q} \sum_{i=1}^n |C_{1+q}(\hat{x}_i)|, \quad (21.126)$$

where we use the following notations:  $C_q(\hat{x}_i)$  denotes the  $q$ -order cumulants of the signal  $\hat{x}_i$  and  $\mathbf{C}_{p,q}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$  denotes the cross-cumulant matrix whose elements are  $[\mathbf{C}_{pq}(\hat{\mathbf{x}}, \hat{\mathbf{x}})]_{ij} = \text{Cum}(\underbrace{\hat{x}_i, \hat{x}_i, \dots, \hat{x}_i}_p, \underbrace{\hat{x}_j, \hat{x}_j, \dots, \hat{x}_j}_q)$ .

**Table 21.1** Basic Equivariant Adaptive Learning Algorithms for ICA.  $\hat{\mathbf{x}}(t) = \mathbf{W}\mathbf{y}(t) = \hat{\mathbf{A}}^\dagger \mathbf{y}(t)$ 

No.	Learning Algorithm	References
1.	$\Delta\mathbf{W} = \eta[\Lambda - \langle \mathbf{f}(\hat{\mathbf{x}})\mathbf{g}^T(\hat{\mathbf{x}}) \rangle]\mathbf{W}$ $\Delta\hat{\mathbf{A}} = -\eta\hat{\mathbf{A}}[\Lambda - \langle \mathbf{f}(\hat{\mathbf{x}})\mathbf{g}^T(\hat{\mathbf{x}}) \rangle]$	[93, 95]
2.	$\Delta\mathbf{W} = \eta[\Lambda - \langle \mathbf{f}(\hat{\mathbf{x}})\hat{\mathbf{x}}^T \rangle]\mathbf{W}, \quad f(y_i) = -p'(y_i)/p(y_i)$ $\lambda_{ii} = \langle f(y_i(k))y_i(k) \rangle$ or $\lambda_{ii} = 1, \forall i$	[91] [98]
3.	$\Delta\mathbf{W} = \eta[\mathbf{I} - \langle \hat{\mathbf{x}}\hat{\mathbf{x}}^T \rangle - \langle \mathbf{f}(\hat{\mathbf{x}})\hat{\mathbf{x}}^T \rangle + \langle \hat{\mathbf{x}}\mathbf{f}^T(\hat{\mathbf{x}}) \rangle]\mathbf{W}$	[100]
4.	$\Delta\mathbf{W} = \eta[\mathbf{I} - \langle \hat{\mathbf{x}}\hat{\mathbf{x}}^T \rangle - \langle \mathbf{f}(\hat{\mathbf{x}})\hat{\mathbf{x}}^T \rangle + \langle \mathbf{f}(\hat{\mathbf{x}})\mathbf{f}^T(\hat{\mathbf{x}}) \rangle]\mathbf{W}$	[45]
5.	$\tilde{\mathbf{W}} = \mathbf{W} + \eta[\Lambda - \langle \mathbf{f}(\hat{\mathbf{x}})\hat{\mathbf{x}}^T \rangle]\mathbf{W}, \quad \lambda_{ii} = \langle f(y_i)y_i \rangle$ $\eta_{ii} = [\lambda_{ii} + \langle f'(y_i) \rangle]^{-1}; \quad \mathbf{W} = (\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)^{-1/2}\tilde{\mathbf{W}}$	[17]
6.	$\Delta\mathbf{W} = \eta[\mathbf{I} - \Lambda^{-1}\langle \hat{\mathbf{x}}\hat{\mathbf{x}}^T \rangle]\mathbf{W}$ $\lambda_{ii}(k) = \langle \hat{x}_i^2(t) \rangle$	[101] [28]
7.	$\mathbf{W}(k+1) = [\mathbf{I} \mp \eta[\mathbf{I} - \langle \mathbf{f}(\hat{\mathbf{x}})\mathbf{g}^T(\hat{\mathbf{x}}) \rangle]]^{\mp 1}\mathbf{W}(k)$	[102]
8.	$\Delta\mathbf{W} = \eta[\mathbf{I} - \mathbf{C}_{1,q}(\hat{\mathbf{x}}, \hat{\mathbf{x}})\mathbf{S}_{q+1}(\hat{\mathbf{x}})]\mathbf{W}$ $\mathbf{C}_{1,q}(x_i, x_j) = \text{Cum}(y_i, \underbrace{x_j, \dots, x_j}_q)$	[99]

The first term in (21.126) assures that the determinant of the global matrix will not approach zero. By including this term we avoid the trivial solution  $\hat{x}_i = 0, \forall i$ . The second term forces the output signals to be as far as possible from Gaussianity, since the higher order cumulants are a natural measure of non-Gaussianity and they will vanish for Gaussian signals. It can be shown that for such a cost function, we can derive the following equivariant and robust (with respect to the Gaussian noise) algorithm [87, 99, 102]

$$\Delta\mathbf{W}(t) = \mathbf{W}(k+1) - \mathbf{W}(k) = \eta_l[\mathbf{I} - \mathbf{C}_{1,q}(\hat{\mathbf{x}}, \hat{\mathbf{x}})\mathbf{S}_{q+1}(\hat{\mathbf{x}})]\mathbf{W}(k), \quad (21.127)$$

where  $\mathbf{S}_{q+1}(\hat{\mathbf{x}}) = \text{sign}(\text{diag}(\mathbf{C}_{1,q}(\hat{\mathbf{x}}, \hat{\mathbf{x}})))$  and  $\mathbf{F}(\hat{\mathbf{x}}) = \mathbf{I} - \mathbf{C}_{1,q}(\hat{\mathbf{x}}, \hat{\mathbf{x}})\mathbf{S}_{q+1}(\hat{\mathbf{x}})$ .

A wide class of equivariant algorithms for ICA can be expressed in a general form as (see Table 21.1) [14]

$$\nabla\mathbf{W}(k) = \mathbf{W}(k+1) - \mathbf{W}(k) = \eta\mathbf{F}(\hat{\mathbf{x}})\mathbf{W}(k), \quad (21.128)$$

where  $\hat{\mathbf{x}}(t) = \mathbf{W}\mathbf{y}(t)$  and the matrix  $\mathbf{F}(\hat{\mathbf{x}})$  can take different forms, for example  $\mathbf{F}(\hat{\mathbf{x}}) = \Lambda - \langle \mathbf{f}(\hat{\mathbf{x}})\mathbf{g}^T(\hat{\mathbf{x}}) \rangle$  with suitably chosen nonlinearities  $\mathbf{f}(\hat{\mathbf{x}}) = [f(\hat{x}_1), \dots, f(\hat{x}_J)]$  and  $\mathbf{g}(\hat{\mathbf{x}}) = [g(\hat{x}_1), \dots, g(\hat{x}_J)]$  [14, 95, 96, 99].

It should be noted that ICA based on HOS can perform blind source separation, i.e., allows us to estimate the true sources only if they are all statistically independent and are non Gaussian (except possibly of one) [14, 91].

### 1.21.3.4 Blind source extraction

There are two main approaches to solve the problem of blind source separation. The first approach, which was mentioned briefly in the previous section, is to simultaneously separate all sources. In the second

approach, we extract the sources sequentially in a blind fashion, one by one, rather than separating them all simultaneously. In many applications a large number of sensors (electrodes, sensors, microphones or transducers) are available but only a very few source signals are the subject of interest. For example, in the modern EEG or MEG devices, we typically observe more than 100 sensor signals, but only a few source signals are interesting; the rest can be considered as interfering noise. In another example, the cocktail party problem, it is usually essential to extract the voices of specific persons rather than separate all the source signals of all speakers available (in mixing form) from an array of microphones. For such applications it is essential to apply reliable, robust and effective learning algorithms which enable us to extract only a small number of source signals that are potentially interesting and contain useful information.

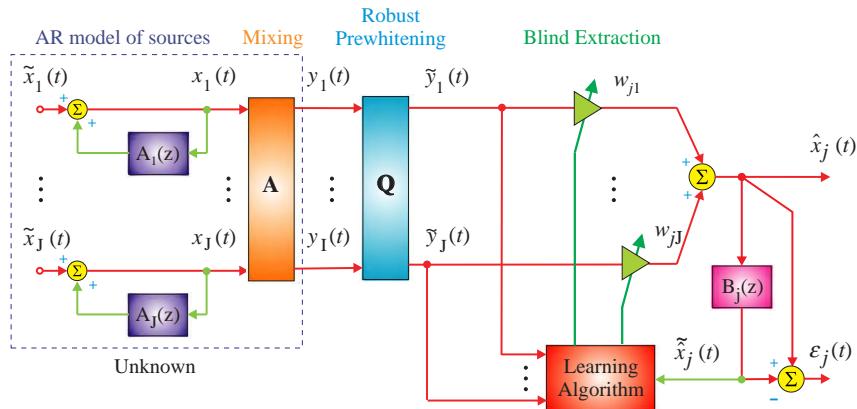
The blind source extraction (BSE) approach has several advantages over the simultaneous blind separation/deconvolution, such as

- Only the “interesting” signals need to be extracted. For example, if the source signals are mixed with a large number of noise terms, we may extract only the specific signals which possess some desired statistical properties.
- The signals can be extracted in a *specified order* according to the statistical features of the source signals, e.g., in the order determined by the absolute values of the generalized normalized kurtosis. Blind extraction of sources can be considered as a generalization of the sequential extraction of principal components, where uncorrelated output signals are extracted according to the decreasing order of their variances.
- The available unsupervised learning algorithms for BSE are local, stable and typically biologically plausible.

We can use two different criteria. The first criterion is based on higher order statistics (HOS), which assumes that the sources are mutually statistically independent and non-Gaussian (at most one can be Gaussian). For independence criteria we will use some measure of non-Gaussianity [14]. The second criterion is based on the concept of linear predictability and it assumes that the source signals have some temporal structure, i.e., the sources are colored with different autocorrelation functions, or equivalently they have different spectral shapes.

#### 1.21.3.4.1 *Blind extraction of sources with temporal structures*

In this approach we exploit the temporal structure of the signals rather than their statistical independence [103, 104]. Intuitively speaking, the source signals  $x_j(t)$  have less complexity than the mixed sensor signals  $y_j(t)$ . In other words, the degree of temporal predictability of any source signal is higher than (or equal to) that of any mixture. For example, waveforms of a mixture of two sine waves with different frequencies are more complex (or less predictable) than either of the original sine waves. This means that applying the standard linear predictor model and minimizing the mean squared error  $E\{\varepsilon^2\}$ , which is a measure of predictability, we can separate (or extract) signals with different temporal structures. More precisely, by minimizing the error, we maximize a measure of temporal predictability for each recovered signal [105].

**FIGURE 21.5**

Block diagram illustrating the implementation of learning algorithm for the blind extraction of temporally correlated sources.

It is worth noting that two criteria used in BSE, namely temporal linear predictability and non-Gaussianity based on kurtosis, may lead to different results. Temporal predictability forces the extracted signals to be smooth and possibly less structurally complex, while the non-Gaussianity measure forces the extracted signals to be as independent as possible, with sparse representation for the sources that have positive kurtosis.

Let us assume that temporally correlated source signals are modeled by autoregressive processes (AR) (see Figure 21.5) as

$$x_j(t) = \tilde{x}_j(t) + \sum_{p=1}^L \tilde{a}_{jp} x_j(t-p) = \tilde{x}_j(t) + A_j(z)x_j(t), \quad (j = 1, 2, \dots, J) \quad (21.129)$$

where \$A\_j(z) = \sum\_{p=1}^L \tilde{a}\_{jp} z^{-p}\$, \$z^{-p}x(t) = x(t-p)\$ and \$\tilde{x}\_j(t)\$ are i.i.d. unknown (innovative) processes. In practice, the AR model can be extended to more general models such as the Auto Regressive Moving Average (ARMA) model or the Hidden Markov Model (HMM) [14, 17, 101].

For ill-conditioned problems (when a mixing matrix is ill-conditioned and/or the source signals have different amplitudes), we can apply optional pre-whitening to the sensor signals \$\mathbf{y}(t)\$ in the form of \$\tilde{\mathbf{y}}(t) = \mathbf{Q}\mathbf{y}(t)\$, where \$\mathbf{Q} \in \mathbb{R}^{J \times J}\$ is a whitening matrix that ensures that the auto-correlation matrix is an identity matrix: \$\mathbf{R}\_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\} = \mathbf{I}\_J\$.

To model temporal structures of source signals, we consider a linear processing unit with an adaptive filter with the transfer function \$B\_j(z)\$ (which estimates one \$A\_j(z)\$) as illustrated in Figure 21.5. Let us assume for simplicity, that we want to extract only one source signal, e.g., \$\hat{x}\_j(t)\$, from the available

sensor vector  $\tilde{\mathbf{y}}(t)$ . For this purpose, we employ a single processing unit described as

$$\hat{x}_j(t) = \mathbf{w}_j^T \tilde{\mathbf{y}}(t) = \sum_{i=1}^J w_{ji} \tilde{y}_j(t), \quad (21.130)$$

$$\varepsilon_j(t) = \hat{x}_j(t) - \sum_{p=1}^L b_{jp} \hat{x}_j(t-p) = \mathbf{w}^T \tilde{\mathbf{y}}(t) - \mathbf{b}_j^T \hat{\mathbf{x}}_j(t), \quad (21.131)$$

where  $\mathbf{w}_j = [w_{j1}, w_{j2}, \dots, w_{jJ}]^T$ ,  $\hat{\mathbf{x}}_j(t) = [\hat{x}_1(t-1), \hat{x}_2(t-2), \dots, \hat{x}_J(t-L)]^T$ ,  $\mathbf{b}_j = [b_{j1}, b_{j2}, \dots, b_{jL}]^T$  and  $B_j(z) = \sum_{p=1}^L b_{jp} z^{-p}$  is the transfer function of the corresponding FIR filter. It should be noted that the FIR (Finite Impulse Response) filter can have a sparse vector  $\mathbf{b}_j$ . In particular, only one single processing unit, e.g., with delay  $p$  and  $b_{jp} \neq 0$  can be used instead of the  $L$  parameters. The processing unit has two outputs:  $\hat{x}_j(t)$  which estimates the extracted source signals, and  $\varepsilon_j(t)$ , which represents a linear prediction error or estimation of the innovation, after passing the output signal  $\hat{x}_j(t)$  through the  $B_j(z)$  filter. Our objective is to estimate the optimal values of the vectors  $\mathbf{w}_1$  and  $\mathbf{b}_1$ , in such a way that the processing unit successfully extracts one of the sources. This is achieved if the global vector defined as  $\mathbf{g}_j = \mathbf{A}^T \mathbf{w}_j = (\mathbf{w}_j^T \mathbf{A})^T = c_j \mathbf{e}_i$  contains only one nonzero element, e.g., in the  $j$ th row, such that  $\hat{x}_j(t) = c_j x_i(t)$ , where  $c_j$  is an arbitrary nonzero scaling factor. For this purpose we reformulate the problem as a minimization of the cost function

$$D_j(\mathbf{w}_j, \mathbf{b}_j) = E\{\varepsilon_j^2\}. \quad (21.132)$$

The main motivation for applying such a cost function is the assumption that the primary source signals (signals of interest) have temporal structures and can be modeled by an autoregressive model [14, 106, 107].

According to the AR model of source signals, the filter output can be represented as  $\varepsilon_j(t) = \hat{x}_j(t) - \tilde{x}_j(t)$ , where  $\tilde{x}_j(t) = \sum_{p=1}^L b_{jp} \hat{x}_j(t-p)$  is defined as an error or estimator of the innovation source  $\tilde{x}_j(t)$ . The mean squared error  $E\{\varepsilon_j^2(t)\}$  achieves a minimum  $c_j^2 E\{\tilde{x}_j^2(t)\}$ , where  $c_j$  is a positive scaling constant, if and only if  $\hat{x}_1 = \pm c_1 x_j$  for any  $j \in \{1, 2, \dots, m\}$  or  $\hat{x}_j = 0$  holds.

The cost function (21.132) can be evaluated as follows:

$$E\{\varepsilon_j^2\} = \mathbf{w}_j^T \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} \mathbf{w}_j - 2\mathbf{w}_j^T \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} \mathbf{b}_j + \mathbf{b}_j^T \widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j} \mathbf{b}_j, \quad (21.133)$$

where  $\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} \approx E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\}$ ,  $\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} \approx E\{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j^T\}$  and  $\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j} \approx E\{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j^T\}$  are estimators of the true values of correlation and cross-correlation matrices:  $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}}$ ,  $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j}$ ,  $\mathbf{R}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}$ , respectively.

In order to estimate the vectors  $\mathbf{w}_j$  and  $\mathbf{b}_j$ , we evaluate the gradients of the cost function and make them equal to zero as follows:

$$\frac{\partial D_j(\mathbf{w}_j, \mathbf{b}_j)}{\partial \mathbf{w}} = 2\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} \mathbf{w}_j - 2\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} \mathbf{b}_j = \mathbf{0}, \quad (21.134)$$

$$\frac{\partial D_j(\mathbf{w}_j, \mathbf{b}_j)}{\partial \mathbf{b}_j} = 2\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j} \mathbf{b}_j - 2\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{y}}} \mathbf{w}_j = \mathbf{0}. \quad (21.135)$$

Solving the above matrix equations we obtain the iterative formulas:

$$\tilde{\mathbf{w}}_j = \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}}^{-1} \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} \mathbf{b}_j, \quad \mathbf{w}_j = \frac{\tilde{\mathbf{w}}_j}{\|\tilde{\mathbf{w}}_j\|_2}, \quad (21.136)$$

$$\mathbf{b}_j = \widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}^{-1} \widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{y}}} \mathbf{w}_j = \widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{y}}}^{-1} \widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_jx_j}, \quad (21.137)$$

where in order to avoid the trivial solution  $\mathbf{w}_j = \mathbf{0}$ , we normalize the vector  $\mathbf{w}_j$  to unit length in each iteration step as  $\mathbf{w}_j(k+1) \leftarrow \mathbf{w}_j(k+1)/\|\mathbf{w}_j(k+1)\|_2$  (which ensures that  $E\{x_j^2\} = 1$ ). It is worth to note that in our derivation the matrices  $\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}$  and  $\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}$  are assumed to be independent of the vector  $\mathbf{w}_j(k+1)$ , i.e., they are estimated based on  $\mathbf{w}_j(k)$  from the previous iteration step. This two-phase procedure is similar to the expectation maximization (EM) scheme: (i) Freeze the correlation and cross-correlation matrices and learn the parameters of the processing unit ( $\mathbf{w}_j, \mathbf{b}_j$ ); (ii) freeze  $\mathbf{w}_j$  and  $\mathbf{b}_j$  and learn the new statistics (i.e., matrices  $\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}$  and  $\widehat{\mathbf{R}}_{\tilde{\mathbf{x}}_j\tilde{\mathbf{x}}_j}$ ) of the estimated source signal, then go back to (i) and repeat. Hence, in phase (i) our algorithm extracts a source signal, whereas in phase (ii) it learns the statistics of the source.

The above algorithm can be further simplified. It should be noted that in order to avoid the inversion of the autocorrelation matrix  $\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}}$ , in each iteration step we can perform the standard prewhitening or standard PCA as a preprocessing step and then normalize the sensor signals to have unit variance. In such cases  $\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = \mathbf{I}$  and the algorithm is considerably simplified to [106]

$$\mathbf{w}_j = \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} \mathbf{b}_j = \widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j}, \quad \mathbf{w}_j \leftarrow \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|_2}, \quad (21.138)$$

where  $\widehat{\mathbf{R}}_{\tilde{\mathbf{y}}\tilde{\mathbf{x}}_j} = \frac{1}{T} \sum_{t=1}^T \tilde{\mathbf{y}}(t) \tilde{\mathbf{x}}_j(t) = \langle \tilde{\mathbf{y}}(t) \tilde{\mathbf{x}}_j(t) \rangle$  or in equivalent form

$$\boxed{\mathbf{w}_j(k+1) = \frac{\langle \tilde{\mathbf{y}}(t) \tilde{\mathbf{x}}_j(t) \rangle}{\langle \tilde{\mathbf{x}}_j^2(t) \rangle}}. \quad (21.139)$$

From (21.138) and (21.139) it follows that the derived update rules are similar to the power method in finding the eigenvector associated with the maximal eigenvalue of the matrix  $\mathbf{R}_{\tilde{\mathbf{y}}}(\mathbf{b}_j) = E\{\sum_{p=1}^L b_{jp} \tilde{\mathbf{y}}(t) \tilde{\mathbf{y}}^T(t-p)\}$ . This observation suggests that it is often not necessary to minimize the cost function with respect to the parameters  $\{b_{jp}\}$  but it is enough to choose a random set of them for which the largest eigenvalue is unique (single). More generally, if all eigenvalues of the generalized covariance matrix  $\mathbf{R}_{\tilde{\mathbf{y}}}(\mathbf{b}_j)$  are distinct, then we can extract all sources simultaneously by estimating the principal eigenvectors of  $\mathbf{R}_{\tilde{\mathbf{y}}}(\mathbf{b}_j)$ .

## 1.21.4 NMF and related problems

NMF has been investigated by many researchers, e.g., Paatero and Tapper [108], but it has gained popularity through the works of Lee and Seung published in Nature and NIPS [20, 109]. Based on the argument that the nonnegativity is important for human perception, they proposed simple algorithms (often called the Lee-Seung algorithms) for finding nonnegative representations of nonnegative data and images.

### 1.21.4.1 Why nonnegativity and sparsity constraints?

Many real-world data are nonnegative and the corresponding hidden components have a physical meaning only when nonnegative. In practice, both nonnegative and sparse decompositions of the data are often either desirable or necessary, when the underlying components have a physical interpretation. For example, in image processing and computer vision, the involved variables and parameters may correspond to pixels, and the nonnegative sparse decomposition is related to the extraction of relevant parts from the images [20, 109]. In computer vision and graphics, we often encounter multi-dimensional data, such as images, videos, and medical data, one type of which is MRI (magnetic resonance imaging). A color image can be considered as a 3D nonnegative data sets, two of the dimensions (rows and columns) being spatial, and the third one being a color plane (channel) depending on its color space, while a color video sequence can be considered as a 4D nonnegative data sets with time being the fourth dimension. A sparse representation of the data by a limited number of components is an important research problem. In machine learning sparseness is closely related to feature selection and certain generalizations of learning algorithms, while nonnegativity relates to probability distributions. Generally speaking, compositional data (i.e., positive sum of components or real vectors) are natural representations when the variables (features) are essentially the probabilities of complementary and mutually exclusive events. Furthermore, we may note that NMF is an additive model which does not allow subtraction; therefore it often describes quantitatively the parts that comprise the entire entity. In other words, NMF can be considered as a part-based representation in which a zero-value represents the absence and a positive number represents the presence of some event or component. Specifically, in the case of facial image data, the additive or part-based nature of NMF has been shown to result in a basis of facial features, such as eyes, nose, and lips [20]. Furthermore, the low-rank matrix factorization methods that exploit nonnegativity and sparsity constraints usually lead to estimation of the hidden components with specific structures and physical interpretations, in contrast to other blind source separation methods. In economics, variables and data such as volume, price and many other factors are nonnegative and sparse. Sparseness constraints may increase the efficiency of a portfolio, while nonnegativity both increases the efficiency and reduces the risk [110, 111]. In information retrieval the documents are usually represented as relative frequencies of words form a prescribed vocabulary. In environmental science the scientists investigate a relative proportion of different pollutants in water or air [112]. In biology each coordinate axis may correspond to a specific gene, and the sparseness is necessary for finding local patterns hidden in the data, whereas the nonnegativity is required to give physical or physiological meaning. This is also important for the robustness of biological systems, where any observed change in the expression level of a specific gene emerges from either positive or a negative influence, rather than a combination of both, which partly cancel each other [109, 111]. It is clear, however, that with constraints such as sparsity and nonnegativity some of the explained variance (fit) may decrease. In other words, it is natural to seek a trade-off between the two goals of interpretability (making sure that the estimated components have physical or physiological meaning) and statistical fidelity (explaining most of the variance of the data) if the data are consistent and do not contain too much noise.

### 1.21.4.2 Basic NMF models

The basic NMF problem can be stated as follows: Given a nonnegative data matrix  $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$  (with  $y_{it} \geq 0 \forall it$  or compactly  $\mathbf{Y} \geq \mathbf{0}$ ) and a reduced rank- $J$  ( $J \leq \min(I, T)$ ), find two nonnegative

matrices  $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J] \in \mathbb{R}_+^{I \times J}$  and  $\mathbf{X} = \mathbf{B}^T = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_J]^T \in \mathbb{R}_+^{J \times T}$ , which factorize  $\mathbf{Y} = \mathbf{AX} + \mathbf{E} = \mathbf{AB}^T + \mathbf{E}$ , as well as possible in the sense that a norm or divergence of error matrix  $\mathbf{E}$  is minimized (see Figure 21.1). Since we usually operate on the column vectors of matrices (in order to avoid a complex or confusing notation) it is often convenient to use the matrix  $\mathbf{B} = \mathbf{X}^T$  instead of the matrix  $\mathbf{X}$ . The factors  $\mathbf{A}$  and  $\mathbf{B}$  may have different physical meaning in different applications. In a BSS problem,  $\mathbf{A}$  plays the role of a mixing matrix, while the columns of  $\mathbf{B}$  express source signals. In clustering problems,  $\mathbf{A}$  is the basis matrix while  $\mathbf{B}$  denotes the weight matrix. In acoustic analysis,  $\mathbf{A}$  represents the basis patterns, while each column of  $\mathbf{B}$  expresses the time points (positions) when the sound patterns are activated. In standard NMF we only assume nonnegativity of the factor matrices  $\mathbf{A}$  and  $\mathbf{B}$ . Unlike blind source separation methods based on independent component analysis (ICA), here we do not assume that the sources are independent, although we often impose additional constraints on  $\mathbf{A}$  and/or  $\mathbf{B}$ .

The NMF model can also be represented as a special form of the bilinear model (see Figure 21.2):  $\mathbf{Y} = \sum_{j=1}^J \mathbf{a}_j \circ \mathbf{b}_j + \mathbf{E} = \sum_{j=1}^J \mathbf{a}_j \mathbf{b}_j^T + \mathbf{E}$ . Thus, we can build an approximate representation of the nonnegative data matrix  $\mathbf{Y}$  as a sum of rank-1 nonnegative matrices  $\mathbf{a}_j \mathbf{b}_j^T$ . If such decomposition is exact (i.e.,  $\mathbf{E} = \mathbf{0}$ ) then it is called the Nonnegative Rank Factorization (NRF) [113]. Among the many possible series representations of the data matrix  $\mathbf{Y}$  by nonnegative rank-1 matrices, the smallest integer  $J$  for which such a nonnegative rank-1 series representation is attained is called the nonnegative rank of the nonnegative matrix  $\mathbf{Y}$  and it is denoted by  $\text{rank}_+(\mathbf{Y})$ . The nonnegative rank satisfies the following bounds [113]:

$$\text{rank}(\mathbf{Y}) \leq \text{rank}_+(\mathbf{Y}) \leq \min\{I, T\}. \quad (21.140)$$

Although the NMF can be applied to BSS problems for nonnegative sources and nonnegative mixing matrices, its application is not limited to BSS and it can be used for other diverse applications far beyond BSS. In many applications we require additional constraints on the elements of the matrices  $\mathbf{A}$  and/or  $\mathbf{B}$ , such as smoothness, sparsity, symmetry, and orthogonality.

#### 1.21.4.2.1 Special forms of NMF

The NMF models may take several different forms:

**Symmetric NMF:** In the special case when  $\mathbf{A} = \mathbf{B} \in \mathbb{R}_+^{I \times J}$  the NMF is called a symmetric NMF, given by

$$\mathbf{Y} = \mathbf{AA}^T + \mathbf{E}. \quad (21.141)$$

This model is also considered equivalent to Kernel K-means clustering and Laplace spectral clustering [114].

If the exact symmetric NMF exists ( $\mathbf{E} = \mathbf{0}$ ) then a nonnegative matrix  $\mathbf{Y} \in \mathbb{R}_+^{I \times I}$  is said to be completely positive (cp) and the smallest number of columns of  $\mathbf{A} \in \mathbb{R}_+^{I \times J}$  satisfying the exact factorization  $\mathbf{Y} = \mathbf{AA}^T$  is called the cp-rank of the matrix  $\mathbf{Y}$ , denoted by  $\text{rank}_{cp}(\mathbf{Y})$ . If  $\mathbf{Y}$  is cp, then the upper bound estimate for the cp-rank is given by [113]:

$$\text{rank}_{cp}(\mathbf{Y}) \leq \frac{\text{rank}(\mathbf{Y})(\text{rank}(\mathbf{Y}) + 1)}{2} - 1, \quad (21.142)$$

provided that  $\text{rank}(\mathbf{Y}) > 1$ .

**Orthogonal NMF:** The orthogonal or semi-orthogonal NMF can be defined as

$$\mathbf{Y} = \mathbf{AB}^T + \mathbf{E}, \quad (21.143)$$

subject to nonnegativity constraints  $\mathbf{A} \geq \mathbf{0}$  and  $\mathbf{B} \geq \mathbf{0}$  (component-wise) and an additional orthogonality constraint:  $\mathbf{A}^T \mathbf{A} = \mathbf{I}_J$  or  $\mathbf{B}^T \mathbf{B} = \mathbf{I}_J$ . It can be proved that the orthogonal NMF is equivalent to a weighted variant of spherical K-means clustering.

Probably the simplest and most efficient way to impose orthogonality for the matrix  $\mathbf{A}$  (or  $\mathbf{B}$ ) is to perform the following transformation after each iteration

$$\mathbf{A} \leftarrow \mathbf{A}[\mathbf{A}^T \mathbf{A}]^{-1/2} \quad \text{or} \quad \mathbf{B} \leftarrow [\mathbf{B}^T \mathbf{B}]^{-1/2} \mathbf{B}^T. \quad (21.144)$$

**Semi-NMF and Nonnegative Factorization of an Arbitrary Matrix:** In some applications the observed input data is unsigned (unconstrained or bipolar) as indicated by  $\mathbf{Y} = \mathbf{Y}_{\pm} \in \mathbb{R}^{I \times T}$  which allows us to relax the constraints regarding nonnegativity of one factor (or only specific vectors of a matrix). This leads to the approximative semi-NMF which can take the following form

$$\mathbf{Y}_{\pm} = \mathbf{A}_{\pm} \mathbf{B}_{\pm}^T + \mathbf{E} \quad \text{or} \quad \mathbf{Y}_{\pm} = \mathbf{A}_{+} \mathbf{B}_{\pm}^T + \mathbf{E}, \quad (21.145)$$

where the subscript in  $\mathbf{A}_{+}$  indicates that the matrix is constrained to be nonnegative. Usually, in order to provide uniqueness of the semi-NMF additional constraints must be imposed.

**Nonnegative Factorization (NF):** In some applications the data matrix  $\mathbf{Y}$  has some negative entries. Such case when the matrices  $\mathbf{A}$  and/or  $\mathbf{B}$  are restricted to contain nonnegative entries, but the data matrix  $\mathbf{Y}$  may have entries with mixed signs, is referred to as the Nonnegative Factorization (NF) [115, 116].

**NMF with Offset (Affine NMF):** In NMF with offset (also called affine NMF, aNMF), the goal is to remove the base line or constant bias from the matrix  $\mathbf{Y}$  by using a slightly modified NMF model:

$$\mathbf{Y} = \mathbf{AB}^T + \mathbf{a}_0 \mathbf{1}^T + \mathbf{E}, \quad (21.146)$$

where  $\mathbf{1} \in \mathbb{R}^T$  is a vector of all ones and  $\mathbf{a}_0 \in \mathbb{R}_+^I$  is a vector which is selected in such a way that the matrix  $\mathbf{B}$  is zero-grounded, that is, with a possibly large number of zero entries in each column (or for noisy data close to zero entries). The term  $\mathbf{Y}_0 = \mathbf{a}_0 \mathbf{1}^T$  denotes the offset, which together with the nonnegativity constraint often ensures the sparseness of the factored matrices. The main role of the offset is to absorb the constant values of a data matrix, thereby making the factorization sparser and therefore improving (relaxing) conditions for the uniqueness of NMF (see next sections).

**Three-factor NMF:** Three-factor NMF (also called the tri-NMF) can be considered as a special case of the multi-layer NMF, and it can take the following general form [117, 118]

$$\mathbf{Y} = \mathbf{ASB}^T + \mathbf{E}, \quad (21.147)$$

where the nonnegativity constraints are imposed on all or only on selected factor matrices:  $\mathbf{A} \in \mathbb{R}^{I \times J}$ ,  $\mathbf{S} \in \mathbb{R}^{J \times R}$ , and/or  $\mathbf{B} \in \mathbb{R}^{T \times R}$ .

It should be noted that if we do not impose any additional constraints to the factors (besides nonnegativity), the three-factor NMF can be reduced to the standard (two-factor) NMF by the transformation  $\mathbf{A} \leftarrow \mathbf{AS}$  or  $\mathbf{B} \leftarrow \mathbf{BS}^T$ . However, the three-factor NMF is not equivalent to the standard NMF if we apply special constraints or conditions as illustrated by the following special cases.

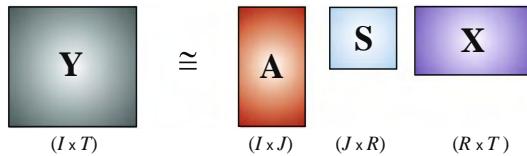
**FIGURE 21.6**

Illustration of three factor NMF (tri-NMF). The goal is to estimate two matrices  $\mathbf{A} \in \mathbb{R}_+^{I \times J}$  and  $\mathbf{B} = \mathbf{X}^T \in \mathbb{R}_+^{T \times R}$ , assuming that the matrix  $\mathbf{S} \in \mathbb{R}^{J \times R}$  is given, or to estimate all three factor matrices  $\mathbf{A}$ ,  $\mathbf{S}$ , and  $\mathbf{X}$ , subject to additional constraints such as orthogonality, smoothness or sparsity.

**Non-Smooth NMF:** Non-smooth NMF (nsNMF) was proposed by [119] and it is a special case of the three-factor NMF model in which the matrix  $\mathbf{S}$  is fixed and known, and it is used for controlling the sparsity or the smoothness of the matrix  $\mathbf{B}$  and/or  $\mathbf{A}$ . Typically, the smoothing matrix  $\mathbf{S} \in \mathbb{R}^{J \times J}$  takes the form:

$$\mathbf{S} = (1 - \Theta)\mathbf{I}_J + \frac{\Theta}{J}\mathbf{1}_{J \times J}, \quad (21.148)$$

where  $\mathbf{I}_J$  is the  $J \times J$  identity matrix and  $\mathbf{1}_{J \times J}$  is the square matrix of all ones. The scalar parameter  $0 \leq \Theta \leq 1$  controls the smoothness of the matrix operator  $\mathbf{S}$ . For  $\Theta = 0$ ,  $\mathbf{S} = \mathbf{I}_J$ , and the model reduces to the standard NMF for  $\Theta \rightarrow 1$  strong smoothing is imposed on  $\mathbf{S}$  causing increased sparseness of both  $\mathbf{A}$  and  $\mathbf{B}$ , in order to maintain the faithfulness of the model.

**CUR Decomposition:** In the CUR decomposition, a given data matrix  $\mathbf{Y} \in \mathbb{R}^{I \times T}$  is decomposed as follows [120–124]:

$$\mathbf{Y} = \mathbf{C}\mathbf{U}\mathbf{R} + \mathbf{E}, \quad (21.149)$$

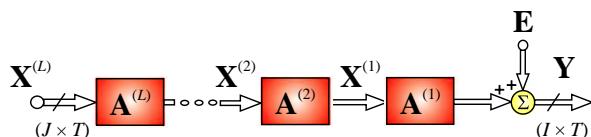
where  $\mathbf{C} \in \mathbb{R}^{I \times C}$  is a matrix constructed from  $C$  selected columns of  $\mathbf{Y}$ ,  $\mathbf{R} \in \mathbb{R}^{R \times T}$  consists of  $R$  rows of  $\mathbf{Y}$  and the matrix  $\mathbf{U} \in \mathbb{R}^{C \times R}$  is chosen to minimize the error  $\mathbf{E} \in \mathbb{R}^{I \times T}$ .

The CUR decomposition for data analysis can be considered as an alternative as low-rank approximation to PCA and SVD, specially for large-scale and sparse data sets. On this regard, the idea of the CUR decomposition is to provide a representation of the data as a linear combination of a few “meaningful” components which are exact replicas of columns and rows of the original data matrix. [121, 123–125]. Since typically,  $C \ll T$  and  $R \ll I$ , the objective is to find a matrix  $\mathbf{U}$  and select rows and columns of  $\mathbf{Y}$  such that the error cost function  $\|\mathbf{E}\|_F^2$  is minimized. There are several strategies for the selection of suitable columns and rows. The main principle is to select columns and rows that exhibit high “statistical leverage” and provide the best low-rank fit to the data matrix [120–122].

The core matrix  $\mathbf{U}$  can be easily computed as  $\mathbf{U} = \mathbf{W}^\dagger$ , where  $\mathbf{W} \in \mathbb{R}^{C \times R}$  is matrix build from entries of  $\mathbf{Y}$  located on the cross-intersections of selected rows and columns. In other words, the core matrix  $\mathbf{U}$  is the Moore–Penrose pseudo-inverse of a matrix  $\mathbf{W} \in \mathbb{R}^{R \times C}$ , i.e.,  $\mathbf{U} = \mathbf{W}^\dagger$ , which is defined by the intersections of the selected rows and columns. Alternatively, we can compute a core matrix  $\mathbf{U}$  as  $\mathbf{U} = \mathbf{C}^\dagger \mathbf{Y} \mathbf{R}^\dagger$ , but in this case the knowledge of the whole data matrix  $\mathbf{Y}$  is necessary (see Figure 21.6).

In the special case, when we assume that  $\mathbf{U}\mathbf{R} = \mathbf{X}$ , we have the CX decomposition:

$$\mathbf{Y} = \mathbf{C}\mathbf{X} + \mathbf{E}. \quad (21.150)$$

**FIGURE 21.7**

Multilayer NMF model. In this model the global factor matrix  $\mathbf{A} = \mathbf{A}^{(1)}\mathbf{A}^{(2)}\dots\mathbf{A}^{(L)}$  has a distributed sparse representation in which each matrix  $\mathbf{A}^{(l)}$  can be sparse.

The CX and CUR decompositions are low-rank matrix decompositions that are explicitly expressed in terms of a small number of actual columns and/or actual rows of the original data matrix and they have recently received increasing attention within the data analysis community, especially for nonnegative data, due to its many applications [121–124]. The CUR decomposition has the advantage that components (factor matrices  $\mathbf{C}$  and  $\mathbf{R}$ ) are directly obtained from the rows and columns of the data matrix  $\mathbf{Y}$ , preserving desired properties such as nonnegativity or sparsity. Because they are constructed from actual data elements, the CUR decomposition is often more easily interpretable by practitioners from the field from which the data are collected (to the extent that the original data points and/or features are interpretable) [123].

**Orthogonal Three-Factor NMF:** Orthogonal three-factor NMF imposes additional constraints upon the two matrices:  $\mathbf{A}^T\mathbf{A} = \mathbf{I}_J$  and  $\mathbf{B}^T\mathbf{B} = \mathbf{I}_R$  while the matrix  $\mathbf{S}$  can be an arbitrary unconstrained matrix (i.e., it has both positive and negative entries) [117, 118].

For the uni-orthogonal three-factor NMF only one matrix ( $\mathbf{A}$  or  $\mathbf{B}$ ) is orthogonal and all three matrices are usually nonnegative.

**Multi-layer NMF:** In multi-layer NMF the basic matrix  $\mathbf{A}$  is replaced by a set of cascaded (factor) matrices. Thus, the model can be described as (see Figure 21.7)

$$\mathbf{Y} = \mathbf{A}^{(1)}\mathbf{A}^{(2)}\dots\mathbf{A}^{(L)}\mathbf{X} + \mathbf{E}. \quad (21.151)$$

Since the model is linear, all the matrices can be merged into a single matrix  $\mathbf{A}$  if no special constraints are imposed upon the individual matrices  $\mathbf{A}^{(l)}$  ( $l = 1, 2, \dots, L$ ). However, multi-layer NMF can be used to considerably improve the performance of standard NMF algorithms due to its distributed structure and the alleviation the problem of local minima.

To improve the performance of the NMF algorithms (especially for ill-conditioned and badly-scaled data) and to reduce the risk of converging to local minima of a cost function due to nonconvex alternating minimization, we have developed a simple hierarchical multi-stage procedure [126] combined with a multi-start initialization, in which we perform a sequential decomposition of nonnegative matrices as follows. In the first step, we perform the basic approximate decomposition  $\mathbf{Y} \cong \mathbf{A}^{(1)}\mathbf{X}^{(1)} \in \mathbb{R}^{I \times T}$  using any available NMF algorithm. In the second stage, the results obtained from the first stage are used to build up an input data matrix  $\mathbf{Y} \leftarrow \mathbf{X}^{(1)}$ , that is, in the next step, we perform a similar decomposition  $\mathbf{X}^{(1)} \cong \mathbf{A}^{(2)}\mathbf{X}^{(2)} \in \mathbb{R}^{J \times T}$ , using the same or different update rules. We continue our decomposition by taking into account only the last obtained components. The process can be repeated for an arbitrary number of times until some stopping criterion is satisfied.

Thus, our multi-layer NMF model has the form:

$$\mathbf{Y} \cong \mathbf{A}^{(1)} \mathbf{A}^{(2)} \cdots \mathbf{A}^{(L)} \mathbf{X}^{(L)}, \quad (21.152)$$

with the final results  $\mathbf{A} = \mathbf{A}^{(1)} \mathbf{A}^{(2)} \cdots \mathbf{A}^{(L)}$  and  $\mathbf{X} = \mathbf{X}^{(L)}$ . Physically, this means that we build up a distributed system that has many layers or cascade connections of  $L$  mixing subsystems. The key point of this approach is that the learning (update) process to find the parameters of the matrices  $\mathbf{X}^{(l)}$  and  $\mathbf{A}^{(l)}$  ( $l = 1, 2, \dots, L$ ) is performed sequentially, layer-by-layer, where each layer is randomly initialized with different initial conditions. We have found that the hierarchical multi-layer approach can improve the performance of most NMF algorithms discussed in this chapter [8, 127].

**Simultaneous NMF:** In simultaneous NMF (siNMF) we have available two or more linked input data matrices (say,  $\mathbf{Y}_1$  and  $\mathbf{Y}_2$ ) and the objective is to decompose them into nonnegative factor matrices in such a way that one of the factor matrices is common (which is a special form of the Nonnegative Tensor Factorization (NTF) model, for example [8]

$$\mathbf{Y}_1 = \mathbf{A}_1 \mathbf{B}^T + \mathbf{E}_1, \quad \mathbf{Y}_2 = \mathbf{A}_2 \mathbf{B}^T + \mathbf{E}_2. \quad (21.153)$$

Such a problem arises, for example, in bio-informatics if we combine gene expression and transcription factor regulation [128]. In this application the data matrix  $\mathbf{Y}_1 \in \mathbb{R}^{I_1 \times T}$  is the expression level of gene  $t$  for the data sample  $i_1$  (i.e., the index  $i_1$  denotes the samples, while  $t$  stands for genes) and  $\mathbf{Y}_2 \in \mathbb{R}^{I_2 \times T}$  is the transcription matrix (which is 1 whenever transcription factor  $i_2$  regulates gene  $t$ ).

**Quadratic NMF:** Quadratic NMF (QNMF) has several forms [129]:

- 1) The asymmetric form (AQNMF)  $\mathbf{Y} \approx \mathbf{A}\mathbf{A}^T\mathbf{X}$ ,
- 2) The symmetric form (SQNMF):  $\mathbf{Y} \approx \mathbf{A}\mathbf{X}\mathbf{A}^T$ ,
- 3) Projective NMF (PNMF):  $\mathbf{Y} \approx \mathbf{A}\mathbf{A}^T\mathbf{Y}$ , for  $\mathbf{X} = \mathbf{Y}$ ,
- 4) Symmetric NMF (SNMF)  $\mathbf{Y} \approx \mathbf{A}\mathbf{A}^T$ , for  $\mathbf{X} = \mathbf{I}$ .

#### 1.21.4.3 Basic approaches to estimate parameters of the standard NMF

In order to estimate factor matrices  $\mathbf{A}$  and  $\mathbf{B}$  for the standard NMF, we need to consider the similarity measure or divergence to quantify the difference between the data matrix  $\mathbf{Y}$  and the approximative NMF model matrix  $\hat{\mathbf{Y}} = \mathbf{AB}^T$ . The choice of the similarity measure (also referred to as distance, divergence or measure of dissimilarity) depends mostly on the probability distribution of the estimated signals or components and on the structure of the data or the distribution of the noise [130]. The simplest and most often used measure is based on the Frobenius norm:

$$D_F(\mathbf{Y} \|\mathbf{AB}^T) = \frac{1}{2} \|\mathbf{Y} - \mathbf{AB}^T\|_F^2, \quad (21.154)$$

which is also referred to as the squared Euclidean distance. It should be noted that the above cost function is convex with respect to either one of the elements of the matrix  $\mathbf{A}$  or the matrix  $\mathbf{B}$ , but not with respect to both. Although the NMF optimization problem is not convex, the objective functions are separately convex in each of the two factors  $\mathbf{A}$  and  $\mathbf{B}$ , which implies that finding the optimal factor matrix  $\mathbf{A}$  corresponding to a fixed matrix  $\mathbf{B}$  reduces to a convex optimization problem and vice versa. However, the convexity is lost as soon as we try to optimize both factor matrices simultaneously [116].

Alternating minimization of such cost function leads to the ALS (Alternating Least Squares) algorithm which can be described as follows:

1. Initialize  $\mathbf{A}$  randomly or by using a specific deterministic strategy.
2. Estimate  $\mathbf{B}$  from the matrix equation  $\mathbf{A}^T \mathbf{AB}^T = \mathbf{A}^T \mathbf{Y}$  by solving

$$\min_{\mathbf{B}} \{D_F(\mathbf{Y} \|\mathbf{AB}^T)\} = \frac{1}{2} \|\mathbf{Y} - \mathbf{AB}^T\|_F^2, \quad \text{with fixed } \mathbf{A}. \quad (21.155)$$

3. Set all the negative elements of  $\mathbf{B}$  to zero or to some small positive value  $\varepsilon$ .
4. Estimate  $\mathbf{A}$  from the matrix equation  $\mathbf{B}^T \mathbf{BA}^T = \mathbf{B}^T \mathbf{Y}^T$  by solving

$$\min_{\mathbf{A}} \{D_F(\mathbf{Y} \|\mathbf{AB}^T)\} = \frac{1}{2} \|\mathbf{Y}^T - \mathbf{BA}^T\|_F^2, \quad \text{with fixed } \mathbf{B}. \quad (21.156)$$

5. Set all negative elements of  $\mathbf{A}$  to zero or to some small positive value  $\varepsilon$ .
6. Repeat the procedure till we achieve convergence.

The above ALS algorithm can be written in the following form (Note that the max operator is applied element-wise, that is, each element of a matrix is compared with the scalar parameter  $\varepsilon$ ):

$$\mathbf{B} \leftarrow \max\{\varepsilon, (\mathbf{Y}^T \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1})\}, \quad \mathbf{A} \leftarrow \max\{\varepsilon, \mathbf{Y}\mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1}\}, \quad (21.157)$$

where  $\varepsilon$  is a small constant (typically,  $10^{-16}$ ) used to enforce positive entries. Various additional constraints (such as sparsity or smoothness) can also be imposed on either  $\mathbf{A}$  or  $\mathbf{B}$ .

Today the ALS method is considered as the basic “workhorse” approach, however it is not guaranteed to converge to a global minimum nor even to a stationary point, but only to a solution where the cost functions cease to decrease [5, 112]. Moreover, it is often not sufficiently accurate. The ALS method can be dramatically improved and its computational complexity reduced as it will be shown later.

It is interesting to note that the NMF problem can be considered as a natural extension of a Nonnegative Least Squares (NLS) problem formulated as the following optimization problem: given a matrix  $\mathbf{A} \in \mathbb{R}^{I \times J}$  and a set of observed values given by the vector  $\mathbf{y} \in \mathbb{R}^I$ , find a nonnegative vector  $\mathbf{x} \in \mathbb{R}^J$  which minimizes the cost function  $J(\mathbf{x}) = \frac{1}{2} \|\mathbf{y} - \mathbf{Ax}\|_2^2$ , i.e.,

$$\min_{\mathbf{x} \geq \mathbf{0}} \frac{1}{2} \|\mathbf{y} - \mathbf{Ax}\|_2^2, \quad (21.158)$$

subject to  $\mathbf{x} \geq \mathbf{0}$ . There is a large volume of literature devoted to NLS problems which can be exploited and adopted to NMF [8].

Most of the existing approaches minimize only one kind of cost function by alternately switching between the sets of parameters. However, we can adopt a more general and flexible approach in which, instead of one cost function, we exploit two or more cost functions (with the same global minima); one of them is minimized with respect to  $\mathbf{A}$  and the other one is minimized with respect to  $\mathbf{B}$ . Such an approach is fully justified as  $\mathbf{A}$  and  $\mathbf{B}$  may have different distributions or different statistical properties and therefore different cost functions may be optimal for each one of them.

Let us consider a basic penalized cost function by considering additional penalty terms:

$$\begin{aligned} D_{Fr}(\mathbf{Y} \|\mathbf{AB}^T) &= \frac{1}{2} \|\mathbf{Y} - \mathbf{AB}^T\|_F^2 + \alpha_{\mathbf{A}} D_{\mathbf{A}}(\mathbf{A}) + \alpha_{\mathbf{B}} D_{\mathbf{B}}(\mathbf{B}) \\ \text{s.t. } a_{ij} &\geq 0, \quad b_{tj} \geq 0, \quad \forall i, j, t, \end{aligned} \quad (21.159)$$

where  $\alpha_{\mathbf{A}}$  and  $\alpha_{\mathbf{B}}$  are nonnegative regularization parameters and the penalty terms  $D_{\mathbf{B}}(\mathbf{B})$ , and  $D_{\mathbf{A}}(\mathbf{A})$  are used to enforce certain application-dependent characteristics for the desired solution. As a special practical case, we have  $D_{\mathbf{B}}(\mathbf{B}) = \sum_{tj} \varphi_{\mathbf{B}}(b_{tj})$ , where  $\varphi(b_{tj})$  are suitably chosen functions which are measures of smoothness or sparsity. To achieve a sparse representation we usually chose  $\varphi_{\mathbf{B}}(b_{tj}) = |b_{tj}|$  or simply  $\varphi_{\mathbf{B}}(b_{tj}) = b_{tj}$  (since,  $b_{tj}$  are nonnegative). Similar regularization terms can also be implemented for the matrix  $\mathbf{A}$ . Note that we treat both matrices  $\mathbf{A}$  and  $\mathbf{B}$  in a similar way.

Applying the standard gradient descent approach, we have

$$a_{ij} \leftarrow a_{ij} - \eta_{ij} \frac{\partial D_{Fr}(\mathbf{A}, \mathbf{B})}{\partial a_{ij}}, \quad b_{tj} \leftarrow b_{tj} - \eta_{tj} \frac{\partial D_{Fr}(\mathbf{A}, \mathbf{B})}{\partial b_{tj}}, \quad (21.160)$$

where  $\eta_{ij}$  and  $\eta_{tj}$  are positive learning rates. The gradients above can be expressed in compact matrix form as

$$\frac{\partial D_{Fr}(\mathbf{A}, \mathbf{B})}{\partial a_{ij}} = [-\mathbf{Y}\mathbf{B} + \mathbf{AB}^T\mathbf{B}]_{ij} + \alpha_{\mathbf{A}} \frac{\partial D_{\mathbf{A}}(\mathbf{A})}{\partial a_{ij}}, \quad (21.161)$$

$$\frac{\partial D_{Fr}(\mathbf{A}, \mathbf{B})}{\partial b_{tj}} = [-\mathbf{Y}^T\mathbf{A} + \mathbf{BA}^T\mathbf{A}]_{tj} + \alpha_{\mathbf{B}} \frac{\partial D_{\mathbf{B}}(\mathbf{B})}{\partial b_{tj}}. \quad (21.162)$$

At this point we can follow the Lee and Seung approach to choose the learning rates [20, 109]

$$\eta_{ij} = \frac{a_{ij}}{[\mathbf{AB}^T\mathbf{B}]_{ij}}, \quad \eta_{tj} = \frac{b_{tj}}{[\mathbf{BA}^T\mathbf{A}]_{jt}}, \quad (21.163)$$

leading to multiplicative updates (refereed to as NMF\_MU):

$$a_{ij} \leftarrow a_{ij} \frac{[[\mathbf{Y}\mathbf{B}]_{ij} - \alpha_{\mathbf{A}} \Psi_{\mathbf{A}}(a_{ij})]_+}{[\mathbf{AB}^T\mathbf{B}]_{ij}}, \quad b_{tj} \leftarrow b_{tj} \frac{[[\mathbf{Y}^T\mathbf{A}]_{tj} - \alpha_{\mathbf{B}} \Psi_{\mathbf{B}}(b_{tj})]_+}{[\mathbf{BA}^T\mathbf{A}]_{jt}}, \quad (21.164)$$

where the functions  $\Psi_{\mathbf{A}}(a_{ij})$  and  $\Psi_{\mathbf{B}}(b_{tj})$  are defined as

$$\Psi_{\mathbf{A}}(a_{ij}) = \frac{\partial D_{\mathbf{A}}(\mathbf{A})}{\partial a_{ij}}, \quad \Psi_{\mathbf{B}}(b_{tj}) = \frac{\partial D_{\mathbf{B}}(\mathbf{B})}{\partial b_{tj}} \quad (21.165)$$

and the regularization parameters  $\alpha_{\mathbf{A}}$  and  $\alpha_{\mathbf{B}}$  should be sufficiently small to ensure the nonnegativity in (21.164) at each iteration, or we can avoid this problem by a half-wave rectifier  $[x]_+ = \max\{x, \varepsilon\}$  with a small  $\varepsilon$ .

The above update rules (21.164), (for  $\alpha_{\mathbf{A}} = \alpha_{\mathbf{B}} = 0$ ) often called Lee-Seung NMF algorithm can be considered as an extension of the well known ISRA (Image Space Reconstruction Algorithm) algorithm proposed first by Daube-Witherspoon and Muehllehner [131] and investigated by many researchers, especially by De Pierro and Byrne [132–136].

In order to impose sparsity, instead of using the  $\ell_2$ -norm, we can use regularization terms based on the  $\ell_1$ -norm, and minimize the cost function:

$$\begin{aligned} D_{F1}(\mathbf{A}, \mathbf{B}) &= \frac{1}{2} \|\mathbf{Y} - \mathbf{AB}^T\|_F^2 + \alpha_{\mathbf{A}} \|\mathbf{A}\|_1 + \alpha_{\mathbf{B}} \|\mathbf{B}\|_1, \\ \text{s.t. } a_{ij} &\geq 0, \quad b_{tj} \geq 0, \quad \forall i, j, t, \end{aligned} \quad (21.166)$$

where the  $\ell_1$ -norms for matrices are defined as  $\|\mathbf{A}\|_1 = \sum_{ij} |a_{ij}|$  and  $\|\mathbf{B}\|_1 = \sum_{tj} |b_{tj}|$ .

This way, the multiplicative learning rules (21.164) for sparse NMF (with controlled sparseness) can be simplified as

$$\boxed{a_{ij} \leftarrow a_{ij} \frac{[(\mathbf{YB})_{ij} - \alpha_{\mathbf{A}}]_+}{[\mathbf{AB}^T \mathbf{B}]_{ij}}, \quad b_{tj} \leftarrow b_{tj} \frac{[(\mathbf{Y}^T \mathbf{A})_{jt} - \alpha_{\mathbf{B}}]_+}{[\mathbf{B} \mathbf{A}^T]_{jt}}}, \quad (21.167)$$

where the normalization of the columns of the matrix  $\mathbf{A}$  at each iteration is performed as  $a_{ij} \leftarrow a_{ij}/\sum_p a_{pj}$ . This algorithm provides a sparse representation for the estimated matrices; the degree of sparseness increases with an increase in the values of the regularization coefficients (typically,  $\alpha_{\mathbf{A}} = \alpha_{\mathbf{B}} = 0.01 - 0.5$ ).

The above multiplicative update rules can be represented in compact matrix form

$$\boxed{\mathbf{A} \leftarrow \mathbf{A} \circledast [(\mathbf{YB} - \alpha_{\mathbf{A}} \mathbf{1}_{I \times J})]_+ \oslash (\mathbf{AB}^T \mathbf{B})}, \quad (21.168)$$

$$\boxed{\mathbf{B} \leftarrow \mathbf{B} \circledast [(\mathbf{Y}^T \mathbf{A} - \alpha_{\mathbf{B}} \mathbf{1}_{T \times J})]_+ \oslash (\mathbf{B} \mathbf{A}^T \mathbf{A})}, \quad (21.169)$$

where the symbols “ $\circledast$ ” and “ $\oslash$ ” denote the Hadamard element-wise product and division, respectively.

#### 1.21.4.4 The HALS algorithm and its extensions

This section focuses on a brief description of a simple local ALS method referred to as the HALS (Hierarchical Alternating Least Squares) algorithm. We call such ALS algorithm hierarchical, since we minimize sequentially a set of simple cost functions which are linked to each other hierarchically via the residual matrices  $\mathbf{Y}^{(j)}$ , which approximate rank-1 bilinear decomposition. Moreover, the HALS algorithm is usually used for multi-layer models in order to improve performance. We highlight the suitability of this method for large-scale NMF problems, and sparse nonnegative coding or representation [126, 137].

We can generalize the concept of ALS by using not only one or two cost functions but rather a set of cost functions to be minimized sequentially or simultaneously. For  $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_1, \dots, \mathbf{a}_J]$  and  $\mathbf{B} = \mathbf{X}^T = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_J]$ , we can express the squared Euclidean cost function as

$$D(\mathbf{a}_1, \mathbf{a}_1, \dots, \mathbf{a}_J, \mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_J) = \frac{1}{2} \|\mathbf{Y} - \mathbf{AB}^T\|_F^2 = \frac{1}{2} \|\mathbf{Y} - \sum_{j=1}^J \mathbf{a}_j \mathbf{b}_j^T\|_F^2. \quad (21.170)$$

An underlying idea is to define the residuals as:

$$\mathbf{Y}^{(j)} = \mathbf{Y} - \sum_{p \neq j} \mathbf{a}_p \mathbf{b}_p^T = \mathbf{Y} - \mathbf{AB}^T + \mathbf{a}_j \mathbf{b}_j^T = \mathbf{E} + \mathbf{a}_j \mathbf{b}_j^T \quad (j = 1, 2, \dots, J), \quad (21.171)$$

and alternately minimize the set of cost functions with respect to the unknown variables  $\mathbf{a}_j, \mathbf{b}_j$ :

$$D_A^{(j)}(\mathbf{a}) = \frac{1}{2} \|\mathbf{Y}^{(j)} - \mathbf{a}\mathbf{b}_j^T\|_F^2, \quad \text{for a fixed } \mathbf{b}_j, \quad (21.172a)$$

$$D_B^{(j)}(\mathbf{b}) = \frac{1}{2} \|\mathbf{Y}^{(j)} - \mathbf{a}_j\mathbf{b}^T\|_F^2, \quad \text{for a fixed } \mathbf{a}_j, \quad (21.172b)$$

for  $j = 1, 2, \dots, J$  subject to  $\mathbf{a} \geq \mathbf{0}$  and  $\mathbf{b} \geq \mathbf{0}$ , respectively.

In other words, we perform alternatively the minimization of the set of cost functions

$$D_F^{(j)}(\mathbf{Y}^{(j)} \|\mathbf{a}_j\mathbf{b}_j^T) = \frac{1}{2} \|\mathbf{Y}^{(j)} - \mathbf{a}_j\mathbf{b}_j^T\|_F^2, \quad (21.173)$$

for  $j = 1, 2, \dots, J$  subject to  $\mathbf{a}_j \geq \mathbf{0}$  and  $\mathbf{b}_j \geq \mathbf{0}$ .

In order to estimate the stationary points we simply compute the gradients of the local cost functions (21.173) with respect to the unknown vectors  $\mathbf{a}_j$  and  $\mathbf{b}_j$  (assuming that the other vectors are fixed) as follows:

$$\nabla_{\mathbf{a}_j} D_F^{(j)}(\mathbf{Y}^{(j)} \|\mathbf{a}_j\mathbf{b}_j^T) = \frac{\partial D_F^{(j)}(\mathbf{Y}^{(j)} \|\mathbf{a}_j\mathbf{b}_j^T)}{\partial \mathbf{a}_j} = \mathbf{a}_j\mathbf{b}_j^T\mathbf{b}_j - \mathbf{Y}^{(j)}\mathbf{b}_j, \quad (21.174)$$

$$\nabla_{\mathbf{b}_j} D_F^{(j)}(\mathbf{Y}^{(j)} \|\mathbf{a}_j\mathbf{b}_j^T) = \frac{\partial D_F^{(j)}(\mathbf{Y}^{(j)} \|\mathbf{a}_j\mathbf{b}_j^T)}{\partial \mathbf{b}_j} = \mathbf{a}_j^T\mathbf{a}_j\mathbf{b}_j - \mathbf{Y}^{(j)T}\mathbf{a}_j. \quad (21.175)$$

$$\mathbf{a}_j \leftarrow \frac{1}{\mathbf{b}_j^T\mathbf{b}_j} [\mathbf{Y}^{(j)}\mathbf{b}_j]_+, \quad \mathbf{b}_j \leftarrow \frac{1}{\mathbf{a}_j^T\mathbf{a}_j} [\mathbf{Y}^{(j)T}\mathbf{a}_j]_+ \quad (j = 1, 2, \dots, J), \quad (21.176)$$

where  $[\mathbf{A}]_+ = \max\{\epsilon, \mathbf{A}\}$ . We refer to these update rules as the HALS algorithm which was first introduced for the NMF in [126]. The same or similar update rules for the NMF have been proposed, extended or rediscovered independently in other publications [116, 138–140]. However, our practical implementations of the HALS algorithm are quite different and allow various extensions to sparse and smooth NMF, and also for the  $N$ -order NTF [8]. It was proved in that for every constant  $\varepsilon > 0$  the limit points of the HALS algorithm initialized with positive matrices and applied to the optimization problem (21.173) are stationary points (see also [139]).

The nonlinear projections can be imposed individually for each source  $\mathbf{b}_j$  and/or vector  $\mathbf{a}_j$ , so the algorithm can be directly extended to a semi-NMF, in which some parameters are relaxed to be unconstrained (by removing the half-wave rectifying  $[\cdot]_+$  operator, if necessary). In practice it is necessary to normalize the column vectors  $\mathbf{a}_j$  and/or  $\mathbf{b}_j$  to unit length vectors (in  $\ell_p$ -norm sense ( $p = 1, 2, \dots, \infty$ )) at each iteration step. In the special case of the  $\ell_2$ -norm, the above algorithm can be further simplified by ignoring the denominators in (21.176) and by imposing a vector normalization after each iterative step, to give a simplified scalar form of the HALS algorithm:

$b_{tj} \leftarrow \left[ \sum_{i=1}^I a_{ij} y_{it}^{(j)} \right]_+, \quad a_{ij} \leftarrow \left[ \sum_{t=1}^T b_{tj} y_{it}^{(j)} \right]_+,$

(21.177)

with  $a_{ij} \leftarrow a_{ij}/\|\mathbf{a}_j\|_2$ , where  $y_{it}^{(j)} = [\mathbf{Y}^{(j)}]_{it} = y_{it} - \sum_{p \neq j} a_{ip}b_{tp}$ .

The above simple algorithm can be further extended or improved with respect to the convergence rate and performance by imposing additional constraints such as sparsity and smoothness. Firstly, observe that the residual matrix  $\mathbf{Y}^{(j)}$  can be rewritten as

$$\begin{aligned}\mathbf{Y}^{(j)} &= \mathbf{Y} - \sum_{p \neq j} \mathbf{a}_p \mathbf{b}_p^T = \mathbf{Y} - \mathbf{AB}^T + \mathbf{a}_j \mathbf{b}_j^T, \\ &= \mathbf{Y} - \mathbf{AB}^T + \mathbf{a}_{j-1} \mathbf{b}_{j-1}^T - \mathbf{a}_{j-1} \mathbf{b}_{j-1}^T + \mathbf{a}_j \mathbf{b}_j^T.\end{aligned}\quad (21.178)$$

It then follows that instead of computing explicitly the residual matrix  $\mathbf{Y}^{(j)}$  at each iteration step, we can just perform smart updates (see [8, 141]).

Different cost functions can be used for the estimation of the columns of matrix  $\mathbf{A}$  and/or  $\mathbf{B}$ , possibly with various additional regularization terms [141]. Furthermore, the columns of  $\mathbf{A}$  can be estimated simultaneously, and the columns of  $\mathbf{B}$  sequentially. In other words, by minimizing the set of cost functions in (21.173) with respect to  $\mathbf{b}_j$ , and simultaneously the cost functions (21.154) with normalization of the columns  $\mathbf{a}_j$  to unit  $\ell_2$ -norm, we obtain a very efficient NMF learning algorithm in which the individual columns of  $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_J]$  are updated locally (column-by-column) and the matrix  $\mathbf{A}$  is updated globally using the global nonnegative ALS (all columns  $\mathbf{a}_j$  simultaneously):

$$\boxed{\mathbf{b}_j \leftarrow [\mathbf{Y}^{(j)T} \mathbf{a}_j]_+, \quad \mathbf{A} \leftarrow [\mathbf{YB}(\mathbf{B}^T \mathbf{B})^{-1}]_+}, \quad (21.179)$$

with the normalization (scaling) of the columns of  $\mathbf{A}$  to unit length in the sense of the  $\ell_2$ -norm after each iteration.

In order to impose sparseness and smoothness constraints for the vectors  $\mathbf{b}_j$  (source signals), we can minimize the following set of cost functions [34, 139]:

$$D_F^{(j)}(\mathbf{Y}^{(j)} \| \mathbf{a}_j \mathbf{b}_j^T) = \frac{1}{2} \|\mathbf{Y}^{(j)} - \mathbf{a}_j \mathbf{b}_j^T\|_F^2 + \alpha_{sp} \|\mathbf{b}_j\|_1 + \alpha_{sm} \|\varphi(\mathbf{Lb}_j)\|_1, \quad (21.180)$$

for  $j = 1, 2, \dots, J$  subject to  $\mathbf{a}_j \geq \mathbf{0}$  and  $\mathbf{b}_j \geq \mathbf{0}$ , where  $\alpha_{sp} > 0, \alpha_{sm} > 0$  are regularization parameters controlling the levels of sparsity and smoothness, respectively,  $\mathbf{L}$  is a suitably designed matrix (the Laplace operator) which measures the smoothness (by estimating the differences between neighboring samples of  $\mathbf{b}_j$ ) and  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  is an edge-preserving function applied component-wise [8].

### 1.21.4.5 Large-scale NMF

In many applications, especially for dimension reduction applications, the data matrix  $\mathbf{Y} \in \mathbb{R}^{I \times T}$  can be very large (with millions of entries), but it can be approximately factorized using a much smaller number of nonnegative components ( $J$ ), that is,  $J \ll I$  and  $J \ll T$ . Then the problem  $\mathbf{Y} \approx \mathbf{AB}^T$  becomes highly redundant and we do not need to use the information about all entries of  $\mathbf{Y}$  in order to estimate precisely the factor matrices  $\mathbf{A} \in \mathbb{R}^{I \times J}$  and  $\mathbf{B} \in \mathbb{R}^{T \times J}$ . In other words, to solve the large-scale NMF problem we do not need to know the whole data matrix, but only a small random part of it. Such an approach can outperform considerably the standard NMF methods, especially for extremely overdetermined systems.

There are several strategies to choose the columns and rows of the input data matrix [120, 123, 142, 143]. In Eqs. (21.168), (21.169), and (21.176), the major bottleneck is caused by the matrix multiplications with the large matrices  $\mathbf{Y}$  and  $\mathbf{Y}_i$ . If these large matrices can be replaced by much smaller matrices, the efficiency of the NMF\_MU and HALS algorithm can be improved. For this reason we consider the following cost function:

$$\begin{aligned} \min_{\tilde{\mathbf{A}}, \tilde{\mathbf{B}}; \mathbf{A}, \mathbf{B}} D(\tilde{\mathbf{A}}, \tilde{\mathbf{B}}; \mathbf{A}, \mathbf{B}) &= \|\mathbf{Y} - \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T\|_F^2 + \|\tilde{\mathbf{A}}\tilde{\mathbf{B}}^T - \mathbf{AB}^T\|_F^2, \\ \text{s.t. } \mathbf{A} &\in \mathbb{R}_+^{I \times J}, \mathbf{B} \in \mathbb{R}_+^{T \times J}, \tilde{\mathbf{A}} \in \mathbb{R}^{I \times R}, \tilde{\mathbf{B}} \in \mathbb{R}^{T \times R}, \end{aligned} \quad (21.181)$$

where the data matrix is first approximated by any low-rank approximation  $\mathbf{Y} \approx \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T$  [144]. Therefore, we can solve (21.181) instead of (21.154) with the same global minimum.

We call this procedure low-rank approximation based NMF (LRA-NMF) [144], which can be performed as follows:

- Step 1: Perform the low-rank approximation using  $\min\{\|\mathbf{Y} - \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T\|_F^2\}$ , where  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{B}}$  are of low-rank  $J$ , without imposing any nonnegativity constraints;
- Step 2: Perform nonnegative matrix factorization using  $\min\{\|\tilde{\mathbf{A}}\tilde{\mathbf{B}}^T - \mathbf{AB}^T\|_F^2\}$  with fixed and known  $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ , where  $\mathbf{A}, \mathbf{B}$  are nonnegative and possibly with additional constraints such as sparsity, smoothness, etc.

The Step 1 of the LRA-NMF can be solved efficiently by using standard PCA or truncated SVD or any other efficient low-rank approximation algorithms such as CUR decomposition. In the Step 2, we assume that the optimal  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{B}}$  have been obtained and then  $\min\{\|\tilde{\mathbf{A}}\tilde{\mathbf{B}}^T - \mathbf{AB}^T\|_F^2\}$  can be solved.

From (21.168)–(21.169) by simply substituting the low-rank approximation  $\mathbf{Y} \approx \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T$ , it yields

$$\mathbf{A} \leftarrow \mathbf{A} \circledast \left[ (\tilde{\mathbf{A}}(\tilde{\mathbf{B}}^T \mathbf{B}) - \alpha_{\mathbf{A}} \mathbf{1}_{I \times J}) \right]_+ \oslash (\mathbf{A}\mathbf{B}^T \mathbf{B}), \quad (21.182)$$

$$\mathbf{B} \leftarrow \mathbf{B} \circledast \left[ (\tilde{\mathbf{B}}(\tilde{\mathbf{A}}^T \mathbf{A}) - \alpha_{\mathbf{B}} \mathbf{1}_{T \times J}) \right]_+ \oslash (\mathbf{B}\mathbf{A}^T \mathbf{A}). \quad (21.183)$$

The algorithm based on (21.182)–(21.183) is named as LRA-NMF\_MU. At a first glance there is not a big difference between it (21.168)–(21.169). However, we should note that the dimensionality of  $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$  is much smaller than that of  $\mathbf{Y}$ . As a matter of fact, LRA-NMF\_MU has much lower time complexity of  $\mathcal{O}(TJ^2)$  and space complexity of  $\mathcal{O}(TJ)$ . In other words, LRA-NMF\_MU is about  $I/J$  times faster than NMF\_MU. In addition, we do not need to load the whole matrix  $\mathbf{Y}$  in the memory during the iterations. Instead,  $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$  are employed explicitly during the NMF iterations.

Similarly to the HALS algorithm, let  $\mathbf{Y}_j = \tilde{\mathbf{A}}\tilde{\mathbf{B}} - \sum_{p \neq j} \mathbf{a}_p \mathbf{b}_p^T$  and the formulas in (21.176) become

$$\mathbf{b}_j \leftarrow \frac{1}{\mathbf{a}_j^T \mathbf{a}_j} \left[ \tilde{\mathbf{B}}(\tilde{\mathbf{A}}^T \mathbf{a}_j) - \mathbf{B}_j(\mathbf{A}_j^T \mathbf{a}_j) \right]_+, \quad (21.184)$$

$$\mathbf{a}_j \leftarrow \frac{1}{\mathbf{b}_j^T \mathbf{b}_j} \left[ \tilde{\mathbf{A}}(\tilde{\mathbf{B}}^T \mathbf{b}_j) - \mathbf{A}_j(\mathbf{B}_j^T \mathbf{b}_j) \right]_+, \quad (21.185)$$

where  $\mathbf{A}_j \in \mathbb{R}^{I \times (J-1)}$  and  $\mathbf{B}_j \in \mathbb{R}^{T \times (J-1)}$  are the sub-matrices of  $\mathbf{A}$  and  $\mathbf{B}$  obtained by removing their  $j$ th columns. The algorithm based on (21.184) is called LRANMF\_HALS. The time complexity of LRA-NMF\_HALS per iteration is  $\mathcal{O}(NR)$  and the space complexity is only  $\mathcal{O}(NR)$ .

Similar approach can be applied to nonnegative tensor factorization (NTF) (see Section 1.21.5).

### 1.21.4.6 Robust NMF algorithms

In many applications the observed data sets are corrupted by large noise (not necessary with a Gaussian distribution). In such cases we need to use a suitably chosen cost function or divergence.

#### 1.21.4.6.1 The alpha-beta divergences

For positive measures  $\mathbf{Y}$  and  $\hat{\mathbf{Y}}$  consider the following generalized dissimilarity measure, which we will refer to as the AB-divergence [130]:

$$D_{AB}^{(\alpha, \beta)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = -\frac{1}{\alpha\beta} \sum_{it} \left( y_{it}^\alpha \hat{y}_{it}^\beta - \frac{\alpha}{\alpha + \beta} y_{it}^{\alpha+\beta} - \frac{\beta}{\alpha + \beta} \hat{y}_{it}^{\alpha+\beta} \right) \quad (21.186)$$

for  $\alpha, \beta, \alpha + \beta \neq 0$ ,

or equivalently

$$D_{AB}^{(\alpha, \lambda-\alpha)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = \frac{1}{(\alpha - \lambda)\alpha} \sum_{it} \left( y_{it}^\alpha \hat{y}_{it}^{\lambda-\alpha} - \frac{\alpha}{\lambda} y_{it}^\lambda - \frac{\lambda - \alpha}{\lambda} \hat{y}_{it}^\lambda \right), \quad (21.187)$$

for  $\alpha \neq 0, \alpha \neq \lambda, \lambda = \alpha + \beta \neq 0$ ,

We shall now illustrate how a suitable choices of the  $(\alpha, \beta)$  parameters simplify the AB-divergence into some existing divergences, including the well-known Alpha- and Beta-divergences [8, 145, 146].

When  $\alpha + \beta = 1$  the AB-divergence reduces to the Alpha-divergence [130, 145, 147].

On the other hand, when  $\alpha = 1$ , it reduces to the Beta-divergence [19, 146, 148, 149].

The AB-divergence reduces to the standard Kullback–Leibler (KL) divergence ( $D_{KL}(\cdot, \cdot)$ ) for  $\alpha = 1$  and  $\beta = 0$

$$D_{AB}^{(1,0)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = D_{KL}(\mathbf{Y} \| \hat{\mathbf{Y}}) = \sum_{it} \left( y_{it} \ln \frac{y_{it}}{\hat{y}_{it}} - y_{it} + \hat{y}_{it} \right), \quad (21.188)$$

and it reduces to the standard Itakura-Saito divergence ( $D_{IS}(\cdot, \cdot)$ ) for  $\alpha = 1$  and  $\beta = -1$  [19, 148]

$$D_{AB}^{(1,-1)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = D_{IS}(\mathbf{Y} \| \hat{\mathbf{Y}}) = \sum_{it} \left( \ln \frac{\hat{y}_{it}}{y_{it}} + \frac{y_{it}}{\hat{y}_{it}} - 1 \right). \quad (21.189)$$

Using the  $1 - \alpha$  deformed logarithm defined as

$$\ln_{1-\alpha}(z) = \begin{cases} \frac{z^\alpha - 1}{\alpha}, & \alpha \neq 0, \\ \ln z, & \alpha = 0, \end{cases} \quad (21.190)$$

where  $z > 0$ , we observe that the AB-divergence is symmetric with respect to both arguments for  $\alpha = \beta \neq 0$  and it takes the form of a metric distance

$$D_{AB}^{(\alpha,\alpha)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = D_E(\ln_{1-\alpha}(\mathbf{Y}) \| \ln_{1-\alpha}(\hat{\mathbf{Y}})) = \frac{1}{2} \sum_{it} (\ln_{1-\alpha}(y_{it}) - \ln_{1-\alpha}(\hat{y}_{it}))^2, \quad (21.191)$$

in the transform domain  $\phi(x) = \ln_{1-\alpha}(x)$ . As particular cases, it includes the scaled squared Euclidean distance (for  $\alpha = 1$ ) and the Hellinger distance (for  $\alpha = 0.5$ ).

#### 1.21.4.6.2 The derivation of robust NMF algorithms based on the AB-divergence

We can easily derive NMF algorithms by employing the AB-divergence with  $\mathbf{Y} = [y_{it}] \in \mathbb{R}_+^{I \times T}$ ,  $\hat{\mathbf{Y}} = [\hat{y}_{it}] = \mathbf{AB}^T \in \mathbb{R}_+^{I \times T}$ , where  $\hat{y}_{it} = [\mathbf{AB}^T]_{it} = \sum_j a_{ij} b_{tj}$ . In this case the gradient of the AB-divergence (21.187) can be expressed in a compact form (for any  $\alpha, \beta \in \mathbf{R}$ ) in terms of  $1 - \alpha$  deformed logarithm (see (21.190))

$$\frac{\partial D_{AB}^{(\alpha, \beta)}}{\partial b_{tj}} = - \sum_{i=1}^I \hat{y}_{it}^{\lambda-1} a_{ij} \ln_{1-\alpha} \left( \frac{y_{it}}{\hat{y}_{it}} \right), \quad (21.192)$$

$$\frac{\partial D_{AB}^{(\alpha, \beta)}}{\partial a_{ij}} = - \sum_{t=1}^T \hat{y}_{it}^{\lambda-1} b_{tj} \ln_{1-\alpha} \left( \frac{y_{it}}{\hat{y}_{it}} \right). \quad (21.193)$$

The multiplicative learning rules are gradient descent based in the natural parameter space  $\phi(x)$  of the proposed divergence:

$$a_{ij} \leftarrow \phi^{-1} \left( \phi(a_{ij}) - \eta_{ij} \frac{\partial D_{AB}^{(\alpha, \beta)}}{\partial \phi(a_{ij})} \right), \quad b_{tj} \leftarrow \phi^{-1} \left( \phi(b_{tj}) - \eta_{tj} \frac{\partial D_{AB}^{(\alpha, \beta)}}{\partial \phi(b_{tj})} \right). \quad (21.194)$$

These updates can be considered as a generalization of the exponentiated gradient (EG) algorithm [130, 150].

In general, such a nonlinear scaling (or transformation) provides a stable solution and the resulting gradients are much better behaved in the  $\phi$  space. We use the  $1 - \alpha$  deformed logarithm transformation  $\phi(z) = \ln_{1-\alpha}(z)$ , whose inverse transformation is the  $1 - \alpha$  deformed exponential

$$\phi^{-1}(z) = \exp_{1-\alpha}(z) = \begin{cases} \exp(z) & \text{for } \alpha = 0, \\ (1 + \alpha z)^{\frac{1}{\alpha}} & \text{for } \alpha \neq 0 \text{ and } 1 + \alpha z \geq 0, \\ 0 & \text{for } \alpha \neq 0 \text{ and } 1 + \alpha z < 0. \end{cases} \quad (21.195)$$

For positive measures  $z > 0$ , the direct transformation  $\phi(z)$  and the composition  $\phi^{-1}(\phi(z))$  are bijective functions which define a one to one correspondence, so we have  $\phi^{-1}(\phi(z)) = z$ .

By choosing suitable learning rates

$$\eta_{tj} = \frac{b_{tj}^{2\alpha-1}}{\sum_{i=1}^I a_{ij} \hat{y}_{it}^{\lambda-1}}, \quad \eta_{ij} = \frac{a_{ij}^{2\alpha-1}}{\sum_{t=1}^T b_{tj} \hat{y}_{it}^{\lambda-1}}. \quad (21.196)$$

a generalized multiplicative NMF algorithm (refereed to as the AB-multiplicative NMF algorithm) is obtained as [130]:

$$\begin{aligned} b_{tj} &\leftarrow b_{tj} \exp_{1-\alpha} \left( \sum_{i=1}^I \frac{a_{ij} \hat{y}_{it}^{\lambda-1}}{\sum_{i=1}^I a_{ij} \hat{y}_{it}^{\lambda-1}} \ln_{1-\alpha} \left( \frac{y_{it}}{\hat{y}_{it}} \right) \right), \\ a_{ij} &\leftarrow a_{ij} \exp_{1-\alpha} \left( \sum_{t=1}^T \frac{b_{tj} \hat{y}_{it}^{\lambda-1}}{\sum_{t=1}^T b_{tj} \hat{y}_{it}^{\lambda-1}} \ln_{1-\alpha} \left( \frac{y_{it}}{\hat{y}_{it}} \right) \right), \end{aligned} \quad (21.197)$$

In these equations, the deformed logarithm of order  $1 - \alpha$  of the quotients  $y_{it}/\hat{y}_{it}$  plays a key role in controlling the relative error terms whose weighted mean provides the multiplicative corrections. This deformation in the relative error is controlled by the parameter  $\alpha$ . The parameter  $\alpha > 1$  gives more relevance to large values of the quotient, while the case of  $\alpha < 1$  puts more emphasis on smaller values of the quotient. On the other hand, the parameter  $\lambda - 1$  (where  $\lambda = \alpha + \beta$ ) controls the influence of the values of the approximation ( $\hat{y}_{it}$ ) on the weighing of the deformed error terms. For  $\lambda = 1$ , this influence disappears.

It is interesting to note that the multiplicative term of the main updates

$$M_\alpha(\mathbf{z}, \mathbf{w}, S) = \exp_{1-\alpha} \left( \frac{1}{\sum_{i \in S} w_i} \sum_{i \in S} w_i \ln_{1-\alpha}(z_i) \right), \quad (21.198)$$

can be interpreted as a weighted generalized mean across the elements with indices in the set  $S$ .

Depending on the value of  $\alpha$  we obtain the following particular cases: the minimum of the vector  $\mathbf{z}$  (for  $\alpha \rightarrow -\infty$ ), its weighted harmonic mean ( $\alpha = -1$ ), the weighted geometric mean ( $\alpha = 0$ ), the arithmetic mean ( $\alpha = 1$ ), the weighted quadratic mean ( $\alpha = 2$ ) and the maximum of the vector ( $\alpha \rightarrow \infty$ ), i.e.,

$$M_\alpha(\mathbf{z}, \mathbf{w}, \{1, \dots, n\}) = \begin{cases} \min\{z_1, \dots, z_n\}, & \alpha \rightarrow -\infty, \\ (\sum_{i=1}^n w_i) \left( \sum_{i=1}^n \frac{w_i}{z_i} \right)^{-1}, & \alpha = -1, \\ \prod_{i=1}^n z^{\frac{w_i}{\sum_{i=1}^n w_i}}, & \alpha = 0, \\ \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i z_i, & \alpha = 1, \\ \left( \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i z_i^2 \right)^{1/2}, & \alpha = 2, \\ \max\{z_1, \dots, z_n\}, & \alpha \rightarrow \infty. \end{cases} \quad (21.199)$$

The generalized weighted means are monotonically increasing functions of  $\alpha$ , i.e., if  $\alpha_1 < \alpha_2$ , then

$$M_{\alpha_1}(\mathbf{z}, \mathbf{w}, S) < M_{\alpha_2}(\mathbf{z}, \mathbf{w}, S). \quad (21.200)$$

Thus, by increasing the values of  $\alpha$ , we put more emphasis on large relative errors in the update formulas (21.197).

In the special case of  $\alpha \neq 0$ , the above update rules can be simplified as:

$$\boxed{a_{ij} \leftarrow a_{ij} \left( \frac{\sum_{t=1}^T b_{tj} y_{it}^\alpha \hat{y}_{it}^{\beta-1}}{\sum_{t=1}^T b_{tj} \hat{y}_{it}^{\alpha+\beta-1}} \right)^{1/\alpha}, \quad b_{tj} \leftarrow b_{tj} \left( \frac{\sum_{i=1}^I a_{ij} y_{it}^\alpha \hat{y}_{it}^{\beta-1}}{\sum_{i=1}^I a_{ij} \hat{y}_{it}^{\alpha+\beta-1}} \right)^{1/\alpha}}, \quad (21.201)$$

where  $\hat{y}_{it} = [\mathbf{A}\mathbf{B}^T]_{it}$  and at every iteration the columns of  $\mathbf{A}$  are normalized to the unit length. The above multiplicative update rules can be written in a compact matrix form as

$$\mathbf{A} \leftarrow \mathbf{A} \circledast [(\mathbf{Z}\mathbf{B}) \oslash (\hat{\mathbf{Y}}^{[\alpha+\beta-1]} \mathbf{B})]^{[1/\alpha]}, \quad (21.202)$$

$$\mathbf{B} \leftarrow \mathbf{B} \circledast [(\mathbf{Z}^T \mathbf{A}) \oslash ((\hat{\mathbf{Y}}^{[\alpha+\beta-1]})^T \mathbf{A})]^{[1/\alpha]}, \quad (21.203)$$

where  $\hat{\mathbf{Y}} = \max\{\mathbf{A}\mathbf{X}, \varepsilon\}$  and  $\mathbf{Z} = \mathbf{Y}^{[\alpha]} \circledast \hat{\mathbf{Y}}^{[\beta-1]}$ .

In order to address the scaling indeterminacy between the columns of  $\mathbf{A}$  and the rows of  $\mathbf{X}$ , in practice, after each iteration, we can usually evaluate the  $l_1$ -norm of the columns of  $\mathbf{A}$  and normalize the elements of the matrices as  $a_{ij} \leftarrow a_{ij} / \sum_p a_{pj}$ . This normalization does not alter  $\hat{\mathbf{Y}} = \mathbf{AX}$ , thus, preserving the value of the AB-divergence.

The above multiplicative update rules are natural extensions of many existing algorithms for NMF, including the ISRA, EMML, Lee-Seung algorithms and Alpha- and Beta-multiplicative NMF algorithms [8, 130]. For example, by selecting  $\alpha + \beta = 1$ , we obtain the Alpha-NMF algorithm, for  $\alpha = 1$ , we have Beta-NMF algorithms, for  $\alpha = -\beta \neq 0$  we obtain a family of multiplicative NMF algorithms based on the extended Itakura-Saito distance [19, 148]. Furthermore, for  $\alpha = 1$  and  $\beta = 1$ , we obtain the ISRA algorithm and for  $\alpha = 1$  and  $\beta = 0$  we obtain the EMML (Expectation Maximization Maximum Likelihood) algorithm.

It is important to note that in low-rank approximations, we do not need access to all input data  $y_{it}$ . In other words, the above algorithms can be applied for low-rank approximations even if some data are missing or they are purposely omitted or ignored. For large-scale problems the learning rules can be written in a more efficient form by restricting the generalized mean only to those elements whose indices belong to the preselected subsets  $S_T \subset \{1, \dots, T\}$  and  $S_I \subset \{1, \dots, I\}$  of the whole set of indices [130].

Using a duality property ( $D_{AB}^{(\alpha, \beta)}(\mathbf{Y} \| \hat{\mathbf{Y}}) = D_{AB}^{(\beta, \alpha)}(\hat{\mathbf{Y}} \| \mathbf{Y})$ ) of the AB-divergence, we obtain now the dual update rules for  $\beta \neq 0$

$$\boxed{a_{ij} \leftarrow a_{ij} \left( \frac{\sum_{t \in S_T} b_{tj} y_{it}^{\alpha-1} \hat{y}_{it}^\beta}{\sum_{t \in S_T} b_{tj} y_{it}^{\alpha+\beta-1}} \right)^{1/\beta}, \quad b_{tj} \leftarrow b_{tj} \left( \frac{\sum_{i \in S_I} a_{ij} y_{it}^{\alpha-1} \hat{y}_{it}^\beta}{\sum_{i \in S_I} a_{ij} y_{it}^{\alpha+\beta-1}} \right)^{1/\beta}.} \quad (21.204)$$

#### 1.21.4.6.3 Why is the AB-divergence potentially robust?

To illustrate the role of the hyperparameters  $\alpha$  and  $\beta$  on the robustness of the AB-divergence with respect to errors and noises we will compare the behavior of the AB-divergence with the standard Kullback–Leibler divergence [130]. We will assume, without loss of generality, that the proposed factorization model  $\hat{\mathbf{Y}}$  (for given noisy (observed)  $\mathbf{Y}$ ) is a function of the vector of parameters  $\theta$  and that each of its elements  $\hat{y}_{it}(\theta) > 0$  is non-negative for a certain range of the parameters, say  $\Theta$ .

The estimator  $\hat{\theta}$  obtained for the Kullback-Leibler divergence between two discrete positive measures  $\mathbf{Y}$  and  $\hat{\mathbf{Y}}$ , is the solution of

$$\frac{\partial D_{KL}(\mathbf{Y} \| \hat{\mathbf{Y}})}{\partial \theta} = - \sum_{it} \frac{\partial \hat{y}_{it}}{\partial \theta} \ln_0 \left( \frac{y_{it}}{\hat{y}_{it}} \right) = \mathbf{0}, \quad (21.205)$$

while, for the Beta-divergence, the estimator solves

$$\frac{\partial D_B^{(\beta)}(\mathbf{Y} \| \hat{\mathbf{Y}})}{\partial \theta} = - \sum_{it} \frac{\partial \hat{y}_{it}}{\partial \theta} \hat{y}_{it}^\beta \ln_0 \left( \frac{y_{it}}{\hat{y}_{it}} \right) = \mathbf{0}. \quad (21.206)$$

The main difference between these equations is in the weighting factors  $\hat{y}_{it}^\beta$  for the Beta-divergence which are controlled by the parameter  $\beta$ . In the context of probability distributions, these weighting

factors may control the influence of the likelihood ratios  $y_{it}/\hat{y}_{it}$ . The parameter  $\beta$  determines a tradeoff between robustness to outliers (for  $\beta > 0$ ) and efficiency (for  $\beta$  near 0) (see e.g., [130]). In the special case of  $\beta = 1$  the Euclidean distance is obtained, which is known to be more robust and less efficient than the Kullback–Leibler divergence (for  $\beta = 0$ ).

On the other hand, for the Alpha-divergence, the estimating equation takes a different form

$$\frac{\partial D_A^{(\alpha)}(\mathbf{Y} \parallel \hat{\mathbf{Y}})}{\partial \theta} = - \sum_{it} \frac{\partial \hat{y}_{it}}{\partial \theta} \ln_{1-\alpha} \left( \frac{y_{it}}{\hat{y}_{it}} \right) = \mathbf{0}. \quad (21.207)$$

In this case, the influence of the values of individual ratios  $y_{it}/\hat{y}_{it}$  is controlled not by weighting factors but by the deformed logarithm of order  $1 - \alpha$ . This feature can be interpreted as a zoom or over-weight of the interesting details of the likelihood ratio. For  $\alpha > 1$  (a zoom-out), we emphasize the relative importance of larger values of the ratio  $y_{it}/\hat{y}_{it}$ , whereas for  $\alpha < 1$  (a zoom-in), we put more emphasis on smaller values of  $y_{it}/\hat{y}_{it}$ . The major consequence is the inclusive ( $\alpha \rightarrow \infty$ ) and exclusive ( $\alpha \leftarrow -\infty$ ) behavior of the Alpha-divergence discussed by [147].

The estimating equation for the AB divergence combines both effects:

$$\frac{\partial D_{AB}^{(\alpha, \beta)}(\mathbf{Y} \parallel \hat{\mathbf{Y}})}{\partial \theta} = - \sum_{it} \frac{\partial \hat{y}_{it}}{\partial \theta} \underbrace{\hat{y}_{it}^{\alpha+\beta-1}}_{\text{weights}} \underbrace{\ln_{1-\alpha}(y_{it}/\hat{y}_{it})}_{\alpha\text{-zoom}} = \mathbf{0}, \quad (21.208)$$

and therefore is much more flexible and powerful regarding robustness to error and noise. Depending on the value of  $\alpha$  we can zoom-in or zoom-out the interesting sets of the ratios  $y_{it}/\hat{y}_{it}$  and simultaneously weight these ratios by scaling factors  $\hat{y}_{it}^{\lambda-1}$  controlled by the parameter  $\lambda = \alpha + \beta$ . Therefore, the parameter  $\alpha$  can be used to control the influence of large or small ratios in the estimator, while the parameter  $\beta$  provides some control on the weighting of the ratios depending on the demand to provide better fit to larger or smaller values of the model [130].

#### 1.21.4.7 Convulsive NMF

The Convulsive NMF (CNMF) is a natural extension and generalization of the standard NMF. In the CNMF, we process a set of nonnegative matrices or patterns which are shifted (or time delayed) versions of the primary matrix  $\mathbf{X}$  [151]. In its simplest form the CNMF can be described as

$$\mathbf{Y} = \sum_{r=0}^{R-1} \mathbf{A}_r \mathbf{X} \mathbf{T}_r + \mathbf{E} = \sum_{r=0}^{R-1} \mathbf{A}_r \overset{r \rightarrow}{\mathbf{X}} + \mathbf{E}, \quad (21.209)$$

where  $\mathbf{Y} \in \mathbb{R}_+^{I \times T}$  is a given input data matrix,  $\mathbf{A}_r \in \mathbb{R}_+^{I \times J}$  is a set of unknown nonnegative basis matrices,  $\mathbf{X} = \overset{0 \rightarrow}{\mathbf{X}} \in \mathbb{R}_+^{J \times T}$  is a matrix representing primary sources or patterns, and  $\overset{r \rightarrow}{\mathbf{X}}$  is a shifted by  $r$  columns version of  $\mathbf{X}$ . In other words,  $\overset{r \rightarrow}{\mathbf{X}}$  means that the columns of  $\mathbf{X}$  are shifted to the right  $r$  spots (columns), while the entries in the columns shifted into the matrix from the outside are set to zero. We denote a shift matrix  $\mathbf{T}_r$  of size  $T \times T$  by a binary matrix with ones only on the  $r$ th superdiagonal for

$r > 0$ , or on the  $r$ th subdiagonal for  $r < 0$ , and zeroes elsewhere.  $\overset{r \rightarrow}{\mathbf{X}} = \mathbf{X}\mathbf{T}_r$  is an  $r$  column shifted version of  $\mathbf{X}$  to the right, with the columns shifted in from outside the matrix set to zero. Note that,  $\overset{0 \rightarrow}{\mathbf{X}} = \overset{\leftarrow 0}{\mathbf{X}} = \mathbf{X}$ .

The shift operator can be illustrated by the following simple example:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \overset{1 \rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 4 & 5 \end{bmatrix}, \quad \overset{2 \rightarrow}{\mathbf{X}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 4 \end{bmatrix}, \quad \overset{\leftarrow 1}{\mathbf{X}} = \begin{bmatrix} 2 & 3 & 0 \\ 5 & 6 & 0 \end{bmatrix}.$$

The goal is to estimate the input sources represented by the nonnegative matrix  $\mathbf{X} \in \mathbb{R}_+^{J \times T}$  (typically,  $T \gg J$ ) and to identify the convoluting system, i.e., to estimate a set of nonnegative matrices  $\{\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{R-1}\}$  ( $\mathbf{A}_r \in \mathbb{R}_+^{I \times J}$ ,  $r = 0, 1, \dots, R-1$ ) knowing only the input data matrix  $\mathbf{Y} \in \mathbb{R}^{I \times T}$ . Each operator  $\mathbf{T}_r$  ( $r = 0, 1, \dots, R-1$ ) performs a horizontal shift of the columns in  $\mathbf{X}$  by one spot.

In the simplest case scenario, update rules for  $\mathbf{A}_r$  and  $\mathbf{X}$  can be derived by minimizing a standard cost function:

$$D(\mathbf{Y} \|\widehat{\mathbf{Y}}) = \frac{1}{2} \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_F^2 = \frac{1}{2} \|\mathbf{Y} - \sum_{r=0}^{R-1} \mathbf{A}_r \mathbf{X} \mathbf{T}_r\|_F^2. \quad (21.210)$$

Note that, the approximation of  $\mathbf{Y}$  can be expressed as a standard NMF problem with rank- $JR$  as

$$\mathbf{Y} = [\mathbf{A}_0 \ \mathbf{A}_1 \ \dots \ \mathbf{A}_{R-1}] \begin{bmatrix} \mathbf{X} \\ \overset{1 \rightarrow}{\mathbf{X}} \\ \vdots \\ \overset{(R-1) \rightarrow}{\mathbf{X}} \end{bmatrix} + \mathbf{E} = \overline{\mathbf{AB}}^T + \mathbf{E}, \quad (21.211)$$

where  $\overline{\mathbf{A}} = [\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{R-1}] \in \mathbb{R}^{I \times RJ}$  and  $\overline{\mathbf{B}} = [\mathbf{X}, \overset{1 \rightarrow}{\mathbf{X}}, \dots, \overset{(R-1) \rightarrow}{\mathbf{X}}]^T \in \mathbb{R}^{RT \times J}$ .

From (21.209) and (21.210), we can apply any existing standard NMF algorithm. For example, we can employ a simple multiplicative update rule using the ISRA algorithm [8]:

$$\overline{\mathbf{A}} \leftarrow \overline{\mathbf{A}} \circledast (\mathbf{Y}\overline{\mathbf{B}}) \oslash (\widehat{\mathbf{Y}}\overline{\mathbf{B}}) = \overline{\mathbf{A}} \circledast [\mathbf{Y}\mathbf{T}_r^T \mathbf{X}^T]_{r=0}^{R-1} \oslash [\widehat{\mathbf{Y}}\mathbf{T}_r^T \mathbf{X}^T]_{r=0}^{R-1}, \quad (21.212)$$

which can be rewritten for the component matrices  $\mathbf{A}_r$ ,

$$\mathbf{A}_r \leftarrow \mathbf{A}_r \circledast (\mathbf{Y}\mathbf{T}_r^T \mathbf{X}^T) \oslash (\widehat{\mathbf{Y}}\mathbf{T}_r^T \mathbf{X}^T) \quad (r = 0, 1, \dots, R-1). \quad (21.213)$$

$$\mathbf{X} \leftarrow \mathbf{X} \circledast \left( \sum_{r=0}^{R-1} \mathbf{A}_r^T \mathbf{Y} \mathbf{T}_r^T \right) \oslash \left( \sum_{r=0}^{R-1} \mathbf{A}_r^T \widehat{\mathbf{Y}} \mathbf{T}_r^T \right). \quad (21.214)$$

The update rules (21.213)–(21.214) are particular cases of the multiplicative algorithm for CNMF2D proposed first in [152] and extended in [153].

For the more general scenarios, the CNMF searches for basis patterns which shift vertically [154] or horizontally [155] or in both directions [152] in the nonnegative data matrix. More advanced and efficient algorithms for generalized and flexible CNMF models can be found in [153, 156].

The CNMF has found a number of applications in music analysis, source detection, image processing [155, 157]. In the CNMF model, temporal continuity exhibited by many audio signals are usually expressed efficiently in the time-frequency domain, especially for signals whose frequencies vary with time.

## 1.21.5 Future directions: constrained multi-block tensor factorizations and multilinear blind source separation

In this section, we discuss an important problem: how to extend established and efficient algorithms for constrained matrix factorization techniques (especially, PCA, ICA and NMF) to tensor scenarios. In what follows, this approach is referred to as multiway Generalized Component Analysis (GCA) or multilinear blind source separation (MBSS).

The constrained matrix factorization techniques are widely applied to PCA, ICA, and NMF, dimensionality reduction, data compression and feature extraction. Although standard 2D BSS (constrained matrix factorizations) approaches, such as PCA, ICA, NMF, and their variants, are invaluable tools for BSS, feature extraction and selection, dimensionality reduction, noise reduction, and data mining, it should be noted that they have only two modes or two-way representations (typically, space and time), and their use is therefore limited. In many applications the data structures often contain higher-order ways (modes) such as subjects, groups, trials, classes, and conditions, together with the intrinsic dimensions of space, time, and frequency. For example, studies in neuroscience often involve multiple subjects (people or animals) and trials, leading to experimental data structures conveniently represented by multiway arrays or blocks of multiway data.

How to find informative and sparse/compact representations of experimental or measured multi-dimensional large tensor data is a fundamental and challenging problem in data mining and data analysis.

### 1.21.5.1 Basic tensor multilinear operations

Tensors are denoted by underlined capital boldface letters, e.g.,  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ . The order of a tensor is the number of modes, also known as ways or dimensions (e.g., space, time, frequency, subjects, trials, classes, groups, and conditions). In contrast, matrices (second-order tensors) are denoted by boldface capital letters, e.g.,  $\mathbf{Y}$ ; vectors (first-order tensors) are denoted by boldface lowercase letters, e.g., the columns of the matrix  $\mathbf{A}$  are denoted by  $\mathbf{a}_j$  and scalars are denoted by lowercase letters, e.g.,  $a_{ij}$ .

Throughout this section, standard notations and basic tensor operations proposed in the literature [5, 8] are used. Specifically, the mode- $n$  product

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_n \mathbf{A} \quad (21.215)$$

of a tensor  $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_N}$  and a matrix  $\mathbf{A} \in \mathbb{R}^{I \times J_n}$  is a tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{J_1 \times \dots \times J_{n-1} \times I \times J_{n+1} \times \dots \times J_N}$ , with elements

$$y_{j_1, j_2, \dots, j_{n-1}, i, j_{n+1}, \dots, j_N} = \sum_{j_n=1}^{J_n} (g_{j_1, j_2, \dots, j_N})(a_{i, j_n}). \quad (21.216)$$

The mode- $n$  multiplication of a tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  by a vector  $\mathbf{a} \in \mathbb{R}^{I_n}$  is denoted by

$$\underline{\mathbf{Y}} \bar{\times}_n \mathbf{a} \quad (21.217)$$

and has dimension  $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$ , that is,

$$\underline{\mathbf{Z}} = \underline{\mathbf{Y}} \bar{\times}_n \mathbf{a} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}, \quad (21.218)$$

and element-wise, we have

$$z_{i_1, i_2, \dots, i_{n-1}, i_{n+1}, \dots, i_N} = \sum_{i_n=1}^{I_n} y_{i_1, i_2, \dots, i_N} a_{i_n}. \quad (21.219)$$

A bar over the operator  $\times$  indicates a contracted product. For example, using this notation we can write:  $\mathbf{Y}\mathbf{w} = \underline{\mathbf{Y}} \bar{\times}_2 \mathbf{w}$  and  $\mathbf{v}^T \mathbf{Y}\mathbf{w} = \underline{\mathbf{Y}} \bar{\times}_1 \mathbf{v} \bar{\times}_2 \mathbf{w}$ . Multiplying a third-order tensor by vectors in two modes results in a first-order tensor (a vector); multiplying it in all modes results in a scalar. We can exchange the order of multiplication by the using following rule:

$$\underline{\mathbf{Y}} \bar{\times}_m \mathbf{a} \bar{\times}_n \mathbf{b} = (\underline{\mathbf{Y}} \bar{\times}_m \mathbf{a}) \bar{\times}_n \mathbf{b} = (\underline{\mathbf{Y}} \bar{\times}_n \mathbf{b}) \bar{\times}_m \mathbf{a}, \quad (21.220)$$

For example, the mode- $n$  multiplication of a tensor  $\underline{\mathbf{G}} \in \mathbb{R}^{J \times R \times P}$  by vectors  $\mathbf{a} \in \mathbb{R}^J$ ,  $\mathbf{b} \in \mathbb{R}^R$  and  $\mathbf{c} \in \mathbb{R}^P$  can be expressed as

$$z = \underline{\mathbf{G}} \bar{\times}_1 \mathbf{a} \bar{\times}_2 \mathbf{b} \bar{\times}_3 \mathbf{c} = \sum_{j=1}^J \sum_{r=1}^R \sum_{p=1}^P g_{jrp} a_j b_r c_p.$$

More generally, for  $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$  and  $\mathbf{a}^{(n)} \in \mathbb{R}^{J_n}$ , the multiplication by all vectors in all modes ( $n = 1, 2, \dots, N$ ) gives a scalar:

$$y = \underline{\mathbf{G}} \bar{\times}_1 \mathbf{a}^{(1)} \bar{\times}_2 \mathbf{a}^{(2)} \cdots \bar{\times}_N \mathbf{a}^{(N)} = \underline{\mathbf{G}} \bar{\times} \{\mathbf{a}\} \in \mathbb{R}, \quad (21.221)$$

whereas the multiplication in every mode except mode- $n$  results in a vector  $\mathbf{x}$  of length  $J_n$ :

$$\begin{aligned} \mathbf{x} &= \underline{\mathbf{G}} \bar{\times}_1 \mathbf{a}^{(1)} \cdots \bar{\times}_{n-1} \mathbf{a}^{(n-1)} \bar{\times}_{n+1} \mathbf{a}^{(n+1)} \cdots \bar{\times}_N \mathbf{a}^{(N)} \\ &= \underline{\mathbf{G}}_{(n)}(\mathbf{a}^{(N)} \otimes \cdots \otimes \mathbf{a}^{(n+1)} \otimes \mathbf{a}^{(n-1)} \otimes \cdots \otimes \mathbf{a}^{(1)}) = \underline{\mathbf{G}} \bar{\times}_{-n} \{\mathbf{a}\} \in \mathbb{R}^{J_n}. \end{aligned} \quad (21.222)$$

Also note that multiplication in every mode except mode- $n$  and mode- $m$  results in a matrix of size  $J_n \times J_m$ .

The mode-1 product of two tensors:  $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  and  $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ , e.g., with a common first mode  $I_1 = J_1$  is defined as a tensor:

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \times_1 \underline{\mathbf{B}} \in \mathbb{R}^{I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M}, \quad (21.223)$$

with entries  $c_{\mathbf{i}_{2:N}, \mathbf{j}_{2:M}} = \sum_{i=1}^{I_1} a_{i, \mathbf{i}_{2:N}} b_{i, \mathbf{j}_{2:M}}$ , (where we use the shorthand MATLAB notation  $\mathbf{i}_{p:q} = \{i_p, i_{p+1}, \dots, i_{q-1}, i_q\}$ ).

The outer product  $\underline{\mathbf{C}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}}$  of two tensors  $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$  and  $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times \dots \times J_M}$  is a tensor  $\underline{\mathbf{C}} \in \mathbb{R}^{I_1 \times \dots \times I_N \times J_1 \times \dots \times J_M}$  with entries  $c_{i_1, \dots, i_N, j_1, \dots, j_M} = a_{i_1, \dots, i_N} b_{j_1, \dots, j_M}$ . Specifically, the outer product of two nonzero vectors  $\mathbf{a} \in \mathbb{R}^I$ ,  $\mathbf{b} \in \mathbb{R}^J$  produces a rank-1 matrix  $\mathbf{X} = \mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^T \in \mathbb{R}^{I \times J}$  and the outer product of three nonzero vectors:  $\mathbf{a} \in \mathbb{R}^I$ ,  $\mathbf{b} \in \mathbb{R}^J$  and  $\mathbf{c} \in \mathbb{R}^K$  produces a 3rd order rank-1 tensor:  $\underline{\mathbf{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} \in \mathbb{R}^{I \times J \times K}$ , whose entries are  $x_{ijk} = a_i b_j c_k$ . A tensor  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  is said to be rank-1 if it can be expressed exactly by  $\underline{\mathbf{X}} = \mathbf{b}_1 \circ \mathbf{b}_2 \circ \dots \circ \mathbf{b}_N$  with entries  $x_{i_1, i_2, \dots, i_N} = b_{i_1} b_{i_2} \cdots b_{i_N}$ , where  $\mathbf{b}_n \in \mathbb{R}^{I_n}$  are nonzero vectors.

The symbol  $\otimes$  denotes the Kronecker product, i.e.,  $\mathbf{A} \otimes \mathbf{B} = [a_{ij} \mathbf{B}]$ , and the symbol  $\odot$  denotes the Khatri-Rao product or column-wise Kronecker product, i.e.,  $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_J \otimes \mathbf{b}_J]$ .

Subtensors are formed when a subset of indices is fixed. Particularly, a fiber is defined as a vector by fixing every index but one, while a tensor slice is a two-dimensional section (matrix) of a tensor, obtained by fixing all indices but two. Unfolding (matricization, flattening) of a tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  in the  $n$ -mode is denoted as  $\underline{\mathbf{Y}}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ , which consists of arranging all possible  $n$ -mode fibers as the columns of a matrix [5]. For simplicity, we define  $\check{I}_k = \Pi_{n \neq k} I_n$ ,  $\check{J}_k = \Pi_{n \neq k} J_n$ .  $\bigotimes_{n \neq k} \mathbf{A}^{(n)} = \mathbf{A}^{(N)} \otimes \dots \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \dots \otimes \mathbf{A}^{(1)}$  and  $\bigodot_{n \neq k} \mathbf{A}^{(n)} = \mathbf{A}^{(N)} \odot \dots \mathbf{A}^{(k+1)} \odot \mathbf{A}^{(k-1)} \dots \odot \mathbf{A}^{(1)}$ . An *n-mode fiber* is obtained by fixing all indices except the index for one dimension. For a third-order tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times K}$  its  $n$ -mode fibers are called *column fibers* ( $n = 1$ ), *row fibers* ( $n = 2$ ) and *tube fibers* ( $n = 3$ ).<sup>1</sup> In other words, the unfolding operation consists of arranging the fibers of an  $N$ -way array into a matrix in a specific order, e.g., in reverse lexicographical order [5]. Given an  $N$ th-order tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  we define the *n-mode unfolding matrix* by  $\underline{\mathbf{Y}}_{(n)} \in \mathbb{R}^{I_n \times \prod_{m \neq n} I_m}$ , which is obtained by grouping all indices for dimensions  $m \neq n$ . For example, given a third-order tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times K}$  the corresponding 1-mode unfolding matrix is obtained by creating a new index, corresponding to the grouping of indices  $t$  and  $k$ , which is denoted by  $\underline{\mathbf{Y}}_{(1)}(i, (tk)) = y_{itk}$ .

We use calligraphic style letters to denote a subset of indices for a given dimension, for example, for 3D tensors with indices  $i = 1, 2, \dots, I$ ,  $t = 1, 2, \dots, T$  and  $k = 1, 2, \dots, K$ , we define the subsets containing  $P_1 \leq I$ ,  $P_2 \leq T$ ,  $P_3 \leq K$  indices as:  $\mathcal{I} = [i_1, i_2, \dots, i_{P_1}]$ ,  $\mathcal{T} = [t_1, t_2, \dots, t_{P_2}]$ , and  $\mathcal{K} = [k_1, k_2, \dots, k_{P_3}]$ . Then we define the *subtensor*  $\underline{\mathbf{W}} \in \mathbb{R}^{P_1 \times P_2 \times P_3}$  as the one determined by indices  $\mathcal{I}$ ,  $\mathcal{T}$  and  $\mathcal{K}$ , i.e.,  $\underline{\mathbf{W}} = \underline{\mathbf{Y}}(\mathcal{I}, \mathcal{T}, \mathcal{K})$  and its unfolding matrices will be denoted by  $\mathbf{W}_{(1)}$ ,  $\mathbf{W}_{(2)}$  and  $\mathbf{W}_{(3)}$ . We may also extract matrices containing the tensor fibers determined by these subsets of indices as follows:  $\mathbf{C}^{(1)} = \underline{\mathbf{Y}}_{(1)}(:, \mathcal{T} \times \mathcal{K}) \in \mathbb{R}^{I \times P_2 P_3}$ ,  $\mathbf{C}^{(2)} = \underline{\mathbf{Y}}_{(2)}(:, \mathcal{I} \times \mathcal{K}) \in \mathbb{R}^{T \times P_1 P_3}$  and  $\mathbf{C}^{(3)} = \underline{\mathbf{Y}}_{(3)}(:, \mathcal{I} \times \mathcal{J}) \in \mathbb{R}^{K \times P_1 P_2}$  for the column, row, and tube fibers respectively, where “ $:$ ” denotes all the indices in a dimension and  $\mathcal{J} \times \mathcal{K}$  denotes all indices produced as a combination of an index in  $\mathcal{J}$  and an index in  $\mathcal{K}$  [120]. Readers can refer to [5,8] for more details regarding the notations and tensor operations.

### 1.21.5.2 Basic tensor decompositions models

There are two basic models for tensor decompositions: Tucker and CP (CANDECOMP/PARAFAC) decomposition. The Tucker- $N$  decomposition or rank- $J_1, J_2, \dots, J_N$  Tucker model, the data is decomposed as the product of a core tensor with  $N$  mode factor (component) matrices [6], i.e., a given data

<sup>1</sup>For convenience, for third-order tensors we use simplified indices  $I = I_1$ ,  $T = I_2$  and  $K = I_3$ .

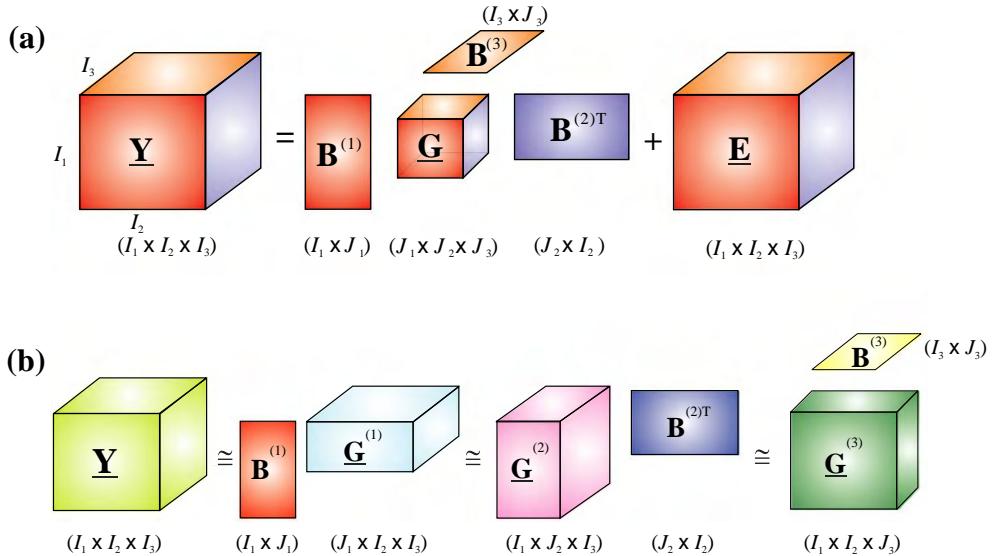


FIGURE 21.8

Illustration of a third-order tensor decomposition (a) using a constrained Tucker-3 model:  $\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \times_3 \mathbf{B}^{(3)} + \underline{\mathbf{E}} = [\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}] + \underline{\mathbf{E}} = \sum_{j_1, j_2, j_3} g_{j_1 j_2 j_3} (\mathbf{b}_{j_1}^{(1)} \circ \mathbf{b}_{j_2}^{(2)} \circ \mathbf{b}_{j_3}^{(3)}) + \underline{\mathbf{E}}$ . The objective is to estimate the components  $\mathbf{b}_{j_n}^{(n)}$ , i.e., the columns of the component matrices  $\mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \dots, \mathbf{b}_{J_n}^{(n)}] \in \mathbb{R}^{I_n \times J_n}$ , with desired diversities or statistical properties and a possibly sparse core tensor  $\underline{\mathbf{G}} \in \mathbb{R}^{I_1 \times J_2 \times J_3}$ , typically with  $J_n \ll I_n$ , ( $n = 1, 2, 3$ ). Note that some selected component matrices from the Tucker-3 model can be absorbed in the core tensor  $\underline{\mathbf{G}}$ , which leads to Tucker-1 models (b):  $\underline{\mathbf{Y}} = \underline{\mathbf{G}}^{(n)} \times_n \mathbf{B}^{(n)} + \underline{\mathbf{E}}_n$ . Instead of applying the standard Alternating Least Squares (ALS) algorithms to the Tucker-3 model, we can apply the unfolding of the data tensor according to Tucker-1 models and then perform constrained matrix factorizations for the unfolded matrices (multilinear BSS) by imposing desired constraints (nonnegativity, sparseness, independence, smoothness, or decorrelation, etc.). More generally, we wish to model complex interactions embedded in multidimensional data and/or predict missing entries.

tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ , is decomposed as (see Figure 21.8a)

$$\begin{aligned} \underline{\mathbf{Y}} &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} + \underline{\mathbf{E}} = \widehat{\mathbf{Y}} + \underline{\mathbf{E}} \\ &= \sum_{j_1=1}^{J_1} \cdots \sum_{j_N=1}^{J_N} g_{j_1 j_2 \cdots j_N} (\mathbf{b}_{j_1}^{(1)} \circ \mathbf{b}_{j_2}^{(2)} \circ \cdots \circ \mathbf{b}_{j_N}^{(N)}) + \underline{\mathbf{E}}, \end{aligned} \quad (21.224)$$

where  $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_N}$  is the core tensor,  $\mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \dots, \mathbf{b}_{J_n}^{(n)}] \in \mathbb{R}^{I_n \times J_n}$  is the mode- $n$  component matrix for  $n = 1, 2, \dots, N$ ,  $\circ$  denotes the outer product, and the tensor  $\underline{\mathbf{E}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  represents errors or noise. For simplicity, we also use a shorter notation for (21.224) as  $\widehat{\mathbf{Y}} = [\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)}]$ .

The Tucker-N model with orthogonal factor matrices  $\mathbf{B}^{(n)}$  is also known Multilinear SVD or Higher-order SVD (HOSVD) [9]. In fact, the original Tucker model made the assumptions of orthogonality of the factor matrices (analogous to SVD) and orthogonality of the core tensor [8, 138, 158, 159]. However, in many practical applications we ignore or relax such constraints to make model more flexible. For example, by imposing nonnegativity constraints the problem of estimating the component matrices and the core tensor is converted into a generalized NMF problem called the Nonnegative Tucker Decomposition (NTD). Several implementations of Tucker decomposition algorithms with nonnegativity constraints together with a number of other constraints are given in [8, 138, 158, 160]. The NTD imposes nonnegativity constraints for all component matrices and the core tensor, while a semi-NTD (analogous to semi-NMF) imposes nonnegativity constraints to only some component matrices and/or some elements of the core tensor [8].

Note that the Tucker- $N$  model (21.224) can be represented equivalently by a set of  $N$  different matrix factorizations with three factors:

$$\mathbf{Y}_{(n)} \approx \mathbf{B}^{(n)} \mathbf{G}_{(n)} \mathbf{Z}^{(n)} \quad (n = 1, 2, \dots, N), \quad (21.225)$$

where the matrix  $\mathbf{Z}^{(n)} = [\mathbf{B}^{(N)} \otimes \dots \otimes \mathbf{B}^{(n+1)} \otimes \mathbf{B}^{(n-1)} \dots \otimes \mathbf{B}^{(1)}]^T$  was the Kronecker structure. In many applications, we usually need to estimate only a pre-selected number of component matrices. Without loss of generality, let us assume for simplicity that we are interested in extractions only the first  $K$  component matrices, with  $K \leq N$ . In that case, we can use a simplified Tucker- $K$  model described as

$$\underline{\mathbf{Y}} = \check{\underline{\mathbf{G}}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \dots \times_K \mathbf{B}^{(K)} + \underline{\mathbf{E}}, \quad (21.226)$$

where the core tensor  $\check{\underline{\mathbf{G}}} \in \mathbb{R}^{J_1 \times \dots \times J_K \times I_{K+1} \dots \times I_N}$  is expressed as

$$\check{\underline{\mathbf{G}}} = \underline{\mathbf{G}} \times_{K+1} \mathbf{B}^{(K+1)} \times_{K+2} \mathbf{B}^{(K+2)} \dots \times_N \mathbf{B}^{(N)}. \quad (21.227)$$

Furthermore, the Tucker decomposition (21.224) can be represented by using simple Tucker-1 decompositions, with only one component matrix  $\mathbf{B}^{(n)}$  in each mode (see Figure 21.8b):

$$\underline{\mathbf{Y}} \approx \underline{\mathbf{G}}^{(\bar{n})} \times_n \mathbf{B}^{(n)} \quad \text{or in matrix form } \mathbf{Y}_{(n)} \approx \mathbf{B}^{(n)} \mathbf{G}_{(n)}^{(\bar{n})}, \quad (21.228)$$

where  $\underline{\mathbf{G}}^{(\bar{n})} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \dots \times_{n-1} \mathbf{B}^{(n-1)} \times_{n+1} \mathbf{B}^{(n+1)} \dots \times_N \mathbf{B}^{(N)}$  is a tensor,  $(n = 1, 2, \dots, N)$ . This means that Tucker decomposition problems can be converted to a set of matrix factorization problems, with suitable constraints imposed on the component matrices in order to achieve essential uniqueness.

In the special case, when the core tensor  $\underline{\mathbf{G}}$  is a hypercube, i.e.,  $J_1 = J_2 = \dots = J_N = J$ , and it has nonzero entries only on its diagonal, the Tucker model is reduced to the CP model (CANDECOMP/PARAFAC) decomposition, analyzed by two independent research groups [2, 4, 161]). In the CP model a data tensor is represented as a linear combination of outer products of columns (components) of the component matrices:

$$\begin{aligned} \underline{\mathbf{Y}} &= \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \dots \times_N \mathbf{B}^{(N)} + \underline{\mathbf{E}} \\ &= \sum_{j=1}^J \lambda_j \mathbf{b}_j^{(1)} \circ \mathbf{b}_j^{(2)} \dots \circ \mathbf{b}_j^{(N)} + \underline{\mathbf{E}}, \end{aligned} \quad (21.229)$$

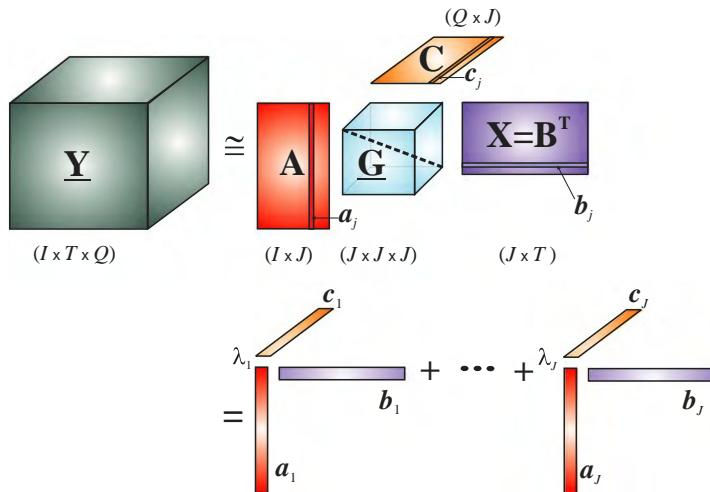
**FIGURE 21.9**

Illustration of a 3-way CP model performing tensor decomposition  $\underline{Y} \cong \underline{G} \times_1 \underline{A} \times_2 \underline{B} \times_3 \underline{C} = \sum_{j=1}^J \lambda_j (\underline{a}_j \circ \underline{b}_j \circ \underline{c}_j)$  with a diagonal core tensor  $\underline{G} = \underline{\Lambda} \in \mathbb{R}^{J \times J \times J}$ , with entries  $\lambda_{jjj} = \lambda_j$ .

where  $\underline{G} = \underline{\Lambda} \in \mathbb{R}^{J \times J \times \dots \times J}$  is a diagonal tensor with all entries zero except the diagonal entries  $\lambda_j$ . We usually assume that the vectors  $\underline{b}_j^{(n)}$  (for  $n = 1, 2, \dots, N$ ) are normalized to unit length. In other words, the CP model represents multiway data as a sum of rank-1 components [5, 162].

The CP model decomposes a given tensor into a sum of multilinear terms, in a way analogous to the bilinear matrix decomposition (see Figure 21.9). Unlike standard SVD, usually CP does not impose any constraints. Since model is unique under mild conditions. A model which imposes nonnegativity on factor matrices is called a Nonnegative Tensor Factorization (NTF) or Nonnegative PARAFAC. A nonnegative version of CP was first introduced by Carroll et al. [163]. Later, more efficient approaches were developed [164, 165], based on the modified NLS (Nonnegative Least Squares) by Paatero [166, 167] who extended his earlier 2-way positive matrix factorization (PMF) method to the three-way CP model, referring to the result as PMF3 (three-way positive matrix factorization). Although such constrained nonnegativity based model may not match perfectly the input data (i.e., it may have larger residual errors  $\mathbf{E}$  than the standard CP without any constraints), such decompositions are often more meaningful and have clearer physical interpretations [8, 34].

The advantage of the Tucker model over the CP model is that the Tucker model is more general since the number of components in each mode can be different, and the components are linked via a core tensor, and hence it allows us to model more complex hidden data structures. Both the CP and the Tucker decompositions attempt to obtain optimal low-rank ( $J$  or  $\{J_1, J_2, \dots, J_N\}$ , respectively) approximations to the original data. However, quite different from the matrix case, the optimal low-rank approximation may not exist at all or, if it exists, it may not be unique for high-order tensors [168, 169]. Particularly, an unconstrained Tucker decomposition is in general non-unique, as it has arbitrary rotational ambiguity.

Due to many degrees of freedom, the results of such decompositions are difficult to interpret because the components do not have a clear physical meaning. The CP model in contrast is essentially unique if the fundamental Kruskal uniqueness conditions are satisfied, which was addressed by Kruskal for third-order tensors and then extended to  $N$ th-order tensors by Sidiropoulos et al. [170, 171]. These are sufficient (but not necessary) conditions. Basically, algebraic properties, e.g., the ranks of the tensors and their component matrices, play a central role for such uniqueness analyze.

If we impose additional constraints to the CP model, such as nonnegativity, sparsity, or orthogonality, the conditions for uniqueness can be relaxed. Moreover, by imposing the nonnegativity or orthogonality constraints the CP model has an optimal solution, i.e., there is no risk for degenerate solutions [172]. Imposing nonnegativity constraints makes degenerate solutions impossible, since no factor can counteract the effect of another factor, and usually it improves the convergence of the learning algorithms since the search space is greatly reduced. For the Tucker and CP decompositions there exist many algorithms [5, 8]. Most of them are based on the ALS (Alternating Least Squares), HALS (Hierarchical ALS) [8, 34, 173, 174] and CUR tensor decompositions [120]. However, the ALS algorithms are often slow for noisy or ill-conditioned data and they may get stuck in local minima due to the non-convexity of the optimization problem, especially when nonnegativity or other constraints are imposed on the component matrices.

In contrast to most of the existing tensor decomposition methods, where only the best fit of a decomposed tensor to the original data tensor is pursued, in this section, the multilinear Blind Source Separation (MBSS) approach is briefly described and we show how to perform constrained tensor decompositions in which both the satisfactory fit and a wide variety of desired properties/constraints and diversities of estimated components can be pursued simultaneously.

### 1.21.5.3 Sparse CP decomposition using the generalized power method

Although the CP tensor decomposition is in general unique without any constraints, in some applications, especially for large-scale problems, we need components  $\mathbf{b}_j^{(n)}$  that are sparse in one specific mode or in all modes. Let us consider now the basic CP model. In order to estimate the first set of components we can consider the following optimization problem (see Section 1.21.2.6.3)

$$\min_{\mathbf{b}_j^{(n)}} \left\{ \left\| \underline{\mathbf{Y}} - \sum_{j=1}^J \lambda_j (\mathbf{b}_j^{(1)} \circ \mathbf{b}_j^{(2)} \cdots \circ \mathbf{b}_j^{(N)}) \right\|_F^2 \right\}.$$

Assuming that we want to extract the desired components sequentially one by one, we can first consider the constrained optimization problem

$$\begin{aligned} & \min_{\mathbf{b}_1^{(n)}} \{ \| \underline{\mathbf{Y}} - \lambda_1 (\mathbf{b}_1^{(1)} \circ \mathbf{b}_1^{(2)} \cdots \circ \mathbf{b}_1^{(N)}) \|_F^2 \}, \\ & \text{s.t. } \|\mathbf{b}_1^{(n)}\|_2 \leq 1, \|\mathbf{b}_1^{(n)}\|_1 \leq 1, P_1(\mathbf{b}_1^{(n)}) \forall C_1 \leq c_1, P_2(\mathbf{b}_1^{(n)}) \leq c_2 \quad \forall n. \end{aligned}$$

**Algorithm 5.** Power Method for Sparse CP Decomposition

**Require:** Data tensor  $\underline{\mathbf{Y}}^{(1)} = \underline{\mathbf{Y}}$ , sparsity coefficients  $\alpha_n \geq 0$ , ( $n = 1, \dots, N$ ) and initial vectors  $\mathbf{b}_j^{(n)}$ .

1: For  $j = 1, 2, \dots, J$  and  $n = 1, 2, \dots, N$ :

(a) Repeat until convergence of  $\mathbf{b}_j^{(n)}$ :

$$2: \quad \mathbf{b}_j^{(\bar{n})} = \underline{\mathbf{Y}}^{(j)} \tilde{\times}_1 \mathbf{b}_1^{(1)} \dots \tilde{\times}_{n-1} \mathbf{b}_1^{(n-1)} \tilde{\times}_{n+1} \mathbf{b}_1^{(n+1)} \dots \tilde{\times}_N \mathbf{b}_1^{(N)};$$

$$3: \quad \mathbf{b}_j^{(n)} = \frac{\mathcal{S}_1(\mathbf{b}_j^{(\bar{n})}, \alpha_n)}{\|\mathcal{S}_1(\mathbf{b}_j^{(\bar{n})}, \alpha_n)\|_2} \quad (\text{for } \alpha_n = 0 \text{ take } \mathbf{b}_j^{(n)} = \frac{\mathbf{b}_j^{(\bar{n})}}{\|\mathbf{b}_j^{(\bar{n})}\|_2});$$

$$4: (b) \quad \lambda_j = |\underline{\mathbf{Y}}^{(j)} \tilde{\times}_1 \mathbf{b}_j^{(1)} \tilde{\times}_2 \mathbf{b}_j^{(2)} \dots \tilde{\times}_N \mathbf{b}_j^{(N)}|;$$

$$5: (c) \quad \underline{\mathbf{Y}}^{(j+1)} = \underline{\mathbf{Y}}^{(j)} - \lambda_j (\mathbf{b}_j^{(1)} \circ \mathbf{b}_j^{(2)} \dots \circ \mathbf{b}_j^{(N)});$$

$$6: \text{return } \mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \dots, \mathbf{b}_J^{(n)}], \quad \Lambda^* = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_J\}.$$

The problem is equivalent to the following optimization problem

$$\max_{\mathbf{b}_1^{(n)}} \{ \underline{\mathbf{Y}} \tilde{\times}_1 \mathbf{b}_1^{(1)} \tilde{\times}_2 \mathbf{b}_1^{(2)} \dots \tilde{\times}_N \mathbf{b}_1^{(N)} \}, \quad (21.230)$$

$$\text{s.t. } \|\mathbf{b}_1^{(n)}\|_2 \leq 1, \quad \|\mathbf{b}_1^{(n)}\|_1 \leq 1, \quad P_1(\mathbf{b}_1^{(n)}) \leq c_1, \quad P_2(\mathbf{b}_1^{(n)}) \leq c_2 \quad \forall n.$$

If we impose sparsity constraints, we can formulate the following optimization problem

$$\max_{\mathbf{b}_1^{(n)}} \{ \underline{\mathbf{Y}} \tilde{\times}_1 \mathbf{b}_1^{(1)} \tilde{\times}_2 \mathbf{b}_1^{(2)} \dots \tilde{\times}_N \mathbf{b}_1^{(N)} - \sum_n \alpha_n \|\mathbf{b}_1^{(n)}\|_1 \}, \quad \text{s.t. } \|\mathbf{b}_1^{(n)}\|_2 \leq 1 \quad \forall n.$$

This optimization leads to the following simple update rules

$$\mathbf{b}_1^{(n)} = \frac{\mathcal{S}_1(\mathbf{b}_1^{(\bar{n})}, \alpha_n)}{\|\mathcal{S}_1(\mathbf{b}_1^{(\bar{n})}, \alpha_n)\|_2}, \quad (21.231)$$

where

$$\mathcal{S}_1(\mathbf{b}, \alpha) = \text{sign}(\mathbf{b})[|\mathbf{b}| - \alpha]_+, \quad (21.232)$$

and

$$\mathbf{b}_1^{(\bar{n})} = \underline{\mathbf{Y}} \tilde{\times}_1 \mathbf{b}_1^{(1)} \dots \tilde{\times}_{n-1} \mathbf{b}_1^{(n-1)} \tilde{\times}_{n+1} \mathbf{b}_1^{(n+1)} \dots \tilde{\times}_N \mathbf{b}_1^{(N)}. \quad (21.233)$$

This update formula simplifies in the absence of sparsity constraints (i.e.,  $\alpha_n = 0 \forall n$ ) as

$$\mathbf{b}_1^{(n)} = \frac{\mathbf{b}_1^{(\bar{n})}}{\|\mathbf{b}_1^{(\bar{n})}\|_2}. \quad (21.234)$$

In order to extract the next set of components, we need to apply deflation approach for  $j = 1, 2, \dots, J$

$$\lambda_j = |\underline{\mathbf{Y}}^{(j)} \tilde{\times}_1 \mathbf{b}_j^{(1)} \tilde{\times}_2 \mathbf{b}_j^{(2)} \dots \tilde{\times}_N \mathbf{b}_j^{(N)}|, \quad (21.235)$$

$$\underline{\mathbf{Y}}^{(j+1)} = \underline{\mathbf{Y}}^{(j)} - \lambda_j (\mathbf{b}_j^{(1)} \circ \mathbf{b}_j^{(2)} \dots \circ \mathbf{b}_j^{(N)}), \quad (21.236)$$

with  $\underline{\mathbf{Y}}^{(1)} = \underline{\mathbf{Y}}$  (see Algorithm 5).

### 1.21.5.4 Multilinear BSS via constrained unique Tucker decompositions

There are two possible approaches to represent constrained Tucker decompositions for the MBSS. In the first approach the columns of the component matrices  $\mathbf{B}^{(n)}$  represent the desired components or latent variables, and the core tensor  $\mathbf{G}$  represents some kind of a "mixing process". More precisely, the core tensor shows the links among the components from different modes, while the data tensor  $\underline{\mathbf{Y}}$  represents a collection of 1-D or 2-D mixing signals. In the second approach the core tensor represents the desired (but unknown hidden)  $N$ -dimensional signal (e.g., 3D MRI image or 4D video), and the component matrices represent specific transformations, e.g., time frequency transformations or wavelets dictionaries (mixing or filtering processes). In this case the data tensor  $\underline{\mathbf{Y}}$  represents observed  $N$ -dimensional signal, which may be distorted, transformed, compressed, or mixed, depending on the specific applications. In this section, we will only consider the first interpretation or approach.

We can implement multilinear BSS algorithms in several ways. First of all, to estimate desired components, we can minimize a global cost function with suitable penalty and regularization terms (see Eq. (21.224)):

$$D_F(\underline{\mathbf{Y}} \parallel \widehat{\underline{\mathbf{Y}}}) = \|\underline{\mathbf{Y}} - \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}\|_F^2 + \sum_n \alpha_n D_n(\mathbf{B}^{(n)}), \quad (21.237)$$

where  $\|\underline{\mathbf{Y}}\|_F = (\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} y_{i_1 \cdots i_N}^2)^{\frac{1}{2}}$ ;  $\alpha_n \geq 0$  are penalty coefficients and  $D_n(\mathbf{B}^{(n)})$  are penalty terms, which are added to achieve desired characteristics of the components. For example, if we need to impose mutual independence constraints the penalty terms can take the following form:  $D_n(\mathbf{B}^{(n)}) = \sum_{j=1}^J \sum_{p \neq j} \mathbf{b}_j^{(n)T} f_n(\mathbf{b}_n^{(p)})$ , where  $f_n(u)$  are suitable nonlinear functions.

In principle, this way leads to penalized ALS algorithms which allow us to find the component matrices  $\mathbf{B}^{(n)}$  and the associated core tensor. However, the method involves heavy computations and it is very time consuming.

A simpler, practical, and much more efficient approach is to exploit unfolding in each mode of the data tensor  $\underline{\mathbf{Y}}$ , according to the Tucker- $N$  or Tucker-1 decompositions (21.228), to allow us to directly apply suitable constrained low-rank matrix factorizations algorithms (PCA/SVD, ICA, NMF, SCA).

Note that in practice, for large-scale problems, we do not need to perform explicitly the unfolding of the full data tensor. Instead, we may apply fast sampling of tubes (columns) of data tensors [120].

For the Tucker decomposition, let  $\underline{\mathbf{Y}} \approx \underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)}$  and assume that only the factor matrices (whose columns represent the components)  $\mathbf{B}^{(n)}$ , ( $n = 1, 2, \dots, N$ ), are subject of interest. Then we have

$$\mathbf{Y}_n = \mathbf{Y}_{(n)}^T \cong \mathbf{A}_{\bar{n}} [\mathbf{B}^{(n)}]^T \quad (n = 1, 2, \dots, N), \quad (21.238)$$

where the basis matrices have a Kronecker structure:

$$\begin{aligned} \mathbf{A}_{\bar{n}} &= (\mathbf{B}^{(N)} \otimes \cdots \otimes \mathbf{B}^{(n+1)} \otimes \mathbf{B}^{(n-1)} \otimes \cdots \otimes \mathbf{B}^{(1)}) \mathbf{G}_{(n)}^T \\ &= \left( \bigotimes_{p=1, p \neq n}^N \mathbf{B}^{(p)} \right) \mathbf{G}_{(n)}^T. \end{aligned} \quad (21.239)$$

For the CP model, the basis (mixing) matrices  $\mathbf{A}_{\bar{n}}$  take much simpler forms (with Khatri-Rao structure):

$$\mathbf{A}_{\bar{n}} = (\mathbf{B}^{(N)} \odot \cdots \odot \mathbf{B}^{(n+1)} \odot \mathbf{B}^{(n-1)} \cdots \odot \mathbf{B}^{(1)}) \mathbf{\Lambda} = \left( \bigodot_{p=1, p \neq n}^N \mathbf{B}^{(p)} \right) \mathbf{\Lambda}, \quad (21.240)$$

where  $\mathbf{\Lambda}$  is a diagonal matrix whose diagonal elements consist of the diagonal entries  $\lambda_j$  of the core tensor  $\underline{\mathbf{G}} = \underline{\mathbf{\Lambda}}$  (see Figure 21.9).

Note that the above models correspond to group BSS/ICA models (21.7) with the following correspondences:  $\mathbf{Y}_n = \mathbf{Y}_{(n)}^T$ ,  $\mathbf{A}_n = \mathbf{A}_{\bar{n}}$ , and  $\mathbf{B}_n = \mathbf{B}^{(n)}$ , for all  $n$ , if we impose desired constraints on the component matrices  $\mathbf{B}^{(n)}$ . From (21.238), we can observe that the columns of the mode- $n$  matricization of  $\underline{\mathbf{Y}}$  are just the linear mixtures of the mode- $n$  component matrices  $\mathbf{B}^{(n)}$ , ( $n = 1, 2, \dots, N$ ). This suggests that we can use various available BSS algorithms to directly extract factor matrices with specific properties and diversities. For example, if apply SVD in each mode, we obtain HOSVD, and if apply ICA algorithms we have multilinear ICA [32, 175].

Once all desired component matrices  $\mathbf{B}^{(n)}$ ,  $n = 1, 2, \dots, N$  have been estimated, the core tensor can be computed using any suitable matrix factorization method which provides an essentially unique decomposition method. In the simplest scenario we can apply the following formula:

$$\widehat{\underline{\mathbf{G}}} = \underline{\mathbf{Y}} \times_1 \mathbf{B}^{(1)\dagger} \times_2 \mathbf{B}^{(2)\dagger} \cdots \times_N \mathbf{B}^{(N)\dagger}, \quad (21.241)$$

where  $\mathbf{B}^\dagger$  denotes the Moore–Penrose inverse of a matrix  $\mathbf{B}$ .

In general,  $\widehat{\underline{\mathbf{G}}} \neq \underline{\mathbf{G}}$  due to the unavoidable ambiguities of the BSS methods. However, the links between the extracted components remain unchanged, except for the ambiguities due to permutation and scaling. The approach described above is called multilinear BSS (MBSS), where multiple sets of components  $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)}$  can be extracted separately from multiple modes of a tensor.

The MBSS approach has two major advantages [32]:

1. The extracted component matrices  $\mathbf{B}^{(n)}$  are essentially unique (neglecting scaling and permutation ambiguities) and have specific desired properties or feature information, under conditions that apply a suitable BSS/PMD algorithm which provide a unique decomposition in each mode.
2. The component matrices of interest can be extracted directly, i.e., independently of the knowledge of other component matrices, due to application of a standard BSS for each mode separately or independently. In other words, using the MBSS, tensor decompositions can be achieved without ALS iterations. Note that in the ALS-based algorithms tensors have to be unfolded to each mode frequently, which is the main bottleneck to achieve high efficiency, especially for large-scale data. In contrast, the MBSS approach allows us to dramatically reduce the computational complexity, since each component matrix can be estimated independently on other modes by employing efficient BSS algorithms, particularly in the case where only a few preselected component matrices are of interest [32].

It should be noted that the unfolding matrices  $\mathbf{Y}_{(n)}^T \in \mathbb{R}^{\check{I}_n \times I_n}$  and the bases matrices  $\tilde{\mathbf{A}}_n \in \mathbb{R}^{\check{I}_n \times J_n}$  are usually very large-scale (very tall) matrices with  $\check{I}_n \gg I_n$  and  $I_n \geq J_n$ . Thus the MBSS

problems (21.238) are usually highly over-determined constrained matrix factorization problems and we need to perform dimensionality reduction first. Moreover, we need to estimate the number of components in each mode  $J_n$ , which in general are unknown. We can solve both problems by applying low-rank approximations, especially PCA/SVD decompositions or CUR decompositions.

### 1.21.5.5 CUR tensor decomposition for dimensionality reduction and compression of large-scale data

Although the PCA/SVD method can be relatively fast, optimal in the least squares sense, and quite robust with respect to Gaussian noise, unfortunately, it does not preserve the properties of original raw tensor data and it is rather time consuming for very large-scale problems. For example, the data tensor can be nonnegative while the reduced unfolding matrices may have in general positive and negative entries. In many applications, for example in NTF and NTD decompositions, we need to preserve in the reduced unfolded matrices the basic characteristic of the original data tensor. To overcome the above problem, we adopted the CUR algorithm for tensor compression and dimensionality reduction [120].

The concept of CUR decomposition has been successfully generalized to tensors in [120, 124]. A practical and fast technique is called Fiber Sampling Tucker Decomposition (FSTD) [120]. Since real-life data have often good low multilinear rank approximations, FSTD provides such a low-rank Tucker decomposition that is directly expressed in terms of a relatively small number of fibers (vector columns) of the data tensor. By virtue of its construction from raw data elements, FSTD preserves the original characteristics and features of tensor data and can be used for efficient compressed representations of the original data tensor.

For a given tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  for which an exact rank- $(R_1, R_2, \dots, R_N)$  Tucker representation exists, FSTD selects  $R$  indices in each mode, which determine an intersection sub-tensor  $\underline{\mathbf{W}} \in \mathbb{R}^{R \times R \times \dots \times R}$  so that the following exact Tucker representation is obtained:

$$\underline{\mathbf{Y}} = \underline{\mathbf{U}}; \mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \dots, \mathbf{C}^{(N)}, \quad (21.242)$$

in which the core tensor is computed as  $\underline{\mathbf{U}} = \underline{\mathbf{G}} = \underline{\mathbf{W}}; \mathbf{W}_{(1)}^\dagger, \mathbf{W}_{(2)}^\dagger, \dots, \mathbf{W}_{(n)}^\dagger$ , and the matrices  $\mathbf{C}^{(n)} \in \mathbb{R}^{I_n \times I_{\bar{n}}}$  contain the mode- $n$  fibers defined by restricting the indices in modes  $m \neq n$  to belong to the selected subsets. This can also be written as a rank- $(R, R, \dots, R)$  Tucker representation:

$$\underline{\mathbf{Y}} = \underline{\mathbf{W}}; \mathbf{C}^{(1)} \mathbf{W}_{(1)}^\dagger, \mathbf{C}^{(2)} \mathbf{W}_{(2)}^\dagger, \dots, \mathbf{C}^{(N)} \mathbf{W}_{(N)}^\dagger. \quad (21.243)$$

Observe that for  $N = 2$  this model simplifies into the CUR matrix case,  $\mathbf{X} = \mathbf{CUR}$ , where  $\mathbf{C}^{(1)} = \mathbf{C}, \mathbf{C}^{(2)} = \mathbf{R}^T$  and the core matrix is  $\mathbf{U} = \mathbf{W}; \mathbf{W}_{(1)}^\dagger, \mathbf{W}_{(2)}^\dagger = \mathbf{W}^\dagger \mathbf{W} \mathbf{W}^\dagger = \mathbf{W}^\dagger$ .

An efficient strategy for the selection of suitable fibers, only requiring access to a partial (small) subset of entries of a data tensor through identifying the entries with maximum modulus within single fibers [120]. The indices are selected sequentially using a deflation approach making the FSTD algorithm suitable for very large scale tensors (including tensors with missing fibers or entries). There are several strategies for the selection of suitable columns and rows. The main idea is to select columns and rows that exhibit high “statistical leverage” and provide the best low-rank fit for the data matrix [120, 122, 123].

CUR decompositions are low-rank tensor decompositions that are explicitly expressed in terms of a relatively small number of actual fibers (vector columns) of the data tensor. Because they are constructed from actual raw data elements, CUR tensor decompositions preserve the ordinal characteristics and features of data and can be used as efficient compressed representations of the original data tensor.

Multilinear BSS and generalized component analysis algorithms, e.g., a combination of PCA, ICA, SCA, NMF, and MCA, are often considered as pure mathematical formulas, powerful, but rather mechanical procedures. There is a misconception that there is not very much left for the user to do after the machinery has been optimally implemented. However, the successful and efficient use of such tools strongly depends on *a priori* knowledge, common sense, and appropriate use of preprocessing and postprocessing tools. In other words, it is the preprocessing of data and postprocessing of models where expertise is truly needed in order to extract and identify physically significant and meaningful hidden components.

### 1.21.5.6 Common component analysis: feature extraction for multi-block tensor data

Dimensionality reduction, feature extraction and future selection are essential problems in the analysis of multi-dimensional data sets with large number of variables [12]. We will first illustrate the basic concepts of dimensionality reduction and feature extraction for a set of large-scale sample matrices. Let us consider that we have available a set of  $K$  matrices (2-D samples)  $\mathbf{Y}_k \in \mathbb{R}^{I_1 \times I_2}$ , ( $k = 1, 2, \dots, K$ ) that represent multi-subjects or multi-trials 2D data, which belong to  $C$  different classes or categories (e.g., multiple mental task/state data or different mental diseases). In order to perform dimensionality reduction and extract essential features for all the training samples, we apply simultaneous (approximative and constrained) matrix factorizations (see Figure 21.10a):

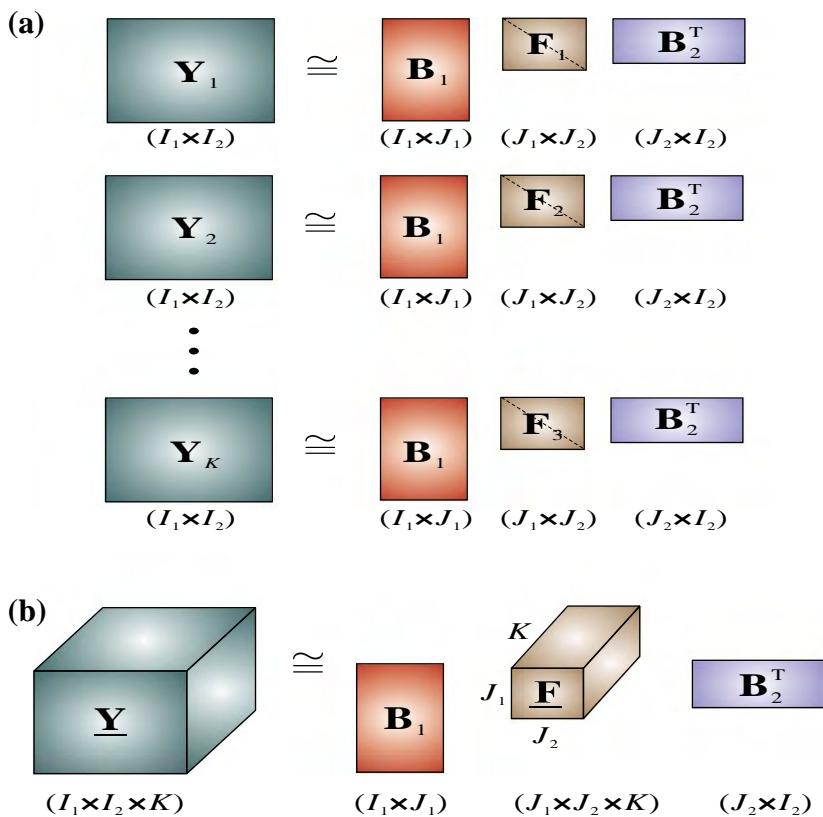
$$\mathbf{Y}_k = \mathbf{B}_1 \mathbf{F}_k \mathbf{B}_2^T + \mathbf{E}_k \quad (k = 1, 2, \dots, K), \quad (21.244)$$

where the two common factors (basis matrices)  $\mathbf{B}_1 \in \mathbb{R}^{I_1 \times J_1}$  and  $\mathbf{B}_2 \in \mathbb{R}^{I_2 \times J_2}$ ,  $J_n \leq I_n$ , code (explain) each sample  $\mathbf{Y}_k$  simultaneously along the horizontal and vertical dimensions and the extracted features are represented by matrices  $\mathbf{F}_k \in \mathbb{R}^{J_1 \times J_2}$ , typically, with  $J_1 \ll I_1$  and  $J_2 \ll I_2$ . In special cases  $\mathbf{F}_k$  are squared diagonal matrices. Then this problem can be considered as a generalization of Joint Approximative Diagonalization (JAD) [8,84].

The common method to solve such matrix factorizations problem is to minimize a set of cost functions  $\|\mathbf{Y}_k - \mathbf{B}_1 \mathbf{F}_k \mathbf{B}_2^T\|_F^2$ ,  $\forall k$ , sequentially or in parallel, with respect to all the factor matrices involved. Alternatively, we can solve the problem by concatenation or tensorization of all matrices  $\mathbf{Y}_k$  along the third dimension to form an  $I_1 \times I_2 \times K$  dimensional data tensor  $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times K}$  and perform the constrained Tucker decomposition (see Figure 21.10b).

In order to obtain meaningful components and a unique decomposition it is often convenient to impose additional constraints [8,12]. In the special case when the feature matrices  $\mathbf{F}_k$  are diagonal the Tucker-2 model is reduced to the special form of the CP model [8].

This approach can be naturally and quite straightforwardly extended to multi-dimensional data. Assume that we have available a multi-block of multi-dimensional tensors  $\underline{\mathbf{Y}}^{(k)} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$ , ( $k = 1, 2, \dots, K$ ), representing training data [12]. In such a case we can apply simultaneous

**FIGURE 21.10**

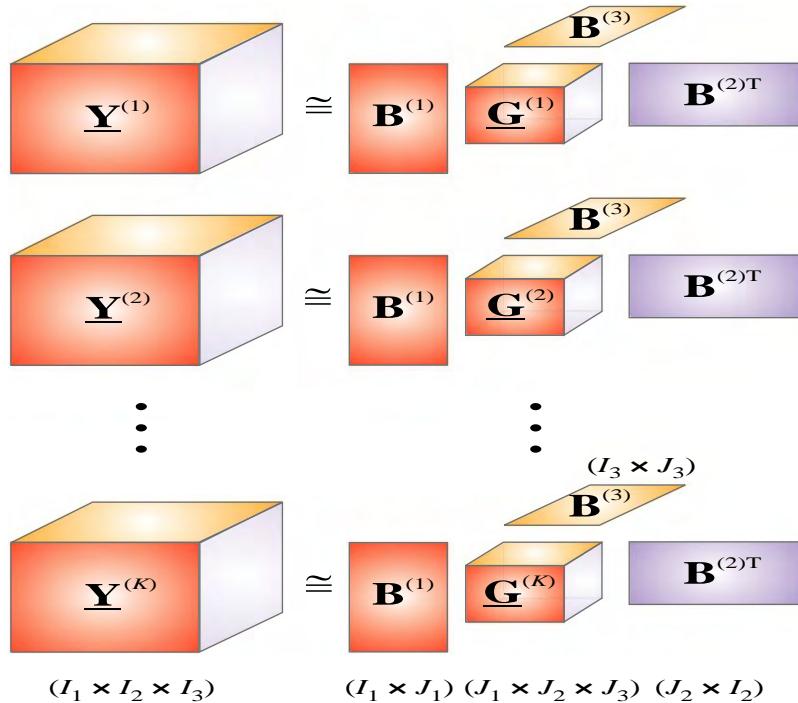
Common Component Analysis for a multi-block set of matrices. (a) Simultaneous matrix factorizations for dimensionality reduction and feature extraction. (b) This model is equivalent to a Tucker-2 decomposition of a third-order tensor as  $\underline{\mathbf{Y}} = \mathbf{F} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2$ , where  $\mathbf{F} \in \mathbb{R}^{J_1 \times J_2 \times K}$  is a constrained core tensor (representing features). In the special case matrices  $\mathbf{F}_k$  are diagonal metrices.

approximative tensor decompositions (see Figure 21.11)

$$\underline{\mathbf{Y}}^{(k)} = \underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} + \underline{\mathbf{E}}^{(k)}, \quad (21.245)$$

$(k = 1, 2, \dots, K)$ , where the compressed core tensors  $\underline{\mathbf{G}}^{(k)} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_N}$  representing features are of lower dimensions than the original data tensors  $\underline{\mathbf{Y}}^{(k)}$ , and we assume that the factors (basis matrices)  $\mathbf{B}^{(n)} = [\mathbf{b}_1^{(n)}, \mathbf{b}_2^{(n)}, \dots, \mathbf{b}_{J_n}^{(n)}] \in \mathbb{R}^{I_n \times J_n}$ ,  $(n = 1, 2, \dots, N)$  are common factors for all data tensors.

To compute the factor matrices  $\mathbf{B}^{(n)}$  and the core tensors  $\underline{\mathbf{G}}^{(k)}$ , we concatenate all training (sample) tensors into one  $N + 1$  order training data tensor

**FIGURE 21.11**

Multiway Common Component Analysis for a multi-block set of tensors for dimensionality reduction and feature extraction from a set of third-order data tensors  $\underline{\mathbf{Y}}^{(k)} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ ,  $(k = 1, 2, \dots, K)$ . The objective is to estimate the common factor matrices  $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times J_n}$  ( $n = 1, 2, 3$ ) and the core tensors  $\underline{\mathbf{G}}^{(k)} \in \mathbb{R}^{J_1 \times J_2 \dots \times J_N}$ , typically with  $J_n \ll I_n$ , representing features.

$\underline{\mathbf{Y}} = \text{cat}(\underline{\mathbf{Y}}^{(1)}, \underline{\mathbf{Y}}^{(2)}, \dots, \underline{\mathbf{Y}}^{(K)}, N + 1) \in \mathbb{R}^{I_1 \times I_2 \dots \times I_N \times I_{N+1}}$ , with  $N + 1 = K$  and perform the Tucker- $N$  decomposition [12]:

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \dots \times_N \mathbf{B}^{(N)} + \underline{\mathbf{E}}, \quad (21.246)$$

where the sample tensors  $\underline{\mathbf{Y}}^{(k)}$  can be extracted back from the concatenated tensor by fixing the  $(N + 1)$ th index at a value  $k$ , i.e.,  $\underline{\mathbf{Y}}_{\cdot, \cdot, \dots, \cdot, k}^{(1, 2, \dots, N, N+1)} = \underline{\mathbf{Y}}^{(k)}$ , and the individual features (corresponding to different classes) are extracted from the core tensor  $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \dots \times J_N \times I_{N+1}}$  as  $\underline{\mathbf{G}}^{(k)} = \underline{\mathbf{G}}_{\cdot, \cdot, \dots, \cdot, k}^{(1, 2, \dots, N, N+1)}$ , with  $I_{N+1} = K$ . In other words, the features of a specific training data  $\underline{\mathbf{Y}}^{(k)}$  are represented by the  $k$ th row of the mode- $(N + 1)$  matricized version of the core tensor  $\underline{\mathbf{G}}$ . We can apply the procedure described above to various feature extraction classification problems [12].

### 1.21.5.7 Linked multilinear BSS/ICA for multi-block data

In many applications, we need to perform a so called group analysis or multi-block data analysis which seeks to identify hidden components, for example, stimuli driven brain patterns that are common in two or more subjects in a group. In many scenarios links or relationships need to be considered to analyze variability and consistency of the components across multi-block data sets. Furthermore, some components do not need to be necessarily independent, they can instead be sparse, smooth or non-negative (e.g., for spectral components). Moreover, it is often necessary to impose some additional constraints in order to estimate some components, which are identical or maximally correlated (across multi-block data sets) with regard to their spatial distributions, spectral, or temporal patterns. In the simplest scenario a multi-block dataset consists set of matrices  $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_K$  with  $K \geq 2$ . In general, each matrix may have different numbers of rows or different number of columns. Our objective is to simultaneously perform matrix decompositions (in the case when the number of columns of the data matrices is the same):

$$\mathbf{Y}_k = \mathbf{A}_k \mathbf{B}_C^T + \tilde{\mathbf{A}}_k \mathbf{B}_k^T + \mathbf{E}_k, \quad (21.247)$$

consisting of three terms. The first term  $\mathbf{A}_k \mathbf{B}_C^T$  represents joint structure via the common matrix  $\mathbf{B}_C \in \mathbb{R}^{I_k \times R}$ , the second term  $\tilde{\mathbf{A}}_k \mathbf{B}_k^T$  represents individual structure and the last term  $\mathbf{E}_k$  represents error.

Alternatively, for a set of data matrices  $\mathbf{Y}_k$  with common number of rows, but possibly different number of columns, we can consider the dual model (see Figure 21.12):

$$\mathbf{Y}_k = \mathbf{A}_k \mathbf{B}_k^T + \mathbf{E}_k = \mathbf{A}_C \mathbf{B}_{C,k}^T + \tilde{\mathbf{A}}_k \mathbf{B}_{I,k}^T + \mathbf{E}_k, \quad (21.248)$$

where  $\mathbf{A}_k = [\mathbf{A}_C, \tilde{\mathbf{A}}_k]$ .

In the special case, when  $\mathbf{A}_k = \mathbf{A}_C \in \mathbb{R}^{I_1 \times J_1}$  for  $k = 1, 2, \dots, K$  the problem simplifies to Common Component Analysis or Population Value Decomposition (PVD) [176]. In more general case, in order to estimate desired common factor matrices  $\mathbf{A}_C \in \mathbb{R}^{I_1 \times R}$  with  $R < J_k$ , we can formulate the following constrained optimization problem, with orthogonality constraints [177]:

$$\min \sum_{k=1}^K \|\mathbf{Y}_k - \mathbf{A}_C \mathbf{B}_{C,k}^T\|_F^2 \quad (21.249)$$

$$\text{s.t. } \mathbf{A}_C^T \mathbf{A}_C = \mathbf{I}, \quad \tilde{\mathbf{A}}_k^T \tilde{\mathbf{A}}_k = \mathbf{I}, \quad \mathbf{A}_C^T \tilde{\mathbf{A}}_k = \mathbf{0}, \quad \mathbf{Y}_k = \mathbf{A}_k \mathbf{B}_k^T \quad \forall k.$$

These 2-way models can be naturally extended to linked multilinear BSS and in the special case of linked multilinear ICA (assuming that mutual statistical independence constraints are used in all modes).

In the linked multilinear BSS the problem is formulated as approximative decompositions of a set of data tensors  $\underline{\mathbf{Y}}^{(k)} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ , ( $k = 1, 2, \dots, K$ ) (see Figure 21.13):

$$\underline{\mathbf{Y}}^{(k)} = \underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1,k)} \times_2 \mathbf{B}^{(2,k)} \cdots \times_N \mathbf{B}^{(N,k)} + \underline{\mathbf{E}}^{(k)}, \quad (21.250)$$

where each factor (component) matrix  $\mathbf{B}^{(n,k)} = [\mathbf{B}_C^{(n)}, \mathbf{B}_I^{(n,k)}] \in \mathbb{R}^{I_n \times J_n}$  is composed of two bases:  $\mathbf{B}_C^{(n)} \in \mathbb{R}^{I_n \times R_n}$  (with  $0 \leq R_n \leq J_n$ ), which are common bases for all available data or more generally

$$\begin{aligned}
 \mathbf{Y}_1 &\approx \mathbf{A}_1 \mathbf{B}_c^T + \tilde{\mathbf{A}}_1 \mathbf{B}_1^T \\
 (I_1 \times T) & \quad (I_1 \times R) \quad (R \times T) \quad (I_1 \times J_1) \quad (J_1 \times T) \\
 \mathbf{Y}_2 &\approx \mathbf{A}_2 \mathbf{B}_c^T + \tilde{\mathbf{A}}_2 \mathbf{B}_2^T \\
 (I_2 \times T) & \quad (I_2 \times R) \quad (R \times T) \quad (I_2 \times J_2) \quad (J_2 \times T) \\
 \vdots & \quad \vdots \\
 \mathbf{Y}_K &\approx \mathbf{A}_K \mathbf{B}_c^T + \tilde{\mathbf{A}}_K \mathbf{B}_K^T \\
 (I_K \times T) & \quad (I_K \times R) \quad (R \times T) \quad (I_K \times J_K) \quad (J_K \times T)
 \end{aligned}$$

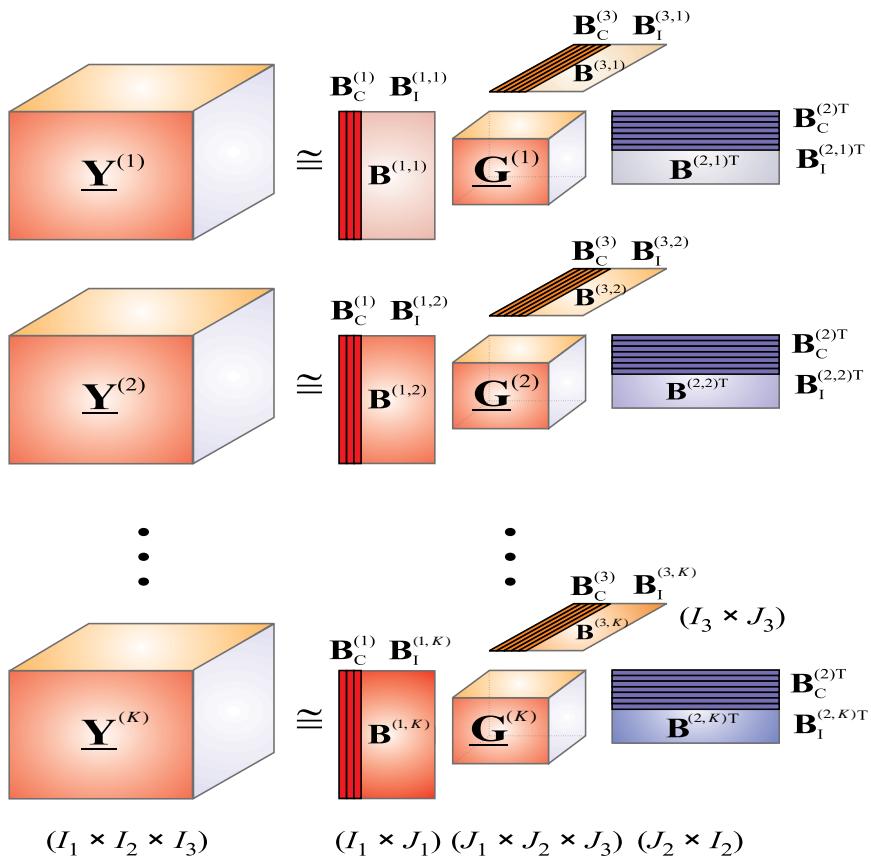
(a)

$$\begin{aligned}
 \mathbf{Y}_1 &\approx \underbrace{\mathbf{A}_C \widetilde{\mathbf{A}}_1}_{(I_1 \times J_1)} \mathbf{B}_1^T \\
 (I_1 \times T_1) & \quad (J_1 \times T_1) \\
 \mathbf{Y}_2 &\approx \underbrace{\mathbf{A}_C \widetilde{\mathbf{A}}_2}_{(I_1 \times J_2)} \mathbf{B}_2^T \\
 (I_1 \times T_2) & \quad (J_2 \times T_2) \\
 \vdots & \quad \vdots \\
 \mathbf{Y}_K &\approx \underbrace{\mathbf{A}_C \widetilde{\mathbf{A}}_K}_{(I_1 \times J_K)} \mathbf{B}_K^T \\
 (I_1 \times T_K) & \quad (J_K \times T_K)
 \end{aligned}$$

(b)

**FIGURE 21.12**

Simple models of linked multi-block matrix decomposition. (a) The objective is to find the constrained matrix  $\mathbf{B}_C \in \mathbb{R}^{R \times T}$  representing common components (e.g., independent, nonnegative, or sparse components) and individual components represented by matrices  $\mathbf{B}_k \in \mathbb{R}^{T \times J_k}$ . (b) Dual model, in which the objective is to find the common matrix  $\mathbf{A}_C \in \mathbb{R}^{I_1 \times R}$  and a set of individual matrices  $\widetilde{\mathbf{A}}_k \in \mathbb{R}^{I_1 \times (J_k - R)}$ . These models can be considered as generalization of low-rank matrix approximation methods, including, CCA/PLS, ICA, SCA and NMF.

**FIGURE 21.13**

A generalized model of multi-block tensor decompositions for linked multiway component analysis (especially, Linked Multilinear ICA). The objective is to find the core tensors  $\underline{\mathbf{G}}^{(k)} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$  and constrained factor matrices  $\mathbf{B}^{(n,k)} = [\mathbf{B}_C^{(n)}, \mathbf{B}_I^{(n,k)}] \in \mathbb{R}^{I_n \times J_n}$ , ( $n = 1, 2, 3$ ), which are partially linked or locally maximally correlated, i.e., they have the same common (or highly correlated) components  $b_{j_n}^{(n,k)}$  for some selected indexes  $j_n$ ,  $n$ ,  $k$ .

represent similar or maximally correlated components, and  $\mathbf{B}_I^{(n,k)} \in \mathbb{R}^{I_n \times (J_n - R_n)}$ , which are generally different and mutually independent, for example, they correspond to stimuli/tasks independent individual characteristics. Based on such a decomposition model, our objective is to estimate the common factors  $\mathbf{B}_C^{(n)}$ , the independent factors  $\mathbf{B}_I^{(n,k)}$  and the interactions between them via the core tensors  $\underline{\mathbf{G}}^{(k)}$  for  $k = 1, 2, \dots, K$  and  $n = 1, 2, \dots, N$ .

If  $\mathbf{B}^{(n,k)} = \mathbf{B}_C^{(n)} \in \mathbb{R}^{I_n \times J_n}$  for all modes  $n$ , then we can concatenate all data tensors along this mode, perform tensor decomposition by applying any standard BSS algorithm for unfolding matrices to

compute the desired components in each mode (e.g., to estimate independent components, we apply any standard ICA algorithm) [12, 14]. In the more general case, when the number of common components  $R_n$  are unknown, we can perform the unfolding of each data tensor  $\underline{\mathbf{Y}}^{(k)}$  in each mode, and then apply suitable constrained matrix factorizations (e.g., by applying standard algorithms for ICA, NMF, SCA, PCA/SVD) for the reduced unfolding matrices:

$$\mathbf{Y}_{(n)}^{(k)} \cong \mathbf{B}^{(n,k)} \mathbf{G}_{(n)}^{(k)} \mathbf{Z}^{(n,k)} = \mathbf{B}_{n,k} \mathbf{A}_{n,k}^T, \quad (21.251)$$

for  $n = 1, 2, \dots, N$  and  $k = 1, 2, \dots, K$ , where the matrices  $\mathbf{B}^{(n,k)}$  represent individual components  $\mathbf{b}_j^{(n,k)}$  (some of them being linked), while the matrices  $\mathbf{A}_{n,k}^T = \mathbf{G}_{(n)}^{(k)} \mathbf{Z}^{(n,k)}$  represent basis (mixing) vectors or linked mixing processes, and  $\mathbf{Z}^{(n,k)} = [\mathbf{B}^{(N,k)} \otimes \dots \otimes \mathbf{B}^{(n+1,k)} \otimes \mathbf{B}^{(n-1,k)} \dots \otimes \mathbf{B}^{(1,k)}]^T$ .

In order to identify the common components  $\mathbf{b}_j^{(n)}$ , in the next stage we need to perform the statistical analysis and to rank and compare the individual components  $\mathbf{b}_j^{(n,k)}$  extracted in each mode  $n$ , for all blocks  $K$ , by performing clustering and ordering the components to identify common or similar (highly correlated) components (see e.g., [8, 14, 178]). In the last step, we compute the core tensors which describe the functional links between common components.

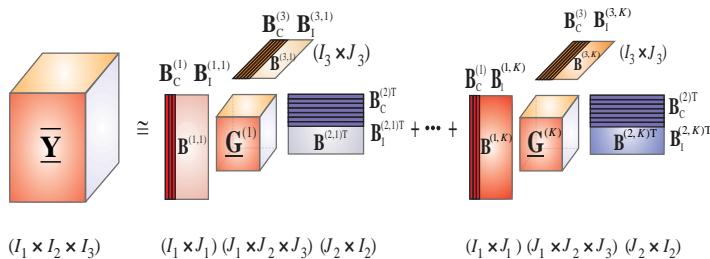
Alternative approach in the case when  $R_n < J_n$ , is to unfold each data tensor  $\underline{\mathbf{Y}}^{(k)}$  in each mode, and to perform a set of linked and constrained matrix factorizations:  $\mathbf{Y}_{(n)}^{(k)} \cong \mathbf{B}_C^{(n)} \mathbf{A}_C^{(n,k)T} + \mathbf{B}_I^{(n,k)} \mathbf{A}_I^{(n,k)T}$  through solving constrained optimization problems:

$$\begin{aligned} \min \sum_{k=1}^K & \| \mathbf{Y}_{(n)}^{(k)} - \mathbf{B}_C^{(n)} \mathbf{A}_C^{(n,k)T} + \mathbf{B}_I^{(n,k)} \mathbf{A}_I^{(n,k)T} \|_F^2 \\ & + f_n(\mathbf{B}_C^{(n)}), \quad \text{s.t. } \mathbf{B}_C^{(n)T} \mathbf{B}_I^{(n,k)} = \mathbf{0} \quad \forall k, n, \end{aligned} \quad (21.252)$$

where  $f_n$  denotes the penalty terms which impose additional constraints on common components  $\mathbf{B}_C^{(n)}$ , in order to extract unique and desired components. In the special case of orthogonality constraints, the problem can be transformed to a generalized eigenvalue problem and solved by the power method [177]. The key point is to assume that common factor sub-matrices  $\mathbf{B}_C^{(n)}$  are present in all multiple data blocks and hence reflect structurally complex (hidden) latent and intrinsic links between them. In practice, the number of common components  $C_n$  in each mode is unknown and should be estimated (see [177] for detail).

Note that our linked multilinear BSS is different from the linked ICA [38] and group NMF/ICA, since we do not limit components diversity by constraining them to be only statistically independent or only nonnegative components and our model is based on constrained Tucker models instead of the rather quite restrictive trilinear CP model.

We conclude that the proposed linked multilinear BSS model provides a framework that is quite flexible and general and it may substantially supplement many of the currently available techniques for group ICA and feature extraction for multi-block data. Moreover, the tensors decompositions models can be extended to a multiway Canonical Correlation Analysis (MCCA), and higher order Partial Least Squares, in which we impose maximal correlations among normalized factor matrices (or subsets of components) and/or core tensors [77].

**FIGURE 21.14**

A Block Term Decomposition (BTD) model of tensors decomposition for linked multiway component analysis (e.g., Linked Multilinear ICA).

For the linked multilinear BSS, we can also exploit the constrained Block Tensor Decomposition (BTD) models [179–181] for an averaging data tensor across subjects (see Figure 21.14):

$$\bar{\mathbf{Y}} = \sum_{k=1}^K (\underline{\mathbf{G}}^{(k)} \times_1 \mathbf{B}^{(1,k)} \times_2 \mathbf{B}^{(2,k)} \cdots \times_N \mathbf{B}^{(N,k)}) + \bar{\mathbf{E}}, \quad (21.253)$$

where  $\bar{\mathbf{Y}} = \sum_{k=1}^K \mathbf{Y}^{(k)}$ . Such model may provide us with additional information.

For group and linked multilinear BSS and ICA applied to the analysis of EEG/MEG data, we usually seek stimuli driven ERPs (event related responses) and/or task related common components (or common basis) reflecting both intra subject and inter subject features as basis, which are independent by involving individual on going brain activities. In other words, we seek event/task-related components  $\mathbf{B}_C^{(n)}$ , that are similar in each mode or maximally correlated across subjects, and event/task independent individual bases  $\mathbf{B}_I^{(s,n)}$ , which are independent or even as far apart as possible (anti-correlated).

How to select common components will depend on the validity of the underlying assumptions and *a priori* knowledge. For instance, identical spatial distributions can well be assumed for homogeneous subject groups, but may be unacceptable in studies that include patients with different ages, mental diseases, or abnormalities. Temporal components may be the same for stimulus- or task-evoked responses that are related to a specific mental task or paradigm, but these will vary for the spontaneous fluctuations that occur in resting state experiments. In some experiments, responses may be assumed similar or identical within, but different across subgroups [37].

### 1.21.5.8 Multiway sparse CCA/PLS using the generalized power method

The concept of standard (matrix) CCA and PLS can be extended to multiway (higher-order) models by using the normalized and standardized tensors  $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$  and  $\underline{\mathbf{Y}} \in \mathbb{R}^{K_1 \times K_2 \cdots \times K_M}$ , with  $I_1 = K_1 = I$ , instead of the matrices  $\mathbf{X} \in \mathbb{R}^{I \times N}$  and  $\mathbf{Y} \in \mathbb{R}^{I \times M}$  (see Section 1.21.2.6.6). We assume without any loss of generality that the tubes in mode-1 are standardized to have mean zero and standard deviation one. In such a case we can apply (after additional constraints) unified approach to multiway CCA and PLS.

The key factor is to find projection vectors  $\mathbf{w}_x^{(n)} \in \mathbb{R}^{I_n}$  and  $\mathbf{w}_y^{(m)} \in \mathbb{R}^{K_m}$  such that the canonical vectors, defined as

$$\mathbf{t}_1 = \underline{\mathbf{X}} \bar{\times}_2 \mathbf{w}_x^{(1)} \bar{\times}_3 \mathbf{w}_x^{(2)} \cdots \bar{\times}_N \mathbf{w}_x^{(N-1)} \in \mathbb{R}^{I_1}, \quad (21.254)$$

$$\mathbf{u}_1 = \underline{\mathbf{Y}} \bar{\times}_2 \mathbf{w}_y^{(1)} \bar{\times}_3 \mathbf{w}_y^{(2)} \cdots \bar{\times}_M \mathbf{w}_y^{(M-1)} \in \mathbb{R}^{I_1}, \quad (21.255)$$

are maximally correlated, that is

$$\begin{aligned} \max\{\mathbf{u}_1^T \mathbf{t}_1\} &= \\ \max\{(\underline{\mathbf{X}} \bar{\times}_2 \mathbf{w}_x^{(1)} \bar{\times}_3 \mathbf{w}_x^{(2)} \cdots \bar{\times}_N \mathbf{w}_x^{(N-1)})^T (\underline{\mathbf{Y}} \bar{\times}_2 \mathbf{w}_y^{(1)} \bar{\times}_3 \mathbf{w}_y^{(2)} \cdots \bar{\times}_M \mathbf{w}_y^{(M-1)})\}, \end{aligned} \quad (21.256)$$

subject to suitable constraints, typically  $\|\mathbf{w}_x^{(n)}\|_2 \leq 1$  and  $\|\mathbf{w}_y^{(m)}\|_2 \leq 1$ .

It can be shown that a such optimization problem is equivalent to the following maximization problem

$$\max_{\mathbf{w}_1^{(p)}} \{\underline{\mathbf{Z}} \bar{\times}_1 \mathbf{w}_1^{(1)} \bar{\times}_2 \mathbf{w}_1^{(2)} \cdots \bar{\times}_P \mathbf{w}_1^{(P)}\}, \quad \text{s.t. } \|\mathbf{w}_1\|_2 \leq 1, \quad (21.257)$$

where  $\mathbf{w}_1^{(p)} = \mathbf{w}_x^{(p)}$  for  $p = 1, 2, \dots, N-1$  and  $\mathbf{w}_1^{(p)} = \mathbf{w}_y^{(p-N+1)}$  for  $p = N, N+1, \dots, P$ , and the generalized covariance of two tensors is defined as

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_1 \underline{\mathbf{Y}} = COV_{(1,1)}(\underline{\mathbf{X}} \underline{\mathbf{Y}}) \in \mathbb{R}^{I_2 \times I_3 \cdots \times I_N \times K_2 \times K_3 \cdots \times K_M}, \quad (21.258)$$

with entries

$$z_{i_2, \dots, i_N, k_2, \dots, k_M} = \sum_{i=1}^I x_{i, i_2, \dots, i_N} y_{i, k_2, \dots, k_M}. \quad (21.259)$$

Note that the tensor  $\underline{\mathbf{Z}}$  has order  $P = N + M - 2$ . In the special case when the tensor  $\underline{\mathbf{Y}}$  is reduced to the matrix  $\mathbf{Y}$ , we have

$$\underline{\mathbf{Z}} = \underline{\mathbf{X}} \times_1 \mathbf{Y} \quad (21.260)$$

or both data are matrices and we can use the standards notation  $\mathbf{Z} = \mathbf{X}^T \mathbf{Y}$ .

If we need to impose sparsity constraints we can use the following cost function

$$\max_{\mathbf{w}_1^{(p)}} \{(\underline{\mathbf{Z}} \bar{\times}_1 \mathbf{w}_1^{(1)} \bar{\times}_2 \mathbf{w}_1^{(2)} \cdots \bar{\times}_P \mathbf{w}_1^{(P)})^2 - \sum_p \alpha_p \|\mathbf{w}_1^{(p)}\|_1\}, \quad \text{s.t. } \|\mathbf{w}_1^{(p)}\|_2 \leq 1. \quad (21.261)$$

This optimization of the cost function (21.261) leads to the following simple update rule

$$\mathbf{w}_1^{(p)} = \frac{\mathcal{S}_1(\mathbf{w}_1^{(\bar{p})}, \alpha_p)}{\|\mathcal{S}_1(\mathbf{w}_1^{(\bar{p})}, \alpha_p)\|_2}, \quad (21.262)$$

where

$$\mathbf{w}_1^{(\bar{p})} = \underline{\mathbf{Z}} \bar{\times}_1 \mathbf{w}_1^{(1)} \cdots \bar{\times}_{p-1} \mathbf{w}_1^{(p-1)} \bar{\times}_{p+1} \mathbf{w}_1^{(p+1)} \cdots \bar{\times}_P \mathbf{w}_1^{(P)}. \quad (21.263)$$

In the special case when  $\alpha_n = 0$  the above updated formula simplifies to (without sparsity constraints)

$$\mathbf{w}_1^{(p)} = \frac{\mathbf{w}_1^{(\bar{p})}}{\|\mathbf{w}_1^{(\bar{p})}\|_2}. \quad (21.264)$$

Our objective in a Multiway CCA is to estimate the set of factor matrices or weights  $\mathbf{W}^{(p)} = [\mathbf{w}_1^{(p)}, \mathbf{w}_2^{(p)}, \dots, \mathbf{w}_J^{(p)}] \in \mathbb{R}^{Q_p \times J}$  for  $p = 1, 2, \dots, P$ , that maximize the following cost function (which can be considered as a generalization of the variance used in the standard sparse PCA and CCA).

$$\begin{aligned} \max_{\{\mathbf{w}_j^{(p)}\}} & \left\{ (\underline{\mathbf{Z}}^{(j)} \tilde{\times}_1 \mathbf{w}_j^{(1)} \tilde{\times}_2 \mathbf{w}_j^{(2)} \cdots \tilde{\times}_P \mathbf{w}_j^{(P)})^2 - \sum_p \alpha_p \|\mathbf{w}_j^{(p)}\|_1 \right\}, \\ \text{s.t. } & \|\mathbf{w}_j^{(p)}\|_2 \leq 1 \quad (j = 1, 2, \dots, J), \end{aligned} \quad (21.265)$$

where the tensor  $\underline{\mathbf{Z}}^{(j)}$  is estimated iteratively using a simple deflation approach

$$\lambda_j = |\underline{\mathbf{Z}}^{(j)} \tilde{\times}_1 \mathbf{w}_j^{(1)} \tilde{\times}_2 \mathbf{w}_j^{(2)} \cdots \tilde{\times}_P \mathbf{w}_j^{(P)}|, \quad (21.266)$$

$$\underline{\mathbf{Z}}^{(j+1)} = \underline{\mathbf{Z}}^{(j)} - \lambda_j (\mathbf{w}_j^{(1)} \circ \mathbf{w}_j^{(2)} \cdots \circ \mathbf{w}_j^{(P)}), \quad (21.267)$$

with  $\underline{\mathbf{Z}}^{(1)} = \underline{\mathbf{Z}}$ .

The above set of optimization problems leads to the following update rule (see Algorithm 6) which is very similar to the sparse sequential factorization for the CP model (see Section 1.21.5.3):

$$\mathbf{w}_j^{(p)} = \frac{\mathcal{S}_1(\mathbf{w}_j^{(\bar{p})}, \alpha_p)}{\|\mathcal{S}_1(\mathbf{w}_j^{(\bar{p})}, \alpha_p)\|_2} \quad (j = 1, 2, \dots, J),$$

where

$$\mathbf{w}_j^{(\bar{p})} = \underline{\mathbf{Z}}^{(j)} \tilde{\times}_1 \mathbf{w}_j^{(1)} \cdots \tilde{\times}_{p-1} \mathbf{w}_j^{(p-1)} \tilde{\times}_{p+1} \mathbf{w}_j^{(p+1)} \cdots \tilde{\times}_P \mathbf{w}_j^{(P)}$$

and

$$\mathcal{S}_1(\mathbf{w}, \alpha) = \text{sign}(\mathbf{w})[|\mathbf{w}| - \alpha 1]_+. \quad (21.268)$$

In multiway PLS our objective is to perform simultaneous constrained Tucker-1 decompositions, with common or maximally correlated factor matrices (see Figure 21.15):

$$\boxed{\underline{\mathbf{X}} = \mathbf{G}_{\mathbf{X}} \times_1 \mathbf{T}^{(1)} + \mathbf{E}_{\mathbf{X}},} \quad (21.269)$$

$$\boxed{\underline{\mathbf{Y}} = \mathbf{G}_{\mathbf{Y}} \times_1 \mathbf{U}^{(1)} + \mathbf{E}_{\mathbf{Y}},} \quad (21.270)$$

where the additional constraints imposed are that:  $\mathbf{T}^{(1)} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_J]$  and  $\mathbf{U}^{(1)} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_J]$  should be maximally correlated, while the other factor matrices should be essentially different (e.g., orthogonal, or mutually independent).

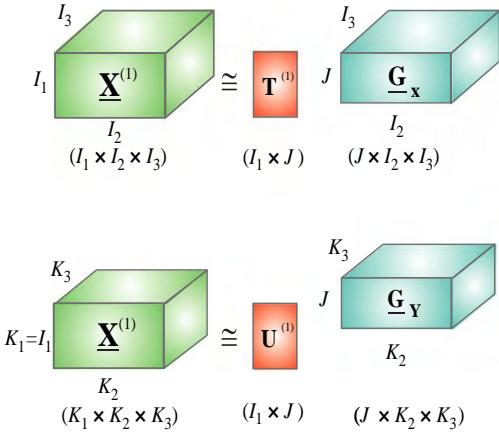
**FIGURE 21.15**

Illustration of a generalized Multiway PLS model for third-order tensors (see Eqs. 21.269, 21.270). The objective is to perform approximative decompositions of the independent tensor  $\underline{\mathbf{X}} \cong \mathbf{G}_{\mathbf{X}} \times_1 \mathbf{T}^{(1)} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$  and dependent tensor  $\underline{\mathbf{Y}} \cong \mathbf{G}_{\mathbf{Y}} \times_1 \mathbf{U}^{(1)} \in \mathbb{R}^{K_1 \times K_2 \times K_3}$ , with  $I_1 = K_1 = I$ , by imposing additional constraints, specifically that the two factor matrices  $\mathbf{T}^{(1)} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_J]$  and  $\mathbf{U}^{(1)} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_J]$  are maximally correlated in the sense that the covariances between all components are maximized, i.e.,  $\max\{\mathbf{u}_j^T \mathbf{t}_j\}, \forall j$ .

**Algorithm 6.** Power Method for Sparse Multiway CCA

**Require:** Data tensors  $\underline{\mathbf{X}}^{(1)} = \underline{\mathbf{X}}$ ,  $\underline{\mathbf{Y}}^{(1)} = \underline{\mathbf{Y}}$ , sparsity coefficients  $\alpha_p \geq 0$ , and initial vectors  $\mathbf{w}_j^{(p)}$  for  $(p = 1, 2, \dots, P)$ .

- 1: Compute  $\underline{\mathbf{Z}}^{(1)} = \underline{\mathbf{X}} \times_1 \underline{\mathbf{Y}} = COV_{(1,1)}(\underline{\mathbf{X}} \underline{\mathbf{Y}}) \in \mathbb{R}^{I_2 \times I_3 \dots \times I_N \times K_2 \times K_3 \dots \times K_M};$
- 2: For  $p = 1, 2, \dots, P$  and  $j = 1, 2, \dots, J$ 
  - (a) Repeat until convergence of  $\mathbf{w}_j^{(p)}$ ;
  - 3:  $\mathbf{w}_j^{(\bar{p})} = \underline{\mathbf{Z}}^{(j)} \bar{\times}_1 \mathbf{w}_j^{(1)} \dots \bar{\times}_{p-1} \mathbf{w}_j^{(p-1)} \bar{\times}_{p+1} \mathbf{w}_j^{(p+1)} \dots \bar{\times}_P \mathbf{w}_j^{(P)};$
  - 4:  $\mathbf{w}_j^{(p)} = \frac{\mathcal{S}_1(\mathbf{w}_j^{(\bar{p})}, \alpha_p)}{\|\mathcal{S}_1(\mathbf{w}_j^{(\bar{p})}, \alpha_p)\|_2}$  (for  $\alpha_p = 0$  take  $\mathbf{w}_j^{(p)} = \frac{\mathbf{w}_j^{(\bar{p})}}{\|\mathbf{w}_j^{(\bar{p})}\|_2}$ );
  - 5: (b) Perform optional Gram-Schmidt orthogonalization;
  - 6: (c)  $\lambda_j = |\underline{\mathbf{Z}}^{(j)} \bar{\times}_1 \mathbf{w}_j^{(1)} \bar{\times}_2 \mathbf{w}_j^{(2)} \dots \bar{\times}_P \mathbf{w}_j^{(P)}|;$
  - 7: (d)  $\underline{\mathbf{Z}}^{(j+1)} = \underline{\mathbf{Z}}^{(j)} - \lambda_j (\mathbf{w}_j^{(1)} \circ \mathbf{w}_j^{(2)} \dots \circ \mathbf{w}_j^{(P)});$
  - 8: **return**  $\mathbf{W}^{(p)} = [\mathbf{w}_1^{(p)}, \mathbf{w}_2^{(p)}, \dots, \mathbf{w}_J^{(p)}] \in \mathbb{R}^{Q_p \times J}$ ,  $\Lambda^* = \text{diag}(\lambda_1, \dots, \lambda_J);$
  - 9: **return**  $\mathbf{t}_j = \underline{\mathbf{X}} \bar{\times}_2 \mathbf{w}_j^{(1)} \bar{\times}_3 \mathbf{w}_j^{(2)} \dots \bar{\times}_N \mathbf{w}_j^{(N-1)}$ ,  
 $\mathbf{u}_j = \underline{\mathbf{Y}} \bar{\times}_2 \mathbf{w}_j^{(N)} \bar{\times}_3 \mathbf{w}_j^{(N+1)} \dots \bar{\times}_K \mathbf{w}_j^{(P)}$ ; ( $j = 1, 2, \dots, J$ ).

Such models allow for different types of structures on  $\mathbf{X}$  and  $\mathbf{Y}$  and provides a general framework for solving multiway regression problems that explore complex relationships between multi-dimensional dependent and independent variables. For example, tensor decompositions can be applied to emerging neuroimaging genetics studies to investigate the links between biological parameters measured with brain imaging and genetic variability.

### 1.21.6 Summary

In this chapter, we have discussed several important latent variables models, especially PCA, ICA, NMF, CCA and PLS and their extensions. We have tried to unify different approaches and methods by applying constrained matrix factorizations or penalized matrix decompositions. Imposing various constraints on the factors, components, or latent variables allows us to extract the components with desired properties. This approach has been extended to the important case of multi-dimensional data by using constrained tensor decompositions. Moreover, the concept of blind source separation has been extended to multilinear or multiway BSS using tensor factorizations.

*Relevant Theory:* Machine Learning

See this Volume, [Chapter 22](#) Semi-Supervised Learning

---

## References

- [1] R. Bro, PARAFAC, Tutorial and applications, in: Second International Conference in Chemometrics (INCINC'96), vol. 38, Chemometr. Intell. Lab. Syst., 1997, pp. 149–171 (special issue).
- [2] J. Carroll, Jih-Jie Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of ‘Eckart-Young’ decomposition, *Psychometrika* 35 (3) (1970) 283–319.
- [3] R.A. Harshman, Foundations of the PARAFAC procedure: models and conditions for an ‘explanatory’ multimodal factor analysis, UCLA Working Papers in Phonetics 16 (1), 1970.
- [4] F.L. Hitchcock, Multiple invariants and generalized rank of a p-way matrix or tensor, *J. Math. Phys.* 7 (1927) 39–79.
- [5] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, *SIAM Rev.* 51 (3) (2009) 455–500.
- [6] L. Tucker, Some mathematical notes on three-mode factor analysis, *Psychometrika* 31 (3) (1966) 279–311.
- [7] L.R. Tucker, The extension of factor analysis to three-dimensional matrices, in: H. Gulliksen, N. Frederiksen (Eds.), *Contributions to Mathematical Psychology*, Holt, Rinehart and Winston, New York, 1964, pp. 110–127.
- [8] A. Cichocki, R Zdunek, A.-H. Phan, S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, Wiley, Chichester, 2009.
- [9] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, *SIAM J. Matrix Anal. Appl.* 24 (2000) 1253–1278.
- [10] L. De Lathauwer, B. De Moor, J. Vandewalle, On the best rank-1 and rank-(R1, R2, ..., RN) approximation of higher-order tensors, *SIAM J. Matrix Anal. Appl.* 21 (4) (2000) 1324–1342.
- [11] F. Miwakeichi, E. Martnez-Montes, P. Valds-Sosa, N. Nishiyama, H. Mizuhara, Y. Yamaguchi, Decomposing EEG data into space-time-frequency components using parallel factor analysis, *NeuroImage* 22 (3) (2004) 1035–1045.

- [12] A.H. Phan, A. Cichocki, Tensor decompositions for feature extraction and classification of high dimensional datasets, *Nonlinear Theory Appl. IEICE* 1 (1) (2010) 37–68.
- [13] P. Comon, C. Jutten (Eds.), *Independent Component Analysis and Applications*, Handbook of Blind Source Separation, Academic Press, 2010.
- [14] A. Cichocki, S. Amari, *Adaptive Blind Signal and Image Processing*, John Wiley & Sons Ltd., New York, 2003.
- [15] A. Cichocki, S.C. Douglas, S. Amari, Robust techniques for independent component analysis (ICA) with noisy data, *Neurocomputing* 23 (1–3) (1998) 113–129.
- [16] A. Cichocki, P. Georgiev, Blind source separation algorithms with matrix constraints, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E86-A (1) (2003) 522–531. <<http://www.bsp.brain.riken.jp/publications/2003/CichockiGEIEICE.pdf>>.
- [17] A. Hyvärinen, J. Karhunen, E. Oja, *Independent Component Analysis*, John Wiley & Sons Ltd., New York, 2001.
- [18] Xi-Lin Li, T. Adali, Independent component analysis by entropy bound minimization, *IEEE Trans. Signal Process.* 58 (10) (2010) 5151–5164.
- [19] A. Cichocki, R. Zdunek, S. Amari, Csiszár’s divergences for non-negative matrix factorization: family of new algorithms, *LNCS*, vol. 3889, Springer, 2006, pp. 32–39.
- [20] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, *Nature* 401 (6755) (1999) 788–791.
- [21] Z. He, A. Cichocki, An efficient K-hyperplane clustering algorithm and its application to sparse component analysis, *Lect. Notes Comput. Sci.* 4492 (2007) 1032–1041.
- [22] Z. He, S. Xie, L. Zhang, A. Cichocki, A note on Lewicki-Sejnowski gradient for learning overcomplete representations, *Neural Comput.* 20 (3) (2008) 636–643.
- [23] Y. Li, S. Amari, A. Cichocki, D. Ho, S. Xie, Underdetermined blind source separation based on sparse representation, *IEEE Trans. Signal Process.* 54 (2006) 423–437.
- [24] Y. Li, A. Cichocki, S. Amari, Blind estimation of channel parameters and source components for EEG signals: a sparse factorization approach, *IEEE Trans. Neural Networks* 17 (2006) 419–431.
- [25] Y. Washizawa, A. Cichocki, On line K-plane clustering learning algorithm for sparse component analysis, in: Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP2006, Toulouse, France, May 14–19, 2006, pp. 681–684.
- [26] J. Bobin, Y. Moudden, J. Fadili, J. Starck, Morphological diversity and sparsity for multichannel data restoration, *J. Math. Imag. Vis.* 33 (2) (2009) 149–168.
- [27] Y. Moudden, J. Bobin, Hyperspectral BSS using GMCA with spatio-spectral sparsity constraints, *IEEE Trans. Image Process.* 20 (3) (2011) 872–879.
- [28] S. Amari, A. Cichocki, Adaptive blind signal processing – neural network approaches, *Proc. IEEE* 86 (1998) 1186–1187.
- [29] Z. He, A. Cichocki, K-EVD clustering and its applications to sparse component analysis, *Lect. Notes Comput. Sci.* 3889 (2006) 438–445.
- [30] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons Ltd., New York, 1994.
- [31] K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T.-W. Lee, T.J. Sejnowski, Dictionary learning algorithms for sparse representation, *Neural Comput.* 15 (2) (2003) 349–396.
- [32] G. Zhou, A. Cichocki, Fast and unique Tucker decompositions via multiway blind source separation, *Bull. Pol. Acad. Sci.* 60 (3) (2012) 389–407.
- [33] A. Cichocki, Generalized component analysis and blind source separation methods for analyzing multichannel brain signals, in: *Statistical and Process Models for Cognitive Neuroscience and Aging*, Lawrence Erlbaum Associates, 2007, pp. 201–272.

- [34] A. Cichocki, H.A. Phan, Fast local algorithms for large scale nonnegative matrix and tensor factorizations, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E92-A (3) (2009) 708–721.
- [35] V.D. Calhoun, J. Liu, T. Adali, A review of group ICA for fMRI data and ICA for joint inference of imaging, genetic, and ERP data, *Neuroimage* 45 (2009) 163–172.
- [36] Y. Guo, G. Pagnoni, A unified framework for group independent component analysis for multi-subject fMRI data, *NeuroImage* 42 (3) (2008) 1078–1093.
- [37] D.M.G. Langers, Unbiased group-level statistical assessment of independent component maps by means of automated retrospective matching, *Human Brain Map.* 31 (2010) 727–742.
- [38] A.R. Groves, C.F. Beckmann, S.M. Smith, M.W. Woolrich, Linked independent component analysis for multimodal data fusion, *NeuroImage* 54 (1) (2011) 2198–2217.
- [39] I.T. Jolliffe, *Principal Component Analysis*, Springer Verlag, New York, 1986.
- [40] M. Moonen, B. de Moor (Eds.), *SVD and Signal Processing. III. Algorithms, Applications and Architecture*, Elsevier Science Publishing Co. Inc., 1994.
- [41] R. Rosipal, M. Girolami, L.J. Trejo, A. Cichocki, Kernel PCA for feature extraction and de-noising in nonlinear regression, *Neural Comput. Appl.* 10 (2001) 231–243. <<http://www.bsp.brain.riken.jp/publications/2001/nca.pdf>>.
- [42] Z. He, A. Cichocki, S. Xie, K. Choi, Detecting the number of clusters in N-way probabilistic clustering, *IEEE Trans. Pattern Anal. Mach. Intell.* 32 (11) (2010) 2006–2021.
- [43] T.P. Minka, Automatic choice of dimensionality for PCA, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), *Advances in Neural Information Processing Systems*, vol. 13, MIT Press, 2001, pp. 556–562.
- [44] M.O. Ulfarsson, V. Solo, Dimension estimation in noisy PCA with SURE and random matrix theory, *IEEE Trans. Signal Process.* 56 (12) (2008) 5804–5816.
- [45] J. Karhunen, A. Cichocki, W. Kasprzak, P. Pajunen, On neural blind separation with noise suppression and redundancy reduction, *Int. J. Neural Syst.* 8 (2) (1997) 219–237.
- [46] M. Wax, T. Kailath, Detection of signals by information theoretic criteria, *IEEE Trans. Signal Process.* 33 (2) (1985) 387–392.
- [47] J. Niesing, *Simultaneous Component and Factor Analysis Methods for Two or More Groups: A Comparative Study*, second ed., DSWO Press, Leiden University, Leiden, The Netherlands, 1997.
- [48] P. Comon, G.H. Golub, Tracking of a few extreme singular values and vectors in signal processing, in: *Proc. IEEE* 78 (8) (1990) 1327–1343 (published from Stanford Report 78NA-89-01, February 1989).
- [49] S. Lipovetsky, PCA and SVD with nonnegative loadings, *Pattern Recognit.* 99 (9) (2009) (in print).
- [50] B. Yang, Projection approximation subspace tracking, *IEEE Trans. Signal Process.* 43 (1) (1995) 95–107.
- [51] V. Zipunnikov, B.C. Caffo, D.M. Yousem, C. Davatzikos, B.S. Schwartz, C.M. Crainiceanu, Functional principal component model for high-dimensional brain imaging, *NeuroImage* 58 (3) (2011) 772–784.
- [52] A. Cichocki, R. Unbehauen, Robust estimation of principal components in real time, *Electron. Lett.* 29 (21) (1993) 1869–1870.
- [53] A. Cichocki, W. Kasprzak, W. Skarbek, Adaptive learning algorithm for principal component analysis with partial data, in: R. Trappl (Ed.), *Cybernetics and Systems '96, 13th European Meeting on Cybernetics and Systems Research*, vol. 2, Austrian Society for Cybernetic Studies, Vienna, 1996, pp. 1014–1019.
- [54] M. Journée, Y. Nesterov, P. Richtarik, R. Sepulchre, Generalized power method for sparse principal component analysis, *J. Mach. Learn. Res.* 1 (2010) 517–553.
- [55] S.T. Roweis, EM algorithms for PCA and SPCA, in: *Advances in Neural Information processing Systems NIPS-98*, vol. 10, 1998, pp. 452–456.
- [56] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, *J. Roy. Stat. Soc. Ser. B* 3 (1999) 600–616.
- [57] A. d'Aspremont, F. Bach, L. El Ghaoui, Optimal solutions for sparse principal component analysis, *J. Mach. Learn. Res.* 9 (2008) 1269–1294.

- [58] R. Jenatton, G. Obozinski, F. Bach, Structured sparse principal component analysis, *J. Mach. Learn. Res. Proc. Track 9* (2010) 366–373.
- [59] I.T. Jolliffe, N.T. Trendafilov, M. Uddin, A modified principal component technique based on the LASSO, *J. Comput. Graph. Stat.* 12 (3) (2003) 1269–1294.
- [60] R. Luss, M. Teboulle, Conditional gradient algorithms for rank-one matrix approximations with a sparsity constraint, submitted for publication. Report: <<http://archiv.org/pdf/1107.1163.pdf>>.
- [61] D.M. Witten, R. Tibshirani, T. Hastie, A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis, *Biostatistics* 10 (3) (2009) 515–534.
- [62] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, *J. Comput. Graph. Stat.* 15 (2) (2006) 265–286.
- [63] G.I. Allen, M. Maletic-Savatic, Sparse non-negative generalized PCA with applications to metabolomics, *Bioinformatics* 27 (21) (2011) 3029–3035.
- [64] D.M. Witten. A penalized matrix decomposition, and its applications, PhD Dissertation, Department of Statistics, Stanford University, 2010.
- [65] H. Shen, J.Z. Huang, Sparse principal component analysis via regularized low rank matrix approximation, *J. Multivariable Anal.* 99 (6) (2008) 1015–1034. <<http://dx.doi.org/10.1016/j.jmva.2007.06.007>>.
- [66] J.Z. Huang, H. Shen, A. Buja, The analysis of two-way functional data using two-way regularized singular value decompositions, *J. Am. Stat. Assoc.* 104 (488) (2009) 1609–1620. <<http://EconPapers.repec.org/RePEc:bes:jnlasa:v:104:i:488:y:2009:p:1609-1620>>.
- [67] M. Lee, H. Shen, J.Z. Huang, J.S. Marron, Biclustering via sparse singular value decomposition, *Biometrics* 66 (4) (2010) 1087–1095. <<http://www.ncbi.nlm.nih.gov/pubmed/20163403>>.
- [68] G.I. Allen, Sparse higher-order principal components analysis, *J. Mach. Learn. Res. – Proc. Track 22* (2012) 27–36.
- [69] C. Croux, P. Filzmoser, H. Fritz, Robust Sparse Principal Component Analysis, Open Access Publications from Katholieke Universiteit Leuven, Katholieke Universiteit Leuven, 2011. <<http://EconPapers.repec.org/RePEc:ner:leuven:urn:hdl:123456789/310504>>.
- [70] L. Mackey, Deflation methods for sparse PCA, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), *Advances in Neural Information Processing Systems (NIPS)*, MIT Press, 2009, pp. 1017–1024.
- [71] D. Widjaja, C. Varon, A. Dorado, J.A.K. Suykens, S. Van Huffel, Application of kernel principal component analysis for single-lead-ecg-derived respiration, *IEEE Trans. Biomed. Eng.* 59 (4) (2012) 1169–1176.
- [72] H. Hotelling, Relations between two sets of variates, *Biometrika* 28 (1936) 321–377.
- [73] D.R. Hardoon, J. Shawe-Taylor, Sparse canonical correlation analysis, *Mach. Learn.* 83 (3) (2011) 331–353.
- [74] D.R. Hardoon, S. Szedmák, J. Shawe-Taylor, Canonical correlation analysis: an overview with application to learning methods, *Neural Comput.* 16 (12) (2004) 2639–2664.
- [75] R. Rosipal, N. Krämer, Overview and recent advances in partial least squares, in: *Lecture Notes in Computer Science*, 2005, pp. 34–51.
- [76] A. Krishnan, L.J. Williams, A.R. McIntosh, H. Abdi, Partial least squares (PLS) methods for neuroimaging: a tutorial and review, *NeuroImage* 56 (2) (2011) 455–475.
- [77] Q. Zhao, C. Caiafa, D. Mandic, Z. Chao, Y. Nagasaka, N. Fujii, L. Zhang, A. Cichocki, Higher-order partial least squares (HOPLS): a generalized multi-linear regression method, in: *PAMI*, 2012. <<http://archiv.org/pdf/1207.1230.pdf>>.
- [78] H. Abdi, Partial least square regression, projection on latent structure regression, PLS-regression, *Wiley Interdiscipl. Rev.: Comput. Stat.* 2 (2010) 97–106.
- [79] B. McWilliams, G. Montana, Multi-view predictive partitioning in high dimensions, *Statist. Anal. Data Mining* 5 (3) (2012) 304–321. Available from: <<arXiv:1202.0825v1>>.
- [80] Xi-Lin Li, T. Adali, Blind separation of noncircular correlated sources using gaussian entropy rate, *IEEE Trans. Signal Process.* 59 (6) (2011) 2969–2975.

- [81] L. Tong, V.C. Soon, Y.F. Huang, R. Liu, AMUSE: a new blind identification algorithm, in: Proceedings for the IEEE ISCAS, vol. 3, IEEE, 1990, pp. 1784–1787.
- [82] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, É. Moulines, A blind source separation technique using second-order statistics, *IEEE Trans. Signal Process.* 45 (2) (1997) 434–444.
- [83] A. Ziehe, K.-R. Müller, G. Nolte, B.-M. Mackert, G. Curio, Artifact reduction in biomagnetic recordings based on time-delayed second order correlations, *IEEE Trans. Biomed. Eng.* 47 (2000) 75–87.
- [84] L. De Lathauwer, A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization, *SIAM J. Matrix Anal. Appl.* 28 (2006) 642–666.
- [85] A. Ziehe, M. Kawanabe, S. Harmeling, K.-R. Müller, A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation, *J. Mach. Learn. Res.* 5 (2004) 801–818.
- [86] A. Belouchrani, A. Cichocki, Robust whitening procedure in blind source separation context, *Electron. Lett.* 36 (24) (2000) 2050–2053.
- [87] S.A. Cruces, A. Cichocki, Combining blind source extraction with joint approximate diagonalization: thin algorithms for ICA, in: Proceedings of 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003), Kyoto, Japan, Riken, ICA, April 2003, pp. 463–468. <<http://www.bsp.brain.riken.jp/publications/2003/ICA03CrucesCich.pdf>>.
- [88] P. Tichavsky, Z. Koldovsky, Fast and accurate methods of independent component analysis: a survey, *Kybernetika* 47 (3) (2011) 426–438.
- [89] P. Tichavsky, A. Yeredor, Fast approximate joint diagonalization incorporating weight matrices, *IEEE Trans. Signal Process.* 47 (3) (2009) 878–891.
- [90] Z. Koldovsky, P. Tichavsky, E. Oja, Efficient variant of algorithm fastica for independent component analysis attaining the Cramér-Rao lower bound, *IEEE Trans. Neural Networks* 17 (5) (2006) 1265–1277.
- [91] A.J. Bell, T.J. Sejnowski, An information maximization approach to blind separation and blind deconvolution, *Neural Comput.* 7 (6) (1995) 1129–1159.
- [92] S. Choi, A. Cichocki, L.L. Zhang, S. Amari, Approximate maximum likelihood source separation using the natural gradient, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E86-A (1) (2003) 198–205. <<http://www.bsp.brain.riken.jp/publications/2003/ieice02.pdf>>.
- [93] A. Cichocki, R. Unbehauen, Robust neural networks with on-line learning for blind identification and blind separation of sources, *IEEE Trans. Circ. Syst. I: Fundam. Theory Appl.* 43 (11) (1996) 894–906.
- [94] A. Cichocki, R. Unbehauen, L. Moszczyński, E. Rummert, A new on-line adaptive learning algorithm for blind separation of sources, in: Proceedings of the 1994 International Symposium on Artificial Neural Networks, ISANN-94, Tainan, Taiwan, December 1994, pp. 406–411.
- [95] A. Cichocki, R. Unbehauen, E. Rummert, Robust learning algorithm for blind separation of signals, *Electron. Lett.* 30 (17) (1994) 1386–1387.
- [96] S. Fiori, A fully multiplicative orthogonal-group ICA neural algorithm, *Electron. Lett.* 39 (24) (2003) 1737–1738.
- [97] S.A. Cruces, A. Cichocki, S. Amari, On a new blind signal extraction algorithm: different criteria and stability analysis, *IEEE Signal Process. Lett.* 9 (8) (2002) 233–236.
- [98] S. Amari, A. Cichocki, H.H. Yang, A new learning algorithm for blind signal separation, in: M.C. Mozer, D.S. Touretzky, M.E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, December 1995, vol. 8, MIT Press, Cambridge MA, 1996, pp. 757–763.
- [99] S.A. Cruces, L. Castedo, A. Cichocki, Robust blind source separation algorithms using cumulants, *Neurocomputing* 49 (2002) 87–118.
- [100] J.-F. Cardoso, B. Laheld, Equivariant adaptive source separation, *IEEE Trans. Signal Process.* 44 (1996) 3017–3030.

- [101] S. Choi, A. Cichocki, S. Amari, Equivariant nonstationary source separation, *Neural Networks* 15 (2002) 121–130.
- [102] S.A. Cruces, A. Cichocki, L. Castedo, An iterative inversion approach to blind source separation, *IEEE Trans. Neural Networks* 11 (6) (2000) 1423–1437.
- [103] A. Cichocki, R. Thawonmas, On-line algorithm for blind signal extraction of arbitrarily distributed, but temporally correlated sources using second order statistics, *Neural Process. Lett.* 12 (1) (2000) 91–98.
- [104] J.V. Stone, Blind source separation using temporal predictability, *Neural Comput.* 13 (7) (2001) 1559–1574.
- [105] A. Cichocki, R. Thawonmas, S. Amari, Sequential blind signal extraction in order specified by stochastic properties, *Electron. Lett.* 33 (1) (1997) 64–65.
- [106] A.K. Barros, A. Cichocki, Extraction of specific signals with temporal structure, *Neural Comput.* 13 (9) (2001) 1995–2000.
- [107] H.-Y. Jung, S.-Y. Lee, On the temporal decorrelation of feature parameters for noise-robust speech recognition, *IEEE Trans. Speech Audio Process.* 8 (7) (2000) 407–416.
- [108] P. Paatero, U. Tapper, Positive matrix factorization: a nonnegative factor model with optimal utilization of error estimates of data values, *Environmetrics* 5 (1994) 111–126.
- [109] D.D. Lee, H.S. Seung, Algorithms for Nonnegative Matrix Factorization, vol. 13, MIT Press, 2001.
- [110] A. Shashua, R. Zass, T. Hazan, Multi-way clustering using super-symmetric non-negative tensor factorization, in: European Conference on Computer Vision (ECCV), Graz, Austria, May 2006. <<http://www.cs.huji.ac.il/zass/>>.
- [111] R. Zass, A. Shashua, Nonnegative sparse PCA, in: Neural Information Processing Systems (NIPS), Vancouver, Canada, December 2006. <<http://www.cs.huji.ac.il/zass/>>.
- [112] M. Berry, M. Browne, A. Langville, P. Pauca, R. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, *Comput. Stat. Data Anal.* 52 (1) (2007) 155–173. <<http://www.wfu.edu/~plemmons/papers.htm>>.
- [113] B. Dong, M.M. Lin, M.T. Chu, Nonnegative rank factorization via rank reduction. Report: <<http://www4.ncsu.edu/mtchu/Research/Papers/Readme.html>>.
- [114] C. Ding, X. He, H.D. Simon, On the equivalence of nonnegative matrix factorization and spectral clustering, in: Proceedings of the SIAM International Conference on Data Mining (SDM'05), 2005, pp. 606–610. <<http://crd.lbl.gov/cding/papers/nmfSIAM1.pdf>>.
- [115] N. Gillis, F. Glineur, Using underapproximations for sparse nonnegative matrix factorization, *Pattern Recogn.* 43 (4) (2010) 1676–1687.
- [116] N. Gillis, F. Glineur, Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization, *Neural Comput.* 24 (4) (2012) 1085–1105.
- [117] C. Ding, T. Li, W. Peng, M.I. Jordan, Convex and semi-nonnegative matrix factorizations, *IEEE Trans. PAMI* 32 (2010) 44–51. <<http://www.cs.berkeley.edu/jordan/papers/ding-li-jordan.pdf>>.
- [118] C. Ding, T. Li, W. Peng, H. Park, Orthogonal nonnegative matrix tri-factorizations for clustering, in: KDD06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM Press, 2006, pp. 126–135, ISBN: 1-59593-339-5.
- [119] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmean, R. Pacual-Marqui, Nonsmooth nonnegative matrix factorization (nsNMF), *IEEE Trans. Pattern Anal. Mach. Intell.* 28 (3) (2006) 403–415.
- [120] C. Caiafa, A. Cichocki, Generalizing the column-row matrix decomposition to multi-way arrays, *Linear Algebra Appl.* 433 (3) (2010) 557–573.
- [121] P. Drineas, M.W. Mahoney, S. Muthukrishnan, Relative-error CUR matrix decompositions, *SIAM J. Matrix Anal. Appl.* 30 (2008) 844–881.
- [122] S.A. Goreinov, E.E. Tyrtyshnikov, N.L. Zamarashkin, A theory of pseudo-skeleton approximations, *Linear Algebra Appl.* 261 (1997) 1–21.

- [123] M.W. Mahoney, P. Drineas, CUR matrix decompositions for improved data analysis, in: Proc. Natl. Acad. Sci. 106 (2009) 697–702.
- [124] M.W. Mahoney, M. Maggioni, P. Drineas, Tensor-CUR decompositions and data applications, SIAM J. Matrix Anal. Appl. 30 (2008) 957–987.
- [125] J. Sun, Incremental pattern discovery on streams, graphs and tensors, PhD Thesis, CMU-CS-07-149, 2007.
- [126] A. Cichocki, R. Zdunek, S.-I. Amari, Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization, Lecture Notes on Computer Science, LNCS, vol. 4666, Springer, 2007, pp. 169–176.
- [127] A. Cichocki, R. Zdunek, Multilayer nonnegative matrix factorization, Electron. Lett. 42 (16) (2006) 947–948.
- [128] L. Badea, Extracting gene expression profiles common to Colon and Pancreatic Adenocarcinoma using simultaneous nonnegative matrix factorization, in: Proceedings of Pacific Symposium on Biocomputing PSB-2008, World Scientific, 2008, pp. 267–278.
- [129] Z. Yang, E. Oja, Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization, IEEE Trans. Neural Networks 22 (12) (2011) 1878–1891.
- [130] A. Cichocki, S. Cruces, S. Amari, Generalized alpha-beta divergences and their application to robust non-negative matrix factorization, Entropy 13 (2011) 134–170.
- [131] M.E. Daube-Witherspoon, G. Muehllehner, An iterative image space reconstruction algorithm suitable for volume ECT, IEEE Trans. Med. Imag. 5 (1986) 61–66.
- [132] C.L. Byrne, Accelerating the EMML algorithm and related iterative algorithms by rescaled block-iterative (RBI) methods, IEEE Trans. Image Process. IP-7 (1998) 100–109.
- [133] A.R. De Pierro, On the relation between the ISRA and the EM algorithm for positron emission tomography, IEEE Trans. Med. Imag. 12 (2) (1993) 328–333.
- [134] A.R. De Pierro, M.E. Beleza Yamagishi, Fast iterative methods applied to tomography models with general Gibbs priors, in: Proceedings of the SPIE Conference Mathematical Modeling, Bayesian Estimation and, Inverse Problems, vol. 3816, 1999, pp. 134–138.
- [135] H. Lantéri, M. Roche, C. Aime, Penalized maximum likelihood image restoration with positivity constraints: multiplicative algorithms, Inverse Problems 18 (2002) 1397–1419.
- [136] R.M. Lewitt, G. Muehllehner, Accelerated iterative reconstruction for positron emission tomography based on the EM algorithm for maximum-likelihood estimation, IEEE Trans. Med. Imag. MI-5 (1986) 16–22.
- [137] A. Cichocki, A.H. Phan, R. Zdunek, L.-Q. Zhang, Flexible component analysis for sparse, smooth, nonnegative coding or representation, in: Lecture Notes in Computer Science, LNCS, vol. 4984, Springer, 2008, pp. 811–820.
- [138] R. Bro, Multi-way analysis in the food industry – models, algorithms, and applications, PhD Thesis, University of Amsterdam, Holland, 1998.
- [139] N.-D. Ho, Nonnegative matrix factorization – algorithms and applications, Thesis/Dissertation, Universite Catholique de Louvain, Belgium, FSA/INMA, Departement d'ingenierie mathematique, 2008.
- [140] N.D. Ho, P. Van Dooren, Non-negative matrix factorization with fixed row and column sums, Linear Algebra Appl. 429 (5–6) (2008) 1020–1025, ISSN: 0024–3795.
- [141] A.H. Phan, A. Cichocki, Multi-way nonnegative tensor factorization using fast hierarchical alternating least squares algorithm (HALS), in: Proceedings of the International Symposium on Nonlinear Theory and its Applications, Budapest, Hungary, 2008.
- [142] M. Hasan, F. Pellacini, K. Bala, Matrix row-column sampling for the many-light problem, in: SIGGRAPH, 2007. <<http://www.cs.cornell.edu/~mhasan/>>.
- [143] M. Hasan, E. Velazquez-Armendariz, F. Pellacini, K. Bala, Tensor clustering for rendering many-light animations, in: Eurographics Symposium on Rendering, vol. 27, 2008. <<http://www.cs.cornell.edu/~mhasan/>>.
- [144] G. Zhou, A. Cichocki, S. Xie, Fast nonnegative matrix/tensor factorization based on low-rank approximation, IEEE Trans. Signal Process. 60 (6) (2012) 2928–2940.

- [145] S. Amari, Integration of stochastic models by minimizing alpha-divergence, *Neural Comput.* 19 (2007) 2780–2796.
- [146] M. Minami, S. Eguchi, Robust blind source separation by Beta-divergence, *Neural Comput.* 14 (2002) 1859–1886.
- [147] T.P. Minka, Divergence measures and message passing, Microsoft Research Technical Report (MSR-TR-2005), 2005.
- [148] C. Févotte, N. Bertin, J.-L. Durrieu, Nonnegative matrix factorization with the Itakura-Saito divergence, with application to music analysis, *Neural Comput.* 21 (3) (2009) 793–830.
- [149] R. Kompass, A generalized divergence measure for nonnegative matrix factorization, *Neural Comput.* 19 (3) (2006) 780–791.
- [150] J. Kivinen, M.K. Warmuth, Exponentiated gradient versus gradient descent for linear predictors, *Inform. Comput.* 132 (1997).
- [151] P. Smaragdis, Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs, *Lect. Notes Comput. Sci.* 3195 (2004) 494–499.
- [152] M.N. Schmidt, M. Mørup, Nonnegative matrix factor 2-D deconvolution for blind single channel source separation, LNCS, vol. 3889, Springer, 2006, pp. 700–707.
- [153] A.H. Phan, A. Cichocki, P. Tichavský, Z. Koldovský, On connection between the convolutive and ordinary nonnegative matrix factorizations, in: Latent Variable Analysis and Signal Separation, LNCS, Springer, 2012, pp. 288–296.
- [154] J. Eggert, E. Körner, Sparse coding and NMF, in: Proceedings of IEEE International Joint Conference on Neural Networks, vol. 4, 2004, pp. 2529–2533.
- [155] P. Smaragdis, J.C. Brown, Nonnegative matrix factorization for polyphonic music transcription, in: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, October 2003, pp. 177–180.
- [156] A.H. Phan, P. Tichavský, A. Cichocki, Z. Koldovský, Low-rank blind nonnegative matrix deconvolution, in: Proceedings of 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2012, pp. 1893–1896.
- [157] P. Smaragdis, Convolutive speech bases and their application to supervised speech separation, *IEEE Trans. Audio Speech Lang. Process.* 15 (1) (2007) 1–12.
- [158] H.A.L. Kiers, A.K. Smilde, Constrained three-mode factor analysis as a tool for parameter estimation with second-order instrumental data, *J. Chemometr.* 12 (1998) 125–147.
- [159] M. Mørup, L.K. Hansen, S.M. Arnfred, Algorithms for sparse nonnegative tucker decompositions, *Neural Comput.* 20 (2008). <<http://www2.imm.dtu.dk/pubdb/>>.
- [160] A.H. Phan, A. Cichocki, Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multi-way analysis and classification, *Neurocomputing* 74 (2011) 1956–1969.
- [161] R.A. Harshman, M.E. Lundy, Data preprocessing and the extended PARAFAC model, in: H.G. Law, C.W. Snyder, J.A. Hattic, R.P. McDonald (Eds.), *Research Methods for Multimode Data Analysis*, Praeger, New York, pp. 216–284.
- [162] L. De Lathauwer, Parallel factor analysis by means of simultaneous matrix decompositions, in: Proceedings of the First IEEE International Workshop on Computational Advances in Multi-sensor Adaptive Processing (CAMSAP 2005), Puerto Vallarta, Jalisco State, Mexico, 2005, pp. 125–128.
- [163] J.D. Carroll, G. De Soete, S. Pruzansky, Fitting of the latent class model via iteratively reweighted least squares CANDECOMP with nonnegativity constraints, in: R. Coppi, S. Bolasco (Eds.), *Multiway Data Analysis*, Elsevier, Amsterdam, The Netherlands, 1989, pp. 463–472.
- [164] C.A. Andersson, R. Bro, The N-way toolbox for MATLAB, *Chemometr. Intell. Lab. Syst.* 52 (1) (2000) 1–4.

- [165] H. Kim, L. Eldén, H. Park, Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares, in: Proceedings of IEEE 7th International Conference on Bioinformatics and Bioengineering (BIBE07), vol. II, 2007, pp. 1147–1151.
- [166] P. Paatero, Least-squares formulation of robust nonnegative factor analysis, *Chemometr. Intell. Lab. Syst.* 37 (1997) 23–35.
- [167] P. Paatero, A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis, *Chemometr. Intell. Lab. Syst.* 38 (2) (1997) 223–242.
- [168] V. de Silva, L-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, *SIAM J. Matrix Anal. Appl.* 30 (2008) 1084–1127.
- [169] S. Weiland, F. van Belzen, Singular value decompositions and low rank approximations of tensors, *IEEE Trans. Signal Process.* 58 (3) (2010) 1171–1182.
- [170] J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, *Linear Algebra Appl.* 18 (2) (1977) 95–138.
- [171] N.D. Sidiropoulos, R. Bro, On the uniqueness of multilinear decomposition of N-way arrays, *J. Chemometr.* 14 (3) (2000) 229–239.
- [172] L.-H. Lim, P. Comon, Nonnegative approximations of nonnegative tensors, *J. Chemometr.* 23 (2009) 432–441.
- [173] E. Acar, D.M. Dunlavy, T.G. Kolda, M. Mørup, Scalable tensor factorizations for incomplete data, *Chemometr. Intell. Lab. Syst.* 106 (1) (2011) 41–56. <<http://www2.imm.dtu.dk/pubdb/p.php?5923>>.
- [174] I. Kopriva, I. Jeric, A. Cichocki, Blind decomposition of infrared spectra using flexible component analysis, *Chemometr. Intell. Lab. Syst.* 97 (2009) 170–178.
- [175] M.A.O. Vasilescu, D. Terzopoulos, Multilinear analysis of image ensembles: tensorfaces, in: Proceedings of the European Conference on Computer Vision (ECCV), vol. 2350, Copenhagen, Denmark, May 2002, pp. 447–460.
- [176] C. Crainiceanu, B. Caffo, S. Luo, V. Zipunnikov, N. Punjabi, Population value decomposition, a framework for the analysis of image populations, *J. Am. Stat. Assoc. (with discussion and rejoinder)* 106 (495) (2011) 775–790. <<http://pubs.amstat.org/doi/abs/10.1198/jasa.2011.ap10089>>.
- [177] G. Zhou, A. Cichocki, S. Xie, Common and individual features analysis: beyond canonical correlation analysis, *IEEE Trans. PAMI*. <<http://archiv.org/pdf/1212.3913.pdf>>.
- [178] A. Cichocki, S. Amari, K. Siwek, et al., ICALAB Toolbox, 2007. <<http://www.bsp.brain.riken.jp/ICALAB>>.
- [179] L. De Lathauwer, Decompositions of a higher-order tensor in block terms – Part I and II, *SIAM J. Matrix Anal. Appl. (SIMAX)* 30 (3) (2008) 1022–1066.
- [180] L. De Lathauwer, D. Nion, Decompositions of a higher-order tensor in block terms – Part III: Alternating least squares algorithms, *SIAM J. Matrix Anal. Appl. (SIMAX)* 30 (3) (2008) 1067–1083.
- [181] L. De Lathauwer, Blind separation of exponential polynomials and the decomposition of a tensor in rank-( $L_r, L_r, 1$ ) terms, *SIAM J. Matrix Anal. Appl.* 32 (4) (2011) 1451–1474.

# Semi-Supervised Learning

# 22

Xueyuan Zhou\* and Mikhail Belkin†

\**Department of Computer Science, The University of Chicago, 1100 East 58th Street, Chicago, IL, USA*  
†*Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA*

## 1.22.1 Introduction

Large amounts of data are being generated every second from a wide variety of sources including online news, social media, images from personal digital cameras, images from telescopy, financial data, biological data, such as DNA sequences, just to name a few.

Of course, all that raw data are useless without efficient tools for extracting and analyzing useful information contained within. One of the most standard and useful tasks in machine learning and data analysis is that of labeling, assigning each data point a certain tag, label or a numerical value according to some prespecified criterion. These labeled data, in the form of training sets, can be used to develop algorithms for making automatic inference about the future unseen data, for tasks, such as image classification, predicting disease susceptibility on the basis of genetic information, automatically tagging new topics, and many others.

However, in many situations obtaining high-quality labeled information can be costly and time-consuming as it may require human input (e.g., labeling speech, images, or news stories) or experimentation (testing for disease). Thus, there is a growing need to develop effective and computationally efficient machine learning algorithms utilizing data, which combines a small number of labeled instances with abundant unlabeled data. In addition, understanding the contribution of unlabeled data to learning from examples is closely related to the fundamental problem of understanding human and animal cognitive process, many of them, arguably, unsupervised or weakly supervised. Semi-supervised or partially supervised learning refers to a class of machine learning techniques which combine labeled and unlabeled data to build classifiers.

Two basic questions of semi-supervised learning is whether unlabeled data are helpful for inference, and if yes, how they can be used to improve the inference quality. These two questions have been studied intensively in the last decade. It should be made clear from the outset that effective use of unlabeled data requires certain assumptions. Certainly, it is not possible to have prediction purely on the basis of unlabeled data alone. Thus the role of unlabeled data is to restrict the space of potential inference rules, so that choosing a correct (or nearly correct) predictor requires less labeled data. To make this possible the structure of unlabeled data has to be aligned with the structure provided by the labels. The nature of this alignment can be encoded through several mathematical formulations leading to different algorithms and theoretical analyses.

The three most common assumptions of semi-supervised learning are the cluster assumption, manifold assumption, and what can be called a “compatibility” assumption. For simplicity, we will discuss these in the context of classification, although other setting may also be applicable.

The cluster assumption suggests that the unlabeled data has clusters, with each cluster corresponding to just a single class. Thus, identifying the clusters from unlabeled data dramatically reduces the space of possible classification rules. In the simplest noiseless case, one labeled instance per cluster is enough for classification.

The manifold assumption is a bit harder to formulate informally. Basically, it states that the data lie on or close to low-dimensional submanifolds in a high-dimensional space and that the conditional class likelihood varies smoothly along the manifold.

Finally, the compatibility assumption relies on data having a different set of attributes, such as web pages having both text and link information. The assumption is that the classification obtained separately using these attributes need to be consistent when compared on the unlabeled part of the data set.

Some of the earliest theoretical work on semi-supervised learning is based on the cluster assumption. For example, the papers [1–3] address the relative significance of labeled and unlabeled data under a special type of clustering assumption, when the classes are sampled from different mixture components. It turns out that in that setting the labeled data is exponentially more informative than the unlabeled points. An important early work on semi-supervised learning is based on the Vapnik’s idea of transductive inference [4]. Transductive inference is not completely equivalent to semi-supervised learning as the goal of transduction is to classify the unlabeled points without necessarily building a global classifier (known as the out-of-sample extension problem). Vapnik’s approach consisted in finding a classifier that maximized margin over *unlabeled data* and can be interpreted as a form of cluster assumption. Another important early algorithm was co-training [5], which introduced the idea of semi-supervised learning based on the “compatibility” assumption. A more recent class of algorithms has been based on graph Laplacian, where unlabeled data is utilized to construct a graph representing the underlying space. The Laplacian of that graph is then used for restricting the space of potential classifiers.

In a standard semi-supervised learning (SSL) problem, one is given two sample sets, the labeled set  $X_L$  together with its label set  $Y_L$ , and the unlabeled set  $X_U$ .  $X_L$  consists of  $l$  sample points  $X_L = \{x_1, \dots, x_l\}$ , with the corresponding labels as  $Y_L = \{y_1, \dots, y_l\}$ , and  $X_U$  consists of  $u$  sample points  $X_U = \{x_{l+1}, \dots, x_{l+u}\}$ . The label for  $X_U$  is  $Y_U = \{y_{l+1}, \dots, y_{l+u}\}$ , which is unknown. It is also convenient to denote  $X = X_L \cup X_U$ , and similarly  $Y = Y_L \cup Y_U$ , where all  $x_i \in \mathcal{X}$  and  $y_i \in \mathcal{Y}$ . Let  $n = l + u$ . The  $\mathcal{X}$  can be a regular subset of  $\mathbb{R}^d$ , a smooth manifold in  $\mathbb{R}^d$ , or sequences in language problems. Similarly,  $\mathcal{Y}$  can be binary class labels  $\{\pm 1\}$ , real-valued numbers, or complex structured output. The data in SSL is also called partially observed data.

In inductive inference, the goal is to find a function  $\hat{f}(x)$  on the whole data domain  $\mathcal{X}$  such that for a new data point  $x \in \mathcal{X}$ , its label  $y = \hat{f}(x)$  can be obtained by function evaluation. Instead of requiring  $\hat{f}(x)$  to be a good estimate on the whole domain, in transductive inference,  $\hat{f}(x)$  is only required to be defined on the unlabeled sample points, i.e., samples in  $X_U$ .

To define the setting for some of these problems more formally, let  $P(x, y)$  be a joint distribution on  $\mathcal{X} \times \mathcal{Y}$ , where  $\mathcal{X}$  is the domain for the attributes and  $\mathcal{Y}$  is the domain for the labels. We will assume that the data is sampled i.i.d. from the probability distribution  $P$ , a typical assumption in the machine learning literature. Let the marginal distribution on  $\mathcal{X}$  be  $P(x)$ . Then goal of inference is to reconstruct the conditional probability distribution  $P(y|x)$ . Abundant unlabeled data mean that  $P(x)$  can be assumed

to be given, or at least can be accurately estimated. Thus we need to connect the conditional distribution  $P(x|y)$  to the marginal  $P(x)$ . Thus the three assumptions can be represented as follows:

- *Cluster assumption*: Marginal distribution of the data  $P(x)$  has multiple clusters, and example points of the same cluster are likely to be of the same class. It is possible for one class to have multiple clusters. The cluster assumption is closely related to the low density separation assumption which states that the decision boundary should pass through a region of low density.
- *Manifold assumption*: The marginal distribution of the data  $P(x)$  is supported on a low-dimensional manifold embedded in a high-dimensional ambient space. The conditional distribution  $P(y|x)$  is smooth, as a function of  $x$ , with respect to this low-dimensional manifold.
- *Compatibility assumption*: In multi-view learning, disagreement between views happens with zero probability. Specifically, in the case of two views, for a data point  $x = (x_1, x_2)$ ,  $P(f_1(x_1) \neq f_2(x_2)) = 0$ , where  $f_1(\cdot)$  and  $f_2(\cdot)$  are classifiers in each view. This is one of the important assumptions of co-training [5].

The chapter is organized as follows. A variety of algorithms have been introduced for SSL since 1990s; several of them are reviewed in Section 1.22.2 according to a rough historical order.

One of the earliest SSL algorithms is transductive SVM (TSVM), which is also closely related to low density separation based algorithms. These methods try to find classifiers that lie in the low density regions which are more likely to be the decision boundary. Starting from the cluster assumption and manifold assumption, graph-based SSL methods are another active area. Some of these methods are based on mature function approximation theories, and also show promising results in practice.

Another early SSL algorithm is co-training, which explores the idea of using two different “views” of data to improve learning. Following the co-training framework, multi-view and co-regularization methods further explore the information gain from using more than one view.

SSL for structured outputs, a popular recent topic of research, is briefly discussed in Section 1.22.3. Applications include various natural language processing problems. The need for SSL algorithms is obvious: the difficulty of obtaining high-quality labels is even higher in structured learning, since the output is more complex than a single scalar value.

One key difficulty for a wider application of SSL in practice is the computational cost for most SSL algorithms. Several works on large scale SSL are reviewed in Section 1.22.4. In particular, one technique using graph Laplacian eigenvectors is discussed in detail.

The fundamental question for SSL: whether unlabeled data are helpful, and if so, how to use them is still an interesting question. Several related studies are discussed in Section 1.22.5, and several open issues of SSL are also briefly discussed in Section 1.22.6. For other reviews on SSL, see [6–8].

## 1.22.2 Semi-supervised learning algorithms

A variety of algorithms have been introduced for SSL since 1990s. Several of them are reviewed in this section.

### 1.22.2.1 TSVM and low density separation

Transductive support vector machines (TSVMs) generalize support vector machines to SSL setting with unlabeled data. Similar to SVMs, TSVMs also learn a large margin hyperplane classifier using labeled

training data. However, TSVMs simultaneously force this hyperplane to be far away from both the labeled and the unlabeled data.

The linear TSVMs solve the following optimization problem:

$$\begin{aligned} \min_{w, \xi, y_i, l+1 \leq i \leq n} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i + C^* \sum_{i=l+1}^n \xi_i, \\ \text{s.t.} \quad & y_i (w^T x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad 1 \leq i \leq l, \\ & y_i (w^T x_i) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad l+1 \leq i \leq n, \end{aligned} \quad (22.1)$$

where  $w$  is the coefficient of the hyperplane,  $C$  and  $C^*$  are the regularization penalty coefficients. The problem is similar to a regular SVM problem, while the difference lies in that the labels of unlabeled points are treated as optimization variables. Then a large margin constraint is added to both labeled and unlabeled data, hoping to find a classifier with the maximum classification margin for both labeled and unlabeled examples. Nonlinear classifiers can be obtained by choosing a reproducing Hilbert space as the solution space.

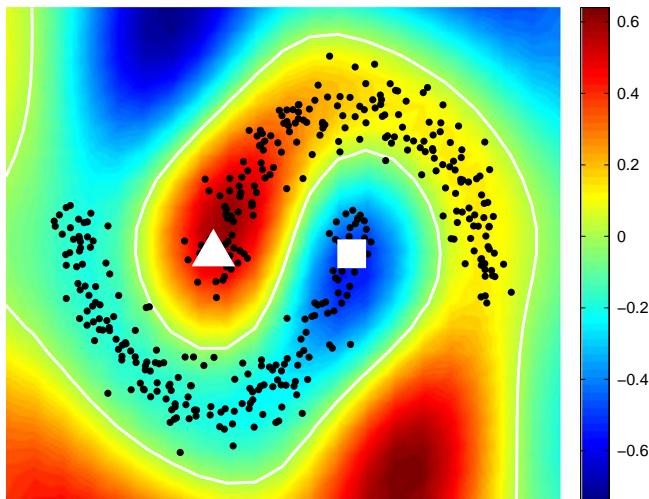
An approximation solution by local search is proposed by Joachims [9]. The algorithm begins with a labeling of the unlabeled data based on the classification of an inductive classifier. Then it improves the solution by switching the labels of the unlabeled data so that the objective function decreases. Notice that TSVMs are transductive, meaning that the algorithm only produces the prediction on unlabeled data, instead of a function over the whole domain. Several other algorithms are introduced to generalize to inductive learning, e.g., Laplacian SVM [10]. Since the optimization problem of TSVMs is non-convex, many approximation methods are introduced. The SVM-light [9] is the first widely used TSVMs implementation. A fast linear SVM (SVMlin) [11] is proposed for large scale application.  $\nabla$ TSVM using gradient descent is proposed in [12]. The concave-convex procedure (CCCP) is used in [13] to solve TSVMs problem. CCCP iteratively optimizes non-convex cost functions that can be expressed as the sum of a convex function and a concave function. The optimization is carried out iteratively by solving a sequence of convex problems obtained by linearly approximating the concave function in the vicinity of the solution of the previous convex problem. See [14] for a review of different optimization techniques.

TSVMs are shown to perform well in text classification tasks [9]. Particularly in small labeled data set cases, it generally outperforms supervised method such as SVMs. However, one drawback of TSVMs is that it is a non-convex optimization problem, which poses a serious hurdle. Chapelle et al. [15] use the branch and bound search method on small data sets to find the global optimal solution. The authors show that the globally optimal solution of the TSVMs problem is actually far better than what most practical methods are able to achieve.

An algorithm combining TSVMs and graph-based distance is proposed in [12]. This graph-based distance has a close relation to the geodesic on smooth manifolds. Many graph-based methods explicitly or implicitly involve the low density separation idea, which will be discussed in detail later.

One way of understanding the TSVM algorithm is closely related to the low density separation assumption, which states that the class separation boundaries should lie among the low density regions, or samples of different classes are separated by large margin. TSVMs try to find a decision boundary that lies in low density regions.

A typical classification result of low density separation algorithms is shown in Figure 22.1 [16, Chapter 2]. The triangle and square are the only labeled points in a binary classification problem and all

**FIGURE 22.1**

Low density separation on two-moon data.

other black dots are unlabeled data. The true decision boundary are the low density regions between the two “moons.” Although the data are distributed in such a way that most supervised classifiers will fail, several low density separation based SSL algorithms, e.g., TSVMs, can take advantage of the unlabeled data and recover the decision boundary.

### 1.22.2.2 Co-training and multi-view

One of the most influential SSL algorithms, so-called co-training, was introduced [5]. The framework of co-training is representative for various types of multi-view learning approaches.

In co-training, the data consist of a small labeled data set and a large unlabeled set, which is the typical setting of SSL. However, in addition, each data sample has two distinct views. One typical example studied in [5] is that the description of a web page can be partitioned into the words occurring on that page, and the words occurring in hyperlinks that point to that page. Another example would be two different feature representations of an image, e.g., SIFT [17] and GIST [18]. Notice that the two views of the first example are more likely created by different people, suggesting higher degree of independence, while for the latter example the two views are created from the same data, in the hope of capturing their different aspects.

Formally, let the instance space be  $\mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$ , where  $\mathcal{X}^{(1)}$  and  $\mathcal{X}^{(2)}$  correspond to the two different “views” of an example. Typically, a data point  $x$  is represented by two different feature vectors,  $x^{(1)}$  and  $x^{(2)}$ , such that  $x^{(1)} \in \mathcal{R}^{d_1}$  and  $x^{(2)} \in \mathcal{R}^{d_2}$ . In this case, instance space  $\mathcal{X}$  is simply  $\mathcal{R}^{d_1+d_2}$ . In co-training, each view is assumed to be sufficient for correct classification in itself. Let  $\mathcal{D}$  be a distribution over the instance space  $\mathcal{X}$ , and let  $\mathcal{C}^{(1)}$  and  $\mathcal{C}^{(2)}$  be the concept classes defined over  $\mathcal{X}^{(1)}$  and  $\mathcal{X}^{(2)}$ , respectively.

The co-training framework proceeds as given below:

- a set  $X_L$  of labeled examples, with the corresponding label  $Y_L$ ;
- a set  $X_U$  of unlabeled examples.

Create a pool  $X_{U'}$  of examples by choosing  $u$  examples at random from  $X_U$ , then loop for  $k$  iterations:

1. Use  $(X_L, Y_L)$  to train a classifier  $h_1$  that considers only the  $x_1$  portion of  $x$ .
2. Use  $(X_L, Y_L)$  to train a classifier  $h_2$  that considers only the  $x_2$  portion of  $x$ .
3. Allow  $h_1$  to label  $p$  positive and  $n$  negative examples from  $X_{U'}$ .
4. Allow  $h_2$  to label  $p$  positive and  $n$  negative examples from  $X_{U'}$ .
5. Add these self-labeled examples to  $X_L$ .
6. Randomly choose  $2p + 2n$  examples from  $X_U$  to replenish  $X_{U'}$ .

In steps 3–5, the  $p$  positive and  $n$  negative examples are the ones with the most confident predictions. There are different variations on how to select candidates from predictions and how to add them to  $X_L$ . For example, “agreement” is a natural criteria to select candidates from two classifiers; only a small fraction of the examples with the most confident predictions are allowed to be added to  $X_L$ , hoping to reduce noise; active learning can also be added to step 5 to further improves the candidate quality. As will be discussed later, many multi-view SSL algorithms follow this co-training framework.

Two important assumptions of co-training are: (i) the *compatibility assumption* and (ii) the *independence assumption*.

The compatibility assumption provides an interesting take on SSL, saying that a target function  $f = (f^{(1)}, f^{(2)}) \in \mathcal{C}^{(1)} \times \mathcal{C}^{(2)}$  is “compatible” with  $\mathcal{D}$  if it satisfies the condition that  $\mathcal{D}$  assigns probability zero to the set of examples  $(x^{(1)}, x^{(2)})$  such that  $f^{(1)}(x^{(1)}) \neq f^{(2)}(x^{(2)})$ . It was pointed out in [5] that  $\mathcal{C}^{(1)}$  and  $\mathcal{C}^{(2)}$  can be large concept classes with high complexity, but for a given  $\mathcal{D}$  the set of compatible target concept classes might be much smaller, which in turn reduces the requirement of labeled samples for learning.

Following a PAC-style argument, combining these two assumptions leads to learnability guarantees.

Many different theoretical aspects and applications of multi-view learning are studied, see, e.g., [19, 20]. One key step in multi-view learning is how to deal with view disagreement, as a result of either the intrinsic error or noise. A multi-view learning approach that uses a conditional entropy criterion to detect view disagreement is proposed in [21]. Once detected, samples with view disagreement are filtered and standard multi-view learning methods can be easily applied to the remaining samples.

### 1.22.2.3 Co-regularization

Following the co-training framework, regularized regression in reproducing kernel Hilbert spaces (RKHSs) is studied in multi-view learning in [22]. This method is called co-regularization. As the name suggests, regularization on different views at the same time plays a key role in the method.

Consider the two views case, let data set  $X$  have two views  $\mathcal{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)}$ . In general, cases with more than two views follow a similar analysis. The classifier spaces for different views are two RKHSs

$\mathcal{H}^{(1)}$  and  $\mathcal{H}^{(2)}$ . Then the co-regularization problem is defined as follows:

$$\begin{aligned} (f^{(1)*}, f^{(2)*}) = & \arg \min_{f^{(1)} \in \mathcal{H}^{(1)}, f^{(2)} \in \mathcal{H}^{(2)}} \gamma_1 \|f^{(1)}\|_{\mathcal{H}^{(1)}}^2 + \gamma_2 \|f^{(2)}\|_{\mathcal{H}^{(2)}}^2 \\ & + \sum_{x_i^{(1)} \in X_L} [y_i - f^{(1)}(x_i^{(1)})]^2 + \mu \sum_{x_i^{(2)} \in X_L} [y_i - f^{(2)}(x_i^{(2)})]^2 \\ & + \gamma_c \sum_{x_i \in X_U} [f^{(1)}(x_i^{(1)}) - f^{(2)}(x_i^{(2)})]^2. \end{aligned} \quad (22.2)$$

The solutions can be found by solving a linear system of equations. See Ref. [22] for details. The solutions are turned into binary classifiers through the sign function. The first two RKHS norms measure the complexity of the solutions in each RKHS. The next two terms are the data fitting error in different spaces. The last term is the “co-regularization” term, or called co-regularizer. It forces the estimators in two spaces  $\mathcal{H}^{(1)}$  and  $\mathcal{H}^{(2)}$  to be close in an  $L^2$  sense, i.e., small square distance.

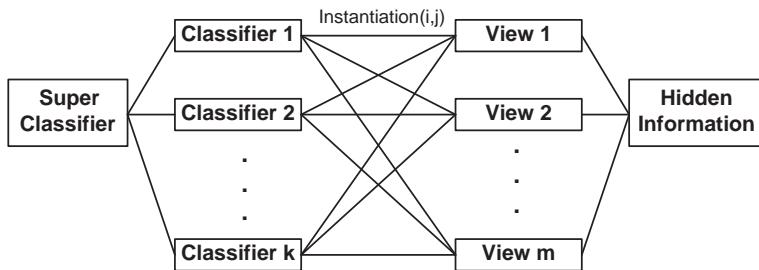
The theoretical properties of the co-regularization problem is studied in detail in [23]. It is shown that the solution space is also an RKHS, called “co-regularization RKHS.” With this result, the kernels can be found explicitly, since the reproducing kernels and the associated RKHS uniquely determine each other. For regularization problems in an RKHS, solutions can be found easily by the Representer Theorem, with the help of an explicit kernel function. Generalization bound based on Rademacher complexity is also given in [23].

Instead of studying each view separately first, then combining the results, co-regularization focuses on a single joint function space, which is generated from two individual function spaces. This provides a new understanding of the multi-view learning problem. The compatibility assumption of co-training can be studied more naturally and potentially easily in this single joint space. For instance, the agreement on unlabeled data by co-regularization is shown to reduce the function space complexity. By taking advantage of the single space view, the method and proof in [23] are elegant and insightful.

The co-regularizer studied in [23] can potentially transform any supervised learning algorithm to an SSL algorithm in a multi-view setting. This can be implemented by simply adding a co-regularizer term to the combined supervised learning problems in each view.

*The ensemble:* Compared to ensemble methods such as AdaBoost which studies how to combine different classifiers on the same data, multi-view learning studies how to combine different views of the same data. Therefore, there exists a natural many-to-many mapping from the entity of classifier to the entity of data. The mapping provides a flexible framework to integrate data and analysis methods, which is illustrated in Figure 22.2. An instantiation means the application of the  $i$ th family classifiers to the  $j$ th view of a data set. There are two “one-to-many” relations in this ensemble. For each specific family of classifiers, multi-view learning can be used to improve learning performance. This step can reduce the classifier space complexity using unlabeled data. The other “one-to-many” relation is that the “Super classifier” chooses a strategy to combined different classifiers on different views. These two steps can be designed and controlled carefully to produce powerful classifiers.

Co-training and multi-view methods provide a flexible and yet powerful framework in practice to combine different information sources and different classifiers. Notice that the specific classifiers do

**FIGURE 22.2**

The ensemble and multi-view.

not need to be from the same family. It is possible to combine artificial neural network in one view and SVM in another view. This can potentially explore the advantage of different classifier families on different views. However, the combination should be carefully controlled to avoid overfitting.

The ensemble of classifiers and data with multi-view has shown impressive performance in practice, e.g., see, Netflix competition grand prize winning algorithm and KDD Cup 2011 Track 1 winning algorithm “A Linear Ensemble of Individual and Blended Models for Music Rating Prediction,” which is a linear ensemble of more than 200 classifiers from different families.

#### 1.22.2.4 EM and self-train

Expectation-Maximization (EM) [24] is a general approach for iterative computation of maximum likelihood or maximum a posteriori estimation when the data is incomplete or have missing values.

First recall the definition of maximum likelihood estimation problem. Consider a probability density function  $p(x|\theta)$  which is governed by the set of parameter  $\theta$ . Assume the i.i.d. data samples  $X = \{x_1, x_2, \dots, x_n\}$  are drawn from  $p(x|\theta)$ . Therefore, the density for the sample set is

$$p(X|\theta) = \prod_{i=1}^n p(x_i|\theta) = \mathcal{L}(\theta|X). \quad (22.3)$$

The function  $\mathcal{L}(\theta|X)$  is called the likelihood function given the data  $X$ . It is a function of  $\theta$  for fixed data  $X$ . In a maximum likelihood problem, the goal is to find a  $\theta$  that maximizes  $\mathcal{L}$ :

$$\theta^* = \arg \max_{\theta} \mathcal{L}(\theta|X). \quad (22.4)$$

For certain family of probability density functions, it is possible to find  $\theta^*$  explicitly, e.g., a single Gaussian distribution. However, for many problems, it is difficult or even impossible to find an analytical expression for  $\theta^*$ . The EM algorithm is such a general method of finding the maximum likelihood estimate of the parameter  $\theta$ .

Let the complete data be  $Z = (X, Y)$ , where  $Y$  is either the missing data, or some hidden variables. Assume a joint density function:

$$p(z|\theta) = p(x, y|\theta) = p(y|x, \theta)p(x|\theta). \quad (22.5)$$

For example,  $y$  can be the cluster label in a clustering problem. The likelihood function for  $Z$  is  $\mathcal{L}(\theta|Z) = p(X, Y|\theta)$ . Due to the missing values, which are random variables,  $\mathcal{L}(\theta|Z)$  is in fact random.

The EM algorithm first finds the expected value of the log-likelihood  $\log p(X, Y|\theta)$  with respect to the unknown data  $Y$ , given the data  $X$  and the current parameter estimate  $\theta^{(t-1)}$ .

$$Q(\theta, \theta^{(t-1)}) = E[\log p(X, Y|\theta)|X, \theta^{(t-1)}]. \quad (22.6)$$

The computation of the expectation is called the *E-step*. The first argument  $\theta$  is the parameter to be optimized in the maximization of the likelihood function. The second argument  $\theta^{(t-1)}$  is the parameter from previous estimate. This means the expectation is taken with respect to the distribution determined by the data  $X$  and the density function with parameter  $\theta^{(t-1)}$ .

The second step is called the *M-step*, which is to maximize the expectation from the *E-step*:

$$\theta^{(t)} = \arg \max_{\theta} Q(\theta, \theta^{(t-1)}). \quad (22.7)$$

These two steps are repeated as necessary. Each iteration is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function.

This is a general framework of EM algorithm. For different problems with different families of probability density functions, the specific steps are different.

The setting of SSL is similar to the general EM setting: a subset of  $Y$  is observed but the values on unlabeled data are missing. Therefore, EM can be applied naturally to SSL. In EM SSL, generative models and certain assumptions are usually used to take advantage of the unlabeled data. For example, one of the earliest successful applications of EM to SSL is text classification [25], which estimates the maximum a posteriori using EM on mixture of multinomials.

In a generative model, it is assumed that data are generated by a parametric model, such as a Gaussian mixture or other exponential families, with unknown parameters. Maximum likelihood estimation using the EM algorithm can be used to find these parameters given the data. For a supervised maximum likelihood estimation problem, the goal is to maximize  $p(X, Y|\theta)$ , where  $X$  and  $Y$  are the data, and  $\theta$  is the parameter for the chosen model. In an SSL setting, the data are  $X_L$ ,  $Y_L$ , and  $X_U$ , therefore, the goal is the maximize  $p(X_L, Y_L, X_U|\theta)$ . The key step is to fill in the missing  $Y_U$ . EM algorithms for SSL include the following two typical steps:

- *E-step*:  $Y_U^{(t+1)} = E[Y_U|X_L, Y_L, X_U, \theta^{(t)}]$ ;
- *M-step*:  $\theta^{(t+1)} = \arg \max_{\theta} p(X_L, Y_L, X_U, Y_U^{(t+1)}|\theta)$ .

The *E-step* is to label the unlabeled data using a model learnt from labeled data and previous estimates. Then the *M-step* is to estimate the parameters by treating all the data as labeled ones as in supervised learning, where the labels of unlabeled data are the predictions from the model in *E-step*. These two steps are similar to *K*-means method, which has a different cost function.

The first question to ask, when using a generative approach, is whether the model is correct. When the chosen model provides a good description of the data, generative approach can be powerful. However, when the true data generating process is very different from the chosen generative model, unlabeled data cannot improvement and, in fact, can even hurt performance.

Another issue for EM in generative model is that, typically it is a non-convex problem with local maxima. Different heuristics such as active learning and random restart can be used to improve the solution.

### 1.22.2.4.1 EM for mixture model

EM algorithm together with mixture models is used frequently in many aspects of machine learning, including supervised, unsupervised and semi-supervised learning. Mixture model assumes that data are sampled from a mixture of more than one distributions with densities  $p_i$ , which are parameterized by  $\theta_i$ :

$$p(x|\Theta) = \sum_{i=1}^m \alpha_i p_i(x|\theta_i), \quad (22.8)$$

where  $\Theta = (\alpha_1, \dots, \alpha_m, \theta_1, \dots, \theta_m)$ , and  $\forall i, \alpha_i > 0, \sum_{i=1}^m \alpha_i = 1$ . This means the data are generated in two steps: first, choose probability density  $p_i$  with probability  $\alpha_i$ ; second, sample from  $p_i$ . The log-likelihood for sample  $X$  then become

$$\log(\mathcal{L}(\Theta|X)) = \log \prod_{i=1}^n p(x_i|\Theta) = \sum_{i=1}^n \log \left( \sum_{j=1}^m \alpha_j p_j(x_i|\theta_j) \right). \quad (22.9)$$

The form of a log of sum is difficult to optimize. The EM algorithm solves this problem by assuming hidden variables  $Y = \{y_i, i = 1, \dots, n\}$ . These hidden variables are the labels of components. For instance,  $y_i \in \{1, \dots, m\}$  and  $y_i = k$  means  $x_i$  is generated by the  $k$ th component with probability density  $p_k$ . If  $Y$  is given, which is the case in supervised learning, the likelihood function for both  $X$  and  $Y$  is

$$\log(\mathcal{L}(\Theta|X, Y)) = \log(p(X, Y|\Theta)) = \sum_{i=1}^n \log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})). \quad (22.10)$$

This likelihood function is of the form of a sum of log functions, which is much easier to optimize.

In clustering,  $Y$  is not observed, given a guess of  $\Theta^t$ , it is not difficult to find the distribution of the unobserved  $Y$ , which is

$$p(y_i|x_i, \Theta^t) = \frac{\alpha_{y_i}^t p_{y_i}(x_i|\theta_{y_i}^t)}{p(x_i|\Theta^t)} = \frac{\alpha_{y_i}^t p_{y_i}(x_i|\theta_{y_i}^t)}{\sum_{j=1}^m \alpha_j^t p_j(x_i|\theta_j^t)}. \quad (22.11)$$

Then

$$p(Y|X, \Theta^t) = \prod_{i=1}^n p(y_i|x_i, \Theta^t). \quad (22.12)$$

Given these explicit results, EM algorithm can proceed without difficulty. The estimated  $\alpha_i$  after  $M$ -step is

$$\alpha_k = \frac{1}{n} \sum_{i=1}^n p(k|x_i, \Theta^t). \quad (22.13)$$

This means, given a guess  $\Theta^t$ , the best estimate for  $\alpha$  is the sum of the probability of all samples belonging to that component. Similarly,  $\theta_i$  can also be obtained. Then we have another better guess  $\Theta^{(t+1)}$ . The EM algorithm repeats as necessary. See, e.g., Ref. [26] for more details on mixture of Gaussians and EM in Hidden Markov Model.

In SSL,  $Y$  is partially observed. Then the likelihood function is

$$\log (\mathcal{L}(\Theta|X_L, X_U, Y_L)) = \log (p(X_L, X_U, Y_L|\Theta)), \quad (22.14)$$

which includes two parts, a likelihood function on unlabeled data with hidden variables and a regular likelihood function on observed data.

#### 1.22.2.4.2 Value of unlabeled data

The value of unlabeled data in parametric models is first studied in [1–3], where  $l$  labeled and  $u$  unlabeled samples were used together for learning. Let  $R(l, u)$  denote the optimal probability of error for  $l$  labeled and  $u$  unlabeled samples, the main results under mild condition in [1,2] are

$$R(0, u) = R(0, \infty) = \frac{1}{2}, \quad (22.15)$$

meaning that the unlabeled sample does not help without labeled points, and

$$R(l, \infty) = R^* + \exp(-\alpha l + o(l)), \quad (22.16)$$

where  $R^*$  is the optimal Bayes error, and  $\alpha$  depends on the weights of the mixture and each mixture density function. This means the labeled sample set reduces the probability of error exponentially fast to the Bayes risk. In a sequel, the value of unlabeled sample set is studied. The main result is that

$$R(l, u) = R^* + O(u^{-1}) + \exp(-\alpha l + o(l)). \quad (22.17)$$

This shows that the unlabeled set reduces the probability of error at the rate of  $u^{-1}$ , a much slower rate compared to labeled data.

A binary classification problem under the Gaussian mixture model is studied in [3]. Given finite labeled and unlabeled data, the goal is to estimate the Gaussian parameters and the mixture weights. The misclassification probability is shown to deviate from the minimal Bayes error rate by  $O(d^{3/5}u^{-1/5}) + O(e^{-cl})$ , where  $d$  is the dimensionality of the Gaussian and  $c$  is a positive constant.

#### 1.22.2.4.3 Self-training

In SSL, the two steps of EM algorithm can also be seen as labeling unlabeled examples first, then use the estimated value to train the classifier again. The process is repeated until certain criteria is met. This type of algorithm is also called self-training. Other relevant names include “bootstrap learning” and “automated labeling.” One of the earliest works on self-training is the application to the word sense disambiguation problem [27]. Instead of using all the estimates from the  $E$ -step in the  $M$ -step in this work, a subset of the result with higher “confidence” is selected and added to the labeled set. Similarly, active learning can also be used in this step to select different subsets to augment the labeled set. Co-training and multi-view methods also explore the idea of augmentation of a small labeled set in SSL, but from a different problem setting.

#### 1.22.2.5 Graph-based SSL

Graph-based SSL is among the most popular SSL methods. As the name suggests, graph plays an important role in this family of algorithms. The data in these problems appear in the form of a weighted

graph. This can be further categorized into two cases. First, the data itself is a weighted graph (directed or undirected), such as various social networks. Second, data are in the form of i.i.d. samples from certain domains other than graphs, then the data samples are converted into the weighted graph forms.

There are diverse algorithms for graph-based SSL methods, with plenty of variations and intuitions, such as, random walk on graphs [28–30], graph cut [31], electronic networks [32], harmonic functions [33,34], Green’s functions [35,36], regularization and reproducing kernels [10], etc. Among these algorithms, some are transductive, meaning the predictions are only on the unlabeled data. In this case, another step is needed for prediction on out-of-sample unseen examples: either add the unseen examples to the unlabeled data set and solve the problem again, or use another step to approximate the prediction, see, e.g., [37]. Others are inductive, meaning the algorithm output an estimator on the whole domain, e.g., [10]. See [8] for more reviews.

Some graph-based SSL algorithms are proposed as a discrete approximation to the corresponding continuous problem, while others are proposed for discrete graphs directly. This leads to an issue in the infinite sample analysis. Due to different problem settings and assumptions, certain graph-based SSL algorithms might not have a meaningful limit in the limit of infinite samples, either labeled or unlabeled. For instance, if the graph is the Internet graph or other social networks, then strictly speaking, even the “limit of infinite sample” is not well defined.

In this section, the focus is a family of graph Laplacian-based SSL methods whose limits are well defined under certain mild conditions. This family includes a large number of different algorithms, most of which fall under the same framework from a functional analysis view. The typical setting of these methods is as follows. The data samples  $x_i$  are from a smooth unknown probability density  $p(x)$  supported on a certain smooth subset  $\Omega$  of  $\mathcal{R}^d$ . This includes the cases when the domain is a low-dimensional smooth submanifold. Notice that most algorithms in this family can still be applied to finite graph data or samples from domains other than  $\mathcal{R}^d$ , such as unit hypercube  $\{0, 1\}^d$  or sequences in natural languages given finite data. However, the limit analysis and interpretation with infinite samples for algorithms on these data will be different, when the limits do exist.

For i.i.d. samples from a subset of  $\mathcal{R}^d$ , the graph construction proceeds as follows. Any sample point  $x_i \in X = X_L \cup X_U$  is mapped into a vertex of a weighted graph  $G(V, E)$ , and the edge weight  $w_{ij}$  is a similarity weight for each pair of example  $x_i$  and  $x_j$  defined by certain weight function on the edge, e.g., Gaussian weight  $w_{ij} = e^{-\|x_i - x_j\|^2/t}$  with parameter  $t$ . In directed graph, the weight is asymmetric. Graph  $G$  with vertex set  $V = X$  and weighted edge set  $E$  acts as a portal to the complex data source. An example is that, the local neighborhood graph constructed from uniform samples on a unit sphere acts as the approximate unit sphere.

### 1.22.2.5.1 Graph Laplacian regularization

Given sample sets  $X_L$ ,  $X_U$ , and  $Y_L$ , one popular family of graph-based SSL method is called the graph Laplacian regularization, which solves a regularized least squares problem as the following:

$$\min_{f \in \mathcal{R}^n} \sum_{i=1}^l (f(x_i) - y_i)^2 + \frac{\mu}{n^2} \sum_{i,j=1}^n w_{ij} (f(x_i) - f(x_j))^2, \quad (22.18)$$

where  $n = l + u$  is the total number of examples,  $w_{ij}$  is a “similarity” value for example  $x_i$  and  $x_j$ , and  $\mu$  is a weight balancing the fidelity to the observations on the labeled data and the second penalty

term. Notice that  $f$  is a vector in  $\mathcal{R}^n$ , and  $f(x_i) = f_i$ . This notation is convenient for the limit analysis, emphasizing that vector  $f$  is a projection of a continuous function on the sample set. Several different algorithms belong to this family of algorithm, see, e.g., [8].

For an inverse problem, there are possibly infinite functions that have zero cost. The second penalty term provides a preference to certain functions. Consider the following typical weight function:

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{t}\right), \quad (22.19)$$

where the closer  $x_i$  and  $x_j$  are, the larger  $w_{ij}$  is. For data points that are far apart, the weight is close to zero. This means the function value differences in the penalty term of the optimization problem (22.18) only contribute to the summation for all data pairs that are near each other. The minimization drives the solution to have small difference for nearby data points, implying that the solution is smooth *locally*. On the other hand, there is little or no penalty for points that are far apart.

The quadratic term can be rewritten as the following matrix form using the graph Laplacian  $L$ :

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (f(x_i) - f(x_j))^2. \quad (22.20)$$

The graph Laplacian matrix  $L$  is defined as follows. Given a graph  $G(V, E)$  with weight matrix  $W$ , and its diagonal degree matrix  $D$  defined as  $D(i, i) = \sum_j W(i, j)$ , where  $W(i, j)$  is the element at the  $i$ th row and  $j$ th column in matrix  $W$ , the commonly used graph Laplacian is defined as the following:

$$L = D - W. \quad (22.21)$$

$L$  is also called the *unnormalized graph Laplacian*, or the *combinatorial Laplacian* for 0/1 weights. There are other version of graph Laplacians, see, e.g., [38, 39].

Then the graph Laplacian regularization problem (22.18) can be rewritten as

$$\min_{f \in \mathcal{R}^n} \sum_{i=1}^l (f(x_i) - y_i)^2 + \frac{\mu}{n^2} f^T L f, \quad (22.22)$$

where the coefficient  $\frac{1}{2}$  is ignored. The solution using pseudoinverse can be found to be

$$\hat{f} = \left(S + \frac{\mu}{n^2} L\right)^+ S Y. \quad (22.23)$$

$Y$  is a column vector with  $Y(i) = y_i$  for  $x_i \in X_L$  and  $Y(i) = 0$  for  $x_i \in X_U$ ,  $S = \text{Diag}(1, \dots, 1, 0, \dots, 0)$  with the first  $l$  diagonal entries as 1 and the rest 0.

Several variations of the problem (22.22) have been proposed. An interpolation problem, instead of a “soft” regularization problem is introduced in [34]. The same regularization using a symmetric normalized version of graph Laplacian is proposed in [40]. Regularization based on the spectrum of the Laplacian is proposed in [33]. Tikhonov regularization interpretation and generalization error are discussed in [41]. The quadratic form  $f^T L f$  and its variations are frequently used in many other machine learning areas. See [8] for more variations.

When data is from the Euclidean space  $\mathcal{R}^d$ , and for a fixed function  $f \in C^2(\mathcal{R}^d)$ , the following limit of the quadratic form was studied in [42] as the number of data examples increases to infinity and

the weight function adjusts accordingly

$$f^T L f \xrightarrow{p} c \int \|\nabla f\|^2 p^2(x) dx, \quad (22.24)$$

where  $c$  is a constant depending on the weight function in the definition of graph Laplacian,  $p(x)$  is the probability density of the random sample, and  $\|\nabla f\|^2$  is the gradient square.

Notice that the limit analysis is based on the assumption that, the graph  $G$  is built on the data sample  $X$  from some distribution in  $\mathcal{R}^d$ . If graph  $G$  does not have this interpretation, e.g., some non-Euclidean space with other distance measure, the limit might not even exist, and the interpretation can be very different even the limit does exist.

The graph-based SSL problem in (22.22) then can be seen as an empirical version of the continuous problem

$$\min_f \sum_{i=1}^l (f(x_i) - y_i)^2 + \mu \int \|\nabla f\|^2 p^2(x) dx. \quad (22.25)$$

Notice that the uniform convergence of solutions on certain smooth function space is needed to show a rigorous equivalence.

Various explanations have been introduced to help the understanding of the graph Laplacian regularization. For example, random walk on graphs is one of the most popular ones [34, 40]. On the other hand, random walk is also used to argue why the method fails in the limit of infinite unlabeled data points in high dimensions [43]. The seemly contradictory views might be better understood from another context: there are pairwise equivalences between random walk (or diffusion), Laplace equation, and energy minimization problems. The Laplace equation governs a diffusion process, which is a random walk in finite sample case; the minimizer of the potential energy  $\int \|\nabla f\|^2 dx$  is the solution of a Laplace equation, which is a harmonic function. See, e.g., [44] for an application in the background of a boundary value problem. This view connects many different variants of the graph Laplacian-based SSL methods.

An alternative view to the graph Laplacian regularization is the reproducing kernel Hilbert space (RKHS) view. Since the graph Laplacian has a zero-valued eigenvalue, it has a null space spanned by the associated eigenvector. For instance, for the unnormalized graph Laplacian  $L = D - W$ , the null space is spanned by the constant eigenvector. The regularizer  $f^T L f$  in fact is an RKHS norm in the subspace that is orthogonal to the null space of  $L$ , see, e.g., [45, Chapter 6] or [35]. The reproducing kernel matrix is the pseudoinverse of  $L$ . This means, by the Representer Theorem, the solution of the graph Laplacian regularization is in the form of a summation of the rows of the kernel matrix.

These functional aspect views provide more insights to the graph Laplacian regularization problem, going beyond the random walk intuition. The idea is further explored in detail next.

### 1.22.2.5.2 Iterated graph Laplacian regularization

The setting of a small labeled set with a large unlabeled set is unique to SSL. This involves two factors in the large sample analysis of SSL: the limit analysis of both the labeled and unlabeled samples. Considering that the unlabeled data are generated much faster than the high-quality labeling process, the limit analysis of algorithms in the setting of a limited labeled set together with infinite unlabeled samples is attractive in practice.

In this SSL setting, the more unlabeled data become available to an SSL algorithm, the better prediction it should produce, according to the philosophy of SSL. However, this may not be the case

for certain SSL algorithms. For example, consider the SSL problem in Eq. (22.22), when the labeled set is fixed, the more unlabeled data involve in the optimization problem, the worse the classification results become, as shown in [43]. The graph Laplacian regularization only works in one-dimensional spaces with infinite unlabeled data. On domains with dimensions higher than 1, the solution will overfit labeled data perfectly. The solution is essentially a signed indicator function of labeled data for binary classifications. This shows there is no generalization in the limit. In practice, the overfitting causes two serious problems: first, the solution shifts to the positive or negative side, which depends greatly on the few labeled data examples; second, the solution is extremely close to zero. These two aspects make the graph Laplacian regularization method numerically unstable.

The reason for this problem is analyzed from a functional analysis point of view in [36]. The reason of the overfitting is that the solution space is too rich. With infinite unlabeled data, the graph Laplacian regularizer becomes the gradient square norm as

$$\int \|\nabla f(x)\|^2 dx, \quad (22.26)$$

where the probability density term  $p^2(x)$  is not important and ignored when it is bounded from below. This norm in fact is one of the Sobolev norms, see, e.g., [46].

The solution to the overfitting problem is solved in [36] by essentially shrinking the solution space. This can be implemented through a higher order Sobolev norm. The norm used in [36] is called the iterated graph Laplacian semi-norm, defined as the following:

$$f^T L^m f, \quad (22.27)$$

where  $m$  is a positive integer for computational purpose, though  $m$  can also be real numbers. The motivation is that the iterated Laplacian  $\Delta^m$  operator is a natural object associated with Sobolev spaces, and in particular, Sobolev spaces on smooth Riemannian submanifolds, see, e.g., [47, Chapter 5]. When  $m = 1$ , it becomes the regular Laplacian, which in  $\mathcal{R}^d$  is defined as

$$\Delta = \sum_{i=1}^d \frac{\partial^2}{\partial x_i^2}. \quad (22.28)$$

By connecting the power of the graph Laplacian matrix and the iterated Laplacian operator, the following iterated graph Laplacian regularization generally provides better solutions than the regular graph Laplacian regularization, as shown empirically in [36]:

$$\min_{f \in \mathcal{R}^n} \sum_{i=1}^l (f(x_i) - y_i)^2 + \mu_n f^T L^m f, \quad (22.29)$$

where  $\mu_n$  depends on  $n$  and the weight function.

The spectral transform of the graph Laplacian in finite sample cases has been discussed and tested in [10, 33, 35]; the limit is analyzed in [36]. Sobolev embedding theorem [46] provides the condition of the spectral transform for the graph Laplacian to give a smooth solution.

The spectral transform requirement is also closely related to RKHS view. In the finite sample case, the norm  $f^T L f$  corresponds to an RKHS, while in the limit of infinite unlabeled data, the Sobolev

normed space may not be an RKHS, depending on the dimensionality of the underlying data domain. Specifically,  $2m > d$  needs to hold in order to make sure the Sobolev space is an RKHS.  $d$  should be replaced with the intrinsic dimensionality when the data domain is a low-dimensional manifold. In non-RKHS cases, the Green's function is a useful tool to study the problem.

*Finite sample case:* In finite data cases, we always have an RKHS since any finite-dimensional Hilbert space is an RKHS, see, e.g., [45]. This means the discrete Green's function is the same as reproducing kernel in the subspace orthogonal to its null. Let the kernel matrix be  $K$  and the discrete Green's function matrix be  $G$ , then for semi-norm  $f^T L f$ , the reproducing kernel in the subspace orthogonal to its null is the pseudoinverse of matrix  $L$ , i.e.,  $K = L^+$  [45, Chapter 6]. Notice that the exact kernel for the semi-norm includes another kernel in its null space. The discrete Green's function should satisfy  $GL = I$  and  $LG = I$ , which implies that  $G = L^+$ . This is also true for symmetric semi-definite matrix  $L^m$ . We can write both discrete Green's function and reproducing kernel for  $L^m$  by eigenfunction expansion as

$$G_m(x, y) = K_m(x, y) = (L^m)^+(x, y) = \sum_{k=2}^n \frac{1}{(\lambda_k)^m} \phi_k(x) \phi_k(y), \quad (22.30)$$

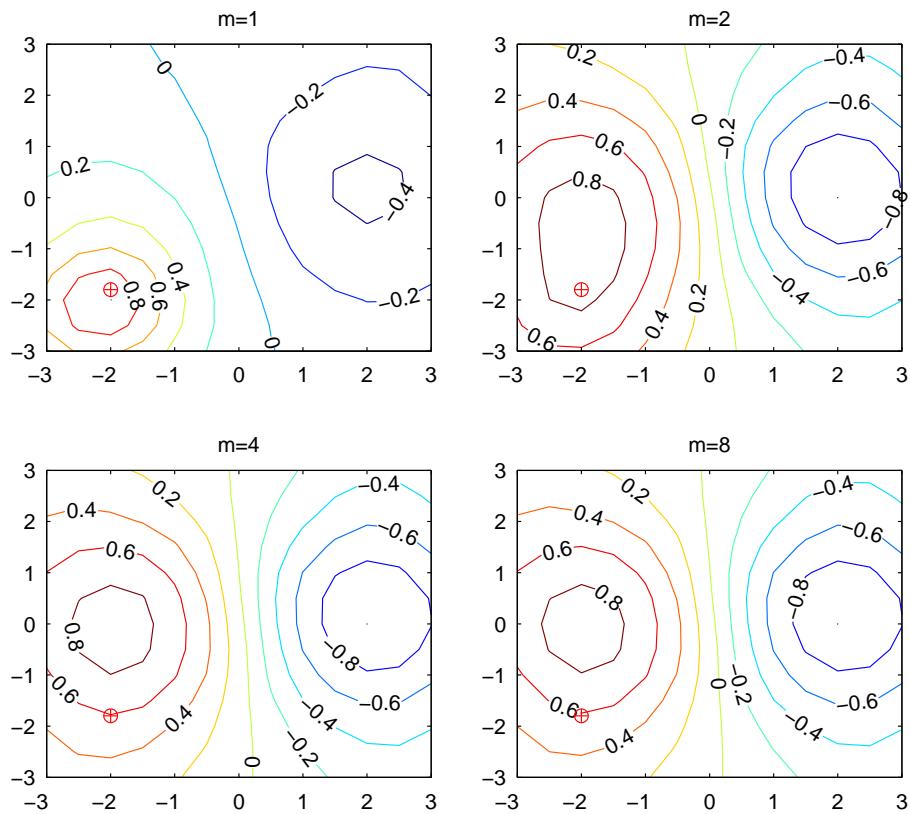
where  $\lambda_k$  and  $\phi_k(x)$  are the  $k$ th eigenvalue and eigenvector of  $L$ .  $\phi_k(x)$  means the  $x$  element of vector  $\phi_k$ . We can see that for a positive integer  $m$ , compared to  $\lambda_n$ , the smaller  $\lambda_k$  is, the larger  $1/\lambda_k^m$  will be relatively. This means  $G_m(x, y)$  will become smoother and smoother as  $m$  increases, since for graph Laplacians, the smaller  $\lambda_k$  is, the smoother the associated eigenvector is, and  $\lambda_k$  are in increasing order.

In order to have an intuitive understanding of the effect of  $m$ , we generate an artificial data set and plot the kernel functions in Figure 22.3. The data set includes a mixture of two Gaussians of unit variance on  $\mathcal{R}^2$  centered at  $(\pm 1.5, 0)$ , and an additional uniform over  $[-3, 3] \times [-3, 3]$  to avoid empty regions. In Figure 22.3, we show the discrete Green's function corresponding to  $L^m$  for two Gaussians with unit variance in  $\mathcal{R}^2$  at  $(-2, -1.8)$ . When  $m$  increases, the Green's function or reproducing kernel “grows” from “spikes” to smooth functions. From the contour plot, even the location of the kernel is not near the means of the Gaussians, when  $m$  increases, the kernel function recovers the true boundary of the two Gaussians. In fact, as long as the centers of kernel functions are in relatively high density regions of the Gaussians, a proper  $m$  value will produce a kernel function which changes little within those regions. This is exactly the basis for several graph Laplacian-based SSL algorithms.

Compared to the thin plate splines [48], the iterated Laplacian regularization can be viewed as a generalization of the thin plate splines from regular domains to unknown submanifolds, from a coordinate dependent Sobolev semi-norm defined by partial derivatives to a coordinate free iterated Laplacian semi-norm using Laplacians, from data independent reproducing kernels to data dependent kernels. One key difference is the null space between the two methods. The null space of  $f^T L^m f$  is spanned only by the first eigenvector, while in thin plate splines the null space is spanned by polynomials of high degrees, whose dimension increases fast as  $d$  and  $m$  increase.

### 1.22.2.5.3 Finite-dimensional approximation

As regularization is a solution for an ill-posed inverse problem, the finite-dimensional approximation is also an effective solution to an infinite-dimensional problem. Consider the graph Laplacian regularization problem (22.22), the solution lives in the span of all the eigenvectors of the graph Laplacian  $L$ . Instead of using all the eigenvectors, a subset of these eigenvectors are chosen and used in [33].

**FIGURE 22.3**

The Green's functions at  $(-2, -1.8)$  for a mixture of two Gaussians.

For a graph Laplacian  $L$ , whose limit is the Laplace operator in  $\mathcal{R}^d$  or Laplacian-Beltrami operator on a submanifold [49], the eigenfunctions form an  $L^2$  basis of square integrable functions. Let  $\phi_i$  and  $\lambda_i$  be the  $i$ th eigenfunction and eigenvalue of the Laplace operator  $\Delta$ , i.e.,  $\Delta\phi_i = \lambda_i\phi_i$ , then any  $L^2$  function  $f$  can be written as

$$f = \sum_{i=1}^{\infty} \alpha_i \phi_i, \quad \text{s.t. } \sum_{i=1}^{\infty} \alpha_i^2 < \infty. \quad (22.31)$$

Assume the eigenvalue and eigenfunctions are ordered in increasing-eigenvalue order. Then the finite-dimensional approximation problem is

$$\min_{\alpha_1, \dots, \alpha_k} \sum_{i=1}^l \left( y_i - \sum_{j=1}^k \alpha_j \phi_j(x_i) \right)^2, \quad (22.32)$$

where  $k$  is a fixed number that can be chosen by validation. This problem is also called Laplacian eigenmaps SSL, since these eigenvectors define the Laplacian eigenmaps [50]. The least squares solution can easily be written down explicitly. Notice that the squares error is only computed on the labeled data set, while the function basis  $\phi_i$  are computed on all the data points. This is a key step for SSL. Function basis can be computed without any label information. These basis are only supported on data samples, which has both pros and cons. One advantage is that they reduce the function class complexity, e.g., on intrinsic spaces; one disadvantage is that they are not defined for out-of-sample data. In finite sample case, the graph Laplacian eigenvectors are used instead of the continuous Laplacian eigenfunctions.

Another important advantage of using these eigenvectors is that they have an intrinsic smoothness ordering, measured by the eigenvalue. By multiplying  $\phi_i(x)$  to both side of  $\Delta\phi_i(x) = \lambda_i\phi_i(x)$ , and integrating both sides, the following relation can be found:

$$\int \phi_i(x)\Delta\phi_i(x)dx = \lambda_i \int \phi_i^2(x)dx = \lambda_i, \quad (22.33)$$

since the eigenfunctions are orthonormal. With proper boundary conditions for the Laplace operator  $\Delta$ , the left-hand side is

$$\int \phi_i(x)\Delta\phi_i(x)dx = \int \|\nabla\phi_i(x)\|^2 dx. \quad (22.34)$$

This means the smaller the eigenvalue is, the smoother the corresponding eigenfunction is, in the sense of gradient squares norm (one type of Sobolev norm). For instance, for the graph Laplacian  $\lambda_1 = 0$ , thus the gradient squares integral should be 0. This is consistent with fact that the first eigenvector of the graph Laplacian is constant. This nice property provides a key step for the asymptotic error analysis of the method, and also provide a flexible guidance for choosing eigenvectors in practice.

The optimal number of eigenfunctions and optimal integrated mean squares errors for the regression problem (22.32) is studied in [51]. In the limit of infinite unlabeled data, the error analysis shows that, when the true regression functions are differentiable, the optimal number of eigenvectors and the optimal integrated mean square error (IMSE) given  $l$  labeled and infinite unlabeled points are

$$k^* \sim \tilde{O}(l^{\frac{d}{d+2}}), \quad (22.35)$$

$$\text{IMSE}^* \sim \tilde{O}(l^{-\frac{2}{d+2}}), \quad (22.36)$$

where  $d$  is the intrinsic dimensionality of the domain, and the notation  $\tilde{O}$  includes a log factor. The results are further generalized to  $m$ -times differentiable functions.

Analysis in [51] also shows that Laplacian Eigenmaps SSL achieves the same error rate using the same optimal dimensions compared to least squares estimates using tensor product spline spaces with equidistant knots [52, Chapter 15.3], and the same rate as local polynomial regression [53].

Notice that the result is in a distribution-free setting. This means in cases when the cluster assumption does not hold, or there is no relationships between the marginal distribution of data  $X$  and its label  $Y$ , unlabeled data still can help learning asymptotically. This is achieved by reducing the function class complexity, which in turn improves learning rate.

#### 1.22.2.5.4 Manifold regularization

Graph Laplacian regularization in problem (22.22) is transductive. It cannot make prediction on out-of-sample data. Manifold regularization [10] is introduced to generalize the graph Laplacian regularization

to inductive inference. The framework introduced a novel regularizer combining the inductive ability and the geometry of the data domain. Manifold regularization includes a family of learning algorithms. Variants generalizing SVM and regularized least square are proposed in [10]. The regularized least square version of the inductive SSL problem is as follows:

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^l (f(x_i) - y_i)^2 + \mu_n \|f\|_K + \gamma_n \|f\|_I, \quad (22.37)$$

where  $\|f\|_K$  is a norm of  $f$  in a chosen reproducing kernel Hilbert space  $\mathcal{H}_K$ ,  $\|f\|_I$  is a norm depending on the intrinsic geometry of the data, e.g.,  $\|f\|_I = f^T L f$ , and  $\mu_n$  and  $\gamma_n$  are parameters to balance each term in the objective function.

A Representer Theorem to problem (22.37) when  $\|f\|_I = f^T L f$  is given as

$$\hat{f}(x) = \sum_{i=1}^{l+u} \alpha_i K(x_i, x). \quad (22.38)$$

The ambient space norm  $\|f\|_K$  provides the inductive ability of this algorithm, while the graph Laplacian regularizer term provides data dependent regularization. For example, RBF Kernels or polynomial kernels can be used as discussed in [10].

Manifold regularization connects spectral graph theory, manifold learning, and RKSH together in one regularization framework. One of the important advantages of the manifold regularization is the induction ability, which most other graph Laplacian regularization methods do not enjoy. The other feature is the regularization framework, i.e., an intrinsic norm plus an extrinsic norm. The combined regularization provides much flexibility from induction in the ambient space to the data dependent domain. More importantly, the problem can be easily solved with the help of the Representer Theorem.

Iterated Laplacian and Sobolev norms can also be used in the manifold regularization framework by simply replacing  $\|f\|_I$  with  $f^T L^m f$ , as suggested in [10, 36]. More Sobolev norms examples can be found in, e.g., [45]. Manifold regularization is further studied in the setting of co-regularization in [23], where different norms are treated to be in different views.

#### 1.22.2.5.5 Measure-based regularization

A density based regularization method is proposed in [42] with the following regularizer:

$$\int \langle \nabla f(x), \nabla f(x) \rangle p(x) dx = \int \|\nabla f(x)\|^2 p(x) dx, \quad (22.39)$$

where  $\nabla f(x)$  is the gradient vector and  $p(x)$  is the probability density of samples. The goal is to use a gradient based regularizer that penalizes variations of the function more in high density regions and less in low density regions. However, the kernel associated with this norm is not straightforward to obtain. As an alternative solution, the gradients of functions in the span of a chosen basis and the density  $p(x)$  are estimated directly. The experiments on real world data sets in [42] are not successful.

There are several difficulties in this method. First, density estimation is difficult in high dimensions. Although unlabeled data points are abundant in SSL, the computation is expensive. Second, the gradient estimated in [42] is in ambient spaces. If data live on or near a smooth submanifold, this gradient can be irrelevant to learning, which is first discussed in [10].

With the help of limit analysis of graph Laplacians and the graph Laplacian regularizer [49, 54, 55] and the two-step normalized graph Laplacian, next we show a simple method that solves both of the above problems.

*Two-step graph Laplacian:* Besides the commonly used graph Laplacians  $L = D - W$  and its variants, there is another family of more general “two-step” graph Laplacians, which is introduced by Coifman and Lafon [54] and further studied by Hein et al. [49]. In this definition, one first normalizes the weight matrix  $W$  as follows:

$$W_\alpha = D^{-\alpha} W D^{-\alpha}, \quad \alpha \in \mathcal{R}. \quad (22.40)$$

Then the element in the matrix, or the normalized weight function is

$$W_\alpha(x_i, x_j) = W_\alpha(i, j) = \frac{W(i, j)}{[D(i, i)D(j, j)]^\alpha}. \quad (22.41)$$

Given this new normalized weight matrix  $W_\alpha$ , one can define empirical Laplacians accordingly. For example,  $L^\alpha = D^\alpha - W^\alpha$ , where we use the superscript  $\alpha$  to emphasize the normalization parameter for the weight matrix. This should not be confused with the power of a matrix. Notice that when  $\alpha = 0$ , these two-step graph Laplacians become the usual ones. In particular, when  $\alpha = \frac{1}{2}$ , with proper scaling the following limit exists [55]:

$$f^T L^\alpha f \xrightarrow{p} \int \|\nabla f(x)\|^2 p(x) dx. \quad (22.42)$$

The gradient here is automatically computed on the data manifold when the data are sampled from a smooth manifold. The density term is implicitly encoded in the two-step normalized graph Laplacian. The reproducing kernel can also be found as the pseudoinverse of  $L^\alpha$ .

The measure associated with the two-step graph Laplacian with different  $\alpha$  brings another issue in using the graph Laplacian. For instance, for the commonly used unnormalized graph Laplacian ( $\alpha = 0$ ), the limit is as follows [42]:

$$f^T L f \xrightarrow{p} \int \|\nabla f(x)\|^2 p^2(x) dx. \quad (22.43)$$

The discrepancy between the measure for the regularizer and the probability of the sample needs more investigations.

### 1.22.2.5.6 Discussions

Graph-based SSL is closely related to two other areas: wavelet on graphs and kriging. Multiresolution analysis and wavelet have been a successful tool for low-dimensional data analysis such as time series and images. With the rising of graph data and high-dimensional data, the multiscale concept can be particularly powerful. One reason is that the concept of location and scale are more abstract and also more important in high-dimensional analysis. This include cases where data are not from a Euclidean space, e.g., social networks. However, the methods and theories for wavelet for graph and high-dimensional data are far behind their counterparts in low-dimensional data analysis. Several recent works in this area are reviewed next, including the discussions of their applications in SSL.

The basis of multiresolution analysis is a hierarchy structure of function spaces. One natural choice is the Sobolev spaces associated with iterated Laplacian. There are a family of generalized function

spaces associated with iterated Laplacian which are closely related to Sobolev spaces, see, e.g., [47, Chapters 4 and 5]. Iterated Laplacian is used in [36] to solve an illness problem of graph Laplacian regularization in a limit setting.

A novel method is introduced in [56] for constructing wavelet transforms of functions defined on the vertices of an arbitrary finite weighted graph. The approach is based on defining scaling using the graph analog of the Fourier domain, with the help of the graph Laplacian.

Diffusion wavelet [57] is another technique which is closely related to graph Laplacian. The method is built on a diffusion operator  $T$ . Functions at different scale corresponds to the diffusion process at different time. The main assumption is reduction of the numerical ranks as we take powers of the operator  $T$ .

Compared to graph spectral wavelet [56], the basis in diffusion wavelet are orthogonal. The orthogonalization step in the construction of wavelets destroys the compact support of the basis. Thus these basis are functions defined over the whole data domain. This might be preferable in SSL since global basis can be used to fit labeled data and make prediction on unlabeled data at the same time. On the other hand, wavelets in [56] are not orthogonal, but have compact support.

Haar-like wavelet is proposed in [58] for tree and graph data. One advantage of the method is that the wavelet coefficients decay exponentially fast, which encourages sparse representation of functions.

Another interesting view to several graph-based SSL algorithms is through kriging, which is a geostatistical technique to interpolate the value of a random field at an unobserved location from observations of its value at nearby locations. This idea resembles transductive SSL problems. In [59], several popular graph Laplacian regularization solutions are shown to be equivalent to kriging predictors with a fixed covariance matrix, which is determined by the data sample density and the geometry of the data domain. Kriging view brings more attention to the correlation among data samples, which is closely related to reproducing kernels and function space view.

Kriging provides an alternative way of interpreting graph Laplacian regularization in finite sample case, and also brings more insight under the Gaussian process framework. These results are consistent with the function analysis view presented earlier.

The kriging view shows that many different graph Laplacian regularization problems use a covariance matrix that is a given function of the graph adjacency matrix  $W$ . This might be a reasonable choice if the vertices of a graph are i.i.d. samples from certain subset of  $\mathcal{R}^d$ , e.g., the surface of a sphere. However, for other types of graphs that do not correspond to i.i.d. samples, e.g., social network graphs, the graph structure itself might not be enough to capture the correlations. One key step toward this covariance matrix design is discussed in [59]. Instead of only using the adjacency matrix  $W$  from data samples, response values are also used in the matrix design. Specifically, the covariance matrix consists of two component: one is stationary based on the vertex similarity, which comes from  $W$ , while the second part depends on vertex and need not to be stationary.

### 1.22.3 Semi-supervised learning for structured outputs

Complex output structures arise in speech recognition, various forms of object recognition tasks, and also in a wide range of natural language tasks such as Part of Speech (POS) tagging and parsing, just to name a few. Different supervised algorithms have been proposed to take advantage of the internal structure through Markovian or graphical model assumptions. Compared to SSL in vector or

scaled-valued function learning tasks, there is less progress in SSL with structured data. For structured data, the labeling process is even more expensive, which makes SSL more attractive in these tasks.

When each sample point  $x_i \in X$  and each label  $y_i \in Y$  are not vectors or scalars, but are sequences, e.g.,  $x_i$  is a sentence, and  $y_i$  is the Part of Speech (POS) tag, or even more complicated objects such as trees, the learning problem is called structured learning. The task is to learn a mapping  $f : \mathcal{X} \rightarrow \mathcal{Y}$ . For each  $x$ , let  $\mathcal{Y}(x)$  be the possible output set, which is finite in most practical problems.

A standard framework of learning structured models is by developing functions  $h : \mathcal{X} \rightarrow \mathcal{Y}$  as

$$h(x) = \arg \max_{y \in \mathcal{Y}(x)} F(x, y), \quad (22.44)$$

where  $F$  is a scoring function for each pair. For computational tractability and feasibility,  $F$  can be broken into scores that are defined on individual positions of the output sequence. This is also done to take advantage of the Markovian dependence that is generally assumed over the output sequences.

The unique feature for structured learning problem is how to model the structure/dependency among elements of data instance (both  $x$  and  $y$ ). With proper models for these dependencies, many SSL algorithms can be applied to the structured output problems. For instance, by defining a graph-based kernel for structured learning problems, kernel-based SSL methods can be applied directly to structured learning. See, e.g., [60, 61]. Manifold regularization for structured learning is discussed by Altun et al. [62]. Brefeld and Scheffer [63] studied a multiple-view regularizer for structured SVM in SSL. More references on structured SSL can be found in [64–71].

## 1.22.4 Large scale semi-supervised learning

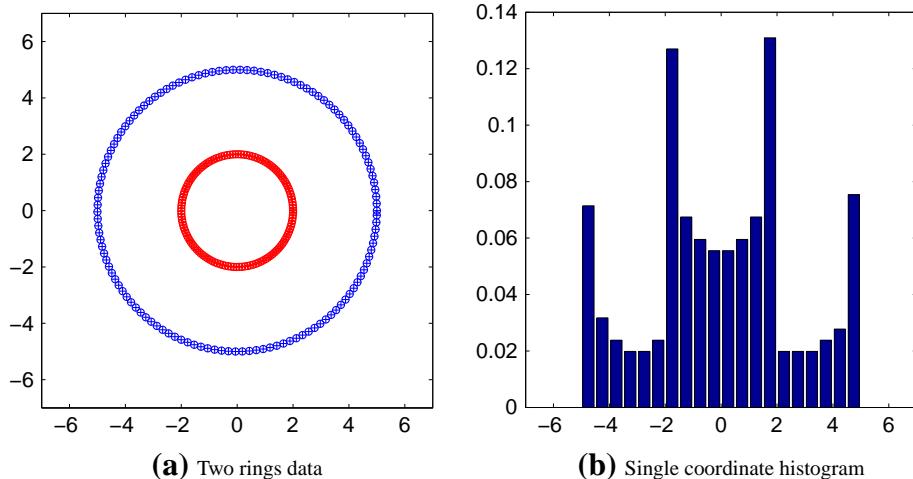
Large scale learning is particularly important for SSL, since the aim of SSL is to use the huge amount of unlabeled data to improve learning. Toward this goal, various new SSL methods and approximations of the existing methods have been proposed for scalable SSL.

From an optimization point of view, especially convex optimization, there are well-studied techniques for large scale computing. For example, Newton method is used in linear semi-supervised SVM in [11, 72]. Sparse grid is used in regularization in SSL [73]. However, the size of the grid increases exponentially fast with dimensions.

One attractive solution for large scale computing is online method. For instance, stochastic gradient descent is used to train TSVM in [74]. This type of algorithms allows training through linear scans of the whole data set. A graph-regularized transductive learning method that is based on minimizing a Kullback-Leibler divergence based loss is introduced in [75], and the algorithm can solve problems with as much as 120 million samples.

For many graph Laplacian-based SSL methods, the unique structure of the graph Laplacian matrix is explored in multi-grid method in [76]. The algorithm run-time and storage are linear in the number of graph edges. For sparse graphs, this is of the order of vertices, or data samples. Multiscale solvers for several (un)weighted graphs can be found in [77], with the software implementation.

There are methods based on the graph Laplacian on a smaller backbone graph, which is chosen in different ways from the whole data set. For example, random sampling is used in [78]. The problem with the approaches based on backbone graphs is that the spectrum of the graph Laplacian can change dramatically with different backbone construction methods, thus the solution is not stable.

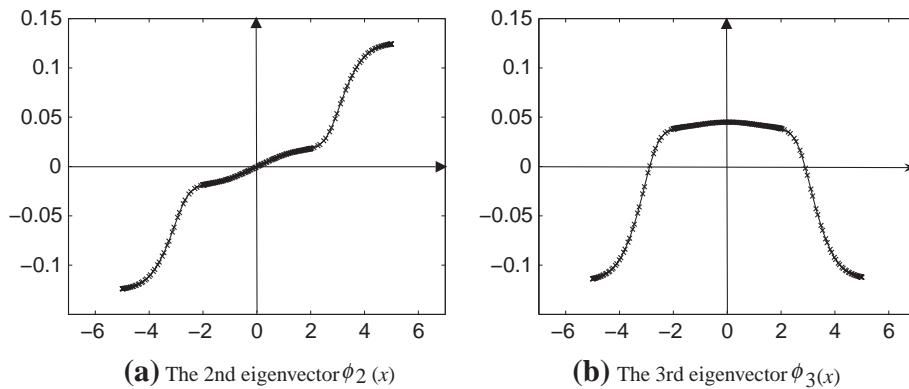
**FIGURE 22.4**

Two-ring data and its marginal distribution histogram.

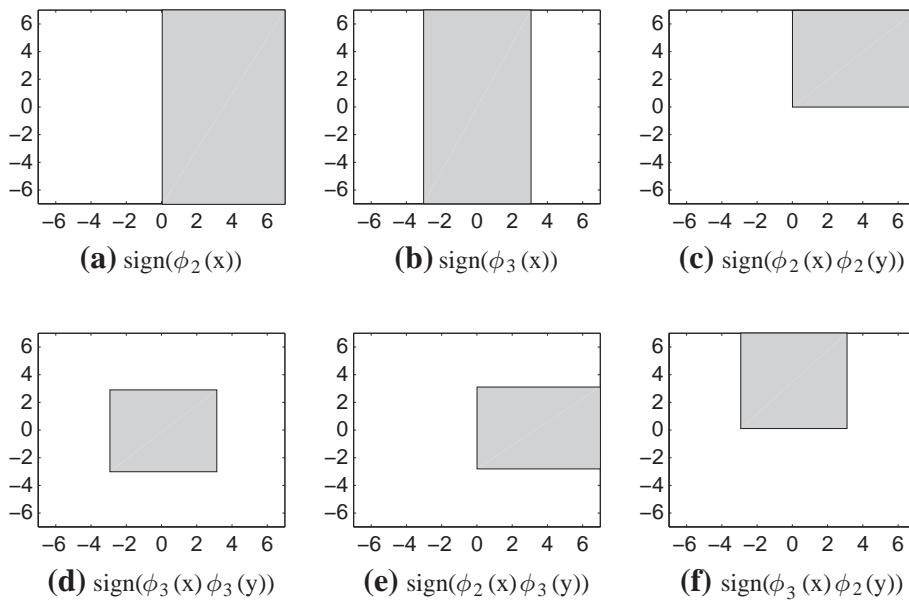
Most of above methods try to attack various approximations to the finite sample problems. On the contrary, starting from a well-studied infinite sample limit, Fergus et al. [79] proposed an attractive scalable SSL algorithm that can learn on much larger data sets. The eigenvectors of the graph Laplacian on data samples can be used as function basis to estimate functions on graphs. In the limit, the Laplacian eigenfunctions are natural square integrable function basis on smooth manifolds. With abundant unlabeled data, the graph Laplacian is close to its limit. Thus [79] starts from the limit, Laplacian on a smooth manifold, and under certain assumptions, takes advantage of a nice property of Laplacian: for product form densities, the Laplacian eigenfunctions on the marginal distribution of a single coordinate is also an eigenfunction of the Laplacian on the whole domain with the same eigenvalue. Then all these single-dimensional eigenvectors can be easily computed using various histogram and interpolation methods on a one-dimensional space.

As an example, a two-ring toy data set is plotted in Figure 22.4. The right panel is the marginal histogram. Since the data set is symmetric, marginal distributions of both dimensions are the same. Notice that this data set does not satisfy the ‘‘product form’’ assumption.

The single-dimensional graph Laplacian eigenvectors are shown in Figure 22.5. Notice that the signs of these single-dimensional eigenvectors decompose the square domain in a way similar to the axis parallel decision stumps. Several examples of such products are illustrated in Figure 22.6. In particular, the product of these eigenvectors provide a more powerful function basis such as squares, which can be obtained by the product of the third eigenvectors from the two dimensions (along  $x$  and  $y$ ). This square is enough to separate the two circles apart. Roughly speaking, the method is similar to decision stumps in AdaBoost. However, there are several differences. First, the products of eigenvectors provide a much richer basis than stumps alone. Second, these eigenvectors are data dependent, and are defined through unlabeled data in general. Third, these basis are real-valued functions instead of step functions.

**FIGURE 22.5**

The second and third graph Laplacian eigenvectors of the marginal.

**FIGURE 22.6**

$\text{Sign}(\cdot)$  functions of the single-dimensional graph Laplacian eigenvectors and their products. Grey areas are  $+1$ , while others are  $0$ .

This approximation transforms a  $d$ -dimensional problem to  $d$  one-dimensional problems. It not only eliminates the need to construct  $k$ NN graphs, which generally costs at least  $O(n^2)$ , but also removes the matrix inversion or eigen-decomposition, which costs  $O(n^3)$ .

Finally, several related themes in large scale SSL include, (a) different approximation methods, (b) large scale optimization techniques and online learning, e.g., [80], and (c) parallel computing, e.g., [81]. These areas also overlap with each other. For example, certain optimization techniques might require certain type of approximation, which might also determines whether it is possible for parallel computing.

## 1.22.5 Theoretical analysis overview

Some of the first work on the theoretical analysis of SSL appeared in [1–3], where the authors analyze parametric models based on the Gaussian mixture distributions. Error analysis of semi-supervised classification is further investigated in [82] under the more general cluster assumption. The main result is consistent with that of [1,2]: unlabeled data reduce the error with a polynomial rate of the form  $O(u^{-\beta})$  for some  $\beta > 0$ , while labeled may reduce the error exponentially fast.

Another result [83] shows that the pointwise mean squares error for a popular graph Laplacian regularization method has the same leading term as a regular supervised kernel smoother, which implies that the unlabeled data set does not help asymptotically.

On the other hand, Niyogi [84] studied SSL under the manifold assumption, and gives a constructive example of a class of distributions supported on manifolds where the error of any supervised learner is bounded from below by a constant, while a semi-supervised learner that can have error bounded from above by  $O(n^{-1/2})$ .

The value of unlabeled data is studied in a finite sample framework in [85], which evaluates the performance gains with SSL under the cluster assumption using finite sample error bounds, which involves finite labeled and unlabeled sets.

In a different approach, the value of unlabeled data was studied in [86] under the PAC model. The compatibility between concept class and the distribution is introduced to link unlabeled data in the analysis.

## 1.22.6 Challenges

In this chapter we have reviewed a number of popular SSL algorithms and various aspects of their analysis and implementation. Here we point out some of the key challenges for research and applications of SSL:

*Mode/parameter selection:* This is a common issue for all SSL algorithms. Model selection methods borrowed from supervised learning, such as cross-validation, may not work well in the SSL setting due to the limited amount of labeled data. Furthermore, many SSL models introduce several parameters which may be hard to choose appropriately for a new data domain.

*Scalability:* The premise of SSL is utilizing large amounts of unlabeled data to improve inference. Thus scalability is one of the most important issues facing SSL and is a key to making these algorithms widely used in various practical applications. While significant amount of research has gone into making SSL algorithms more scalable (see Section 1.22.4), there is a need for more work in that direction.

*Theoretical analysis:* The asymptotic analysis for SSL can shed light on various SSL algorithms and guide their applications. However, existing analyses are far from being satisfactory. There are two important questions for any SSL algorithm. First, what the estimators are. This is the problem of consistency, including how to define consistency. Second, how fast an empirical estimator converges to its limit, including how to define convergence. The rate should involve both  $l$  and  $u$ . Without consistency, no matter how fast the convergence rates are, it does not provide any guarantee. Most existing analyses focus on the large sample asymptotic analysis, which may not be sufficient for explaining the finite sample behaviors of some SSL algorithms.

With the recent advances in semi-supervised learning and as future developments addressing the challenges above come along, we expect that semi-supervised learning algorithms will become a key part in the toolbox of methods for large scale high-dimensional data analysis.

## Glossary

Graph Laplacian	also called the Laplacian matrix, is a matrix representation of a graph
Independent and identically distributed Laplacian	i.i.d.
Multiresolution analysis (MRA)	also called the Laplace operator, a differential operator given by the divergence of the gradient of a function on Euclidean space
Regularization	data analysis methods by constructing orthonormal bases at different scale
Semi-supervised learning (SSL)	any method of preventing overfitting of data by a model. Regularization involves introducing additional information in order to solve an ill-posed problem or to prevent overfitting
	a class of machine learning techniques that make use of both labeled and unlabeled data for training—typically a small amount of labeled data with a large amount of unlabeled data

## Relevant Websites

SVMlin: <http://vikas.sindhwani.org/svmlin.html>

SVM-light: <http://svmlight.joachims.org/>

Low Density Separation: <http://olivier.chapelle.cc/lde/>

Collection “Semi-supervised Learning”: <http://olivier.chapelle.cc/ssl-book/>

Semi-Supervised Learning Literature Survey: <http://pages.cs.wisc.edu/jerryzhu/>

*Relevant Theory:* Signal Processing Theory and Machine Learning

See this Volume, [Chapter 6](#) Digital Filter Structures and Their Implementation

See this Volume, [Chapter 16](#) Kernel Methods and SVMs

See this Volume, [Chapter 20](#) Clustering

---

## References

- [1] Vittorio Castelli, Thomas M. Cover, On the exponential value of labeled samples, *Pattern Recogn. Lett.* 16 (0) (1995) 105–111.
- [2] Vittorio Castelli, Thomas M. Cover, The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter, *IEEE Trans. Inf. Theory* 42 (6) (1996) 2102–2117.
- [3] Joel Ratsaby, Santosh S. Venkatesh, Learning from a mixture of labeled and unlabeled examples with parametric side information, in: Eighth Annual Conference on Computational Learning Theory (COLT95), 1995, pp. 412–417.
- [4] V. Vapnik, Statistical Learning Theory, Wiley-Interscience Press, New York, 1998.
- [5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: COLT: Proceedings of the Workshop on Computational Learning Theory, 1998, pp. 92–100.
- [6] O. Chapelle, B. Schölkopf, A. Zien (Eds.), Semi-Supervised Learning, MIT Press, Cambridge, MA, 2006. <<http://www.kyb.tuebingen.mpg.de/ssl-book>>.
- [7] M. Seeger, Learning with labeled and unlabeled data, Technical Report, University of Edinburgh, 2001.
- [8] X. Zhu, Semi-supervised learning literature survey, Technical Report, University of Wisconsin, Madison, 2008.
- [9] Thorsten Joachims, Transductive inference for text classification using support vector machines, in: 16th International Conference on Machine Learning (ICML), vol. 15, 1999, pp. 200–209.
- [10] M. Belkin, P. Niyogi, S. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, *J. Mach. Learn. Res.* 7 (2006) 2399–2434.
- [11] Vikas Sindhwani, Sathiya S. Keerthi, Large scale semi-supervised linear SVMs, in: 29th Annual International ACM SIGIR, 2006, pp. 477–484.
- [12] O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005, pp. 57–64.
- [13] Ronan Collobert, Fabian Sinz, Jason Weston, Léon Bottou, Large scale transductive SVMs, *J. Mach. Learn. Res.* 7 (2006) 1687–1712.
- [14] Olivier Chapelle, Vikas Sindhwani, Sathiya S. Keerthi, Optimization techniques for semi-supervised support vector machines, *J. Mach. Learn. Res.* 9 (2008) 203–233.
- [15] Olivier Chapelle, Vikas Sindhwani, S. Sathiya Keerthi, Branch and bound for semi-supervised support vector machines, in: B. Schölkopf, J. Platt, T. Hoffman (Eds.), Advances in Neural Information Processing Systems, vol. 19, MIT Press, Cambridge, MA, 2007, pp. 217–224.
- [16] Balaji Krishnapuram, Shipeng Yu, R. Bharat Rao (Eds.), Cost-Sensitive Machine Learning, CRC Press, 2011.
- [17] David G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the International Conference on Computer Vision, 1999, pp. 1150–1157.
- [18] Aude Oliva, Antonio Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, *Int. J. Comput. Vis.* 42 (3) (2001) 145–175.
- [19] Michael Collins, Yoram Singer, Unsupervised models for named entity classification, in: 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999, pp. 100–110.
- [20] Sham M. Kakade, Dean P. Foster, Multi-view regression via canonical correlation analysis, in: Conference on Computational Learning Theory (COLT), 2007, pp. 82–96.
- [21] C. Mario Christoudias, Raquel Urtasun, Trevor Darrell, Multi-view learning in the presence of view disagreement, in: 24th Conference on Uncertainty in Artificial Intelligence (UAI 2008), 2008.

- [22] V. Sindhwani, P. Niyogi, M. Belkin, A co-regularization approach to semi-supervised learning with multiple views, in: Workshop on Learning with Multiple Views, International Conference on Machine Learning (ICML), 2005.
- [23] V. Sindhwani, D. Rosenberg, An RKHS for multi-view learning and manifold co-regularization, in: International Conference on Machine Learning (ICML), 2008.
- [24] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Statist. Soc. Ser. B (Methodol.)* 39 (7) (1977) 1–38.
- [25] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, Tom Mitchell, Text classification from labeled and unlabeled documents using EM, *Mach. Learn.* 39 (2–3) (2000) 103–134.
- [26] Jeff A. Bilmes, A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models, Technical Report, University of Berkeley, 1998.
- [27] D. Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods, in: Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, 1995, pp. 189–196.
- [28] M. Meila, J. Shi, Learning segmentation by random walks, in: Neural Information Processing Systems, vol. 13, 2000, pp. 873–879.
- [29] Marina Meila, Jianbo Shi, A random walks view of spectral segmentation, in: AI and STATISTICS (AISTATS), 2001.
- [30] M. Szummer, T. Jaakkola, Partially labeled classification with Markov random walks, in: Neural Information Processing Systems (NIPS), vol. 14, 2001.
- [31] Avrim Blum, Shuchi Chawla, Learning from labeled and unlabeled data using graph mincuts, in: 18th International Conference on Machine Learning, 2001, pp. 19–26.
- [32] Xiaojin Zhu, Zoubin Ghahramani, Learning from labeled and unlabeled data with label propagation, Technical Report, CMU-CALD-02-107, Carnegie Mellon University, 2002.
- [33] M. Belkin, P. Niyogi, Semi-supervised learning on Riemannian manifolds, *Mach. Learn.* 56 (2004) 209–239 (special issue on Clustering).
- [34] X. Zhu, J. Lafferty, Z. Ghahramani, Semi-supervised learning using Gaussian fields and harmonic function, in: 20th International Conference on Machine Learning, 2003.
- [35] A.J. Smola, R. Kondor, Kernels and regularization on graphs, in: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, 2003, pp. 144–158.
- [36] Xueyuan Zhou, Mikhail Belkin, Semi-supervised learning by higher order regularization, in: Geoffrey Gordon, David Dunson, Miroslav Dudík (Eds.), 14th International Conference on Artificial Intelligence and Statistics, vol. 15, JMLR W&CP, 2011, pp. 892–900.
- [37] Yoshua Bengio, Jean Francois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas Le Roux, Marie Ouimet, Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering, in: Advances in Neural Information Processing Systems, vol. 16, MIT Press, 2004, pp. 177–184.
- [38] Fan R.K. Chung, Spectral graph theory, in: CBMS Regional Conference Series in Mathematics, No. 92, 1992.
- [39] U. von Luxburg, A tutorial on spectral clustering, *Statist. Comput.* 17 (4) (2007) 395–416.
- [40] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, Bernhard Schölkopf, Learning with local and global consistency, in: Sebastian Thrun, Lawrence Saul, Bernhard Schölkopf (Eds.), Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, MA, 2004, pp. 321–328.
- [41] M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in: John Shawe-Taylor, Yoram Singer (Eds.), Computational Learning Theory (COLT), Lecture Notes in Computer Science, vol. 3120, Springer, 2004, pp. 624–638.
- [42] Olivier Bousquet, Olivier Chapelle, Matthias Hein, Measure based regularization, in: Sebastian Thrun, Lawrence Saul, Bernhard Schölkopf (Eds.), Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, MA, 2004.

- [43] Boaz Nadler, Nathan Srebro, Xueyuan Zhou, Statistical analysis of semi-supervised learning: the limit of infinite unlabelled data, in: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 1330–1338.
- [44] B. Daya Reddy, Introductory Function Analysis, with Applications to Boundary Value Problems and Finite Elements, Springer, 1997.
- [45] A. Berlinet, C. Thomas-Agnan, Reproducing Kernel Hilbert Spaces in Probability and Statistics, Kluwer Academic Publishers, 2003.
- [46] R.A. Adams, Sobolev Spaces, Academic Press, New York, 1975.
- [47] Michael E. Taylor, Partial Differential Equations I: Basic Theory, Springer, New York, 1996.
- [48] G. Wahba, Spline models for observational data, in: CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59, SIAM, Philadelphia, 1990.
- [49] Matthias Hein, Jean yves Audibert, Ulrike Von Luxburg, Graph Laplacians and their convergence on random neighborhood graphs, *J. Mach. Learn. Res.* 8 (2007) 1325–1368.
- [50] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, *Neural Comput.* 15 (6) (2003) 1373–1396.
- [51] Xueyuan Zhou, Nathan Srebro, Error analysis of Laplacian eigenmaps for semi-supervised learning, in: Geoffrey Gordon, David Dunson, Miroslav Dudík (Eds.), 14th International Conference on Artificial Intelligence and Statistics, vol. 15, JMLR W&CP, 2011, pp. 901–908.
- [52] László Györfi, Michael Kohler, Adam Krzyżak, Harro Walk, A Distribution-Free Theory of Nonparametric Regression, Springer, 2002.
- [53] Peter J. Bickel, Bo Li, Local polynomial regression on unknown manifolds, *Complex Datasets and Inverse Problems: Tomography, Networks and Beyond, IMS Lecture Notes—Monograph Series*, vol. 54, 2007, pp. 177–186.
- [54] R.R. Coifman, S. Lafon, Diffusion maps, *Appl. Comput. Harmon. Anal.* 21 (1) (2006) 5–30.
- [55] Xueyuan Zhou, Mikhail Belkin, Behavior of graph Laplacians on manifolds with boundary, 2011. arXiv:1105.3931v1 [cs.LG].
- [56] David K. Hammond, Pierre Vandergheynst, Rémi Gribonval, Wavelets on graphs via spectral graph theory, *Appl. Comput. Harmon. Anal.* 30 (2) (2011) 129–150.
- [57] R.R. Coifman, M. Maggioni, Diffusion wavelets, *Appl. Comput. Harmon. Anal.* 21 (1) (2006) 53–94.
- [58] Matan Gavish, Boaz Nadler, Ronald R. Coifman, Multiscale wavelets on trees, graphs and high dimensional data: theory and applications to semi supervised learning, in: International Conference on Machine Learning, 2010, pp. 367–374.
- [59] Y. Xu, J.S. Dyer, A.B. Owen, Empirical stationary correlations for semi-supervised learning on graphs, *Ann. Appl. Statist.* 4 (2) (2010) 589–614.
- [60] John Lafferty, Xiaojin Zhu, Yan Liu, Kernel conditional random fields: representation and clique selection, in: Proceedings of the 21st International Conference on Machine Learning, 2004.
- [61] Ben Taskar, Carlos Guestrin, Daphne Koller, Max-margin Markov networks, in: Sebastian Thrun, Lawrence Saul, Bernhard Schölkopf (Eds.), Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge, MA, 2004, pp. 25–32.
- [62] Yasemin Altun, David McAllester, Mikhail Belkin, Maximum margin semi-supervised learning for structured variables, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems, vol. 18, MIT Press, Cambridge, MA, 2006, pp. 33–40.
- [63] Ulf Brefeld, Tobias Scheffer, Semi-supervised learning for structured output variables, in: William Cohen, Andrew Moore (Eds.), 23rd International Conference on Machine Learning (ICML), Omni Press, 2006, pp. 624–638.
- [64] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, S.V.N. Vishwanathan, Predicting Structured Data, MIT Press, Cambridge, MA, 2007.

- [65] Ulf Brefeld, Semi-Supervised Structured Prediction Models, PhD thesis, Humboldt-Universität zu Berlin, 2008.
- [66] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, Dale Schuurmans, Semi-supervised conditional random fields for improved sequence segmentation and labeling, in: Association for Computational Linguistics (ACL), 2006, pp. 209–216.
- [67] Amarnag Subramanya, Slav Petrov, Fernando Pereira, Efficient graph-based semi-supervised learning of structured tagging models, in: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2010, pp. 167–176.
- [68] Ben Taskar, Simon Lacoste-Julien, Michael Jordan, Structured prediction, dual extragradient and Bregman projections, *J. Mach. Learn. Res.* 7 (2006) 1627–1653.
- [69] I. Tschantzidis, T. Joachims, T. Hofmann, Y. Altun, Large margin methods for structured and interdependent output variables, *J. Mach. Learn. Res.* 6 (2005) 1453–1484.
- [70] Chun-Nam John Yu, T. Joachims, Learning structural SVMs with latent variables, in: Andrea Pohoreckyj Danyluk, Léon Bottou, Michael L. Littman (Eds.), 26th Annual International Conference on Machine Learning (ICML), 2009, pp. 1169–1176.
- [71] Jun Zhu, Eric P. Xing, Maximum entropy discrimination Markov networks, *J. Mach. Learn. Res.* 10 (2009) 2531–2569.
- [72] V. Sindhwani, P. Niyogi, M. Belkin, SVMlin: fast linear SVM solvers for supervised and semi-supervised learning, in: Workshop on Machine Learning Open Source Software, Neural Information Processing Systems (NIPS), 2006.
- [73] Jochen Garcke, Michael Griebel, Semi-supervised learning with sparse grids, in: 22nd ICML Workshop on Learning with Partially Classified Training Data, 2005.
- [74] Michael Karlen, Jason Weston, Ayse Erkan, Ronan Collobert, Large scale manifold transduction, in: Andrew McCallum, Sam Roweis (Eds.), 25th International Conference on Machine Learning (ICML 2008), 2008, pp. 448–455.
- [75] Amarnag Subramanya, Jeff Bilmes, Entropic graph regularization in non-parametric semi-supervised classification, in: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 1803–1811.
- [76] Oren E. Livne, Achi Brandt, Lean algebraic multigrid (LAMG): fast graph Laplacian linear solver, 2011. arXiv:1108.1310v1 [math.NA].
- [77] Ilya Safro, Dorit Ron, Achi Brandt, Multilevel algorithms for linear ordering problems, *J. Exp. Algor.* 13 (2009).
- [78] Ameet Talwalkar, Sanjiv Kumar, Henry Rowley, Large-scale manifold learning, in: Computer Vision and Pattern Recognition, 2008 (CVPR 2008), 2008, pp. 1–8.
- [79] Rob Fergus, Yair Weiss, Antonio Torralba, Semi-supervised learning in gigantic image collections, in: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 522–530.
- [80] Andrew B. Goldberg, Ming Li, Xiaojin Zhu, Online manifold regularization: a new learning setting and empirical study, in: 2008 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML08), 2008, pp. 393–407.
- [81] Amol Ghating, Rajasekar Krishnamurthy, Edwin Pednault, Berthold Reinwald, Vikas Sindhwani, Shirish Tatikonda, Yuanyuan Tian, Shivakumar Vaithyanathan, SystemML: declarative machine learning on MapReduce, in: IEEE International Conference on Data Engineering (ICDE), 2011, pp. 231–242.
- [82] Philippe Rigollet, Generalization error bounds in semi-supervised classification under the cluster assumption, *J. Mach. Learn. Res.* 8 (2007) 1369–1392.

- [83] John Lafferty, Larry Wasserman, Statistical analysis of semi-supervised regression, in: J.C. Platt, D. Koller, Y. Singer, S. Roweis (Eds.), *Advances in Neural Information Processing Systems*, vol. 20, MIT Press, Cambridge, MA, 2008, pp. 801–808.
- [84] Partha Niyogi, Manifold regularization and semi-supervised learning: some theoretical analyses, Technical Report, Department of Computer Science, University of Chicago, 2008.
- [85] Aarti Singh, Robert Nowak, Xiaojin Zhu, Unlabeled data: now it helps, now it doesn't, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), *Advances in Neural Information Processing Systems*, vol. 21, 2009, pp. 1513–1520.
- [86] Maria-Florina Balcan, Avrim Blum, A PAC-style model for learning from labeled and unlabeled data, in: 18th Annual Conference on Computational Learning Theory (COLT), 2005, pp. 111–126.

# Sparsity-Aware Learning and Compressed Sensing: An Overview<sup>\*</sup>

# 23

Sergios Theodoridis<sup>\*</sup>, Yannis Kopsinis<sup>\*</sup>, and Konstantinos Slavakis<sup>†</sup>

<sup>\*</sup>Department of Informatics and Telecommunication, University of Athens, Athens, Greece  
<sup>†</sup>Digital Technology Center, University of Minnesota, Minneapolis, USA

## 1.23.1 Introduction

The notion of regularization has been widely used as a tool to address a number of problems that are usually encountered in Machine Learning. Improving the performance of an estimator by shrinking the norm of the Minimum Variance Unbiased (MVU) estimator, guarding against overfitting, coping with ill-conditioning, providing a solution to an underdetermined set of equations are some notable examples where regularization has provided successful answers. A notable example is the ridge regression concept, where the LS loss function is combined, in a tradeoff rationale, with the Euclidean norm of the desired solution.

In this chapter, our interest will be on alternatives to the Euclidean norms and in particular the focus will revolve around the  $\ell_1$  norm; this is the sum of the absolute values of the components comprising a vector. Although seeking a solution to a problem via the  $\ell_1$  norm regularization of a loss function has been known and used since the 1970s, it is only recently that has become the focus of attention of a massive volume of research in the context of compressed sensing. At the heart of this problem lies an underdetermined set of linear equations, which, in general, accepts an infinite number of solutions. However, in a number of cases, an extra piece of information is available: the true model, whose estimate we want to obtain, is sparse; that is, only a few of its coordinates are nonzero. It turns out that a large number of commonly used applications can be cast under such a scenario and can be benefited by a so-called sparse modeling.

Besides its practical significance, sparsity-aware processing has offered to the scientific community novel theoretical tools and solutions to problems that only a few years ago seemed to be intractable. This is also a reason that this is an interdisciplinary field of research encompassing scientists from, e.g., mathematics, statistics, machine learning, signal processing. Moreover, it has already been applied in many areas ranging from biomedicine, to communications and astronomy. At the time this chapter is compiled, there is a “research happening” in this field, which poses some difficulties in assembling related material together. We have made an effort to put together, in a unifying way, the basic notions and ideas that run across this new field. Our goal is to provide the reader with an overview of the major contributions which took place in the theoretical and algorithmic fronts and have been consolidated over the

\*This paper is based on a chapter of a new book on Machine Learning, by the first and third authors, which is currently under preparation.

last decade or so. Besides the methods and algorithms which are reviewed in this article, there is another path of methods based on the Bayesian learning rationale. Such techniques will be reviewed elsewhere.

### 1.23.2 Parameter estimation

Parameter estimation is at the heart of what is known as *Machine Learning*; a term that is used more and more as an umbrella for a number of scientific topics that have evolved over the years within different communities, such as Signal Processing, Statistical Learning, Estimation/Detection, Control, Neurosciences, Statistical Physics, to name but a few.

In its more general and formal setting, the parameter estimation task is defined as follows. Given a set of data points  $(y_n, \mathbf{x}_n)$ ,  $y_n \in \mathcal{R}$ ,  $\mathbf{x}_n \in \mathcal{R}^l$ ,  $n = 1, 2, \dots, N$ , known as the *training data*, and a parametric set of functions

$$\mathcal{F} := \{f_{\theta}, \theta \in \mathcal{A} \subseteq \mathcal{R}^k\},$$

find a function in  $\mathcal{F}$ , which will be denoted as  $f(\cdot) := f_{\theta_*}(\cdot)$ , such that given the value of  $\mathbf{x} \in \mathcal{R}^l$ ,  $f(\mathbf{x})$  best approximates the corresponding value of  $y \in \mathcal{R}$ . After all, the main goal of Machine Learning is *prediction*. In a more general setting,  $y$  can also be a vector  $\mathbf{y} \in \mathcal{R}^m$ . Most of our discussion here will be limited to real valued variables. Obviously, extensions to complex valued data are readily available.

Having adopted the parametric set of functions and given the training data set, the goal becomes that of *estimating* the values of the parameters  $\theta$  so that to “fit” the data in some (optimal) way. There are various paths to achieve this goal. In this chapter, our approach comprises the adoption of a *loss* function

$$\mathcal{L}(\cdot, \cdot) : \mathcal{R} \times \mathcal{R} \longmapsto [0, \infty),$$

and obtain  $\theta_*$  such that

$$\theta_* := \arg \min_{\theta} J(\theta),$$

where

$$J(\theta) := \sum_{n=1}^N \mathcal{L}(y_n, f_{\theta}(\mathbf{x})). \quad (23.1)$$

Our focus will be on the Least Squares loss function, i.e.,

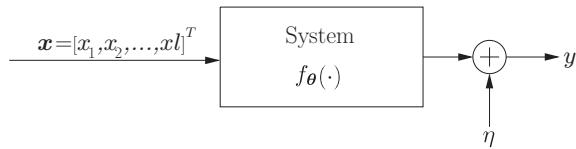
$$\mathcal{L}(y, f_{\theta}(\mathbf{x})) := (y - f_{\theta}(\mathbf{x}))^2.$$

Among the many parametric models, *regression* covers a large class of Machine Learning tasks. In linear regression, one models the relationship of a *dependent* variable  $y$ , which is considered as the output of a system, with a set of *independent* variables,  $x_1, x_2, \dots, x_l$ , which are thought as the respective inputs that activate the system in the presence of a noise (unobserved) disturbance,  $\eta$ , i.e.,

$$y = \theta_1 x_1 + \dots + \theta_l x_l + \theta_0 + \eta,$$

where  $\theta_0$  is known as the *bias* or *intercept*, see Figure 23.1. Very often the previous input-output relationship is written as

$$y = \mathbf{x}^T \boldsymbol{\theta} + \eta, \quad (23.2)$$

**FIGURE 23.1**

Block diagram showing the input-output relation in a regression model.

where

$$\boldsymbol{\theta} := [\theta_1, \dots, \theta_0]^T, \quad \text{and} \quad \mathbf{x} := [x_1, \dots, x_l, 1]^T. \quad (23.3)$$

Often,  $\mathbf{x}$  is called the *regressor*. Given the set of training data points,  $(y_n, \mathbf{x}_n), n = 1, 2, \dots, N$ , (23.2) can compactly written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\eta}, \quad (23.4)$$

where

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_N \end{bmatrix}. \quad (23.5)$$

For such a model, the Least Squares cost function is written as

$$J(\boldsymbol{\theta}) = \sum_{n=1}^N (y_n - \boldsymbol{\theta}^T \mathbf{x}_n)^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2, \quad (23.6)$$

where  $\|\cdot\|$  denotes the Euclidean norm. Minimizing (23.6) with respect to  $\boldsymbol{\theta}$  results to the celebrated LS estimate

$$\hat{\boldsymbol{\theta}}_{LS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \quad (23.7)$$

assuming the matrix inversion is possible. However, for many practical cases, the cost function in (23.6) is augmented with a so called *regularization* term. There are a number of reasons that justify the use of a regularization term. Guarding against overfitting, purposely introducing bias in the estimator in order to improve the overall performance, dealing with the ill conditioning of the task are cases which the use of regularization has successfully addressed. *Ridge regression* is a celebrated example, where the cost function is augmented as

$$L(\boldsymbol{\theta}, \lambda) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2, \quad \lambda \geq 0$$

leading to the estimate

$$\hat{\boldsymbol{\theta}}_R = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y},$$

where  $\mathbf{I}$  is the identity matrix. Note that the first term in a regularized cost function measures the model misfit and the second one the “size” of the model.

The major goal of this review chapter is to focus at alternative norms in place of the Euclidean norm, which was employed in ridge regression. As we will see, there are many good reasons in doing that.

### 1.23.3 Searching for a norm

Mathematicians have been very imaginative in proposing various norms in order to equip linear spaces. Among the most popular norms used in functional analysis are the so-called  $\ell_p$  norms. To tailor things to our needs, given a vector  $\boldsymbol{\theta} \in \mathcal{R}^l$ , its  $\ell_p$  norm is defined as

$$\|\boldsymbol{\theta}\|_p := \left( \sum_{i=1}^l |\theta_i|^p \right)^{\frac{1}{p}}. \quad (23.8)$$

For  $p = 2$ , the Euclidean or  $\ell_2$  norm is obtained, and for  $p = 1$ , (23.8) results in the  $\ell_1$  norm, i.e.,

$$\|\boldsymbol{\theta}\|_1 = \sum_{i=1}^l |\theta_i|. \quad (23.9)$$

If we let  $p \rightarrow \infty$ , then we get the  $\ell_\infty$  norm; let  $|\theta_{i_{\max}}| := \max \{|\theta_1|, |\theta_2|, \dots, |\theta_l|\}$ , and notice that

$$\|\boldsymbol{\theta}\|_\infty := \lim_{p \rightarrow \infty} \left( |\theta_{i_{\max}}|^p \sum_{i=1}^l \left( \frac{|\theta_i|}{|\theta_{i_{\max}}|} \right)^p \right)^{\frac{1}{p}} = |\theta_{i_{\max}}|, \quad (23.10)$$

that is,  $\|\boldsymbol{\theta}\|_\infty$  is equal to the maximum of the absolute values of the coordinates of  $\boldsymbol{\theta}$ . One can show that all the  $\ell_p$  norms are true norms for  $p \geq 1$ ; they satisfy all four requirements that a function  $\mathcal{R}^l \rightarrow [0, \infty)$  must respect in order to be called a norm, i.e.,

1.  $\|\boldsymbol{\theta}\|_p \geq 0$ .
2.  $\|\boldsymbol{\theta}\|_p = 0 \Leftrightarrow \boldsymbol{\theta} = \mathbf{0}$ .
3.  $\|\alpha\boldsymbol{\theta}\|_p = |\alpha| \|\boldsymbol{\theta}\|_p, \forall \alpha \in \mathcal{R}$ .
4.  $\|\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2\|_p \leq \|\boldsymbol{\theta}_1\|_p + \|\boldsymbol{\theta}_2\|_p$ .

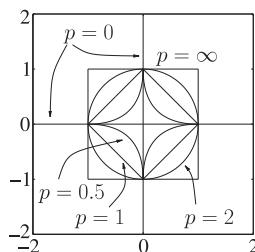
The third condition enforces the norm function to be (*positively*) *homogeneous* and the fourth one is the *triangle inequality*. These properties also guarantee that any function that is a norm is also a convex one. Although strictly speaking, if we allow  $p > 0$  to take values less than one in (23.8), the resulting function is easily shown not to be a true norm, we can still call them norms, albeit knowing that this is an abuse of the definition of a norm. An interesting case, which will be used extensively in this paper, is the  $\ell_0$  norm, which can be obtained as the limit, for  $p \rightarrow 0$ , of

$$\|\boldsymbol{\theta}\|_0 := \lim_{p \rightarrow 0} \|\boldsymbol{\theta}\|_p^p = \lim_{p \rightarrow 0} \sum_{i=1}^l |\theta_i|^p = \sum_{i=1}^l \chi_{(0,\infty)}(|\theta_i|), \quad (23.11)$$

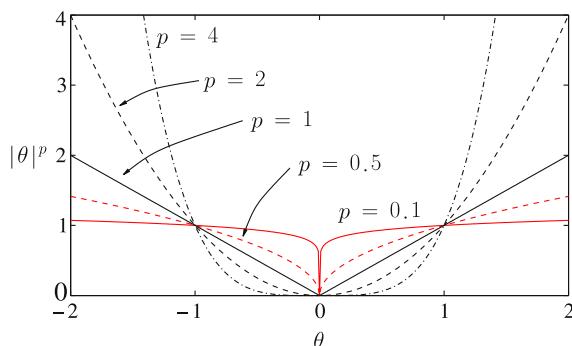
where  $\chi_{\mathcal{A}}(\cdot)$  is the characteristic function with respect to a set  $\mathcal{A}$ , defined as

$$\chi_{\mathcal{A}}(\tau) := \begin{cases} 1, & \text{if } \tau \in \mathcal{A}, \\ 0, & \text{if } \tau \notin \mathcal{A}. \end{cases}$$

That is, the  $\ell_0$  norm is equal to the number of nonzero components of the respective vector. It is very easy to check that this function is not a true norm. Indeed, this function is not homogeneous, i.e.,  $\|\alpha\boldsymbol{\theta}\|_0 \neq |\alpha| \|\boldsymbol{\theta}\|_0, \forall \alpha \neq 1$ . Figure 23.2 shows the isovalue curves, in the two-dimensional space, that correspond

**FIGURE 23.2**

The isovalue curves for  $\|\theta\|_p = 1$  and for various values of  $p$ , in the two dimensional space. Observe that for the  $\ell_0$  norm, the respective values cover the two axes with the exception of the point  $(0, 0)$ . For the  $\ell_1$  norm the isovalue curve is a rhombus and for the  $\ell_2$  (Euclidean) norm, it is a circle.

**FIGURE 23.3**

Observe that the epigraph, that is, the region above the graph, is nonconvex for values  $p < 1$ , indicating the nonconvexity of the respective  $|\cdot|^p$  function. The value  $p = 1$  is the smallest one for which convexity is retained. Also note that, for large values of  $p > 1$ , the contribution of small values of  $\theta$  to the respective norm becomes insignificant.

to  $\|\theta\|_p = \rho \equiv 1$ , for  $p = 0, 0.5, 1, 2$ , and  $\infty$ . Observe that for the Euclidean norm the isovalue curve has the shape of a “ball” and for the  $\ell_1$  norm the shape of a rhombus. We refer to them as the  $\ell_2$  and the  $\ell_1$  balls, respectively, by slightly “abusing” the meaning of a ball.<sup>1</sup> Observe that in the case of the  $\ell_0$  norm, the isovalue curve comprises both the horizontal and the vertical axes, excluding the  $(0, 0)$  element. If we restrict the size of the  $\ell_0$  norm to be less than one, then the corresponding set of points becomes a singleton, i.e.,  $(0, 0)$ . Also, the set of all the two-dimensional points that have  $\ell_0$  norm less than or equal to two, is the  $\mathcal{R}^2$  space. This, slightly “strange” behavior, is a consequence of the discrete nature of this “norm.”

Figure 23.3 shows the graph of  $|\cdot|^p$ , which is the individual contribution of each component of a vector to its  $\ell_p$  norm, for different values of  $p$ . Observe that: (a) for  $p < 1$ , the region which is formed

<sup>1</sup>Strictly speaking, a ball must also contain all the points in the interior.

above the graph (known as epigraph) is not a convex one, which verifies what we have already said; i.e., the respective function is not a true norm, (b) for values of the argument  $|\theta| > 1$ , the larger the value of  $p \geq 1$  and the larger the value of  $|\theta|$  the higher its respective contribution to the norm. Hence, if  $\ell_p$  norms,  $p \geq 1$ , are used to regularize a loss function, such large values become the dominant ones and the optimization algorithm will concentrate on these by penalizing them to get smaller, so that the overall cost to be reduced. On the other hand, for values of the argument  $|\theta| < 1$  and closer to zero, the  $\ell_1$  norm is the only one (among  $p \geq 1$ ) that retains relatively large values and, hence, the respective components can still have a say in the optimization process and can be penalized by being pushed to smaller values. Hence, if the  $\ell_1$  norm is used to replace the  $\ell_2$  one in the regularization equation, *only those components of the vector, that are really significant in reducing the model misfit measuring term in the regularized cost function, will be kept and the rest will be forced to zero.* The same tendency, yet more aggressive, is true for  $0 \leq p < 1$ . The extreme case is when one considers the  $\ell_0$  norm. Even a small increase of a component from zero, makes its contribution to the norm large, so the optimizing algorithm has to be very “cautious” in making an element nonzero.

From all the true norms ( $p \geq 1$ ), the  $\ell_1$  is the only one that shows respect to small values. The rest of the  $\ell_p$  norms,  $p > 1$ , just squeeze them, to make their values even smaller and care, mainly, for the large values. We will return to this point very soon.

### 1.23.4 The least absolute shrinkage and selection operator (LASSO)

We have already discussed some of the benefits in adopting the regularization method for enhancing the performance of an estimator. However, in this paper, we are going to see and study more reasons that justify the use of regularization. The first one refers to what is known as the *interpretation* power of an estimator. For example, in the regression task, we want to select those components,  $\theta_i$ , of  $\boldsymbol{\theta}$  that have the most important say in the formation of the output variable. This is very important if the number of parameters,  $l$ , is large and we want to concentrate on the most important of them. In a classification task [1], not all features are informative, hence one would like to keep the most informative of them and make the less informative ones equal to zero. Another related problem refers to those cases where we know, *a priori*, that a number of the components of a parameter vector are zero but we do not know which ones. The discussion we had at the end of the previous section starts now to become more meaningful. Can we use, while regularizing, an appropriate norm that can assist the optimization process (a) in unveiling such zeros or (b) to put more emphasis on the most significant of its components, those that play a decisive role in reducing the misfit measuring term in the regularized cost function, and set the rest of them equal to zero? Although the  $\ell_p$  norms, with  $p < 1$ , seem to be the natural choice for such a regularization, the fact that they are not convex makes the optimization process hard. The  $\ell_1$  norm is the one that is “closest” to them yet it retains the computationally attractive property of convexity.

The  $\ell_1$  norm has been used for such problems for a long time. In the seventies, it was used in seismology [2,3], where the reflected signal, that indicates changes in the various earth substrates, is a sparse one, i.e., very few values are relatively large and the rest are small and insignificant. Since then, it has been used to tackle similar problems in different applications, e.g., [4,5]. However, one can trace two papers that were really catalytic in providing the spark for the current strong interest around the  $\ell_1$  norm. One came from statistics, [6], which addressed the LASSO task (first formulated, to our

knowledge, in [5]), to be discussed next, and the other from the signal analysis community, [7], which formulated the *Basis Pursuit*, to be discussed in a later section.

We first address our familiar regression task

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\eta}, \quad \mathbf{y}, \boldsymbol{\eta} \in \mathcal{R}^N, \quad \boldsymbol{\theta} \in \mathcal{R}^l,$$

and obtain the estimate of the unknown parameter  $\boldsymbol{\theta}$  via the LS loss, regularized by the  $\ell_1$  norm, i.e., for  $\lambda \geq 0$ ,

$$\hat{\boldsymbol{\theta}} := \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} L(\boldsymbol{\theta}, \lambda) \quad (23.12)$$

$$\begin{aligned} &:= \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left( \sum_{n=1}^N (y_n - \mathbf{x}_n^T \boldsymbol{\theta})^2 + \lambda \|\boldsymbol{\theta}\|_1 \right) \\ &= \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left( (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \right). \end{aligned} \quad (23.13)$$

In order to simplify the analysis, we will assume hereafter, without harming generality, that the data are centered. If this is not the case, the data can be centered by subtracting the sample mean  $\bar{y}$  from each one of the output values. The estimate of the bias term will be equal to the sample mean  $\bar{y}$ . The task in (23.13) can be *equivalently* written in the following two formulations

$$\begin{aligned} \hat{\boldsymbol{\theta}} : \min_{\boldsymbol{\theta} \in \mathcal{R}^l} & (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}), \\ \text{s.t. } & \|\boldsymbol{\theta}\|_1 \leq \rho, \end{aligned} \quad (23.14)$$

or

$$\begin{aligned} \hat{\boldsymbol{\theta}} : \min_{\boldsymbol{\theta} \in \mathcal{R}^l} & \|\boldsymbol{\theta}\|_1, \\ \text{s.t. } & (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \leq \epsilon, \end{aligned} \quad (23.15)$$

given the user-defined parameters  $\rho, \epsilon \geq 0$ . The formulation in (23.14) is known as the LASSO and the one in (23.15) as the *Basis Pursuit Denoising* (BPDN), e.g., [8]. All three formulations can be shown to be equivalent for specific choices of  $\lambda, \epsilon$ , and  $\rho$ . The minimized cost function in (23.13) corresponds to the Lagrangian of the formulations in (23.14) and (23.15). However, this functional dependence is hard to compute, unless the columns of  $\mathbf{X}$  are mutually orthogonal. Moreover, this equivalence does not necessarily imply that all three formulations are equally easy or difficult to solve. As we will see later on, algorithms have been developed along each one of the previous formulations. From now on, we will refer to all three formulations as the LASSO task, in a slight abuse of the standard terminology, and the specific formulation will be apparent from the context, if not stated explicitly.

As it was discussed before, the Ridge regression admits a closed form solution, i.e.,

$$\hat{\boldsymbol{\theta}}_R = \left( \mathbf{X}^T \mathbf{X} + \lambda I \right)^{-1} \mathbf{X}^T \mathbf{y}.$$

In contrast, this is not the case for LASSO and its solution requires iterative techniques. It is straightforward to see that LASSO can be formulated as a standard convex quadratic problem with linear

inequalities. Indeed, we can rewrite (23.13) as

$$\begin{aligned} \min_{\{\theta_i, u_i\}_{i=1}^l} \quad & (\mathbf{y} - X\boldsymbol{\theta})^T(\mathbf{y} - X\boldsymbol{\theta}) + \lambda \sum_{i=1}^l u_i \\ \text{s.t.} \quad & \begin{cases} -u_i \leq \theta_i \leq u_i, & i = 1, 2, \dots, l, \\ u_i \geq 0, \end{cases} \end{aligned}$$

which can be solved by any standard convex optimization method, e.g., [9, 10]. The reason that developing algorithms for the LASSO has been a hot research topic is due to the emphasis in obtaining *efficient* algorithms by exploiting the specific nature of this task, especially for cases where  $l$  is very large, as it is often the case in practice.

In order to get a better insight of the nature of the solution that is obtained by LASSO, let us assume that the regressors are mutually orthogonal and of unit norm, hence  $X^T X = I$ . Orthogonality of the input matrix helps to decouple the coordinates and results to  $l$  one-dimensional problems, that can be solved analytically. For this case, the LS estimator becomes

$$\hat{\boldsymbol{\theta}}_{\text{LS}} = (X^T X)^{-1} X^T \mathbf{y} = X^T \mathbf{y},$$

and the ridge regression gives

$$\hat{\boldsymbol{\theta}}_R = \frac{1}{1 + \lambda} \hat{\boldsymbol{\theta}}_{\text{LS}}, \quad (23.16)$$

that is, every component of the LS estimator is simply shrunk by the *same* factor,  $\frac{1}{1+\lambda}$ .

In the case of the  $\ell_1$  regularization, the minimized Lagrangian function is no more differentiable, due to the presence of the absolute values in the  $\ell_1$  norm. So, in this case, we have to consider the notion of the subdifferential (see Appendix). It is known that if the zero vector belongs to the subdifferential set of a convex function at a point, this means that this point corresponds to a minimum of the function. Taking the subdifferential of the Lagrangian defined in (23.13) and recalling that the subdifferential set of a differentiable function includes *only* the respective gradient, we obtain that

$$\mathbf{0} \in -2X^T \mathbf{y} + 2X^T X \boldsymbol{\theta} + \lambda \partial \|\boldsymbol{\theta}\|_1,$$

where  $\partial$  stands for the subdifferential operator (see Appendix). If  $X$  has orthonormal columns, the previous equation can be written component-wise as follows

$$0 \in -\hat{\theta}_{\text{LS},i} + \hat{\theta}_{1,i} + \frac{\lambda}{2} \partial |\hat{\theta}_{1,i}|, \quad \forall i, \quad (23.17)$$

where the subdifferential of the function  $|\cdot|$ , derived in Appendix, is given as

$$\partial |\theta| = \begin{cases} \{1\}, & \text{if } \theta > 0, \\ \{-1\}, & \text{if } \theta < 0, \\ [-1, 1], & \text{if } \theta = 0. \end{cases}$$

Thus, we can now write for each component of the LASSO optimal estimate

$$\hat{\theta}_{1,i} = \begin{cases} \hat{\theta}_{\text{LS},i} - \frac{\lambda}{2}, & \text{if } \hat{\theta}_{1,i} > 0, \\ \hat{\theta}_{\text{LS},i} + \frac{\lambda}{2}, & \text{if } \hat{\theta}_{1,i} < 0. \end{cases} \quad (23.18)$$

$$(23.19)$$

Notice that (36.18) can only be true if  $\hat{\theta}_{\text{LS},i} > \frac{\lambda}{2}$ , and (36.19) only if  $\hat{\theta}_{\text{LS},i} < -\frac{\lambda}{2}$ . Moreover, in the case where  $\hat{\theta}_{1,i} = 0$ , then (23.17) and the subdifferential of  $|\cdot|$  suggest that necessarily  $|\hat{\theta}_{\text{LS},i}| \leq \frac{\lambda}{2}$ . Concluding, we can write in a more compact way that

$$\hat{\theta}_{1,i} = \text{sgn}(\hat{\theta}_{\text{LS},i}) \left( |\hat{\theta}_{\text{LS},i}| - \frac{\lambda}{2} \right)_+, \quad (23.20)$$

where  $(\cdot)_+$  denotes the “positive part” of the respective argument; it is equal to the argument if this is non-negative, and zero otherwise. This is very interesting indeed. In contrast to the ridge regression that shrinks all coordinates of the unregularized LS solution by the same factor, LASSO forces all coordinates, whose absolute value is less than or equal to  $\lambda/2$ , to zero, and the rest of the coordinates are reduced, in absolute value, by the same amount  $\lambda/2$ . This is known as *soft thresholding*, to distinguish it from the *hard thresholding* operation; the latter is defined as  $\theta \cdot \chi_{(0,\infty)}(|\theta| - \frac{\lambda}{2})$ ,  $\theta \in \mathcal{R}$ , where  $\chi_{(0,\infty)}(\cdot)$  stands for the characteristic function with respect to the set  $(0, \infty)$ . Figure 23.4 shows the graphs illustrating the effect that the ridge regression, LASSO and hard thresholding have on the unregularized LS solution, as a function of its value (horizontal axis). Note that our discussion here, simplified via the orthonormal input matrix case, has quantified what we had said before about the tendency of the  $\ell_1$  norm to push small values to become *exactly zero*. This will be further strengthened, via a more rigorous mathematical formulation, in Section 1.23.6.

**Example 1.** Assume that the unregularized LS solution, for a given regression task,  $y = X\theta + \eta$ , is given by:

$$\hat{\theta}_{\text{LS}} = [0.2, -0.7, 0.8, -0.1, 1.0]^T.$$

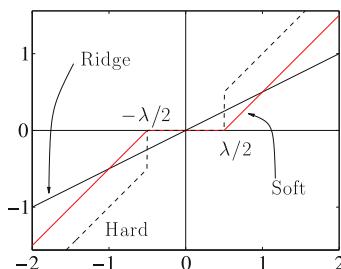


FIGURE 23.4

Output-input curves for the hard thresholding, soft thresholding operators together with the linear operator associated with the ridge regression, for the same value of  $\lambda = 1$ .

Derive the solutions for the corresponding ridge regression and  $\ell_1$  norm regularization tasks. Assume that the input matrix  $X$  has orthonormal columns and that the regularization parameter is  $\lambda = 1$ . Also, what is the result of hard thresholding the vector  $\hat{\theta}_{\text{LS}}$  with threshold equal to 0.5?

We know that the corresponding solution for the ridge regression is

$$\hat{\theta}_R = \frac{1}{1 + \lambda} \hat{\theta}_{\text{LS}} = [0.1, -0.35, 0.4, -0.05, 0.5]^T.$$

The solution for the  $\ell_1$  norm regularization is given by soft thresholding, with threshold equal to  $\lambda/2 = 0.5$ , hence the corresponding vector is

$$\hat{\theta}_1 = [0, -0.2, 0.3, 0, 0.5]^T.$$

The result of the hard thresholding operation is the vector  $[0, -0.7, 0.8, 0, 1.0]^T$ .

### Remarks 1.

- The hard and soft thresholding rules are only two possibilities out of a larger number of alternatives. Note that the hard thresholding operation is defined via a discontinuous function and this makes this rule to be unstable, in the sense of being very sensitive to small changes of the input. Moreover, this shrinking rule tends to exhibit large variance in the resulting estimates. The soft thresholding rule is a continuous function, but, as it is readily seen from the graph in Figure 23.4, it introduces bias even for the large values of the input argument. In order to ameliorate such shortcomings, a number of alternative thresholding operators have been introduced and studied both theoretically and experimentally. Although these are not within the mainstream of our interest, we provide two popular examples for the sake of completeness; the *Smoothly Clipped Absolute Deviation* (SCAD):

$$\hat{\theta}_{\text{SCAD}} = \begin{cases} \text{sgn}(\theta) (|\theta| - \lambda_{\text{SCAD}})_+, & |\theta| \leq 2\lambda_{\text{SCAD}}, \\ \frac{(\alpha-1)\theta - \alpha\lambda_{\text{SCAD}} \text{sgn}(\theta)}{\alpha-2}, & 2\lambda_{\text{SCAD}} < |\theta| \leq \alpha\lambda_{\text{SCAD}}, \\ \theta, & |\theta| > \alpha\lambda_{\text{SCAD}}, \end{cases}$$

and the *non-negative garrote* thresholding rule :

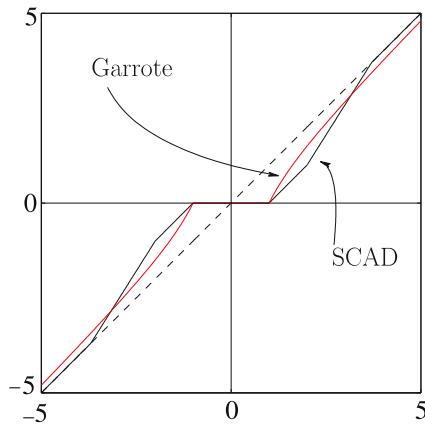
$$\hat{\theta}_{\text{garr}} = \begin{cases} 0, & |\theta| \leq \lambda_{\text{garr}}, \\ \theta - \frac{\lambda_{\text{garr}}^2}{\theta}, & |\theta| > \lambda_{\text{garr}}. \end{cases}$$

Figure 23.5 shows the respective graphs. Observe that, in both cases, an effort has been made to remove the discontinuity (associated with the hard thresholding) and to remove/reduce the bias for large values of the input argument. The parameter  $\alpha$  is a user-defined one. For a more detailed discussion on this topic, the interested reader can refer, for example, to [11].

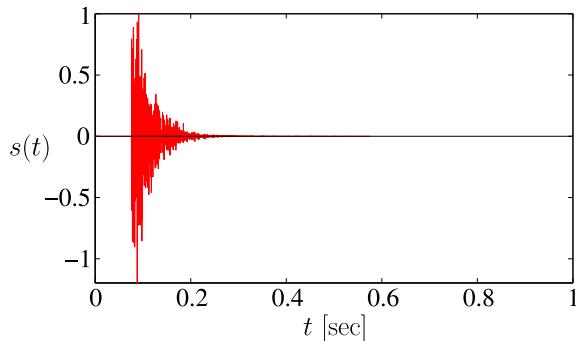
---

### 1.23.5 Sparse signal representation

In the previous section, we brought into our discussion the need for taking special care for zeros. Sparsity is an attribute that is met in a plethora of natural signals, since nature tends to be parsimonious. In this section, we will briefly present a number of application cases, where the existence of zeros in a

**FIGURE 23.5**

Output-input graph for the SCAD and nonnegative garrote rules with parameters  $\alpha = 3.7$ , and  $\lambda_{\text{SCAD}} = \lambda_{\text{garr}} = 1$ . Observe that both rules smooth out the discontinuity associated with the hard thresholding rule. Notice, also, that the SCAD rule removes the bias, associated with the soft thresholding rule, for large values of the input variable. On the contrary, the garrote thresholding rule allows some bias for large input values, which diminishes as  $\lambda_{\text{garr}}$  gets smaller and smaller.

**FIGURE 23.6**

The impulse response function of an echo-path in a telephone network. Observe that although it is of relatively short duration, it is not a priori known where exactly in time will occur.

mathematical expansion is of paramount importance, hence it justifies to further strengthen our search for and developing related analysis tools.

Echo cancelation is a major task in Communications. In a number of cases, the echo path, represented by a vector comprising the values of the impulse response samples, is a sparse one. This is the case, for example, in internet telephony and in acoustic and network environments, e.g., [12–14]. Figure 23.6 shows the impulse response of such an echo path. The impulse response of the echo path is of short duration; however, the delay with which it appears is not known. So, in order to model it, one has to

use a long impulse response, yet only a relatively small number of the coefficients will be significant and the rest will be close to zero. Of course, one could ask why not use an LMS or an RLS [15, 16] and eventually the significant coefficients will be identified. The answer is that this turns out not to be the most efficient way to tackle such problems, since the convergence of the algorithm can be very slow. In contrast, if one embeds, somehow, into the problem the a priori information concerning the existence of (almost) zero coefficients, then the convergence speed can be significantly increased and also better error floors can be attained.

A similar situation, as in the previous case, occurs in wireless communication systems, which involve multipath channels. A typical application is in high definition television (HDTV) systems, where the involved communications channels consist of a few non-negligible echoes, some of which may have quite large time delays with respect to the main signal, see, e.g., [17–20]. If the information signal is transmitted at high symbol rates through such a dispersive channel, then the introduced intersymbol interference (ISI) has a span of several tens up to hundreds of symbol intervals. This in turn implies that quite long channel estimators are required at the receiver's end in order to reduce effectively the ISI component of the received signal, although only a small part of it has values substantially different to zero. The situation is even more demanding whenever the channel frequency response exhibits deep nulls. More recently, sparsity has been exploited in channel estimation for multicarrier systems, both for single antenna as well as for MIMO systems [21, 22]. A thorough and in depth treatment related to sparsity in multipath communication systems is provided in [23].

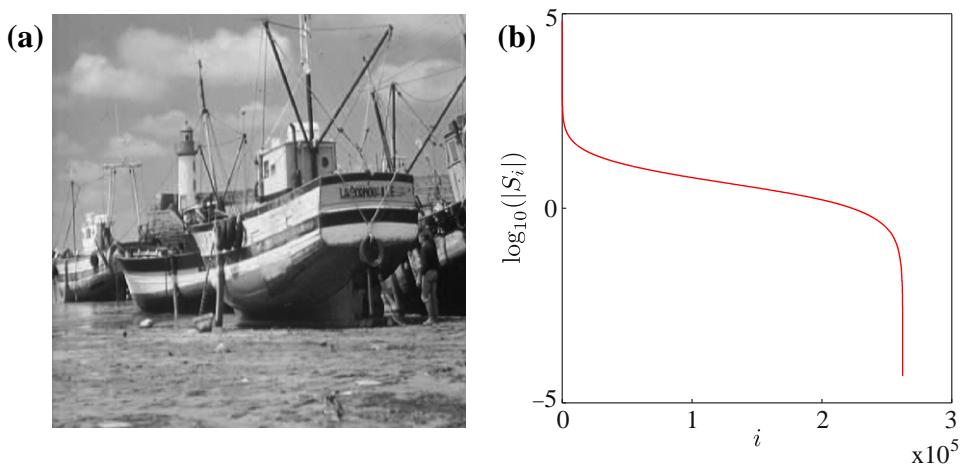
Another example, which might be more widely known, is that of signal compression. It turns out that if the signal modalities, with which we communicate, e.g., speech, and also we sense the world, e.g., images, audio, are transformed into a suitably chosen domain then they are sparsely represented; only a relatively small number of the signal components in this domain are large and the rest are close to zero. As an example, Figure 23.7a shows an image and Figure 23.7b the plot of the magnitude of the obtained Discrete Cosine Transform (DCT) components, which are computed by writing the corresponding image array as a vector in lexicographic order. Note that more than 95% of the total energy is contributed by only the 5% of the largest components. This is at the heart of any compression technique. Only the large coefficients are chosen to be coded and the rest are considered to be zero. Hence, significant gains are obtained in memory/bandwidth requirements while storing/transmitting such signals, without much perceptual loss. Depending on the modality, different transforms are used. For example, in JPEG-2000, an image array, represented in terms of a vector that contains the intensity of the gray levels of the image pixels, is transformed via the discrete wavelet transform (DWT) and results to a transform vector that comprises only a few large components. Such an operation is of the form

$$\mathbf{S} = \Phi^H \mathbf{s}, \quad \mathbf{s}, \mathbf{S} \in \mathcal{C}^l, \quad (23.21)$$

where  $\mathbf{s}$  is the vector of the “raw” signal samples,  $\mathbf{S}$  the (complex-valued) vector of the transformed ones, and  $\Phi$  is the  $l \times l$  transformation matrix. Often, this is an orthonormal matrix,  $\Phi^H \Phi = I$ . Basically, a transform is nothing else than a projection of a vector on a new set of coordinate axes, which comprise the columns of the transformation matrix  $\Phi$ . Celebrated examples of such transforms are the wavelet, the discrete Fourier (DFT) and the discrete cosine (DCT) transforms, e.g., [1]. In such cases, where the transformation matrix is orthonormal, one can write that

$$\mathbf{s} = \Psi \mathbf{S}, \quad (23.22)$$

where  $\Psi = \Phi$ . Eq. (23.21) is known as the *analysis* and (23.22) as the *synthesis* equation.



**FIGURE 23.7**

(a) A  $512 \times 512$  pixel image and (b) The magnitude of its Discrete Cosine Transform components in descending order and logarithmic scale. Note that more than 95% of the total energy is contributed by only the 5% of the largest components.

Compression via such transforms exploit the fact that many signals in nature, which are rich in context, can be *compactly* represented in an appropriately chosen basis, depending on the modality of the signal. Very often, the construction of such bases tries to “imitate” the sensory systems that the human (and not only) brain has developed in order to sense these signals; and we know that nature (in contrast to modern humans) does not like to waste resources. A standard compression task comprises the following stages: (a) Obtain the  $l$  components of  $S$ , via the analysis step (23.21), (b) keep the, say,  $k$  most significant of them, (c) code these values, as well as their respective locations in the transform vector  $S$ , and (d) obtain the (approximate) original signal  $s$ , when needed (after storage or transmission), via the synthesis Eq. (23.22), where in place of  $S$  only its  $k$  most significant components are used, which are the ones that were coded, while the rest are set equal to zero. However, there is something unorthodox in this process of compression, as it has been practised till very recently. One processes (transforms) large signal vectors of  $l$  coordinates, where  $l$  in practice can be quite large, and then uses only a small percentage of the transformed coefficients and the rest are simply ignored. Moreover, one has to store/transmit the location of the respective large coefficients that were finally coded. A natural question that is now raised is the following: Since  $S$  in the synthesis equation is (approximately) sparse, can one compute it via an alternative path than the analysis equation in (23.21)? The issue here is to investigate whether one could use a more informative way of obtaining measurements from the available raw data, so that less than  $l$  measurements are sufficient to recover all the necessary information. The ideal case would be to be able to recover it via a set of  $k$  such measurement samples, since this is the number of the significant free parameters. On the other hand, if this sounds a bit extreme, can one obtain  $N$  ( $k < N \ll l$ ) such signal-related measurements, from which one can obtain the  $k$  needed components of  $S$ ? It turns out that such an approach is possible and it leads to the solution of an *underdetermined* system of linear

equations, under the constraint that the unknown target vector is a sparse one. The importance of such techniques becomes even more apparent when, instead of an orthonormal basis, as discussed before, a more general type of expansion is adopted, in terms of what is known as *overcomplete dictionaries*.

A dictionary [24] is a collection of parameterized waveforms, which are discrete-time signal samples, represented as vectors  $\psi_i \in \mathcal{C}^l$ ,  $i \in \mathcal{I}$ . For example, the columns of a DFT or a DWT matrix comprise a dictionary. These are two examples of what is known as *complete* dictionaries, which consist of  $l$  (orthonormal) vectors, i.e., a number equal to the length of the signal vector. However, in many cases in practice, using such dictionaries is very restrictive. Let us take, for example, a segment of audio signal, from a news media or a video, that needs to be processed. This consists, in general, of different types of signals, namely speech, music, environmental sounds. For each type of these signals, different signal vectors may be more appropriate in the expansion for the analysis. For example, music signals are characterized by a strong harmonic content and the use of sinusoids seems to be best for compression, while for speech signals a Gabor type signal expansion (sinusoids of various frequencies weighted by sufficiently narrow pulses at different locations in time, [1, 25]), may be a better choice. The same applies when one deals with an image. Different parts of an image, e.g., parts which are smooth or contain sharp edges, may demand a different expansion vector set, for obtaining the best overall performance. The more recent tendency, in order to satisfy such needs, is to use *overcomplete* dictionaries. Such dictionaries can be obtained, for example, by concatenating different dictionaries together, e.g., a DFT and a DWT matrix to result in a combined  $l \times 2l$  transformation matrix. Alternatively, a dictionary can be “trained” in order to effectively represent a set of available signal exemplars, a task which is often referred to as dictionary learning [26–29]. While using such overcomplete dictionaries, the synthesis equation takes the form

$$\mathbf{s} = \sum_{i \in \mathcal{I}} \theta_i \psi_i. \quad (23.23)$$

Note that, now, the analysis is an ill-posed problem, since the elements  $\{\psi_i\}_{i \in \mathcal{I}}$  (usually called *atoms*) of the dictionary are not linearly independent, and there is not a unique set of coefficients  $\{\theta_i\}_{i \in \mathcal{I}}$  which generates  $\mathbf{s}$ . Moreover, we expect most of these coefficients to be (nearly) zero. Note that, in such cases, the cardinality of  $\mathcal{I}$  is larger than  $l$ . This necessarily leads to underdetermined systems of equations with infinite many solutions. The question that is now raised is whether we can exploit the fact that most of these coefficients are known to be zero, in order to come up with a unique solution, and if yes, under which conditions such a solution is possible?

Besides the previous examples, there is a number of cases where an underdetermined system of equations is the result of our inability to obtain a sufficiently large number of measurements, due to physical and technical constraints. This is for example the case in MRI imaging, which will be presented in more detail later on.

### 1.23.6 In quest for the sparsest solution

Inspired by the discussion in the previous section, we now turn our attention to the task of solving underdetermined systems of equations, by imposing the sparsity constraint on the solution [30]. We will develop the theoretical set up in the context of the regression task and we will adhere to the notation that has been adopted for this task. Moreover, we will focus on the real-valued data case, in order to

simplify the presentation. The theory can be readily extended to the more general complex data case, see, e.g., [31,32]. We assume that we are given a set of measurements,  $\mathbf{y} := [y_1, y_2, \dots, y_N]^T \in \mathcal{R}^N$ , according to the linear model

$$\mathbf{y} = X\boldsymbol{\theta}, \quad \mathbf{y} \in \mathcal{R}^N, \quad \boldsymbol{\theta} \in \mathcal{R}^l, \quad l > N, \quad (23.24)$$

where  $X$  is the  $N \times l$  input matrix, which is assumed to be of full row rank, i.e.,  $\text{rank}(X) = N$ . Our starting point is the noiseless case. The system in (23.24) is an underdetermined one and accepts an infinite number of solutions. The set of possible solutions lies in the intersection of the  $N$  hyperplanes<sup>2</sup> in the  $l$ -dimensional space,

$$\{\boldsymbol{\theta} \in \mathcal{R}^l : y_n = \mathbf{x}_n^T \boldsymbol{\theta}\}, \quad n = 1, 2, \dots, N.$$

We know from geometry, that the intersection of  $N$  non-parallel hyperplanes (which in our case is guaranteed by the fact that  $X$  has been assumed to be full row rank, hence  $\mathbf{x}_n$  are mutually independent) is a plane of dimensionality  $l - N$  (e.g., the intersection of two (non-parallel) (hyper) planes in the 3-dimensional space is a straight line; that is, a plane of dimensionality equal to one). In a more formal way, the set of all possible solutions, to be denoted as  $\Theta$ , is an *affine* set. An affine set is the translation of a linear subspace by a constant vector. Let us pursue this a bit further, since we will need it later on.

Let the null space of  $X$  be the set  $\text{null}(X)$ , defined as the linear subspace

$$\text{null}(X) = \left\{ \mathbf{z} \in \mathcal{R}^l : X\mathbf{z} = \mathbf{0} \right\}.$$

Obviously, if  $\boldsymbol{\theta}_0$  is a solution to (23.24), i.e.,  $\boldsymbol{\theta}_0 \in \Theta$ , then it is easy to verify that  $\forall \boldsymbol{\theta} \in \Theta, X(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = \mathbf{0}$ , or  $\boldsymbol{\theta} - \boldsymbol{\theta}_0 \in \text{null}(X)$ . As a result,

$$\Theta = \boldsymbol{\theta}_0 + \text{null}(X),$$

and  $\Theta$  is an affine set. We also know from linear algebra basics that the null space of a full row rank matrix,  $N \times l, l > N$ , is a subspace of dimensionality  $l - N$ . Figure 23.8 illustrates the case for one measurement sample in the 2-dimensional space,  $l = 2$  and  $N = 1$ . The set of solutions  $\Theta$  is a line, which is the translation of the linear subspace crossing the origin (the  $\text{null}(X)$ ). Therefore, if one wants to determine a *single* point that lies in the affine set of solutions,  $\Theta$ , then an extra constraint/a priori knowledge has to be imposed

In the sequel, three such possibilities are examined.

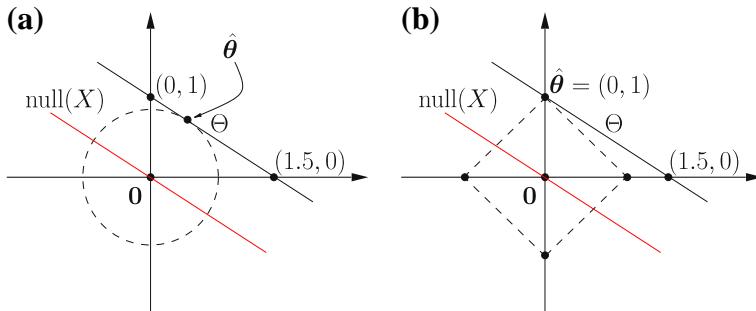
### 1.23.6.1 The $\ell_2$ norm minimizer

Our goal now becomes to pick a point in (the affine set)  $\Theta$ , that corresponds to the minimum  $\ell_2$  norm. This is equivalent to solving the following constrained task

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \quad & \|\boldsymbol{\theta}\|_2^2 \\ \text{s.t. } & \mathbf{x}_n^T \boldsymbol{\theta} = y_n, \quad n = 1, 2, \dots, N. \end{aligned} \quad (23.25)$$

---

<sup>2</sup>In  $\mathcal{R}^l$ , a hyperplane is of dimension  $l - 1$ . A plane has dimension lower than  $l - 1$ .

**FIGURE 23.8**

(a) The  $\ell_2$  norm minimizer. The dotted circle corresponds to the smallest  $\ell_2$  ball that intersects the set  $\Theta$ . As such, the intersection point,  $\hat{\theta}$ , is the  $\ell_2$  norm minimizer of the task (23.25). Notice that the vector  $\hat{\theta}$  contains no zero component. (b) The  $\ell_1$  norm minimizer. The dotted rhombus corresponds to the smallest  $\ell_1$  ball that intersects  $\Theta$ . Hence, the intersection point,  $\hat{\theta}$ , is the solution of the constrained  $\ell_1$  minimization task of (23.28). Notice that the obtained estimate  $\hat{\theta} = (0, 1)$  contains a zero.

The previous optimization task accepts a *unique* solution given in closed form as

$$\hat{\theta} = X^T (X X^T)^{-1} y. \quad (23.26)$$

The geometric interpretation of this solution is provided in Figure 23.8a, for the case of  $l = 2$  and  $N = 1$ . The radius of the Euclidean norm ball keeps increasing, till it touches the plane that contains the solutions. This point is the one with the minimum  $\ell_2$  norm or, equivalently, the point that lies closest to the origin. Equivalently, the point  $\hat{\theta}$  can be seen as the (metric) projection of  $\mathbf{0}$  onto  $\Theta$ .

Minimizing the  $\ell_2$  norm, in order to solve a linear set of underdetermined equations, has been used in various applications. The closest to us is in the context of determining the unknown coefficients in an expansion using an overcomplete dictionary of functions (vectors) [33]. A main drawback of this method is that it is not sparsity preserving. There is no guarantee that the solution in (23.26) will give zeros even if the true model vector  $\theta$  has zeros. Moreover, the method is *resolution limited* [7]. This means that, even if there may be a sharp contribution of specific atoms in the dictionary, this is not portrayed in the obtained solution. This is a consequence of the fact that the information provided by  $X X^T$  is a global one, containing all atoms of the dictionary in an “averaging” fashion, and the final result tends to smooth out the individual contributions, especially when the dictionary is overcomplete.

### 1.23.6.2 The $\ell_0$ norm minimizer

Now we turn our attention to the  $\ell_0$  norm (once more, it is pointed out that this is an abuse of the definition of the norm, as stated before), and we make sparsity our new flag under which a solution will be obtained. Recall from Section 1.23.5 that such a constraint is in line with the natural structure that underlies a number of applications. The task now becomes

$$\min_{\theta \in \mathcal{R}^l} \|\theta\|_0$$

$$\text{s.t. } \mathbf{x}_n^T \boldsymbol{\theta} = y_n, \quad n = 1, 2, \dots, N, \quad (23.27)$$

that is, from all the points that lie on the plane of all possible solutions find the *sparsest* one; i.e., the one with the least number of nonzero elements. As a matter of fact, such an approach is within the spirit of *Occam's razor* rule. It corresponds to the smallest number of parameters that can explain the obtained measurements. The points that are now raised are:

- Is a solution to this problem unique and under which conditions?
- Can a solution be obtained with low enough complexity in realistic time?

We postpone the answer to the first question later on. As for the second one, the news is no good. Minimizing the  $\ell_0$  norm under a set of linear constraints is a task of combinatorial nature and as a matter of fact the problem is, in general, NP-hard [34]. The way to approach the problem is to consider all possible combinations of zeros in  $\boldsymbol{\theta}$ , removing the respective columns of  $X$  in (23.24) and check whether the system of equations is satisfied; keep as solutions the ones with the smallest number of nonzero elements. Such a searching technique exhibits complexity of an exponential dependence on  $l$ . Figure 23.8a illustrates the two points ((1.5, 0) and (0, 1)) that comprise the solution set of minimizing the  $\ell_0$  norm for the single measurement (constraint) case.

### 1.23.6.3 The $\ell_1$ norm minimizer

The current task is now given by

$$\begin{aligned} & \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \|\boldsymbol{\theta}\|_1 \\ & \text{s.t. } \mathbf{x}_n^T \boldsymbol{\theta} = y_n, \quad n = 1, 2, \dots, N. \end{aligned} \quad (23.28)$$

Figure 23.8b illustrates the geometry. The  $\ell_1$  ball is increased till it touches the affine set of the possible solutions. For this specific geometry, the solution is the point (0, 1). In our discussion in Section 1.23.3, we saw that the  $\ell_1$  norm is the one, out of all  $\ell_p$ ,  $p \geq 1$  norms, that bears some similarity with the sparsity favoring (nonconvex)  $\ell_p$ ,  $p < 1$  “norms.” Also, we have commented that the  $\ell_1$  norm encourages zeros, when the respective values are small. In the sequel, we will state one lemma, that establishes this zero-favoring property in a more formal way. The  $\ell_1$  norm minimizer is also known as *Basis Pursuit* and it was suggested for decomposing a vector signal in terms of the atoms of an overcomplete dictionary [7].

The  $\ell_1$  minimizer can be brought into the standard Linear Programming (LP) form and then can be solved by recalling any related method; the simplex method or the more recent interior point methods are two possibilities, see, e.g., [9, 35]. Indeed, consider the (LP) task

$$\begin{aligned} & \min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \\ & \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \quad \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

To verify that our  $\ell_1$  minimizer can be cast in the previous form, notice first that any  $l$ -dimensional vector  $\boldsymbol{\theta}$  can be decomposed as

$$\boldsymbol{\theta} = \mathbf{u} - \mathbf{v}, \quad \mathbf{u} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0}.$$

Indeed, this holds true if, for example,

$$\mathbf{u} := \boldsymbol{\theta}_+, \quad \mathbf{v} := (-\boldsymbol{\theta})_+,$$

where  $\mathbf{x}_+$  stands for the vector obtained after taking the positive parts of the components of  $\mathbf{x}$ . Moreover, notice that

$$\|\boldsymbol{\theta}\|_1 = [1, 1, \dots, 1] \begin{bmatrix} \boldsymbol{\theta}_+ \\ (-\boldsymbol{\theta})_+ \end{bmatrix} = [1, 1, \dots, 1] \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}.$$

Hence, our  $\ell_1$  minimization task can be recast in the LP form, if

$$\begin{aligned} \mathbf{c} &:= [1, 1, \dots, 1]^T, \quad \mathbf{x} := [\mathbf{u}^T, \mathbf{v}^T]^T, \\ \mathbf{A} &:= [X, -X], \quad \mathbf{b} := \mathbf{y}. \end{aligned}$$

#### 1.23.6.4 Characterization of the $\ell_1$ norm minimizer

**Lemma 1.** An element  $\boldsymbol{\theta}$  in the affine set,  $\Theta$ , of the solutions of the underdetermined linear system (23.24), has minimal  $\ell_1$  norm if and only if the following condition is satisfied:

$$\left| \sum_{i:\theta_i \neq 0} \operatorname{sgn}(\theta_i) z_i \right| \leq \sum_{i:\theta_i=0} |z_i|, \quad \forall \mathbf{z} \in \operatorname{null}(X). \quad (23.29)$$

Moreover, the  $\ell_1$  minimizer is unique if and only if the inequality in (23.29) is a strict one for all  $\mathbf{z} \neq \mathbf{0}$  (see, e.g., [36]).

**Remarks 2.** The previous lemma has a very interesting and important consequence. If  $\hat{\boldsymbol{\theta}}$  is the *unique* minimizer of (23.28), then

$$\operatorname{card}\{i : \hat{\theta}_i = 0\} \geq \dim(\operatorname{null}(X)), \quad (23.30)$$

where  $\operatorname{card}\{\cdot\}$  denotes the cardinality of a set. In words, the number of zero coordinates of the unique minimizer cannot be smaller than the dimension of the null space of  $X$ . Indeed, if this is not the case, then the unique minimizer could have less zeros than the dimensionality of  $\operatorname{null}(X)$ . As it can easily be shown, this means that we can always find a  $\mathbf{z} \in \operatorname{null}(X)$ , which has zeros in the same locations where the coordinates of the unique minimizer are zero, and at the same time it is not identically zero, i.e.,  $\mathbf{z} \neq \mathbf{0}$ . However, this would violate (23.29), which in the case of uniqueness holds as a strict inequality.

**Definition 1.** A vector  $\boldsymbol{\theta}$  is called  $k$ -sparse if it has *at most*  $k$  nonzero components.

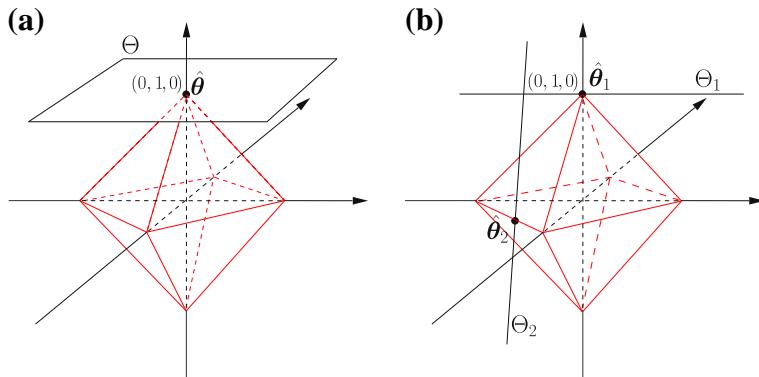
**Remarks 3.** If the minimizer of (23.28) is *unique*, then it is a  $k$ -sparse vector with

$$k \leq N.$$

This is a direct consequence of the Remark 2, and the fact that for the matrix  $X$ ,

$$\dim(\operatorname{null}(X)) = l - \operatorname{rank}(X) = l - N.$$

Hence, the number of the nonzero elements of the unique minimizer must be at most equal to  $N$ . If one resorts to geometry, all the previously stated results become crystal clear.

**FIGURE 23.9**

(a) The  $\ell_1$  ball intersecting with a plane. The only possible scenario, for the existence of a unique common intersecting point of the  $\ell_1$  ball with a plane in the Euclidean  $\mathcal{R}^3$  space, is for the point to be located at one of the corners of the  $\ell_1$  ball, i.e., to be an 1-sparse vector. (b) The  $\ell_1$  ball intersecting with lines. In this case, the sparsity level of the unique intersecting point is relaxed; it could be an 1- or a 2-sparse vector.

### 1.23.6.5 Geometric interpretation

Assume that our target solution resides in the 3-dimensional space and that we are given one measurement

$$y_1 = \mathbf{x}_1^T \boldsymbol{\theta} = x_{11}\theta_1 + x_{12}\theta_2 + x_{13}\theta_3.$$

Then the solution lies in the 2-dimensional (hyper)plane, which is described by the previous equation. To get the minimal  $\ell_1$  solution we keep increasing the size of the  $\ell_1$  ball<sup>3</sup> (the set of all points that have equal  $\ell_1$  norm) till it touches this plane. The only way that these two geometric objects have a single point in common (unique solution) is when they meet at a corner of the diamond. This is shown in Figure 23.9a. In other words, the resulting solution is 1-sparse, having two of its components equal to zero. This complies with the finding stated in Remark 3, since now  $N = 1$ . For any other orientation of the plane, this will either cut across the  $\ell_1$  ball or will share with the diamond an edge or a side. In both cases, there will be infinite many solutions.

Let us now assume that we are given an extra measurement,

$$y_2 = x_{21}\theta_1 + x_{22}\theta_2 + x_{23}\theta_3.$$

The solution now lies in the intersection of the two previous planes, which is a straight line. However, now, we have more alternatives for a unique solution. A line, e.g.,  $\Theta_1$ , can either touch the  $\ell_1$  ball at a corner (1-sparse solution) or, as it is shown in Figure 23.9b, it can touch the  $\ell_1$  ball at one of its edges, e.g.,  $\Theta_2$ . The latter case, corresponds to a solution that lies on a 2-dimensional subspace, hence it will be a 2-sparse vector. This also complies with the findings stated in Remark 3, since in this case, we have  $N = 2, l = 3$  and the sparsity level for a unique solution can be either 1 or 2.

<sup>3</sup>Observe that in the 3-dimensional space the  $\ell_1$  ball looks like a diamond.

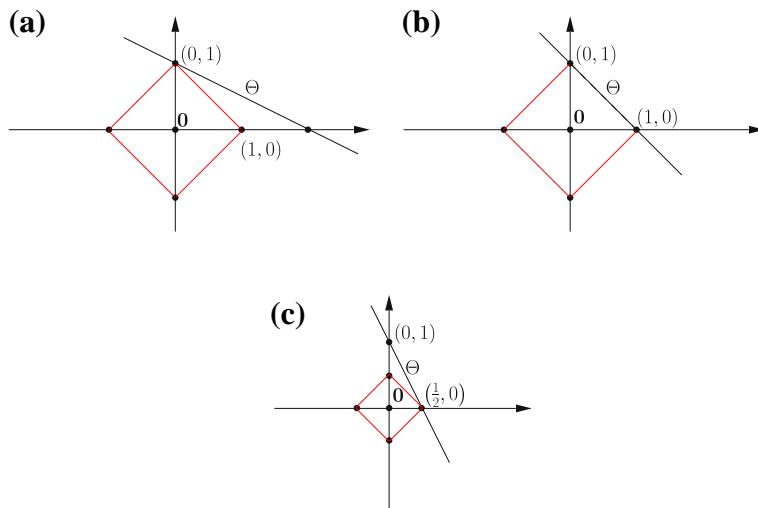
Note that uniqueness is associated with the particular geometry and orientation of the affine set, which is the set of all possible solutions of the underdetermined system of equations. For the case of the square  $\ell_2$  norm, the solution was always unique. This is a consequence of the (hyper)spherical shape formed by the Euclidean norm. From a mathematical point of view, the square  $\ell_2$  norm is a strict convex function. This is not the case for the  $\ell_1$  norm, which is convex, albeit not a strict convex function.

**Example 2.** Consider a sparse vector parameter  $[0, 1]^T$ , which we assume to be unknown. We will use one measurement to *sense* it. Based on this single measurement, we will use the  $\ell_1$  minimizer of (23.28) to recover its true value. Let us see what happens. We will consider three different values of the “sensing” (input) vector  $x$  in order to obtain the measurement  $y = x^T \theta$ : (a)  $x = [\frac{1}{2}, 1]^T$ , (b)  $x = [1, 1]^T$ , and (c)  $x = [2, 1]^T$ . The resulting measurement, after sensing  $\theta$  by  $x$ , is  $y = 1$  for all the three previous cases.

Case (a): The solution will lie on the straight line

$$\Theta = \left\{ [\theta_1, \theta_2]^T \in \mathbb{R}^2 : \frac{1}{2}\theta_1 + \theta_2 = 1 \right\},$$

which is shown in Figure 23.10a. For this setting, expanding the  $\ell_1$  ball, this will touch the line (our solutions’ affine set) at the corner  $[0, 1]^T$ . This is a unique solution, hence it is sparse, and it coincides with the true value.



**FIGURE 23.10**

(a) Sensing with  $x = [\frac{1}{2}, 1]^T$ , (b) sensing with  $x = [1, 1]^T$ , and (c) sensing with  $x = [2, 1]^T$ . The choice of the sensing vector  $x$  is crucial to unveiling the true sparse solution  $(0, 1)$ . Only the sensing vector  $x = [\frac{1}{2}, 1]^T$  identifies uniquely the desired  $(0, 1)$ .

Case (b): The solutions lies on the straight line

$$\Theta = \left\{ [\theta_1, \theta_2]^T \in \mathcal{R}^2 : \theta_1 + \theta_2 = 1 \right\},$$

which is shown in Figure 23.10b. For this set up, there is an infinite number of solutions, including two sparse ones.

Case (c): The affine set of solutions is described by

$$\Theta = \left\{ [\theta_1, \theta_2]^T \in \mathcal{R}^2 : 2\theta_1 + \theta_2 = 1 \right\},$$

which is sketched in Figure 23.10c. The solution in this case is sparse, but it is not the correct one.

This example is quite informative. *If we sense (measure) our unknown parameter vector with appropriate sensing (input) data, the use of the  $\ell_1$  norm can unveil the true value of the parameter vector, even if the system of equations is underdetermined, provided that the true parameter is sparse.* This now becomes our new goal. To investigate whether what we have just said can be generalized, and under which conditions holds true, if it does. In such a case, the choice of the regressors (which we just called them sensing vectors) and hence the input matrix (which, from now on, we will refer to, more and more frequently, as the sensing matrix) acquire an extra significance. It is not enough for the designer to care only for the rank of the matrix, i.e., the linear independence of the sensing vectors. One has to make sure that the corresponding affine set of the solutions has such an orientation, so that the touch with the  $\ell_1$  ball, as this increases from zero to meet this plane, is a “gentle” one, i.e., they meet at a single point, and more important at the correct one; that is, at the point that represents the true value of the sparse parameter, which we are searching for.

#### Remarks 4.

- Often in practice, the columns of the input matrix,  $X$ , are normalized to unit  $\ell_2$  norm. Although  $\ell_0$  norm is insensitive to the values of the nonzero components of  $\theta$ , this is not the case with the  $\ell_1$  and  $\ell_2$  norms. Hence, while trying to minimize the respective norms, and at the same time to fulfill the constraints, components that correspond to columns of  $X$  with high energy (norm) are favored more than the rest. Hence, the latter become more popular candidates to be pushed to zero. In order to avoid such situations, the columns of  $X$  are normalized to unity, by dividing each element of the column vector by the respective (Euclidean) norm.

---

## 1.23.7 Uniqueness of the $\ell_0$ minimizer

Our first goal is to derive *sufficient* conditions that guarantee uniqueness of the  $\ell_0$  minimizer, which has been defined in Section 1.23.6.

**Definition 2.** The *spark* of a full rank  $N \times l$  ( $l \geq N$ ) matrix,  $X$ , denoted as  $\text{spark}(X)$ , is the *smallest* number of its linearly dependent columns.

According to the previous definition, *any*  $m < \text{spark}(X)$  columns of  $X$  are, necessarily, *linearly independent*. The spark of a square,  $N \times N$ , full rank matrix is equal to  $N + 1$ .

**Remarks 5.**

- In contrast to the rank of a matrix, which can be easily determined, its spark can only be obtained by resorting to a combinatorial search over all possible combinations of the columns of the respective matrix, see, e.g., [8, 37]. The notion of the spark was used in the context of sparse representation, under the name of *Uniqueness Representation Property*, in [38]. The name “spark” was coined in [37]. An interesting discussion relating this matrix index with other indices, used in other disciplines, is given in [8].

**Example 3.** Consider the following matrix

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The matrix has rank equal to 4 and spark equal to 3. Indeed, any pair of columns are linearly independent. On the other hand, the first, the second and the fifth columns are linearly dependent. The same is also true for the combination of the second, third and sixth columns.

**Lemma 2.** *If  $\text{null}(X)$  is the null space of  $X$ , then*

$$\|\boldsymbol{\theta}\|_0 \geq \text{spark}(X), \quad \forall \boldsymbol{\theta} \in \text{null}(X), \boldsymbol{\theta} \neq \mathbf{0}.$$

**Proof.** To derive a contradiction, assume that there exists a  $\mathbf{0} \neq \boldsymbol{\theta} \in \text{null}(X)$  such that  $\|\boldsymbol{\theta}\|_0 < \text{spark}(X)$ . Since by definition  $X\boldsymbol{\theta} = \mathbf{0}$ , there exists a number of  $\|\boldsymbol{\theta}\|_0$  columns of  $X$  that are linearly dependent. However, this contradicts the minimality of  $\text{spark}(X)$ , and the claim of Lemma 2 is established.

**Lemma 3.** *If a linear system of equations,  $X\boldsymbol{\theta} = \mathbf{y}$ , has a solution that satisfies*

$$\|\boldsymbol{\theta}\|_0 < \frac{1}{2}\text{spark}(X),$$

*then this is the sparsest possible solution. In other words, this is, necessarily, the unique solution of the  $\ell_0$  minimizer.*

**Proof.** Consider any other solution  $\mathbf{h} \neq \boldsymbol{\theta}$ . Then,  $\boldsymbol{\theta} - \mathbf{h} \in \text{null}(X)$ , i.e.,

$$X(\boldsymbol{\theta} - \mathbf{h}) = \mathbf{0}.$$

Thus, according to Lemma 2,

$$\text{spark}(X) \leq \|\boldsymbol{\theta} - \mathbf{h}\|_0 \leq \|\boldsymbol{\theta}\|_0 + \|\mathbf{h}\|_0. \tag{23.31}$$

Observe that although the  $\ell_0$  “norm” is not a true norm, it can be readily verified by simple inspection and reasoning that the triangular property is satisfied. Indeed, by adding two vectors together, the resulting

number of nonzero elements will always be at most equal to the total number of nonzero elements of the two vectors. Therefore, if  $\|\theta\|_0 < \frac{1}{2}\text{spark}(X)$ , then (23.31) suggests that

$$\|\boldsymbol{h}\|_0 > \frac{1}{2}\text{spark}(X) > \|\theta\|_0.$$

### Remarks 6.

- Lemma 3 is a very interesting result. We have a sufficient condition to check whether a solution is the unique optimal in a, generally, NP-hard problem. Of course, although this is nice from a theoretical point of view, is not of much use by itself, since the related bound (the spark) can only be obtained after a combinatorial search. Well, in the next section, we will see that we can relax the bound by involving another index, in place of the spark, which can be easily computed.
- An obvious consequence of the previous lemma is that if the unknown parameter vector is a sparse one with  $k$  nonzero elements, then if matrix  $X$  is chosen so that to have  $\text{spark}(X) > 2k$ , then the true parameter vector is necessarily the sparsest one that satisfies the set of equations, and the (unique) solution to the  $\ell_0$  minimizer.
- In practice, the goal is to sense the unknown parameter vector by a matrix that has as high a spark as possible, so that the previously stated sufficiency condition to cover a wide range of cases. For example, if the spark of the input matrix is, say, equal to three, then one can check for optimal sparse solutions up to a sparsity level of  $k = 1$ . From the respective definition, it is easily seen that the values of the spark are in the range  $1 < \text{spark}(X) \leq N + 1$ .
- Constructing an  $N \times l$  matrix  $X$  in a random manner, by generating i.i.d. entries, guarantees, with high probability, that  $\text{spark}(X) = N + 1$ ; that is, any  $N$  columns of the matrix are linearly independent.

#### 1.23.7.1 Mutual coherence

Since the spark of a matrix is a number that is difficult to compute, our interest shifts to another index, which can be derived easier and at the same time can offer a useful bound on the spark. The *mutual coherence* of an  $N \times l$  matrix  $X$  [24], denoted as  $\mu(X)$ , is defined as

$$\mu(X) := \max_{1 \leq i < j \leq l} \frac{|\mathbf{x}_i^T \mathbf{x}_j|}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \quad (23.32)$$

where  $\mathbf{x}_i$ ,  $i = 1, 2, \dots, l$ , denote the columns of  $X$  (notice the difference in notation between a row  $\mathbf{x}_i^T$  and a column  $\mathbf{x}_i$  of the matrix  $X$ ). This number reminds us of the correlation coefficient between two random variables. Mutual coherence is bounded as  $0 \leq \mu(X) \leq 1$ . For a square orthogonal matrix,  $X$ ,  $\mu(X) = 0$ . For general matrices, with  $l > N$ ,  $\mu(X)$  satisfies

$$\sqrt{\frac{l-N}{N(l-1)}} \leq \mu(X) \leq 1,$$

which is known as the *Welch bound* [39]. For large values of  $l$ , the lower bound becomes, approximately,  $\mu(X) \geq \frac{1}{\sqrt{N}}$ . Common sense reasoning guides us to construct input (sensing) matrices of mutual

coherence as small as possible. Indeed, the purpose of the sensing matrix is to “measure” the components of the unknown vector and “store” this information in the measurement vector  $\mathbf{y}$ . Thus, this should be done in such a way so that  $\mathbf{y}$  to retain as much information about the components of  $\boldsymbol{\theta}$  as possible. This can be achieved if the columns of the sensing matrix,  $X$ , are as “independent” as possible. Indeed,  $\mathbf{y}$  is the result of a combination of the columns of  $X$ , each one weighted by a different component of  $\boldsymbol{\theta}$ . Thus, if the columns are as much “independent” as possible then the information regarding each component of  $\boldsymbol{\theta}$  is contributed by a different direction making its recovery easier. This is easier understood if  $X$  is a square orthogonal matrix. In the more general case of a non-square matrix, the columns should be made as “orthogonal” as possible.

**Example 4.** Assume that  $X$  is an  $N \times 2N$  matrix, formed by concatenating two orthonormal bases together,

$$X = [I, W],$$

where  $I$  is the identity matrix, having as columns the vectors  $\mathbf{e}_i$ ,  $i = 1, 2, \dots, N$ , with elements equal to

$$\delta_{ir} = \begin{cases} 1, & \text{if } i = r, \\ 0, & \text{if } i \neq r, \end{cases}$$

for  $r = 1, 2, \dots, N$ . The matrix  $W$  is the orthonormal DFT matrix, defined as

$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix},$$

where

$$W_N := \exp\left(-j\frac{2\pi}{N}\right).$$

Such an overcomplete dictionary could be used to represent signal vectors in terms of the expansion in (23.23), that comprise the sum of sinusoids with very narrow spiky-like pulses. The inner products between any two columns of  $I$  and between any two columns of  $W$  are zero, due to orthogonality. On the other hand, it is easy to see that the inner product between any column of  $I$  and any column of  $W$  has absolute value equal to  $\frac{1}{\sqrt{N}}$ . Hence, the mutual coherence of this matrix is  $\mu(X) = \frac{1}{\sqrt{N}}$ . Moreover, observe that the spark of this matrix is  $\text{spark}(X) = N + 1$ .

**Lemma 4.** For any  $N \times l$  matrix  $X$ , the following inequality holds

$$\text{spark}(X) \geq 1 + \frac{1}{\mu(X)}. \quad (23.33)$$

The proof is given in [37] and it is based on arguments that stem from matrix theory applied on the Gram matrix,  $X^T X$ , of  $X$ . A “superficial” look at the previous bound is that for very small values of  $\mu(X)$  the spark can be larger than  $N + 1$ ! Looking at the proof, it is seen that in such cases the spark of the matrix attains its maximum value  $N + 1$ .

The result complies with a common sense reasoning. The smaller the value of  $\mu(X)$  the more independent are the columns of  $X$ , hence the higher the value of its spark is expected to be. Based on this lemma, we can now state the following theorem, first given in [37]. Combining the way that Lemma 3 is proved and (23.33), we come to the following important theorem.

**Theorem 1.** *If the linear system of equations in (23.24) has a solution that satisfies the condition*

$$\|\theta\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(X)} \right), \quad (23.34)$$

*then this solution is the sparsest one.*

**Remarks 7.**

- The bound in (23.34) is “psychologically” important. It relates an easily computed bound to check whether the solution to a NP-hard task is the optimal one. However, it is not a particularly good bound and it restricts the range of values in which it can be applied. As we saw in Example 4, while the maximum possible value of the spark of a matrix was equal to  $N + 1$ , the minimum possible value of the mutual coherence was  $\frac{1}{\sqrt{N}}$ . Therefore, the bound based on the mutual coherence restricts the range of sparsity, i.e.,  $\|\theta\|_0$ , where one can check optimality, to around  $\frac{1}{2}\sqrt{N}$ . Moreover, as the previously stated Welch bound suggests, this  $\mathcal{O}(\frac{1}{\sqrt{N}})$  dependence of the mutual coherence seems to be a more general trend and not only the case for Example 4, see, e.g., [40]. On the other hand, as we have already stated in the Remarks 6, one can construct random matrices with spark equal to  $N + 1$ ; hence, using the bound based on the spark, one could expand the range of sparse vectors up to  $\frac{1}{2}N$ .

## 1.23.8 Equivalence of $\ell_0$ and $\ell_1$ minimizers: sufficiency conditions

We have now come to the crucial point and we will establish the conditions that guarantee the equivalence between the  $\ell_1$  and the  $\ell_0$  minimizers. Hence, under such conditions, a problem, that is in general NP-hard problem, *can be solved via a tractable convex optimization task*. Under these conditions, the zero value encouraging nature of the  $\ell_1$  norm, that has already been discussed, obtains a much higher stature; it provides the sparsest solution.

### 1.23.8.1 Condition implied by the mutual coherence number

**Theorem 2.** *Let the underdetermined system of equations*

$$y = X\theta,$$

*where  $X$  is an  $N \times l$  ( $N < l$ ) full row rank matrix. If a solution exists and satisfies the condition*

$$\|\theta\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(X)} \right), \quad (23.35)$$

*then this is the unique solution of both, the  $\ell_0$  as well the  $\ell_1$  minimizers.*

This is a very important theorem and it was shown independently in [37, 41]. Earlier versions of the theorem addressed the special case of a dictionary comprising two orthonormal bases, [40, 42]. A proof is also summarized in [8]. This theorem established, for a first time, what it was till then empirically known: often, the  $\ell_1$  and  $\ell_0$  minimizers result in the same solution.

### Remarks 8.

- The theory that we have presented so far is very satisfying, since it offers the theoretical framework and conditions that guarantee uniqueness of a sparse solution to an underdetermined system of equations. Now we know that, under certain conditions, the solution, which we obtain by solving the convex  $\ell_1$  minimization task, is the (unique) sparsest one. However, from a practical point of view, the theory, which is based on mutual coherence, does not say the whole story and falls short to predict what happens in practice. Experimental evidence suggests that the range of sparsity levels, for which the  $\ell_0$  and  $\ell_1$  tasks give the same solution, is much wider than the range guaranteed by the mutual coherence bound. Hence, there is a lot of theoretical happening in order to improve this bound. A detailed discussion is beyond the scope of this paper. In the sequel, we will present one of these bounds, since it is the one that currently dominates the scene. For more details and a related discussion the interested reader may consult, e.g., [43].

#### 1.23.8.2 The restricted isometry property (RIP)

**Definition 3.** For each integer  $k = 1, 2, \dots$ , define the *isometry constant*  $\delta_k$  of an  $N \times l$  matrix  $X$  as the *smallest* number such that

$$(1 - \delta_k) \|\boldsymbol{\theta}\|_2^2 \leq \|X\boldsymbol{\theta}\|_2^2 \leq (1 + \delta_k) \|\boldsymbol{\theta}\|_2^2, \quad (23.36)$$

holds true for *all*  $k$ -sparse vectors  $\boldsymbol{\theta}$ .

This definition was introduced in [44]. We loosely say that matrix  $X$  obeys the RIP of order  $k$  if  $\delta_k$  is not too close to one. When this property holds true, it implies that the Euclidean norm of  $\boldsymbol{\theta}$  is approximately *preserved*, after projecting it onto the rows of  $X$ . Obviously, if matrix  $X$  were orthonormal then  $\delta_k = 0$ . Of course, since we are dealing with non-square matrices this is not possible. However, the closer  $\delta_k$  is to zero, the closer to orthonormal *all* subsets of  $k$  columns of  $X$  are. Another view point of (23.36) is that it preserves Euclidean distances between  $k$ -sparse vectors. Let us consider two  $k$ -sparse vectors,  $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$  and apply (23.36) to their difference  $\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2$ , which, in general, is a  $2k$ -sparse vector. Then we obtain

$$(1 - \delta_{2k}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2 \leq \|X(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)\|_2^2 \leq (1 + \delta_{2k}) \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^2. \quad (23.37)$$

Thus, when  $\delta_{2k}$  is small enough, the Euclidean distance is preserved after projection in the lower dimensional measurements' space. In words, if the RIP holds true, this means that searching for a sparse vector in the lower dimensional subspace formed by the measurements,  $\mathcal{R}^N$ , and not in the original  $l$ -dimensional space, one can still recover the vector since distances are preserved and the target vector is not "confused" with others. After projection onto the rows of  $X$ , the discriminatory power of the method is retained. It is interesting to point out that the RIP is also related to the condition

number of the Gramian matrix. In [44,45], it is pointed out that if  $X_r$  denotes the matrix that results by considering only  $r$  of the columns of  $X$ , then the RIP in (23.36) is equivalent with requiring the respective Gramian,  $X_r^T X_r$ ,  $r \leq k$ , to have its eigenvalues within the interval  $[1 - \delta_k, 1 + \delta_k]$ . Hence, the more well conditioned the matrix is, the better is for us to dig out the information hidden in the lower dimensional measurements space.

**Theorem 3.** *Assume that for some  $k$ ,  $\delta_{2k} < \sqrt{2} - 1$ . Then the solution to the  $\ell_1$  minimizer of (23.28), denoted as  $\boldsymbol{\theta}_*$ , satisfies the following two conditions*

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_*\|_1 \leq C_0 \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_1, \quad (23.38)$$

and

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_*\|_2 \leq C_0 k^{-\frac{1}{2}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_1, \quad (23.39)$$

for some constant  $C_0$ . In the previously stated formulas,  $\boldsymbol{\theta}$  is the true (target) vector that generates the measurements in (23.28) and  $\boldsymbol{\theta}_k$  is the vector that results from  $\boldsymbol{\theta}$  if we keep its  $k$  largest components and set the rest equal to zero, [44,46–48].

Hence, if the true vector is a sparse one, i.e.,  $\boldsymbol{\theta} = \boldsymbol{\theta}_k$ , then the  $\ell_1$  minimizer recovers the (unique) exact value. On the other hand, if the true vector is not a sparse one, then the minimizer results in a solution whose accuracy is dictated by a genie-aided procedure that knew in advance the locations of the  $k$  largest components of  $\boldsymbol{\theta}$ . This is a groundbreaking result. Moreover, it is deterministic, it is always true and not with high probability. Note that the isometry property of order  $2k$  is used, since at the heart of the method lies our desire to preserve the norm of the differences between vectors.

Let us now focus on the case where there is a  $k$ -sparse vector that generates the measurements, i.e.,  $\boldsymbol{\theta} = \boldsymbol{\theta}_k$ . Then it is shown in [46] that the condition  $\delta_{2k} < 1$  guarantees that the  $\ell_0$  minimizer has a unique  $k$ -sparse solution. In other words, in order to get the equivalence between the  $\ell_1$  and  $\ell_0$  minimizers, the range of values for  $\delta_{2k}$  has to be decreased to  $\delta_{2k} < \sqrt{2} - 1$ , according to Theorem 3. This sounds reasonable. If we relax the criterion and use  $\ell_1$  instead of  $\ell_0$ , then the sensing matrix has to be more carefully constructed. Although we are not going to provide the proofs of these theorems here, since their formulation is well beyond the scope of this chapter, it is interesting to follow what happens if  $\delta_{2k} = 1$ . This will give us a flavor of the essence behind the proofs. If  $\delta_{2k} = 1$ , the left hand side term in (23.37) becomes zero. In this case, there may exist two  $k$ -sparse vectors  $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$  such that  $X(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) = \mathbf{0}$ , or  $X\boldsymbol{\theta}_1 = X\boldsymbol{\theta}_2$ . Thus, it is not possible to recover all  $k$ -sparse vectors, after projecting them in the measurements space, by any method.

The previous argument also establishes a connection between RIP and the spark of a matrix. Indeed, if  $\delta_{2k} < 1$ , this guarantees that any number of columns of  $X$  up to  $2k$  are linearly independent, since for any  $2k$ -sparse  $\boldsymbol{\theta}$ , (23.36) guarantees that  $\|X\boldsymbol{\theta}\|_2 > 0$ . This implies that  $\text{spark}(X) > 2k$ . A connection between RIP and the coherence is established in [49], where it is shown that if  $X$  has coherence  $\mu(X)$ , and unit norm columns, then  $X$  satisfies the RIP of order  $k$  with  $\delta_k \leq (k-1)\mu(X)$ .

### 1.23.8.2.1 Constructing matrices that obey the RIP of order $k$

It is apparent from our previous discussion that the higher the value of  $k$ , for which the RIP property of a matrix,  $X$ , holds true, the better, since a larger range of sparsity levels can be handled. Hence, a main goal towards this direction is to construct such matrices. It turns out that verifying the RIP for

a matrix of a general structure is a difficult task. This reminds us of the spark of the matrix, which is also a difficult task to compute. However, it turns out that for a certain class of random matrices, the RIP follows fairly easy. Thus, constructing such sensing matrices has dominated the scene of related research. We will present a few examples of such matrices, which are also very popular in practice, without going into details of the proofs, since this is out of our scope and the interested reader may dig this information from the related references.

Perhaps, the most well known example of a random matrix is the Gaussian one, where the entries  $X(i, j)$  of the sensing matrix are i.i.d. realizations from a Gaussian pdf  $\mathcal{N}(0, \frac{1}{N})$ . Another popular example of such matrices is constructed by sampling i.i.d. entries from a Bernoulli, or related, distributions

$$X(i, j) = \begin{cases} \frac{1}{\sqrt{N}}, & \text{with probability } \frac{1}{2}, \\ -\frac{1}{\sqrt{N}}, & \text{with probability } \frac{1}{2}, \end{cases}$$

or

$$X(i, j) = \begin{cases} +\sqrt{\frac{3}{N}}, & \text{with probability } \frac{1}{6}, \\ 0, & \text{with probability } \frac{2}{3}, \\ -\sqrt{\frac{3}{N}}, & \text{with probability } \frac{1}{6}. \end{cases}$$

Finally, one can adopt the uniform distribution and construct the columns of  $X$  by sampling uniformly at random on the unit sphere in  $\mathbb{R}^N$ . It turns out, that such matrices obey the RIP of order  $k$ , with overwhelming probability, provided that the number of measurements,  $N$ , satisfy the following inequality

$$N \geq Ck \ln(l/k), \quad (23.40)$$

where  $C$  is some constant, which depends on the isometry constant  $\delta_k$ . In words, having such a matrix at our disposal, one can recover a  $k$ -sparse vector from  $N < l$  measurements, where  $N$  is larger than the sparsity level by an amount controlled by the inequality (23.40). More on these issues can be obtained from, e.g., [45, 50].

Besides random matrices, one can construct other matrices that obey the RIP. One such example includes the partial Fourier matrices, which are formed by selecting uniformly at random  $N$  rows drawn from the  $l \times l$  DFT matrix. Although the required number of samples for the RIP to be satisfied may be larger than the bound in (23.40) (see, [51]), Fourier-based sensing matrices offer certain computational advantages, when it comes to storage ( $\mathcal{O}(N \ln l)$ ) and matrix-vector products ( $\mathcal{O}(l \ln l)$ ), [52]. In [53], the case of random Toeplitz sensing matrices, containing statistical dependencies across rows, is considered and it is shown that they can also satisfy the RIP with high probability. This is of particular importance in signal processing and communications applications, where it is very common for a system to be excited in its input via a time series, hence independence between successive input rows cannot be assumed. In [54, 55], the case of separable matrices is considered where the sensing matrix is the result of a Kronecker product of matrices, which satisfy the RIP individually. Such matrices are of interest for multidimensional signals, in order to exploit the sparsity structure along each one of the involved

dimensions. For example, such signals may occur while trying to “encode” information associated with an event whose activity spreads across the temporal, spectral, spatial, etc., domains.

In spite of their theoretical elegance, the derived bounds, that determine the number of the required measurements for certain sparsity levels, fall short of what is the experimental evidence, e.g., [43]. In practice, a rule of thumb is to use  $N$  of the order of  $3k - 5k$ , e.g., [46]. For large values of  $l$ , compared to the sparsity level, the analysis in [56] suggests that we can recover most sparse signals when  $N \approx 2k \ln(l/N)$ . In an effort to overcome the shortcomings associated with the RIP, a number of other techniques have been proposed, e.g., [43, 57–59]. Furthermore, in specific applications, the use of an empirical study may be a more appropriate path.

Note that, in principle, the minimum number of measurements that are required to recover a  $k$  sparse vector from  $N < l$  measurements is  $N \geq 2k$ . Indeed, in the spirit of the discussion after Theorem 3, the main requirement that a sensing matrix must fulfill is the following: not to map two different  $k$ -sparse vectors to the same measurement vector  $\mathbf{y}$ . Otherwise, one can never recover both vectors from their (common) measurements. If we have  $2k$  measurements and a sensing matrix that guarantees that any  $2k$  columns are linearly independent, then the previously stated requirement is readily seen that it is satisfied. However, the bounds on the number of measurements set in order the respective matrices to satisfy the RIP are higher. This is because RIP accounts also for the stability of the recovery process. We will come to this issue soon, in Section 1.23.10, where we talk about *stable* embeddings.

### 1.23.9 Robust sparse signal recovery from noisy measurements

In the previous section, our focus was on recovering a sparse solution from an underdetermined system of equations. In the formulation of the problem, we assumed that there is no noise in the obtained measurements. Having acquired some experience and insight from a simpler problem, we now turn our attention to the more realistic task, where uncertainties come into the scene. One type of uncertainty may be due to the presence of noise and our measurements’ model comes back to the standard regression form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\eta}, \quad (23.41)$$

where  $\mathbf{X}$  is our familiar non-square  $N \times l$  matrix. A sparsity-aware formulation for recovering  $\boldsymbol{\theta}$  from (23.41) can be cast as

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \quad & \|\boldsymbol{\theta}\|_1 \\ \text{s.t.} \quad & \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 \leq \epsilon, \end{aligned} \quad (23.42)$$

which coincides with the LASSO task given in (23.15). Such a formulation implicitly assumes that the noise is bounded and the respective range of values is controlled by  $\epsilon$ . One can consider a number of different variants. For example, one possibility would be to minimize the  $\|\cdot\|_0$  norm instead of the  $\|\cdot\|_1$ , albeit loosing the computational elegance of the latter. An alternative route would be to replace the Euclidean norm in the constraints with another one.

Besides the presence of noise, one could see the previous formulation from a different perspective. The unknown parameter vector,  $\boldsymbol{\theta}$ , may not be exactly sparse, but it may consist of a few large components,

while the rest are small and close to, yet not necessarily equal to, zero. Such a model misfit can be accommodated by allowing a deviation of  $\mathbf{y}$  from  $X\boldsymbol{\theta}$ .

In this relaxed setting of a sparse solution recovery, the notions of uniqueness and equivalence, concerning the  $\ell_0$  and  $\ell_1$  solutions, no longer apply. Instead, the issue that now gains in importance is that of *stability* of the solution. To this end, we focus on the computationally attractive  $\ell_1$  task. The counterpart of Theorem 3 is now expressed as follows.

**Theorem 4.** *Assume that the sensing matrix,  $X$ , obeys the RIP with  $\delta_{2k} < \sqrt{2} - 1$ , for some  $k$ . Then the solution  $\boldsymbol{\theta}_*$  of (23.42) satisfies the following ([47, 48]),*

$$\|\boldsymbol{\theta} - \boldsymbol{\theta}_*\|_2 \leq C_0 k^{-\frac{1}{2}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|_1 + C_1 \sqrt{\epsilon}, \quad (23.43)$$

for some constants  $C_1, C_0$ .

This is also an elegant result. If the model is exact and  $\epsilon = 0$  we obtain (23.39). If not, the higher the uncertainty (noise) term in the model, the higher our ambiguity about the solution. Note, also, that the ambiguity about the solution depends on how far the true model is from  $\boldsymbol{\theta}_k$ . If the true model is  $k$ -sparse, the first term on the right hand side of the inequality is zero. The values of  $C_1, C_0$  depend on  $\delta_{2k}$  but they are small, e.g., close to five or six, [48].

The important conclusion, here, is that *the LASSO formulation for solving inverse problems (which in general tend to be ill-conditioned) is a stable one and the noise is not amplified excessively during the recovery process*.

## 1.23.10 Compressed sensing: the glory of randomness

The way in which this paper was deplored followed, more or less, the sequence of developments that took place during the evolution of the sparsity-aware parameter estimation field. We intentionally made an effort to follow such a path, since this is also indicative of how science evolves in most cases. The starting point had a rather strong mathematical flavor: to develop conditions for the solution of an underdetermined linear system of equations, under the sparsity constraint and in a mathematically tractable way, i.e., using convex optimization. In the end, the accumulation of a sequence of individual contributions revealed that the solution can be (uniquely) recovered if the unknown quantity is sensed via randomly chosen data samples. This development has, in turn, given birth to a new field with strong theoretical interest as well as with an enormous impact on practical applications. This new emerged area is known as *compressed sensing* or *compressive sampling* (CS). Although CS builds around the LASSO and Basis Pursuit (and variants of them, as we will soon see), it has changed our view on how to sense and process signals efficiently.

### 1.23.10.1 Compressed sensing

In compressed sensing, the goal is to directly acquire as few samples as possible that encode the minimum information, which is needed to obtain a compressed signal representation. In order to demonstrate this, let us return to the data compression example, which was discussed in Section 1.23.5. There, it was commented that the “classical” approach to compression was rather unorthodox, in the sense that first

all (i.e., a number of  $l$ ) samples of the signal are used, then they are processed to obtain  $l$  transformed values, from which only a small subset is used for coding. In the CS setting, the procedure changes to the following one.

Let  $X$  be an  $N \times l$  sensing matrix, which is applied to the (unknown) signal vector,  $s$ , in order to obtain the measurements,  $y$ , and  $\Psi$  be the dictionary matrix that describes the domain where the signal  $s$  accepts a sparse representation, i.e.,

$$\begin{aligned} s &= \Psi\theta, \\ y &= Xs. \end{aligned} \quad (23.44)$$

Assuming that at most  $k$  of the components of  $\theta$  are nonzero, this can be obtained by the following optimization task

$$\begin{aligned} \min_{\theta \in \mathcal{R}^l} \quad & \|\theta\|_1 \\ \text{s.t. } & y = X\Psi\theta, \end{aligned} \quad (23.45)$$

*provided that the combined matrix  $X\Psi$  complies with the RIP and the number of measurements,  $N$ , satisfies the associated bound given in (23.40).* Note that  $s$  needs not to be stored and can be obtained any time, once  $\theta$  is known. Moreover, as we will soon discuss, the measurements,  $y_n$ ,  $n = 1, 2, \dots, N$ , can be acquired directly from an analog signal  $s(t)$ , prior to obtaining its sample (vector) version,  $s$ ! Thus, from such a perspective, CS fuses the data acquisition and the compression steps together.

There are different ways to obtain a sensing matrix,  $X$ , that leads to a product  $X\Psi$ , which satisfies the RIP. It can be shown, that if  $\Psi$  is orthonormal and  $X$  is a random matrix, which is constructed as discussed at the end of Section 1.23.8.2, then the product  $X\Psi$  obeys the RIP, provided that (23.40) is satisfied, [48]. An alternative way to obtain a combined matrix, that respects the RIP, is to consider another orthonormal matrix  $\Phi$ , whose columns have low coherence with the columns of  $\Psi$  (coherence between two matrices is defined in (23.32), where, now, the pace of  $\mathbf{x}_i$  is taken by a column of  $\Phi$  and that of  $\mathbf{x}_j$  by a column of  $\Psi$ ). For example,  $\Phi$  could be the DFT matrix and  $\Psi = I$  or vice versa. Then choose  $N$  rows of  $\Phi$  uniformly at random to form  $X$  in (23.44). In other words, for such a case, the sensing matrix can be written as  $R\Phi$ , where  $R$  is a  $N \times l$  matrix that extracts  $N$  coordinates uniformly at random. The notion of incoherence (low coherence) between the sensing and the basis matrices is closely related to RIP. The more incoherent the two matrices are, the less the number of the required measurements for the RIP to hold, e.g., [51, 60]. Another way to view incoherence is that the rows of  $\Phi$  cannot be sparsely represented in terms of the columns of  $\Psi$ . It turns out that if the sensing matrix  $X$  is a random one, formed as it has already been described in Section 1.23.8.2.1, then RIP and the incoherence with any  $\Psi$  are satisfied with high probability.

The news get even better to say that all the previously stated philosophy can be extended to the more general type of signals, which are not, necessarily, sparse or sparsely represented in terms of the atoms of a dictionary, and they are known as *compressible*. A signal vector is said to be compressible if its expansion in terms of a basis consists of just a few large coefficients  $\theta_i$  and the rest are small. In other words, the signal vector is *approximately* sparse in some basis. Obviously, this is the most interesting case in practice, where exact sparsity is scarcely (if ever) met. Reformulating the arguments used in

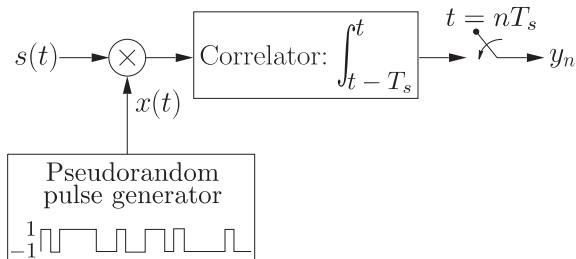
Section 1.23.9, the CS task for this case can be cast as:

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \mathcal{R}^l} & \|\boldsymbol{\theta}\|_1 \\ \text{s.t. } & \|\mathbf{y} - \mathbf{X}\Psi\boldsymbol{\theta}\|_2^2 \leq \epsilon, \end{aligned} \quad (23.46)$$

and everything that has been said in Section 1.23.9 is also valid for this case, if in place of  $\mathbf{X}$  we consider the product  $\mathbf{X}\Psi$ .

### Remarks 9.

- An important property in compressed sensing is that the sensing matrix, which provides the measurements, may be chosen independently on the matrix  $\Psi$ ; that is, the basis/dictionary in which the signal is sparsely represented. In other words, the sensing matrix can be “universal” and can be used to provide the measurements for reconstructing any sparse or sparsely represented signal in any dictionary, provided RIP is not violated.
- Each measurement,  $y_n$ , is the result of an inner product (projection) of the signal vector with a row,  $\mathbf{x}_n^T$ , of the sensing matrix,  $\mathbf{X}$ . Assuming that the signal vector,  $s$ , is the result of a sampling process on an analog signal,  $s(t)$ , then  $y_n$  can be directly obtained, to a good approximation, by taking the inner product (integral) of  $s(t)$  with a sensing waveform,  $x_n(t)$ , that corresponds to  $\mathbf{x}_n$ . For example, if  $\mathbf{X}$  is formed by  $\pm 1$ , as described in Section 1.23.8.2.1, then the configuration shown in Figure 23.11 can result to  $y_n$ . An important aspect of this approach, besides avoiding to compute and store the  $l$  components of  $s$ , is that multiplying by  $\pm 1$  is a relatively easy operation. It is equivalent with changing the polarity of the signal and it can be implemented by employing inverters and mixers. It is a process that can be performed, in practice, at much higher rates than sampling (we will come to it soon). If such a scenario is adopted, one could obtain measurements of an analog signal at much lower rates than required for classical sampling, since  $N$  is much lower than  $l$ . The only condition is that the signal vector must be sparse, in the DFT dictionary, [61]. The dictionary is required only during the reconstruction of  $s$ . Thus, in the CS rationale, processing complexity is removed from the “front end” and is transferred to the “back end,” by exploiting  $\ell_1$  optimization.



**FIGURE 23.11**

Sampling an analog signal  $s(t)$  in order to generate the measurement  $y_n$  at the time instant  $n$ . The sampling period  $T_s$  is much lower than that required by the Nyquist sampling.

One of the very first applications that were inspired by the previous approach, is the so-called *one pixel camera* [62]. This was one among the most catalytic examples, that spread the rumor about the practical power of CS. CS is an example of what is commonly said: “There is nothing more practical than a good theory”!

### 1.23.10.2 Dimensionality reduction and stable embeddings

We are now going to shed light to what we have said so far in this paper from a different view. In both cases, either when the unknown quantity was a  $k$ -sparse vector in a high dimensional space,  $\mathcal{R}^l$ , or if the signal  $s$  was (approximately) sparsely represented in some dictionary ( $s = \Psi\theta$ ), we chose to work in a lower dimensional space ( $\mathcal{R}^N$ ); that is, the space of the measurements,  $y$ . This is a typical task of dimensionality reduction. The main task in any (linear) dimensionality reduction technique is to choose the proper matrix  $X$ , that dictates the projection to the lower dimensional space. In general, there is always a loss of information by projecting from  $\mathcal{R}^l$  to  $\mathcal{R}^N$ , with  $N < l$ , in the sense that we cannot recover any vector,  $\theta_l \in \mathcal{R}^l$ , from its projection  $\theta_N \in \mathcal{R}^N$ . Indeed, take any vector  $\theta_{l-N} \in \text{null}(X)$ , that lies in the  $(l-N)$ -dimensional null space of the (full rank)  $X$  (see Section 1.23.6). Then, all vectors  $\theta_l + \theta_{l-N} \in \mathcal{R}^l$  share the same projection in  $\mathcal{R}^N$ . However, what we have discovered in our tour in this chapter is that if the original vector is sparse then we can recover it exactly. This is because all the  $k$ -sparse vectors do not lie anywhere in  $\mathcal{R}^l$ , but rather in a subset of it; that is, in a *union of subspaces*, each one having dimensionality  $k$ . If the signal  $s$  is sparse in some dictionary  $\Psi$ , then one has to search for it in the union of all possible  $k$ -dimensional subspaces of  $\mathcal{R}^l$ , which are spanned by  $k$  column vectors from  $\Psi$ , [63, 64]. Of course, even in this case, where sparse vectors are involved, not any projection can guarantee unique recovery. The guarantee is provided if the projection in the lower dimensional space is a *stable embedding*. A stable embedding in a lower dimensional space must guarantee that if  $\theta_1 \neq \theta_2$ , then their projections remain also different. Yet this is not enough. A stable embedding must guarantee that distances are (approximately) preserved; that is, vectors that lie far apart in the high dimensional space, have projections that also lie far apart. Such a property guarantees robustness to noise. Well, the sufficient conditions, which have been derived and discussed throughout this paper, and guarantee the recovery of a sparse vector lying in  $\mathcal{R}^l$  from its projections in  $\mathcal{R}^N$ , are conditions that guarantee stable embeddings. The RIP and the associated bound on  $N$  provides a condition on  $X$  that leads to stable embeddings. We commented on this norm-preserving property of RIP in the related section. The interesting fact that came out from the theory is that we can achieve such stable embeddings via random projection matrices.

Random projections for dimensionality reduction are not new and have extensively been used in pattern recognition, clustering and data mining, see, e.g., [1, 65–67]. More recently, the spirit underlying compressed sensing has been exploited in the context of pattern recognition, too. In this application, one needs not to return to the original high dimensional space, after the information-digging activity in the low dimensional measurements subspace. Since the focus in pattern recognition is to identify the class of an object/pattern, this can be performed in the measurements subspace, provided that there is no class-related information loss. In [68], it is shown, using compressed sensing arguments, that if the data is approximately linearly separable in the original high dimensional space and the data has a sparse representation, even in an unknown basis, then projecting randomly in the measurements subspace retains the structure of linear separability.

*Manifold learning* is another area where random projections have been recently applied. A manifold is, in general, a nonlinear  $k$ -dimensional surface, embedded in a higher dimensional (ambient) space. For example, the surface of a sphere is a two-dimensional manifold in a three-dimensional space. More on linear and nonlinear techniques for manifold learning can be found in, e.g., [1]. In [69, 70], the compressed sensing rationale is extended to signal vectors that live along a  $k$ -dimensional submanifold of the space  $\mathcal{R}^l$ . It is shown that choosing a matrix,  $X$ , to project and a sufficient number,  $N$ , of measurements, then the corresponding submanifold has a stable embedding in the measurements subspace, under the projection matrix,  $X$ ; that is, pairwise Euclidean and geodesic distances are approximately preserved after the projection mapping. More on these issues can be found in the given references and in, e.g., [63].

### 1.23.10.3 Sub-Nyquist sampling: analog-to-information conversion

In our discussion in the Remarks presented before, we touched a very important issue; that of going from the analog domain to the discrete one. The topic of analog-to-digital (A/D) conversion has been at the forefront of research and technology since the seminal works of Shannon, Nyquist, Whittaker and Kotelnikof were published, see, for example, [71] for a thorough related review. We all know that if the highest frequency of an analog signal,  $s(t)$ , is less than  $F/2$ , then Shannon's theorem suggests that no loss of information is achieved if the signal is sampled, at least, at the Nyquist rate of  $F = 1/T$ , where  $T$  is the corresponding sampling period, and the signal can be perfectly recovered by its samples

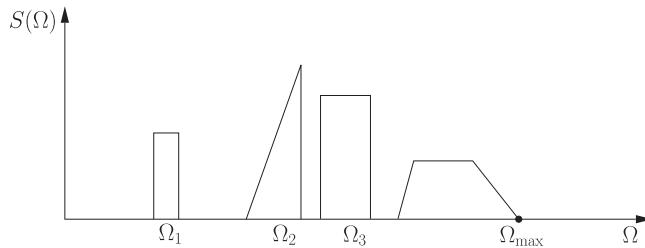
$$s(t) = \sum_n s(nT) \text{sinc}(Ft - n),$$

where  $\text{sinc}(\cdot)$  is the sampling function

$$\text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}.$$

While this has been the driving force behind the development of signal acquisition devices, the increasing complexity of emerging applications demands increasingly higher sampling rates, that cannot be accommodated by today's hardware technology. This is the case, for example, in wideband communications, where conversion speeds, as dictated by Shannon's bound, have become more and more difficult to obtain. Consequently, alternatives to high rate sampling are attracting a strong interest with the goal to reduce the sampling rate by exploiting the *underlying structure* of the signals at hand. In many applications, the signal comprises a few frequencies or bands, see Figure 23.12 for an illustration. In such cases, sampling at the Nyquist rate is inefficient. This is an old problem and it has been addressed by a number of authors, in the case where the locations of the nonzero bands in the frequency spectrum are known, see, e.g., [72–74]. CS theory has inspired research to study cases where the locations (carrier frequencies) of the bands are not known a priori. A typical application of this kind, of high practical interest, lies within the field of Cognitive radio, e.g., [75–77]. In contrast to what we have studied so far in this paper, the sparsity now characterizes the analog signal, and this poses a number of challenges that need to be addressed. In other words, one can consider that

$$s(t) = \sum_{i \in \mathcal{I}} \theta_i \psi_i(t),$$

**FIGURE 23.12**

The Fourier transform of an analog signal,  $s(t)$ , which is sparse in the frequency domain; only a limited number of frequency bands contribute to its spectrum content  $S(\Omega)$ , where  $\Omega$  stands for the angular frequency. Nyquist's theory guarantees that sampling at a frequency larger than or equal to twice the maximum  $\Omega_{\max}$  is sufficient to recover the original analog signal. However, this theory does not exploit information related to the sparse structure of the signal in the frequency domain.

where  $\psi_i(t)$ ,  $i \in \mathcal{I}$ , are the functions that comprise the dictionary, and only a small subset of the coefficients  $\theta_i$  are nonzero. Note that although each one of the dictionary functions can be of high bandwidth, the true number of degrees of freedom of the signal is low. Hence, one would like to sample the signal not at the Nyquist rate but at a rate determined by the sparsity level of the coefficients' set. We refer to such a scenario as *Analog-to-Information* sampling or *sub-Nyquist* sampling.

An approach inspired directly by the theory of CS was first presented in [78] and later improved and theoretically developed in [61]. The approach builds around the assumption that the signal consists of a sum of sinusoids and the *random demodulator* of Figure 23.11 is adopted. In [75, 79], the more general case of a signal consisting of a number of frequency bands, instead of tones, was treated. In addition, the task of extracting each band of the signal from the compressed measurements, that enables (low rate) baseband processing, is addressed. In principle, CS related theory would enable far fewer data samples than traditionally required when capturing signals with relatively high bandwidth, but with a low information rate. However, from a practical point of view, there are still a number of hardware implementation related issues, such as sampling jitter, to be solved first, e.g., [80, 81].

An alternative path to sub-Nyquist sampling embraces a different class of analog signals known as *multipulse* signals; that is, signals that consist of a stream of short pulses. Sparsity now refers to the time domain, and such signals may not even be bandlimited. Signals of this type can be met in a number of applications, such as in radar, ultrasound, bioimaging and neuronal signal processing, see, e.g., [82]. An approach, known as *finite rate of innovation sampling*, passes an analog signal, having  $k$  degrees of freedom per second, through a linear time invariant filter and then samples at a rate of  $2k$  samples per second. Reconstruction is performed via rooting a high-order polynomial, see, e.g., [83, 84] and the references therein. In [85], the task of sub-Nyquist sampling is treated using CS theory arguments and an expansion in terms of Gabor functions; the signal is assumed to consist of a sum of a few pulses of finite duration, yet of unknown shape and time positions.

The task of sparsity-aware learning in the analog domain is still in its early stages and there is currently a lot of on-going activity; more on this topic can be obtained in [86, 87] and the references there in.

### 1.23.11 Sparsity-promoting algorithms

In the previous sections, our emphasis was to highlight the most important aspects underlying the theory of sparse signal/vector recovery from an underdetermined set of linear equations. We now turn our attention to the algorithmic aspects of the problem [30]. The issue now becomes that of discussing *efficient* algorithmic schemes, which can achieve the recovery of the unknown set of parameters. In Sections 1.23.4 and 1.23.6, we saw that the constrained  $\ell_1$  norm minimization (Basis Pursuit) can be solved via Linear Programming techniques and the LASSO task via convex optimization schemes. However, such general purpose techniques tend to be inefficient, since, often, they require many iterations to converge and the respective computational resources can be excessive for practical applications, especially in high dimensional spaces,  $\mathcal{R}^l$ . As a consequence, a huge research effort has been invested with the goal to develop efficient algorithms, that are tailored-made to these specific tasks. This is still a hot on-going area of research and definite conclusions are still risky to be drawn. Our aim here is to provide the reader with some general trends and philosophies that characterize the related activity. We will focus on the most commonly used and cited algorithms, which at the same time are structurally simple and the reader can follow them, without requiring a deeper knowledge on optimization. Moreover, these algorithms involve, in one way or another, arguments that are directly related to points and notions that we have already used while presenting the theory; thus, they can also be exploited from a pedagogical point of view, in order to strengthen the reader's understanding of the topic. We start our review with the class of batch algorithms, where all data are assumed to be available prior to the application of the algorithm, and then we will move onto online/time-adaptive schemes. Furthermore, our emphasis is on algorithms that are appropriate for any sensing matrix. This is stated in order to point out that in the literature efficient algorithms have also been developed for specific forms of highly structured sensing matrices; exploiting their particular structure can lead to reduced computational demands, [88, 89].

There are currently three rough types of families along which this algorithmic activity is growing: (a) greedy algorithms, (b) iterative shrinkage schemes, and (c) convex optimization techniques. We have used the word rough, since, in some cases, it may be difficult to assign an algorithm to a specific family.

#### 1.23.11.1 Greedy algorithms

Greedy algorithms have a long history, see, for example, [90] for a comprehensive list of references. In the context of dictionary learning, a greedy algorithm known as *Matching Pursuit* was introduced in [24]. A greedy algorithm is built upon a series of *locally* optimal *single-term* updates. In our context, the goals are: (a) to unveil the “active” columns of the sensing matrix  $X$ ; that is, those columns that correspond to the nonzero locations of the unknown parameters and (b) to estimate the respective sparse parameter vector. The set of indices which correspond to the nonzero vector components is also known as the *support*. To this end, the set of active columns of  $X$  (and the support) is increased by one at each iteration step. In the sequel, an updated estimate of the unknown sparse vector is obtained. Let us assume that, at the  $(i - 1)$ th iteration step, the algorithm has selected the columns denoted as  $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \dots, \mathbf{x}_{j_{i-1}}$ , with  $j_1, j_2, \dots, j_{i-1} \in \{1, 2, \dots, l\}$ . These indices are the elements of the currently available support,  $S^{(i-1)}$ . Let  $X^{(i-1)}$  be the  $N \times (i - 1)$  matrix having  $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \dots, \mathbf{x}_{j_{i-1}}$  as its columns. Let, also, the current estimate of the solution be  $\boldsymbol{\theta}^{(i-1)}$ , which is a  $(i - 1)$ -sparse vector, with zeros at all locations with index outside the support.

**Algorithm 1 (Orthogonal Matching Pursuit (OMP)).**

The algorithm is initialized with  $\boldsymbol{\theta}^{(0)} := \mathbf{0}$ ,  $\mathbf{e}^{(0)} := \mathbf{y}$  and  $S^{(0)} := \emptyset$ . At iteration step  $i$ , the following computational steps are performed:

1. Select the column  $\mathbf{x}_{j_i}$  of  $X$ , which is *maximally* correlated to (forms the least angle with) the respective error vector,  $\mathbf{e}^{(i-1)} := \mathbf{y} - X\boldsymbol{\theta}^{(i-1)}$ , i.e.,

$$\mathbf{x}_{j_i} : j_i := \arg \max_{j=1,2,\dots,l} \frac{|\mathbf{x}_j^T \mathbf{e}^{(i-1)}|}{\|\mathbf{x}_j\|_2}.$$

2. Update the support and the corresponding set of active columns:  $S^{(i)} = S^{(i-1)} \cup \{j_i\}$ , and  $X^{(i)} = [X^{(i-1)}, \mathbf{x}_{j_i}]$ .
3. Update the estimate of the parameter vector: Solve the Least-Squares (LS) problem that minimizes the norm of the error, using the active columns of  $X$  only, i.e.,

$$\tilde{\boldsymbol{\theta}} := \arg \min_{\mathbf{z} \in \mathcal{R}^i} \left\| \mathbf{y} - X^{(i)} \mathbf{z} \right\|_2^2.$$

Obtain  $\boldsymbol{\theta}^{(i)}$  by inserting the elements of  $\tilde{\boldsymbol{\theta}}$  in the respective locations ( $j_1, j_2, \dots, j_i$ ), which comprise the support (the rest of the elements of  $\boldsymbol{\theta}^{(i)}$  retain their zero values).

4. Update the error vector

$$\mathbf{e}^{(i)} := \mathbf{y} - X\boldsymbol{\theta}^{(i)}.$$

The algorithm terminates if the norm of the error becomes less than a preselected user-defined constant,  $\epsilon_0$ . The following observations are in order.

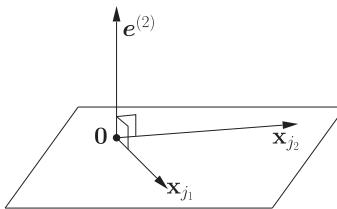
**Remarks 10.**

- Since  $\boldsymbol{\theta}^{(i)}$ , in Step 3, is the result of a LS task, it is known that the error vector is orthogonal to the subspace spanned by the active columns involved, i.e.,

$$\mathbf{e}^{(i)} \perp \text{span} \{ \mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_i} \}.$$

This guarantees that taking the correlation, in the next step, of the columns of  $X$  with  $\mathbf{e}^{(i)}$  none of the previously selected columns will be reselected; they result to zero correlation, being orthogonal to  $\mathbf{e}^{(i)}$ , see Figure 23.13.

- It can be shown that the column, which has maximal correlation (maximum absolute value of the inner product) with the currently available error vector, is the one that maximally reduces (compared to any other column) the  $\ell_2$  norm of the error, when  $\mathbf{y}$  is approximated by linearly combining the currently available active columns. This is the point where the heart of the greedy strategy beats. This minimization is with respect to a *single term*, keeping the rest fixed, as they have been obtained from the previous iteration steps [8].
- Starting with all the components being zero, if the algorithm stops after  $k_0$  iteration steps, the result will be a  $k_0$ -sparse solution.

**FIGURE 23.13**

The error vector at the  $i$ th iteration is orthogonal to the subspace spanned by the currently available set of active columns. Here is an illustration for the case of the 3-dimensional Euclidean space  $\mathcal{R}^3$ , and for  $i = 2$ .

- Note that there is no optimality in this searching strategy. The only guarantee is that the  $\ell_2$  norm of the error vector is decreased at every iteration step. In general, there is no guarantee that the algorithm can obtain a solution close to the true one, see, e.g., [91]. However, under certain constraints on the structure of  $X$ , performance bounds can be obtained, see, e.g., [92–94].
- The complexity of the algorithm amounts to  $\mathcal{O}(k_0 l N)$  operations, which are contributed by the computations of the correlations, plus the demands raised by the solution of the LS task, in Step 3, whose complexity depends on the specific algorithm used. The  $k_0$  is the sparsity level of the delivered solution and, hence, the total number of iteration steps that are performed.

Another more qualitative argument, that justifies the selection of the columns based on their correlation with the error vector, is the following. Assume that the matrix  $X$  is orthonormal. Let also  $\mathbf{y} = X\boldsymbol{\theta}$ . Then,  $\mathbf{y}$  lies in the subspace spanned by the active columns of  $X$ , i.e., those which correspond to the nonzero components of  $\boldsymbol{\theta}$ . Hence, the rest of the columns are orthogonal to  $\mathbf{y}$ , since  $X$  is assumed to be orthonormal. Taking the correlation of  $\mathbf{y}$ , at the first iteration step, with all the columns, it is certain that one among the active columns will be chosen. The inactive columns result in zero correlation. A similar argument holds true for all subsequent steps, since all the activity takes place in a subspace that is orthogonal to all the inactive columns of  $X$ . In the more general case, where  $X$  is not orthonormal, we can still use the correlation as a measure that quantifies geometric similarity. The smaller the correlation/the magnitude of the inner product is, the more orthogonal two vectors are. This brings us back to the notion of mutual coherence, which is a measure of the maximum correlation (least angle) among the columns of  $X$ .

#### 1.23.11.1.1 OMP can recover optimal sparse solutions: sufficiency condition

We have already stated that, in general, there are no guarantees that OMP will recover optimal solutions. However, when the unknown vector is sufficiently sparse, with respect to the structure of the sensing matrix  $X$ , then OMP can exactly solve the  $\ell_0$  minimization task in (23.27) and recover the solution in  $k_0$  steps, where  $k_0$  is the sparsest solution that satisfies the associated linear set of equations.

**Theorem 5.** *Let the mutual coherence (Section 1.23.7.1) of the sensing matrix,  $X$ , be  $\mu(X)$ . Assume, also, that the linear system,  $\mathbf{y} = X\boldsymbol{\theta}$ , accepts a solution such as*

$$\|\boldsymbol{\theta}\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(X)} \right). \quad (23.47)$$

Then, OMP guarantees to recover the sparsest solution in  $k_0 = \|\theta\|_0$  steps.

We know from Section 1.23.7.1 that, under the previous condition, any other solution will be necessarily less sparse. Hence, there is a unique way to represent  $y$  in terms of  $k_0$  columns of  $X$ . Without harming generality, let us assume that the true support corresponds to the first  $k_0$  columns of  $X$ , i.e.,

$$y = \sum_{j=1}^{k_0} \theta_j \mathbf{x}_j, \quad \theta_j \neq 0, \quad \forall j \in \{1, \dots, k_0\}.$$

The theorem is a direct consequence of the following proposition:

**Proposition 1.** *If the condition (23.47) holds true, then the OMP algorithm will never select a column with index outside the true support, see, e.g., [93]. In a more formal way, this is expressed as*

$$j_i = \arg \max_{j=1,2,\dots,l} \frac{|\mathbf{x}_j^T \boldsymbol{\epsilon}^{(i-1)}|}{\|\mathbf{x}_j\|_2} \in \{1, \dots, k_0\}.$$

A geometric interpretation of this proposition is the following: if the angles formed between all the possible pairs among the columns of  $X$  are large enough in the  $\mathcal{R}^l$  space, which guarantees that  $\mu(X)$  is small enough, then  $y$  will lean more (form smaller angle) towards any one of the active columns, which contribute to its formation, compared to the rest, which are inactive and do not participate in the linear combination that generates  $y$ . Figure 23.14 illustrates the geometry, for the extreme case of mutually orthogonal vectors (Figure 23.14a) and for the more general case, where the vectors are not orthogonal, yet the angle between any pair of columns is large enough (Figure 23.14b).

In a nutshell, the previous proposition guarantees that, during the first iteration, a column corresponding to the true support will be selected. In a similar way, this is also true for all subsequent iterations. In the second step, another, different from the previously selected column (as it has already been stated), will be chosen. At step  $k_0$ , the last remaining active column, corresponding to the true support, is selected and this necessarily results to zero error. To this end, it suffices to set  $\epsilon_0$  equal to zero.

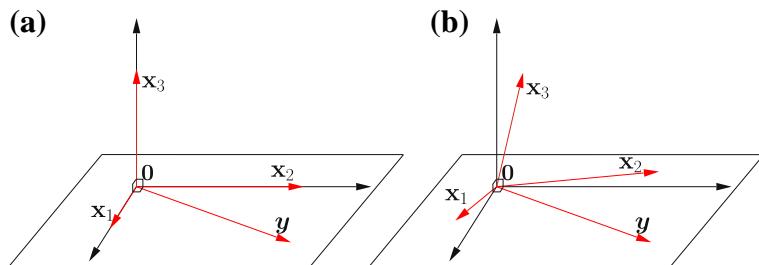


FIGURE 23.14

- (a) In the case of an orthogonal matrix, the measurement vector  $y$  will be orthogonal to any inactive column; here,  $x_3$ .
- (b) In the more general case, it is expected to “lean” closer (form smaller angles) to the active than to the inactive columns.

### 1.23.11.1.2 The LARS algorithm

The Least Angle Regression (LARS) algorithm, [95], shares the first two steps with OMP. It selects  $j_i$  to be an index outside the currently available active set so that to maximize the correlation with the residual vector. However, instead of performing an LS fit to compute the nonzero components of  $\theta^{(i)}$ , these are computed so that the residual to be equicorrelated with all the columns in the active set, i.e.,

$$|\mathbf{x}_j^T(\mathbf{y} - X\theta^{(i)})| = \text{constant}, \quad \forall j \in S^{(i)},$$

where we have assumed that the columns of  $X$  are normalized, as it is common in practice (recall, also, the Remarks 4). In other words, in contrast to the OMP, where the error vector is forced to be orthogonal to the active columns, LARS demands this error to form equal angles with each one of them. Likewise OMP, it can be shown that, provided the target vector is sufficiently sparse and under incoherence of the columns of  $X$ , LARS can exactly recover the sparsest solution, [96].

A further small modification leads to the so-called LARS-LASSO algorithm. According to this version, a previously selected index in the active set can be removed at a later stage. This gives the algorithm the potential to “recover” from a previously bad decision. Hence, this modification departs from the strict rationale that defines the greedy algorithms. It turns out that this version solves the LASSO optimization task. This algorithm is the same as the one suggested in [97] and it is known as *homotopy* algorithm. Homotopy methods are based on a continuous transformation from one optimization task to another. The solutions to this sequence of tasks lie along a continuous parameterized path. The idea is that, while the optimization tasks may be difficult to solve by themselves, one can trace this path of solutions by slowly varying the parameters. For the LASSO task, it is the  $\lambda$  parameter which is varying, see, e.g., [98–100]. Take as an example the LASSO task in its regularized version in (23.13). For  $\lambda = 0$ , the task minimizes the  $\ell_2$  norm and for  $\lambda \rightarrow \infty$  the task minimizes the  $\ell_1$  norm, and for this case the solution tends to zero. It turns out that the solution path, as  $\lambda$  changes from large to small values, is polygonal. Vertices on this solution path correspond to vectors having nonzero elements only on a subset of entries. This subset remains unchanged, till  $\lambda$  reaches the next critical value, which corresponds to a new vertex of the polygonal path and to a new subset of potential nonzero values. Thus, the solution is obtained via this sequence of steps along this polygonal path.

### 1.23.11.1.3 Compressed Sensing Matching Pursuit (CSMP) algorithms

Strictly speaking, these algorithms are not greedy, yet, as it is stated in [89], they are at heart greedy algorithms. Instead of performing a single term optimization per iteration step, in order to increase the support by one, as it is the case with OMP, these algorithms attempt to obtain first an estimate of the support and then use this information to compute a least squares estimate of the target vector, constrained on the respective active columns. The quintessence of the method lies in the near-orthogonal nature of the sensing matrix, assuming that this obeys the RIP.

Assume that  $X$  obeys the RIP for some small enough value  $\delta_k$  and sparsity level,  $k$ , of the unknown vector. Let, also, that the measurements are exact, i.e.,  $\mathbf{y} = X\theta$ . Then,  $X^T\mathbf{y} = X^T X\theta \approx \theta$ . Therefore, intuition indicates that it is not unreasonable to select, in the first iteration step, the  $t$  (a user-defined parameter) largest in magnitude components of  $X^T\mathbf{y}$  as indicative of the nonzero positions of the sparse target vector. This reasoning carries on for all subsequent steps, where, at the  $i$ th iteration, the place of  $\mathbf{y}$  is taken by the residual  $e^{(i-1)} := \mathbf{y} - X\theta^{(i-1)}$ , where  $\theta^{(i-1)}$  indicates the estimate of the target vector

at the  $(i - 1)$ th iteration. Basically, this could be considered as a generalization of the OMP. However, as we will soon see, the difference between the two mechanisms is more substantial.

**Algorithm 2 (The CSMP Scheme).**

1. Select the value of  $t$ .
2. Initialize the algorithm:  $\boldsymbol{\theta}^{(0)} = \mathbf{0}$ ,  $\mathbf{e}^{(0)} = \mathbf{y}$ .
3. For  $i = 1, 2, \dots$ , execute the following.

- a. Obtain the current support:

$$S^{(i)} := \text{supp}(\boldsymbol{\theta}^{(i-1)}) \cup \left\{ \begin{array}{l} \text{indices of the } t \text{ largest in magnitude} \\ \text{components of } X^T \mathbf{e}^{(i-1)} \end{array} \right\}.$$

- b. Select the active columns: Construct  $X^{(i)}$  to comprise the active columns of  $X$  in accordance to  $S^{(i)}$ . Obviously,  $X^{(i)}$  is a  $N \times r$  matrix, where  $r$  denotes the cardinality of the support set  $S^{(i)}$ .
- c. Update the estimate of the parameter vector: solve the LS task

$$\tilde{\boldsymbol{\theta}} := \arg \max_{\mathbf{z} \in \mathcal{R}^r} \left\| \mathbf{y} - X^{(i)} \mathbf{z} \right\|_2^2.$$

Obtain  $\hat{\boldsymbol{\theta}}^{(i)} \in \mathcal{R}^l$  having the  $r$  elements of  $\tilde{\boldsymbol{\theta}}$  in the respective locations, as indicated by the support, and the rest of the elements being zero.

- d.  $\boldsymbol{\theta}^{(i)} := H_k(\hat{\boldsymbol{\theta}}^{(i)})$ . The mapping  $H_k$  denotes the *hard thresholding* operator; that is, it returns a vector with the  $k$  largest in magnitude components of the argument, and the rest are forced to zero.
- e. Update the error vector:  $\mathbf{e}^{(i)} = \mathbf{y} - X \boldsymbol{\theta}^{(i)}$ .

The algorithm requires as input the sparsity level  $k$ . Iterations carry on until a halting criterion is met. The value of  $t$ , that determines the largest in magnitude values in Steps 1 and 3a, depends on the specific algorithm. In CoSaMP (Compressive Sampling Matching Pursuit, [89]),  $t = 2k$  and in the SP (Subspace Pursuit, [101]),  $t = k$ .

Having stated the general scheme, a major difference with OMP becomes readily apparent. In OMP, only one column is selected per iteration step. Moreover, this remains in the active set for all subsequent steps. If, for some reason, this was not a good choice, the scheme cannot recover from such a bad decision. In contrast, the support and hence the active columns of  $X$  are continuously updated in CSMP and the algorithm has the ability to correct a previously bad decision, as more information is accumulated and iterations progress. In [101], it is shown that if the measurements are exact ( $\mathbf{y} = X \boldsymbol{\theta}$ ) then SP can recover the  $k$ -sparse true vector in a finite number of iteration steps, provided that  $X$  satisfies the RIP with  $\delta_{3k} < 0.205$ . If the measurements are noisy, performance bounds have been derived, which hold true for  $\delta_{3k} < 0.083$ . For the CoSaMP, performance bounds have been derived for  $\delta_{4k} < 0.1$ .

### 1.23.11.2 Iterative shrinkage algorithms (ISTA)

This family of algorithms have also a long history, see, e.g., [102–105]. However, in the “early” days, the developed algorithms had some sense of heuristic flavor, without establishing a clear bridge with

optimizing a cost function. Later attempts were substantiated by sound theoretical arguments concerning issues such as convergence and convergence rate, e.g., [106–109].

The general form of this algorithmic family has a striking resemblance with the classical linear algebra iterative schemes for approximating the solution of large linear systems of equations, known as *stationary iterative* or *iterative relaxation* methods. The classical Gauss-Seidel and Jacobi algorithms, e.g., [110], in numerical analysis can be considered as members of this family. Given a linear system of  $l$  equations with  $l$  unknowns,  $\mathbf{z} = A\mathbf{x}$ , the basic iteration at step  $i$  has the following form

$$\begin{aligned}\mathbf{x}^{(i)} &= (I - QA)\mathbf{x}^{(i-1)} + Q\mathbf{z} \\ &= \mathbf{x}^{(i-1)} + Q\mathbf{e}^{(i-1)}, \quad \mathbf{e}^{(i-1)} := \mathbf{z} - A\mathbf{x}^{(i-1)},\end{aligned}$$

which does not come as a surprise. It is of the same form as most of the iterative schemes for numerical solutions! The matrix  $Q$  is chosen so that to guarantee convergence and different choices lead to different algorithms with their pros and cons. It turns out that this algorithmic form can also be applied to underdetermined systems of equations,  $\mathbf{y} = X\boldsymbol{\theta}$ , with a “minor” modification, which is imposed by the sparsity constraint of the target vector. This leads to the following general form of iterative computation

$$\boldsymbol{\theta}^{(i)} = T_i(\boldsymbol{\theta}^{(i-1)} + Q\mathbf{e}^{(i-1)}), \quad \mathbf{e}^{(i-1)} = \mathbf{y} - X\boldsymbol{\theta}^{(i-1)},$$

starting from an initial guess of  $\boldsymbol{\theta}^{(0)}$  (usually  $\boldsymbol{\theta}^{(0)} = \mathbf{0}, \mathbf{e}^{(0)} = \mathbf{y}$ ). In certain cases,  $Q$  can be made to be iteration-dependent. The operator  $T_i(\cdot)$  is a nonlinear thresholding operator, that is applied *entry-wise*, i.e., *component-wise*. Depending on the specific scheme, this can be either the hard thresholding operator, denoted as  $H_k$ , or the soft thresholding operator, denoted as  $S_\alpha$ . Hard thresholding, as we already know, keeps the  $k$  largest components of a vector unaltered and sets the rest equal to zero. Soft thresholding was introduced in Section 1.23.4. All components with magnitude less than  $\alpha$  are forced to zero and the rest are reduced in magnitude by  $\alpha$ ; that is, the  $j$ th component of a vector,  $\boldsymbol{\theta}$ , after soft thresholding becomes

$$(S_\alpha(\boldsymbol{\theta}))_j = \text{sgn}(\theta_j)(|\theta_j| - \alpha)_+.$$

Depending on (a) the choice of  $T_i$ , (b) the specific value of the parameter  $k$  or  $\alpha$ , and (c) the matrix  $Q$ , different instances occur. A most common choice for  $Q$  is  $\mu X^T$  and the generic form of the main iteration becomes

$$\boldsymbol{\theta}^{(i)} = T_i(\boldsymbol{\theta}^{(i-1)} + \mu X^T \mathbf{e}^{(i-1)}), \tag{23.48}$$

where  $\mu$  is a relaxation (user-defined) parameter, which can also be left to vary with each iteration step. The choice of  $X^T$  is intuitively justified, once more, by the near-orthogonal nature of  $X$ . For the first iteration step and for a linear system of the form  $\mathbf{y} = X\boldsymbol{\theta}$ , starting from a zero initial guess, we have  $X^T\mathbf{y} = X^T X\boldsymbol{\theta} \approx \boldsymbol{\theta}$  and we are close to the solution.

Although intuition is most important in scientific research, it is not enough, by itself, to justify decisions and actions. The generic scheme in (23.48) has been reached from different paths, following different perspectives that lead to different choices of the respective parameters. Let us spend some more time on that, with the aim to make the reader more familiar with techniques that address optimization

tasks of non-differentiable loss functions. The term in the parenthesis in (23.48) coincides with the gradient descent iteration step if the cost function were the unregularized LS loss, i.e.,

$$J(\boldsymbol{\theta}) = \frac{1}{2} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2.$$

In this case, the gradient descent rationale leads to

$$\begin{aligned} \boldsymbol{\theta}^{(i-1)} - \mu \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} &= \boldsymbol{\theta}^{(i-1)} - \mu X^T (X\boldsymbol{\theta}^{(i-1)} - \mathbf{y}) \\ &= \boldsymbol{\theta}^{(i-1)} + \mu X^T \mathbf{e}^{(i-1)}. \end{aligned}$$

It is well known and it can easily be shown that the gradient descent can alternatively be viewed as the result of minimizing a regularized version of the linearized loss function, i.e.,

$$\begin{aligned} \boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left\{ J(\boldsymbol{\theta}^{(i-1)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)})^T \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} \right. \\ \left. + \frac{1}{2\mu} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 \right\}. \end{aligned} \quad (23.49)$$

One can adopt this view of the gradient descent philosophy as a kick-off point to minimize iteratively the following LASSO task, i.e.,

$$\min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left\{ L(\boldsymbol{\theta}, \lambda) = \frac{1}{2} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 = J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \right\}.$$

The difference now is that the loss function comprises two terms. One which is smooth (differentiable) and a non-smooth one. Let the current estimate be  $\boldsymbol{\theta}^{(i-1)}$ . The updated estimate is obtained by

$$\begin{aligned} \boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left\{ J(\boldsymbol{\theta}^{(i-1)}) + (\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)})^T \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} \right. \\ \left. + \frac{1}{2\mu} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 \right\}, \end{aligned}$$

which, after ignoring constants, becomes

$$\boldsymbol{\theta}^{(i)} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left\{ \frac{1}{2} \|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|_2^2 + \lambda \mu \|\boldsymbol{\theta}\|_1 \right\}, \quad (23.50)$$

where

$$\tilde{\boldsymbol{\theta}} := \boldsymbol{\theta}^{(i-1)} - \mu \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}}. \quad (23.51)$$

Following exactly the same steps as those that led to the derivation of (23.20) from (23.13) (after replacing  $\hat{\boldsymbol{\theta}}_{LS}$  with  $\tilde{\boldsymbol{\theta}}$ ) we obtain

$$\boldsymbol{\theta}^{(i)} = S_{\lambda\mu}(\tilde{\boldsymbol{\theta}}) = S_{\lambda\mu} \left( \boldsymbol{\theta}^{(i-1)} - \mu \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} \right) \quad (23.52)$$

$$= S_{\lambda\mu} \left( \boldsymbol{\theta}^{(i-1)} + \mu X^T \mathbf{e}^{(i-1)} \right). \quad (23.53)$$

This is very interesting and practically useful. The only effect of the presence of the non-smooth  $\ell_1$  norm in the loss function is an extra simple thresholding operation, which as we know is an operation performed *individually* on each component. It can be shown, e.g., [111], that this algorithm converges to a minimizer  $\boldsymbol{\theta}_*$  of the LASSO (23.13), provided that  $\mu \in (0, 1/\lambda_{\max}(X^T X))$ , where  $\lambda_{\max}(\cdot)$  denotes the maximum eigenvalue of  $X^T X$ . The convergence rate is dictated by the rule

$$L(\boldsymbol{\theta}^{(i)}, \lambda) - L(\boldsymbol{\theta}_*, \lambda) \approx O(1/i),$$

which is known as *sublinear* global rate of convergence. Moreover, it can be shown that

$$L(\boldsymbol{\theta}^{(i)}, \lambda) - L(\boldsymbol{\theta}_*, \lambda) \leq \frac{C \|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}_*\|_2^2}{2i}.$$

The latter result indicates that if one wants to achieve an accuracy of  $\epsilon$ , then this can be obtained by at most  $\left\lfloor \frac{C \|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}_*\|_2^2}{2\epsilon} \right\rfloor$  iterations, where  $\lfloor \cdot \rfloor$  denotes the floor operator.

In [107], (23.48) was obtained from a nearby corner, building upon arguments from the classical *proximal-point* methods in optimization theory, e.g., [112]. The original LASSO regularized cost function is modified to the so called *surrogate objective*,

$$J(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \frac{1}{2} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1 + \frac{1}{2} d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}),$$

where

$$d(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) := c \|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|_2^2 - \|X\boldsymbol{\theta} - X\tilde{\boldsymbol{\theta}}\|_2^2.$$

If  $c$  is appropriately chosen (larger than the largest eigenvalue of  $X^T X$ ), the surrogate objective is guaranteed to be strictly convex. Then it can be shown that the minimizer of the surrogate objective is given by

$$\hat{\boldsymbol{\theta}} = S_{\lambda/c} \left( \tilde{\boldsymbol{\theta}} + \frac{1}{c} X^T (\mathbf{y} - X\tilde{\boldsymbol{\theta}}) \right). \quad (23.54)$$

In the iterative formulation,  $\tilde{\boldsymbol{\theta}}$  is selected to be the previously obtained estimate; in this way, one tries to keep the new estimate close to the previous one. The procedure readily results to our generic scheme in (23.48), using soft thresholding with parameter  $\lambda/c$ . It can be shown that such a strategy converges to a minimizer of the original LASSO problem. The same algorithm was reached in [109], using *majorization-minimization* techniques from optimization theory. So, from this perspective, the IST family has strong ties with algorithms that belong to the convex optimization category.

In [31], the *Sparse Reconstruction by Separable Approximation* (SpaRSA) algorithm is proposed, which is a modification of the standard IST scheme. The starting point is (23.49); however, the multiplying factor,  $\frac{1}{2\mu}$ , instead of being constant is now allowed to change from iteration to iteration according to a rule. This results in a speed up in the convergence of the algorithm. Moreover, inspired by the homotopy family of algorithms, where  $\lambda$  is allowed to vary, SpaRSA can be extended to solve a sequence of problems which are associated with a corresponding sequence of values of  $\lambda$ . Once a solution has been

obtained for a particular value of  $\lambda$ , it can be used as a “warm-start” for a nearby value. Solutions can therefore be computed for a range of values, at a small extra computational cost, compared to solving for a single value from a “cold start.” This technique abides with the so-called *continuation strategy*, which has been used in the context of other algorithms as well, e.g., [113]. Continuation has been shown to be a very successful tool to increase the speed of convergence.

An interesting modification of the basic IST scheme has been proposed in [111], which improves the convergence rate to  $O(1/i^2)$ , by only a simple modification with almost no extra computational burden. The scheme is known as *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA). This scheme is an evolution of [114], which introduced the basic idea for the case of differentiable costs, and consists of the following steps:

$$\begin{aligned}\boldsymbol{\theta}^{(i)} &= S_{\lambda\mu} \left( \mathbf{z}^{(i)} + \mu X^T (\mathbf{y} - X\mathbf{z}^{(i)}) \right), \\ \mathbf{z}^{(i+1)} &:= \boldsymbol{\theta}^{(i)} + \frac{t_i - 1}{t_{i+1}} \left( \boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}^{(i-1)} \right),\end{aligned}$$

where

$$t_{i+1} := \frac{1 + \sqrt{1 + 4t_i^2}}{2},$$

with initial points  $t_1 = 1$  and  $\mathbf{z}^{(1)} = \boldsymbol{\theta}^{(0)}$ . In words, in the thresholding operation,  $\boldsymbol{\theta}^{(i-1)}$  is replaced by  $\mathbf{z}^{(i)}$ , which is a specific linear combination of two successive updates of  $\boldsymbol{\theta}$ . Hence, at a marginal increase of the computational cost, a substantial increase in convergence speed is achieved.

In [115] the hard thresholding version has been used, with  $\mu = 1$  and the thresholding operator  $H_k$  uses the sparsity level  $k$  of the target solution, that is assumed to be known. In a later version, [116], the relaxation parameter is left to change so that, at each iterations step, the error is maximally reduced. It has been shown that the algorithm converges to a local minimum of the cost function  $\|\mathbf{y} - X\boldsymbol{\theta}\|_2$ , under the constraint that  $\boldsymbol{\theta}$  is a  $k$ -sparse vector. Moreover, the latter version is a stable one and it results to a near optimal solution if a form of RIP is fulfilled.

A modified version of the generic scheme given in (23.48), that evolves along the lines of [117], obtains the updates component-wise, one vector component at a time. Thus, a “full” iteration consists of  $l$  steps. The algorithm is known as *coordinate descent* and its basic iteration has the form

$$\theta_j^{(i)} = S_{\lambda/\|\mathbf{x}_j\|_2^2} \left( \theta_j^{(i-1)} + \frac{\mathbf{x}_j^T \boldsymbol{\epsilon}^{(i-1)}}{\|\mathbf{x}_j\|_2^2} \right), \quad j = 1, 2, \dots, l. \quad (23.55)$$

This algorithm replaces the constant  $c$ , in the previously reported soft thresholding algorithm, with the norm of the respective column of  $X$ , if the columns of  $X$  are not normalized to unit norm. It has been shown that the parallel coordinate descent algorithm also converges to a LASSO minimizer of (23.13), [108]. Improvements of the algorithm, using line search techniques to determine the most descent direction for each iteration, have also been proposed, see, [118].

The main contribution to the complexity for the iterative shrinkage algorithmic family comes from the two matrix-vector products, which amounts to  $\mathcal{O}(Nl)$ , unless  $X$  has a special structure, e.g., DFT, that can be exploited to reduce the load.

In [119], the so-called Two Stage Thresholding (TST) scheme is presented, which brings together arguments from the iterative shrinkage family and the OMP. This algorithmic scheme involves two stages of thresholding. The first step is exactly the same as in (23.48). However, this is now used only for determining “significant” nonzero locations, just as in Compressed Sensing Matching Pursuit (CSMP) algorithms, presented in the previous subsection. Then, a LS problem is solved to provide the updated estimate, under the constraint of the available support. This is followed by a second step of thresholding. The thresholding operations in the two stages can be different. If hard thresholding,  $H_k$ , is used in both steps, this results to the algorithm proposed in [120]. For this latter scheme, convergence and performance bounds are derived if the RIP holds for  $\delta_{3k} < 0.58$ . In other words, the basic difference between the TST and CSMP approaches is that, in the latter case, the most significant nonzero coefficients are obtained by looking at the correlation term  $X^T \mathbf{e}^{(i-1)}$  and in the TST family at  $\theta^{(i-1)} + \mu X^T \mathbf{e}^{(i-1)}$ . The differences among different approaches can be minor and the crossing lines among the different algorithmic categories are not necessarily crispy clear. However, from a practical point of view, sometimes small differences may lead to substantially improved performance.

### Remarks 11.

- The minimization in (23.52) bridges the IST algorithmic family with another powerful tool in convex optimization, which builds upon the notion of *proximal mapping* or *Moreau envelopes*, see, e.g., [112, 121]. Given a convex function  $h : \mathcal{R}^l \rightarrow \mathcal{R}$ , and a  $\mu > 0$ , the proximal mapping,  $\text{Prox}_{\mu h} : \mathcal{R}^l \rightarrow \mathcal{R}^l$ , with respect to  $h$ , and of index  $\mu$ , is defined as the (unique) minimizer

$$\text{Prox}_{\mu h}(\mathbf{x}) := \arg \min_{\mathbf{u} \in \mathcal{R}^l} \left\{ h(\mathbf{u}) + \frac{1}{2\mu} \|\mathbf{x} - \mathbf{u}\|_2^2 \right\}, \quad \forall \mathbf{x} \in \mathcal{R}^l. \quad (23.56)$$

Let us now assume that we want to minimize a convex function, which is given as the sum

$$f(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + h(\boldsymbol{\theta}),$$

where  $J(\cdot)$  is convex and differentiable, and  $h(\cdot)$  is also a convex, but not necessarily a smooth one. Then it can be shown that the following iterations converge to a minimizer of  $f(\cdot)$ ,

$$\boldsymbol{\theta}^{(i)} = \text{Prox}_{\mu h} \left( \boldsymbol{\theta}^{(i-1)} - \mu \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} \right), \quad (23.57)$$

where  $\mu > 0$  and it can also be made iteration dependent, i.e.,  $\mu_i > 0$ . If we now use this scheme to minimize our familiar cost, i.e.,

$$J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1,$$

we obtain (23.52); this is so, because the proximal operator of  $h(\boldsymbol{\theta}) := \lambda \|\boldsymbol{\theta}\|_1$  is shown ([106, 121]) to be identical to the soft thresholding operator, i.e.,

$$\text{Prox}_h(\boldsymbol{\theta}) = S_\lambda(\boldsymbol{\theta}).$$

In order to feel more comfortable with this operator, note that if  $h(\mathbf{x}) := 0$ , its proximal operator is equal to  $\mathbf{x}$ , and in this case (23.57) becomes our familiar gradient descent algorithm.

- All the non-greedy algorithms, which have been discussed so far, have been developed to solve the task defined in the formulation (23.13). This is mainly because this is an easier task to solve; once  $\lambda$  has been fixed, it is an unconstrained optimization task. However, there are algorithms which have been developed to solve the alternative formulations.

The NESTA algorithm has been proposed in [122] and solves the task in its (23.15) formulation. Adopting this path can have an advantage since  $\epsilon$  may be given as an estimate of the uncertainty associated with the noise, which can readily be obtained in a number of practical applications. In contrast, selecting a priori the value for  $\lambda$  is more intricate. In [7], the value  $\lambda = \sigma_\eta \sqrt{2 \ln l}$ , where  $\sigma_\eta$  is the noise standard deviation, is argued to have certain optimality properties; however this argument hinges on the assumption of the orthogonality of  $X$ . NESTA relies heavily on Nesterov's generic scheme ([114]), hence its name. The original Nesterov's algorithm performs a constrained minimization of a smooth convex function  $f(\boldsymbol{\theta})$ , i.e.,

$$\min_{\boldsymbol{\theta} \in Q} f(\boldsymbol{\theta}),$$

where  $Q$  is a convex set, and in our case this is associated with the quadratic constraint in (23.15). The algorithm consists of three basic steps. The first one is similar with the step in (23.49), i.e.,

$$\mathbf{w}^{(i)} = \arg \min_{\boldsymbol{\theta} \in Q} \left\{ (\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)})^T \frac{\partial J(\boldsymbol{\theta}^{(i-1)})}{\partial \boldsymbol{\theta}} + \frac{L}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(i-1)}\|_2^2 \right\}, \quad (23.58)$$

where  $L$  is an upper bound on the Lipschitz coefficient, which the gradient of  $f(\cdot)$  has to satisfy. The difference with (23.49) is that the minimization is now a constrained one. However, Nesterov has also added a second step involving another auxiliary variable,  $\mathbf{z}^{(i)}$ , which is computed in a similar way as  $\mathbf{w}^{(i)}$  but the linearized term is now replaced by a weighted cumulative gradient,

$$\sum_{k=0}^{i-1} \alpha_k (\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)})^T \frac{\partial J(\boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\theta}}.$$

The effect of this term is to smooth out the “zig-zagging” of the path towards the solution, whose effect is to increase significantly the convergence speed. The final step of the scheme involves an averaging of the previously obtained variables,

$$\boldsymbol{\theta}^{(i)} = t_i \mathbf{z}^{(i)} + (1 - t_i) \mathbf{w}^{(i)}.$$

The values of the parameters  $\alpha_k$ ,  $k = 0, \dots, i-1$ , and  $t_i$  result from the theory so that convergence is guaranteed. As it was the case with its close relative FISTA, the algorithm enjoys an  $O(1/i^2)$  convergence rate. In our case, where the function to be minimized,  $\|\boldsymbol{\theta}\|_1$ , is not smooth, NESTA uses a smoothed prox-function of it. Moreover, it turns out that close-form updates are obtained for  $\mathbf{z}^{(i)}$  and  $\mathbf{w}^{(i)}$ . If  $X$  is chosen so that to have orthonormal rows, the complexity per iteration is  $O(l)$  plus the computations needed for performing the product  $X^T X$ , which is the most computational thirsty part. However, this complexity can substantially be reduced if the sensing matrix is chosen to

be a submatrix of a unitary transform, which admits fast matrix-vector product computations, e.g., a subsampled DFT matrix. For example, for the case of a subsampled DFT matrix, the complexity amounts to  $O(l)$  plus the load to perform the two Fast Fourier Transforms (FFT). Moreover, the continuation strategy can also be employed to accelerate convergence. In [122], it is demonstrated that NESTA exhibits good accuracy results, while retaining a complexity that is competitive with algorithms developed around the (23.15) formulation and scales in an affordable way for large size problems. Furthermore, NESTA, and in general Nesterov's scheme, enjoy a generality that allows their use to other optimization tasks as well.

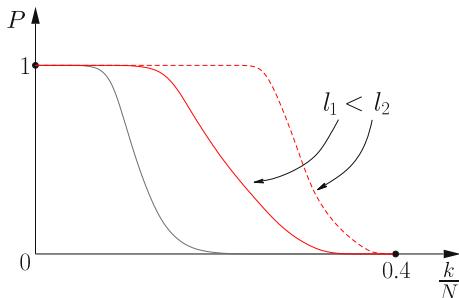
- The task in (23.14) has been considered in [97, 123]. In the former, the algorithm comprises a projection on the  $\ell_1$  ball  $\|\theta\|_1 \leq \rho$  (see also, Section 1.23.13.4) per iteration step. The most computationally dominant part of the algorithm consists of matrix-vector products. In [97], a homotopy algorithm is derived for the same task, where now the bound  $\rho$  becomes the homotopy parameter which is left to vary. This algorithm is also referred as the LARS-LASSO, as it has already been reported before.

### 1.23.11.3 Which algorithm then: some practical hints

We have already discussed a number of algorithmic alternatives to obtain solutions to the  $\ell_0$  or  $\ell_1$  norm minimization tasks. Our focus was on schemes whose computational demands are rather low and they scale well to very large problem sizes. We have not touched more expensive methods such as interior point methods for solving the  $\ell_1$  convex optimization task. A review of such methods is provided in [124]. Interior point methods evolve along the Newton-type recursion and their complexity per iteration step is at least of the order  $\mathcal{O}(l^3)$ . As it is most often the case, there is a trade off. Schemes of higher complexity tend to result in enhanced performance. However, such schemes become impractical in problems of large size. Some examples of other algorithms, that were not discussed, can be found in [31, 123, 125, 126]. Talking about complexity, it has to be pointed out that what really matters at the end is not so much the complexity per iteration step but the overall required resources in computer time/memory for the algorithm to converge to a solution within a specified accuracy. For example, an algorithm may be of low complexity per iteration step but it may need an excessive number of iterations to converge.

Computational load is only one among a number of indices that characterize the performance of an algorithm. Other performance measures, refer to convergence rate, tracking speed (for the adaptive algorithms), and stability with respect to the presence of noise and/or finite word length computations. No doubt, all these performance measures are also of interest here, too. However, there is an additional aspect that is of particular importance when quantifying performance of sparsity-promoting algorithms. This is related to the so called *undersampling-sparsity tradeoff* or the *phase transition curve*.

One of the major issues, on which we focused in this chapter, was to derive and present the conditions that guarantee uniqueness of the  $\ell_0$  minimization and its equivalence with the  $\ell_1$  minimization task, under an underdetermined set of measurements,  $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$ , for the recovery of sparse enough signals/vectors. While discussing the various algorithms in this section, we reported a number of different RIP-related conditions that some of the algorithms have to satisfy in order to recover the target sparse vector. As a matter of fact, it has to be admitted that this was quite confusing, since each algorithm had to satisfy its own conditions. In addition, in practice, these conditions are not easy to be verified. Although such results are, no doubt, important to establish convergence and make us more confident and also

**FIGURE 23.15**

For any algorithm, the transition between the regions of 100% success and of a complete failure is very sharp. For the algorithm corresponding to the middle curve, this transition occurs at higher sparsity values and, from this point of view, it is a better algorithm than the one associated with the curve on the left. Also, given a algorithm, the higher the dimensionality the higher the sparsity level where this transition occurs, as indicated by the middle and the dotted curve on the right.

understand better why and how an algorithm works, one needs further experimental evidence in order to establish good performance bounds for an algorithm. Moreover, all the conditions that we have dealt with, including coherence and RIP, are sufficient conditions. In practice, it turns out that sparse signal recovery is possible with sparsity levels much higher than those predicted by the theory, for given  $N$  and  $l$ . Hence, proposing a new algorithm or selecting an algorithm from an available palette, one has to demonstrate experimentally the range of sparsity levels that can be recovered by the algorithm, as a percentage of the number of measurements and the dimensionality. Thus, in order to select an algorithm, one should cast his/her vote for the algorithm which, for given  $l$  and  $N$ , has the potential to recover  $k$ -sparse vectors with  $k$  being as high as possible, for most of the cases, that is, with *high probability*.

Figure 23.15 illustrates the type of curve that is expected to result in practice. The vertical axis is the probability of exact recovery of a target  $k$ -sparse vector and the horizontal axis shows the ratio  $k/N$ , for a given number of measurements,  $N$ , and the dimensionality of the ambient space,  $l$ . Three curves are shown. The middle curve and the one on the right correspond to the same algorithm, for two different values of the dimensionality,  $l$ , and the curve on the left corresponds to another algorithm. Curves of this shape are expected to result from experiments of the following set up. Assume that we are given a sparse vector,  $\theta_0$ , with  $k$  nonzero components in the  $l$ -dimensional space. Using a sensing matrix  $X$ , we generate  $N$  measurements  $y = X\theta_0$ . The experiment is repeated a number of, say,  $M$  times, each time using a different realization of the sensing matrix and a different  $k$ -sparse vector. For each instance, the algorithm is run to recover the target sparse vector. This is not always possible. We count the number,  $m$ , of successful recoveries, and compute the corresponding percentage of successful recovery (probability),  $m/M$ , which is plotted on the vertical axis of Figure 23.15. The procedure is repeated for a different value of  $k$ ,  $1 \leq k \leq N$ . A number of issues now jump into the stage: (a) How one selects the ensemble of sensing matrices and (b) how one selects the ensemble of sparse vectors. There are different scenarios and some typical examples are described next.

1. The  $N \times l$  sensing matrices  $X$  are formed by:

- a. Different i.i.d. realizations with elements drawn from a Gaussian  $\mathcal{N}(0, 1/N)$ .
  - b. Different i.i.d. realizations from the uniform distribution on the unit sphere in  $\mathcal{R}^N$ , which is also known as the uniform spherical ensemble.
  - c. Different i.i.d. realizations with elements drawn from Bernoulli type distributions.
  - d. Different i.i.d. realizations of partial Fourier matrices, each time using a different set of  $N$  rows.
2. The  $k$ -sparse target vector  $\theta_0$  is formed by selecting the locations of at most  $k$  nonzero elements randomly, by “tossing a coin” with probability  $p = k/l$ , and fill the values of the nonzero elements according to a statistical distribution, e.g., Gaussian, uniform, double exponential, Cauchy.

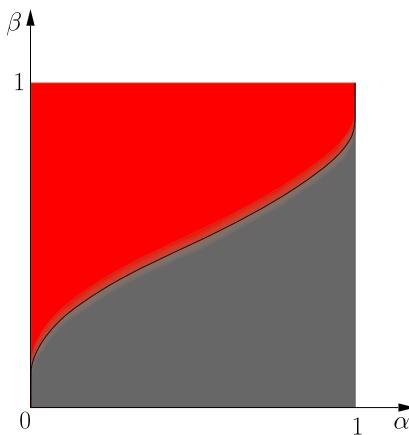
Other scenarios are also possible. Some authors set all nonzero values to one, [66], or to  $\pm 1$ , with the randomness imposed on the choice of the sign. It must be stressed out that the performance of an algorithm may vary significantly under different experimental scenarios, and this may be indicative of the stability of an algorithm. In practice, a user may be interested in a specific scenario, which is more representative of the available data.

Looking at Figure 23.15, the following conclusions are in order. In all curves, there is a sharp transition between two levels. From the 100% success to the 0% success. Moreover, the higher the dimensionality, the sharper the transition is. This has also been shown theoretically in [127]. For the algorithm corresponding to the two curves on the right, this transition occurs at higher values of  $k$ , compared to the algorithm that generates the curve on the left. Provided that the computational complexity of the former algorithm can be accommodated by the resources, which are available for a specific application, this seems to be the more sensible choice between the two algorithms. However, if the resources are limited, concessions are unavoidable.

Another way to “interrogate” and demonstrate the performance of an algorithm, with respect to its robustness to the range of values of sparsity levels that can be successfully recovered, is via the so-called *phase transition curve*. To this end define:

- $\alpha := \frac{N}{l}$ , which is a normalized measure of the problem indeterminacy.
- $\beta := \frac{k}{N}$ , which is a normalized measure of sparsity.

In the sequel, plot a graph having  $\alpha \in [0, 1]$  in the horizontal axis and  $\beta \in [0, 1]$  in the vertical one. For each point,  $(\alpha, \beta)$ , in the  $[0, 1] \times [0, 1]$  region, compute the probability of the algorithm to recover a  $k$ -sparse target vector. In order to compute the probability, one has to adopt one of the previously stated scenarios. In practice, one has to form a grid of points that cover densely enough the region  $[0, 1] \times [0, 1]$  in the graph. Use a varying intensity level scale to color the corresponding  $(\alpha, \beta)$  point. Gray (lower region) corresponds to probability one and red (upper region) to probability zero. Figure 23.16, illustrates the type of graph that is expected to be recovered in practice, for large values of  $l$ . That is, the transition from the region (phase) of “success” (gray/lower region) to that of “fail” (red/upper region) is very sharp. As a matter of fact, there is a curve that separates the two regions. The theoretical aspects of this curve have been studied in the context of combinatorial geometry in [127] for the asymptotic case,  $l \rightarrow \infty$ , and in [128] for finite values of  $l$ . Observe that the larger the value of  $\alpha$  (larger percentage of measurements) the larger the value of  $\beta$  at which the transition occurs. This is in line with what we have said so far in this paper, and the problem gets increasingly harder as one moves

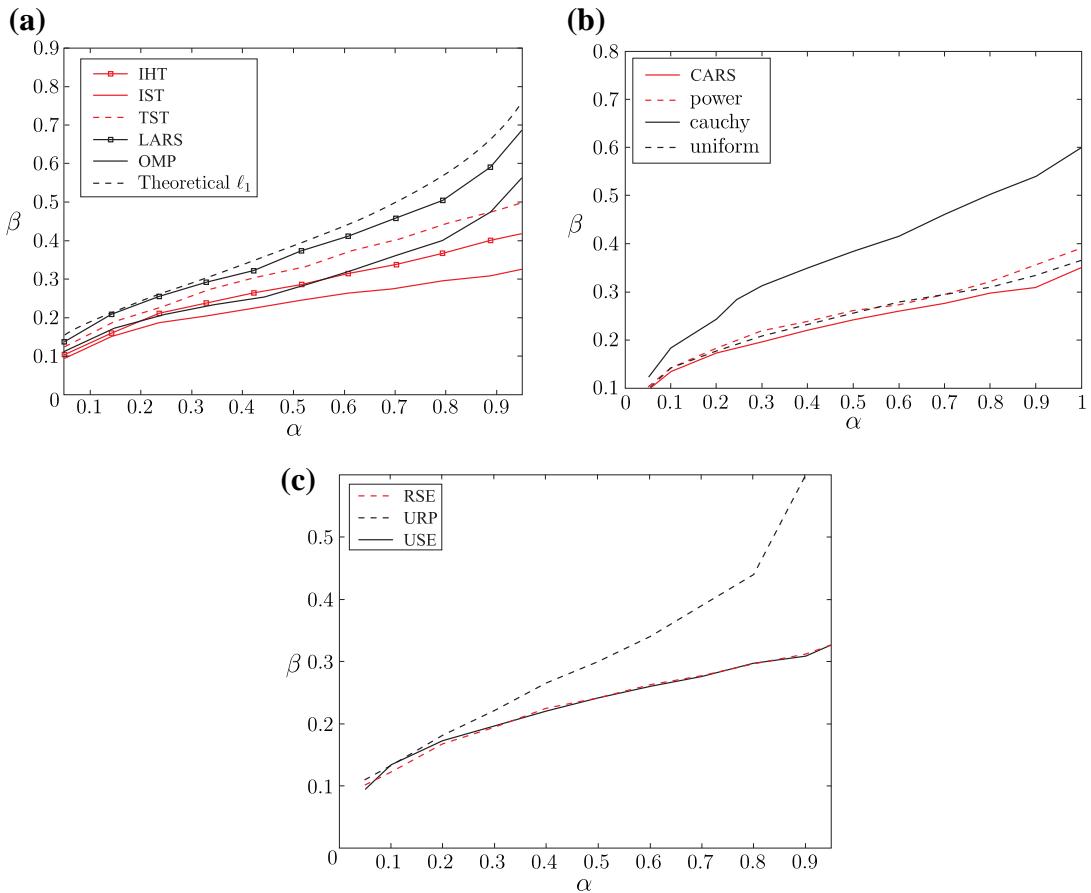
**FIGURE 23.16**

Typical phase transition behavior of a sparsity promoting algorithm. Gray (lower region) corresponds to 100% success of recovering the sparsest solution and red (upper region) to 0%. For high dimensional spaces, the transition is very sharp, as it is the case in the figure. For lower dimensionality spaces, the transition from gray (lower region) to red (upper region) is smoother and involves a region of varying color intensity. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this book.)

up and to the left in the graph. In practice, for smaller values of  $l$ , the transition region between the two regions is smoother, and it gets narrower as  $l$  increases. In practice, one can draw an approximate curve that separates the “success” and “fail” regions, using regression techniques, see, e.g., [119].

The reader may already be aware of the fact that, so far, we have avoided to talk about the performance of individual algorithms. We have just discussed some “typical” behavior that algorithms tend to exhibit in practice. What the reader might have expected is to discuss comparative performance tests and draw related conclusions. We have not done it since we feel that it is early in time to have “definite” performance conclusions, and this field is still in an early stage. Most authors compare their newly suggested algorithm with a few other algorithms, usually within a certain algorithmic family and, more important, under some specific scenarios, where the advantages of the newly suggested algorithm are documented. However, the performance of an algorithm can change significantly by changing the experimental scenario, under which the tests are carried out. The most comprehensive comparative performance study, so far, has been carried out in [119]. However, even in this one, the scenario of exact measurements has been considered and there are no experiments concerning the robustness of individual algorithms to the presence of noise. It is important to say that this study involved a huge effort of computation. We will comment on some of the findings from this study, which will also reveal to the reader that different experimental scenarios can significantly affect the performance of an algorithm.

Figure 23.17a shows the obtained phase transition curves for (a) the iterative hard thresholding (IHT), (b) the iterative soft thresholding scheme of (23.48) (IST), (c) the Two-Stage-Thresholding scheme (TST), as discussed earlier on, (d) the LARS, and (e) the OMP algorithms, together with the

**FIGURE 23.17**

(a) The obtained phase transition curves for different algorithms under the same experimental scenario, together with the theoretical one. (b) Phase transition curve for the IST algorithm under different experimental scenarios for generating the target sparse vector. (c) The phase transition for the IST algorithms under different experimental scenarios for generating the sensing matrix  $X$ .

theoretically obtained one using  $\ell_1$  minimization. All algorithms were tuned with the optimal values, with respect to the required user-defined parameters, after extensive experimentation. The results in the Figure correspond to the uniform spherical scenario, for the generation of the sensing matrices. Sparse vectors were generated according to the  $\pm 1$  scenario, for the nonzero coefficients. The interesting observation is that, although the curves deviate from each other as we move to larger values of  $\beta$ , for smaller values, the differences in their performance become less and less. This is also true for computationally simple schemes, such as the IHT one. The performance of LARS is close to the

optimal one. However, this comes at the cost of computational increase. The required computational time for achieving the same accuracy, as reported in [119], favor the TST algorithm. In some cases, LARS required excessively longer time to reach the same accuracy, in particular when the sensing matrix was the partial Fourier one and fast schemes to perform matrix vector products can be exploited. For such matrices, the thresholding schemes (IHT, IST, TST) exhibited a performance that scales very well to large size problems.

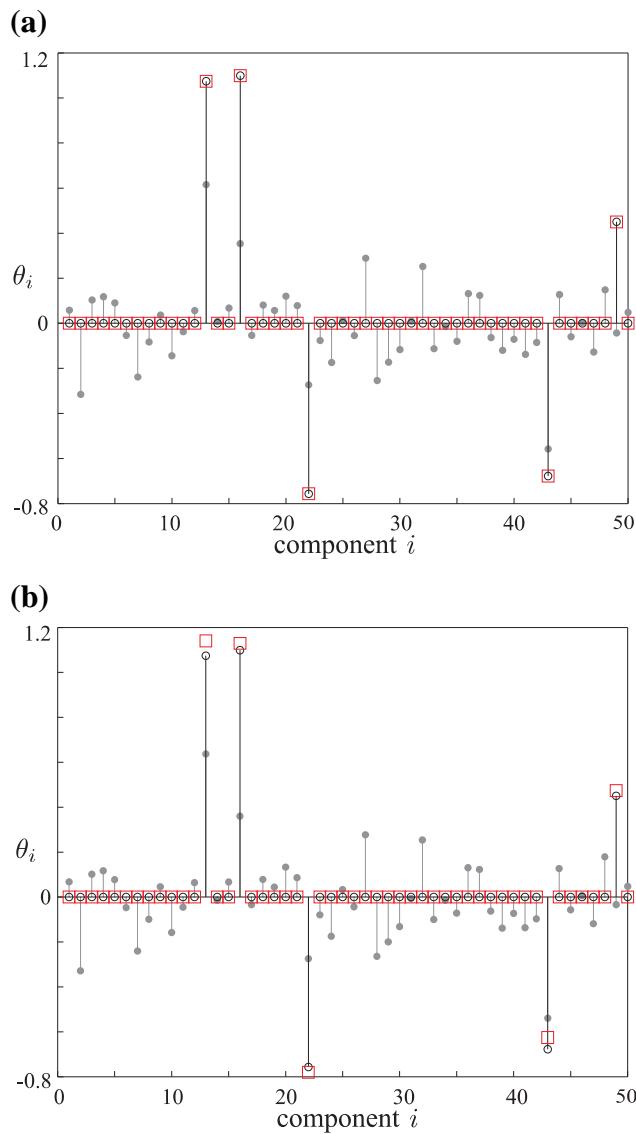
Figure 23.17b indicates the phase transition curve for one of the algorithms (IST) as we change the scenarios for generating the sparse (target) vectors, using different distributions; (a)  $\pm 1$ , with equiprobable selection of signs (Constant Amplitude Random Selection (CARS)), (b) double exponential (power), (c) Cauchy, and (d) uniform in  $[-1, 1]$ . This is indicative and typical for other algorithms as well, with some of them being more sensitive than others. Finally, Figure 23.17c shows the transition curves for the IST algorithm by changing the sensing matrix generation scenario. Three curves are shown corresponding to (a) uniform spherical ensemble (USE), (b) random sign ensemble (RSE), where the elements are  $\pm 1$  with signs uniformly distributed, and (c) the uniform random projection (URP) ensemble. Once more, one can observe the possible variations that are expected due to the use of different matrix ensembles. Moreover, changing ensembles affects each algorithm in a different way.

Concluding this section it must be emphasized that the field of algorithmic development is still an ongoing research field and it is early to come with definite and concrete comparative performance conclusions. Moreover, besides the algorithmic front, existing theories often fall short to predict what is observed in practice, with respect to their phase transition performance. For a related discussion, see, e.g., [43].

**Example 5.** We are given a set of  $N = 20$  measurements stacked in the  $\mathbf{y} \in \mathcal{R}^N$  vector. These were taken by applying a sensing matrix  $X$  on an “unknown” vector in  $\mathcal{R}^{50}$ , which is known to be sparse with  $k = 5$  nonzero components; the location of these nonzero components in the unknown vector are not known. The sensing matrix was a random matrix with elements drawn from a normal distribution  $\mathcal{N}(0, 1)$  and then the columns were normalized to unit norm. There are two scenarios for the measurements. In the first one, we are given the exact measurements while in the second one white Gaussian noise of variance  $\sigma^2 = 0.025$  was added.

In order to recover the unknown sparse vector, the CoSaMP algorithm was used for both scenarios.

The results are shown in Figures 23.18a and 23.18b for the noiseless and noisy scenarios, respectively. The values of the true unknown vector  $\boldsymbol{\theta}$  are represented with black stems topped with open circles. Note that all but five of them are zero. In Figure 23.18a exact recovery of the unknown values is succeeded; the estimated values of  $\theta_i$ ,  $i = 1, 2, \dots, 50$ , are indicated with squares in red color (for the web version). In the noisy case of Figure 23.18b, the resulted estimates, which are denoted with squares, deviate from the correct values. Note that estimated values very close to zero ( $|\theta| \leq 0.01$ ) have been omitted from the figure in order to facilitate visualizing. In both figures, the stemmed gray filled circles correspond to the minimum  $\ell_2$  norm LS solution. The advantages of adopting a sparsity-promoting approach to recover the solution are obvious. The CoSaMP algorithm was provided with the exact number of sparsity. The reader is advised to play with this example by experimenting with different values of the parameters and see how results are affected.

**FIGURE 23.18**

(a) Noiseless case. The values of the true vector, which generated the data for the Example 5, are shown with stems topped with open circles. The recovered points, using the CoSaMP, are shown with squares. An exact recovery of the signal has been obtained. The stems topped with gray filled circles correspond to the minimum Euclidean norm LS solution. (b) This figure corresponds to the noisy counterpart of that in (a). In the presence of noise, exact recovery is not possible and the more the power of the noise is the less accurate the results are.

### 1.23.12 Variations on the sparsity-aware theme

In our tour, so far, we have touched a number of aspects of the sparsity-aware learning that come from the main stream of the theoretical developments. However, more and more variants appear, which are developed with the goal to address problems of a more special structure and/or to propose alternatives, which can be beneficial in boosting the performance in practice, by serving the needs of specific applications. These variants focus either on the regularization term in (23.13) or on the misfit-measuring term or on both. Once more, research activity in this direction is dense and our purpose is to simply highlight possible alternatives and make the reader alert of the various possibilities that spring from the basic theory.

In a number of tasks, it is a priori known that the nonzero coefficients in the target signal/vector occur in groups and they are not randomly spread in all possible positions. Such a typical example is the echo path in internet telephony, where the nonzero coefficients of the impulse response tend to cluster together, see Figure 23.6. Other examples of “structured” sparsity can be traced in DNA microarrays, MIMO channel equalization, source localization in sensor networks, magnetoencephalography or in neuroscience problems, e.g., [63, 129–131]. As it is always the case in Machine Learning, being able to incorporate a priori information in the optimization can only be of benefit for improving performance, since the estimation task is externally assisted in its effort to search for the target solution.

The *group LASSO*, [132–135] addresses the task where it is a priori known that the nonzero components occur in groups. The unknown vector  $\boldsymbol{\theta}$  is divided into, say,  $L$  groups, i.e.,

$$\boldsymbol{\theta}^T = [\boldsymbol{\theta}_1^T, \dots, \boldsymbol{\theta}_L^T]^T,$$

each of them of a predetermined size,  $s_i, i = 1, 2, \dots, L$ , with  $\sum_{i=1}^L s_i = l$ . The regression model can then be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\eta} = \sum_{i=1}^L \mathbf{X}_i \boldsymbol{\theta}_i + \boldsymbol{\eta},$$

where each  $\mathbf{X}_i$  is a submatrix of  $\mathbf{X}$  comprising the corresponding  $s_i$  columns. The solution of the group LASSO is given by the following LS regularized task

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left( \left\| \mathbf{y} - \sum_{i=1}^L \mathbf{X}_i \boldsymbol{\theta}_i \right\|_2^2 + \lambda \sum_{i=1}^L \sqrt{s_i} \|\boldsymbol{\theta}_i\|_2 \right), \quad (23.59)$$

where  $\|\boldsymbol{\theta}_i\|_2$  is the Euclidean norm (not the squared one) of  $\boldsymbol{\theta}_i$ , i.e.,

$$\|\boldsymbol{\theta}_i\|_2 = \sqrt{\sum_{j=1}^{s_i} |\theta_{i,j}|^2}.$$

In other words, the individual components of  $\boldsymbol{\theta}$  that contribute to the formation of the  $\ell_1$  norm, in the standard LASSO formulation are now replaced by the square root of the energy of each individual block. In this setting, it is not the individual components but blocks of them which are forced to zero,

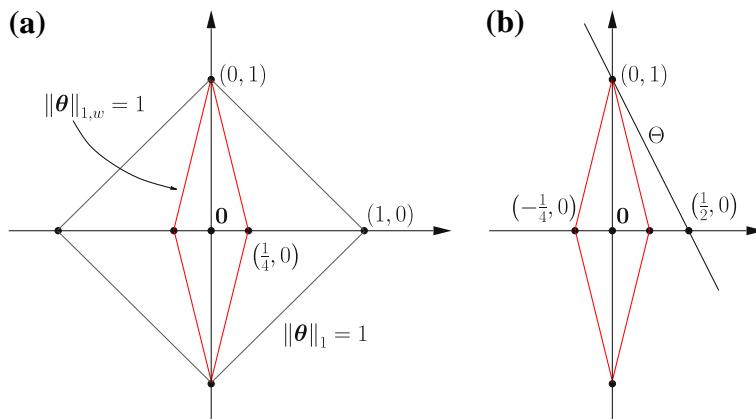
when their contribution to the LS misfit measuring term is not significant. Sometimes, this type of regularization is coined as the  $\ell_1/\ell_2$  regularization. It is straightforward to see that if  $L = l$ , then the group LASSO becomes the standard LASSO method. An alternative formulation of the group sparse model using greedy algorithms is considered in [136]. Theoretical results that extend the RIP to the so-called block RIP have been developed and reported, see, e.g., [64, 137].

In [129, 138], the so-called *model based* Compressed Sensing is addressed. The  $(k, C)$  model allows the significant coefficients of a  $k$ -sparse signal to appear in at most  $C$  clusters, whose size is unknown. This is a major difference with the group LASSO, that was reported before. In Section 1.23.10, it was commented that searching for a  $k$ -sparse solution takes place in a union of subspaces, each one of dimensionality  $k$ . Imposing a certain structure on the target solution restricts the searching in a subset of these subspaces and leaves a number of these out of the game. This obviously facilitates the optimization task. In [138], a dynamic programming technique is adopted to obtain the solution. In [139], structured sparsity is considered in terms of graphical models. An even more advanced block sparsity model is the C-HiLasso, which allows each block to have a sparse structure itself, [140].

In [141], it is suggested to replace the  $\ell_1$  norm by a weighted version of it. To justify such a choice, let us recall Example 2 and the case where the “unknown” system was sensed using  $x = [2, 1]^T$ , shown in Figure 23.10c. We have seen that by “blowing” up the  $\ell_1$  ball, the wrong sparse solution was obtained. Let us now replace the  $\ell_1$  norm in (23.28) with its weighted version

$$\|\theta\|_{1,w} := w_1|\theta_1| + w_2|\theta_2|, \quad w_1, w_2 > 0,$$

and set  $w_1 = 4$  and  $w_2 = 1$ . Figure 23.19a shows the isovalue curve  $\|\theta\|_{1,w} = 1$ , together with that resulting from the standard  $\ell_1$  norm. The weighted one is sharply “pinched” around the vertical axis, and the larger the value of  $w_1$  is, compared to that of  $w_2$ , the sharper the corresponding ball will be.



**FIGURE 23.19**

(a) The isovalue curves for the  $\ell_1$  and the weighted  $\ell_1$  norms for the same value. The weighted  $\ell_1$  is sharply pinched around one of the axis, depending on the weights. (b) Adopting to minimize the weighted  $\ell_1$  norm, for the setup of Figure 23.10 the correct sparse solution is obtained.

Figure 23.19b shows what happens when “blowing” the weighted  $\ell_1$  ball. It will first touch the point  $(0, 1)$ , which is the true solution. Basically, what we have done is to “squeeze” the  $\ell_1$  ball to be aligned more to the axis that contains the (sparse) solution. For the case of our example, any weight  $w_1 > 2$  would do the job.

Considering now the general case of a weighted norm

$$\|\boldsymbol{\theta}\|_{1,w} := \sum_{j=1}^l w_j |\theta_j|, \quad w_j > 0.$$

The ideal choice of the weights would be

$$w_j = \begin{cases} \frac{1}{|\theta_{0,j}|}, & \theta_{0,j} \neq 0, \\ \infty, & \theta_{0,j} = 0, \end{cases}$$

where  $\boldsymbol{\theta}_0$  is the target true vector, and where we have silently assumed that  $0 \cdot \infty = 0$ . In other words, the smaller a coefficient is the larger the respective weight becomes. This is justified, since large weighting will force respective coefficients towards zero during the minimization process. Of course, in practice, the values of the true vector are not known, so it is suggested to use their estimates during each iteration of the minimization procedure. The resulting scheme is of the following form.

### Algorithm 3.

1. Initialize weights to unity,  $w_j^{(0)} = 1, j = 1, 2, \dots, l$ .
2. Minimize the weighted  $\ell_1$  norm,

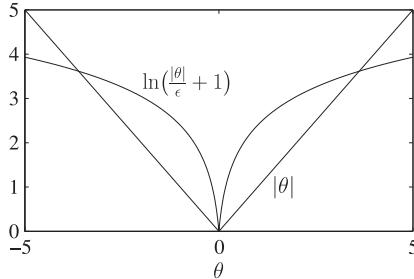
$$\begin{aligned} \boldsymbol{\theta}^{(i)} &= \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \|\boldsymbol{\theta}\|_{1,w} \\ \text{s.t. } \mathbf{y} &= X\boldsymbol{\theta}. \end{aligned}$$

3. Update the weights

$$w_j^{(i+1)} = \frac{1}{|\theta_j^{(i)}| + \epsilon}, \quad j = 1, 2, \dots, l.$$

4. Terminate when a stopping criterion is met, otherwise return to step 2.

The constant  $\epsilon$  is a small user-defined parameter to guarantee stability when the estimates of the coefficients take very small values. Note that if the weights have constant preselected values, the task retains its convex nature; this is no longer true when the weights are changing. It is interesting to point out that this intuitively motivated weighting scheme can result if the  $\ell_1$  norm is replaced by  $\sum_{j=1}^l \ln(|\theta_j| + \epsilon)$  as the regularizing term of (23.13). Figure 23.20 shows the respective graph, in the one-dimensional space together with that of the  $\ell_1$  norm. The graph of the logarithmic function reminds us of the  $\ell_p$ ,  $p < 0 < 1$ , “norms” and the comments made in Section 1.23.3. This is no more a convex function and the iterative scheme, given before, is the result of a majorization-minimization procedure in order to solve the resulting non-convex task, [141].

**FIGURE 23.20**

One-dimensional graphs of the  $\ell_1$  norm and the logarithmic regularizer  $\ln\left(\frac{|\theta|}{\epsilon} + 1\right) = \ln(|\theta| + \epsilon) - \ln\epsilon$ , with  $\epsilon = 0.1$ . The term  $\ln\epsilon$  was subtracted for illustration purposes only and does not affect the optimization. Notice the nonconvex nature of the logarithmic regularizer.

The concept of the iterative weighting, as used before, has also been applied in the context of the *iterative reweighted least squares algorithm*. Observe that the  $\ell_1$  norm can be written as

$$\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^l |\theta_j| = \boldsymbol{\theta}^T \mathcal{W}_\theta \boldsymbol{\theta},$$

where

$$\mathcal{W}_\theta = \begin{bmatrix} \frac{1}{|\theta_1|} & 0 & \cdots & 0 \\ 0 & \frac{1}{|\theta_2|} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{|\theta_l|} \end{bmatrix},$$

and where in the case of  $\theta_i = 0$ , for some  $i \in \{1, 2, \dots, l\}$ , the respective coefficient of  $\mathcal{W}_\theta$  is defined to be 1. If  $\mathcal{W}_\theta$  were a constant weighting matrix, i.e.,  $\mathcal{W}_\theta := \mathcal{W}_{\tilde{\theta}}$ , for some fixed  $\tilde{\theta}$ , then obtaining the minimum

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \|\mathbf{y} - X\boldsymbol{\theta}\|_2^2 + \lambda \boldsymbol{\theta}^T \mathcal{W}_{\tilde{\theta}} \boldsymbol{\theta},$$

is straightforward and similar to the ridge regression. In the iterative reweighted scheme,  $\mathcal{W}_\theta$  is replaced by  $\mathcal{W}_{\theta^{(i)}}$ , formed by using the respected estimates of the coefficients, which have been obtained from the previous iteration, i.e.,  $\tilde{\boldsymbol{\theta}} := \boldsymbol{\theta}^{(i)}$ , as we did before. In the sequel, each iteration solves a weighted ridge regression task. Variants of this basic iteratively weighting scheme have also been proposed, see, e.g., [125] and the references therein.

In [142], the LASSO task is modified by replacing the square error term with one involving correlations and the minimization task becomes

$$\begin{aligned} \hat{\boldsymbol{\theta}} : \min_{\boldsymbol{\theta} \in \mathcal{R}^l} & \quad \|\boldsymbol{\theta}\|_1 \\ \text{s.t. } & \quad \|X^T(\mathbf{y} - X\boldsymbol{\theta})\|_\infty \leq \epsilon, \end{aligned}$$

where  $\epsilon$  is related to  $l$  and the noise variance. This task is known as the *Dantzig selector*. That is, instead of constraining the energy of the error, the constraint, now, imposes an upper limit to the correlation of the error vector with any of the columns of  $X$ . In [57, 143], it is shown that under certain conditions the LASSO estimator and the Dantzig selector become identical.

*Total Variation (TV)* [144] is a closely related to  $\ell_1$  sparsity promotion notion and it has been widely used in image processing. Most of the grayscale image arrays,  $I \in \mathcal{R}^{l \times l}$ , consist of slowly varying pixel intensities except at the edges. As a consequence, the discrete gradient of an image array will be approximately sparse (compressible). The discrete directional derivatives of an image array are defined pixel-wise as

$$\nabla_x(I)(i, j) := I(i+1, j) - I(i, j), \quad \forall i \in \{1, 2, \dots, l-1\}, \quad (23.60)$$

$$\nabla_y(I)(i, j) := I(i, j+1) - I(i, j), \quad \forall j \in \{1, 2, \dots, l-1\}, \quad (23.61)$$

and

$$\nabla_x(I)(l, j) := \nabla_y(I)(i, l) := 0, \quad \forall i, j \in \{1, 2, \dots, l-1\}. \quad (23.62)$$

The discrete gradient transform

$$\nabla : \mathcal{R}^{l \times l} \rightarrow \mathcal{R}^{l \times 2l},$$

is defined in terms of a matrix form as

$$\nabla(I)(i, j) := [\nabla_x(I)(i, j), \nabla_y(I)(i, j)], \quad \forall i, j \in \{1, 2, \dots, l\}. \quad (23.63)$$

The total variation of the image array is defined as the  $\ell_1$  norm of the *magnitudes* of the elements of the discrete gradient transform, i.e.,

$$\|I\|_{TV} := \sum_{i=1}^l \sum_{j=1}^l \|\nabla(I)(i, j)\|_2 = \sum_{i=1}^l \sum_{j=1}^l \sqrt{\nabla_x(I)^2(i, j) + \nabla_y(I)^2(i, j)}. \quad (23.64)$$

Note that this is a mixture of  $\ell_2$  and  $\ell_1$  norms. The sparsity promoting optimization around the total variation is defined as

$$\begin{aligned} I_* &\in \arg \min_I \|I\|_{TV} \\ &\text{s.t. } \|\mathbf{y} - \mathcal{F}(I)\|_2 \leq \epsilon, \end{aligned} \quad (23.65)$$

where  $\mathbf{y} \in \mathcal{R}^N$  is the measurements vector and  $\mathcal{F}(I)$  denotes the result in vectorized form of the application of a linear operator on  $I$ . For example, this could be the result of the action of a partial two-dimensional DFT on the image. Subsampling of the DFT matrix as a means to form sensing matrices has already been discussed in Section 1.23.8.2. The task in (23.65) retains its convex nature and it basically expresses our desire to reconstruct an image which is as smooth as possible given the available measurements. The NESTA algorithm can be used for solving the total variation minimization task; besides it, other efficient algorithms for this task can be found in, e.g., [145, 146].

It has been shown in [52], for the exact measurements case ( $\epsilon = 0$ ), and in [147], for the erroneous measurements case, that conditions and bounds which guarantee recovery of an image array from the task in (23.65) can be derived and are very similar with those that we have discussed for the case of the  $\ell_1$  norm.

**Example 6 (Magnetic Resonance Imaging (MRI)).** In contrast to ordinary imaging systems, which directly acquire pixel samples, MRI scanners sense the image in an encoded form. Specifically, MRI scanners sample components in the spatial frequency domain, known as “ $k$ -space” in the MRI nomenclature. If all the components in this transform domain were available, one could apply the inverse 2D-DFT to recover the exact MR image in the pixel domain. Sampling in the  $k$ -space is realized along particular trajectories in a number of successive acquisitions. This process is time consuming, merely due to physical constraints. As a result, techniques for efficient image recovery from a *limited number of measurements* is of high importance, since they can reduce the required acquisition time for performing the measurements. Long acquisition times are not only inconvenient but even impossible, since the patients have to stay still for long time intervals. Thus, MRI was among the very first applications where compressed sensing found its way to offer its elegant solutions.

Figure 23.21a shows the “famous” Shepp-Logan phantom, and the goal is to recover it via a limited number of (measurements) samples in its frequency domain. The MRI measurements are taken across 17 radial lines in the spatial frequency domain, as shown in Figure 23.21b. A “naive” approach to recover the image from this limited number of measuring samples would be to adopt a zero-filling rationale for the missing components. The recovered image according to this technique is shown in Figure 23.21c. Figure 23.21d shows the recovered image using the approach of minimizing the total variation, as

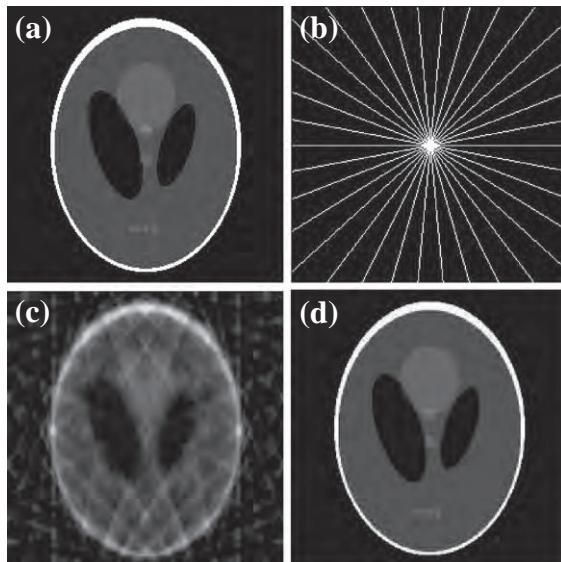


FIGURE 23.21

(a) The original Shepp-Logan image phantom. (b) The white lines indicate the directions across which the sampling in the spatial Fourier transform were obtained. (c) The recovered image after applying the inverse DFT having first filled with zeros the missing values in the DFT transform. (d) The recovered image using the total variation minimization approach.

explained before. Observe that the results for this case are astonishingly good. The original image is almost perfectly recovered. The constrained minimization was performed via the NESTA algorithm. Note that if the minimization of the  $\ell_1$  norm of the image array were used in place of the total variation, the results would not be as good; the phantom image is sparse in the discrete gradient domain, since it contains large sections which share constant intensities.

---

### 1.23.13 Online time-adaptive sparsity-promoting algorithms

In this section, online (time-recursive) schemes for sparsity-aware learning are presented. There is a number of reasons that one has to resort to such schemes in various signal processing tasks, for example, when the data arrive sequentially. Under such a scenario, using batch processing techniques to obtain an estimate of an unknown target parameter vector would be highly inefficient, since the number of training points keeps increasing. Such an approach is prohibited for real time applications. Moreover, time-recursive schemes can easily incorporate the notion of adaptivity, when the learning environment is not stationary but it undergoes changes as time evolves. Besides signal processing applications, there is an increasing number of machine learning applications where online processing is of paramount importance, such as bioinformatics, hyperspectral imaging, and data mining. In such applications, the number of training points easily amounts to a few thousand up to hundred of thousand points. Concerning the dimensionality of the ambient (feature) space, one can claim numbers that lie in similar ranges. For example, in [148], the task is to search for sparse solutions in feature spaces with dimensionality as high as  $10^9$  having access to data sets as large as  $10^7$  points. Using batch techniques, in a single computer, is out of question with today's technology.

Let us assume that there is an unknown parameter vector that generates data according to the standard regression model

$$y_n = \mathbf{x}_n^T \boldsymbol{\theta} + \eta_n, \quad \forall n,$$

and the training samples are received sequentially  $(y_n, \mathbf{x}_n), n = 1, 2, \dots$ . In the case of a stationary environment, we would expect our algorithm to converge, asymptotically as  $n \rightarrow \infty$ , to or “near to” the true parameter vector that gives birth to the measurements,  $y_n$ , when it is sensed by  $\mathbf{x}_n$ . For time varying environments, the algorithms should be able to track the underlying changes as time goes by. Before we proceed, a comment is important. Since the time index,  $n$ , is left to grow, all we have said in the previous sections with respect to underdetermined systems of equations, loses its meaning. Sooner or later we are going to have more measurements than the dimensionality of the space. Our major concern here becomes the issue of asymptotic convergence, for the case of stationary environments. The obvious question, that is now raised, is why not using a standard algorithm, e.g., LMS, RLS, or APSM [15, 16, 149], since we know that these algorithms converge to, or near enough in some sense, to the solution; that is, the algorithm will identify the zeros asymptotically. The answer is that if such algorithms are modified to be aware for the underlying sparsity, convergence is significantly speeded up; in real life applications, one has not the “luxury” to wait long time for the solution. In practice, a good algorithm should be able to provide a good enough solution, and in the case of sparse solutions to *obtain the support*, after a reasonably small number of iteration steps. In this section, the powerful theory around the  $\ell_1$  norm regularization will be used to obtain sparsity-promoting time adaptive schemes.

### 1.23.13.1 LASSO: asymptotic performance

The notions of bias, variance and consistency, are major indices for assessing the performance of an estimator. In a number of cases, such performance measures are derived asymptotically. For example, it is well known that the maximum likelihood estimator is asymptotically unbiased and consistent [1]. Also the LS estimator is asymptotically consistent. Moreover, under the assumption that the noise samples are i.i.d., the LS estimate,  $\hat{\theta}_N$ , that is obtained using  $N$  measurement (training) samples, is itself a random vector, that satisfies the  $\sqrt{N}$ -estimation consistency, e.g., [150], i.e.,

$$\sqrt{N} (\hat{\theta}_N - \theta_0) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sigma^2 \Sigma^{-1}),$$

where  $\theta_0$  is the true vector that generates the measurements,  $\sigma^2$  denotes the variance of the noise source and  $\Sigma$  is the covariance matrix  $E[\mathbf{x}\mathbf{x}^T]$  of the input sequence, which has been assumed to be zero mean and the limit denotes convergence in distribution.

The LASSO in (23.13) is the task of minimizing the  $\ell_1$  norm regularized version of the LS cost. However, nothing has been said, so far, about the statistical properties of this estimator. The only performance measure that we referred to was the error norm bound given in (23.43). However, this bound, although important in the context it was proposed for, does not provide much statistical information. Since the introduction of the LASSO estimator, a number of papers have addressed problems related to its statistical performance, see, e.g., [151–154].

When dealing with sparsity-promoting estimators, such as the LASSO, two crucial issues emerge: (a) whether the estimator, even asymptotically, can obtain the support, if the true vector parameter is a sparse one and (b) quantify the performance of the estimator with respect to the estimates of the nonzero coefficients, i.e., those whose index belongs to the support. Especially for LASSO, the latter issue becomes to study whether LASSO behaves as well as the unregularized LS with respect to these nonzero components. This task was addressed, for a first time and in a more general setting, in [153]. Let the support of the true, yet unknown,  $k$ -sparse parameter vector  $\theta_0$  be denoted as  $S$ . Let also  $\Sigma|_S$  be the  $k \times k$  covariance matrix  $E[\mathbf{x}|_S \mathbf{x}|_S^T]$ , where  $\mathbf{x}|_S \in \mathcal{R}^k$  is the vector that contains only the  $k$  components of  $\mathbf{x}$ , with indices in the support  $S$ . Then, we say that an estimator satisfies asymptotically the *oracle properties* if:

- $\lim_{N \rightarrow \infty} \text{Prob} \left\{ S_{\hat{\theta}_N} = S \right\} = 1$ . This is known as *support consistency*.
- $\sqrt{N} (\hat{\theta}_{N|S} - \theta_{0|S}) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sigma^2 \Sigma|_S^{-1})$ . This is the  $\sqrt{N}$ -estimation consistency.

We denote as  $\theta_{0|S}$  and  $\theta_{N|S}$  the  $k$ -dimensional vectors which result from  $\theta_0, \hat{\theta}_N$ , respectively, if we keep the components whose indices lie in the support  $S$ . In other words, according to the oracle properties, a good sparsity-promoting estimator should be able (a) to predict, asymptotically, the true support and (b) its performance with respect to the nonzero components should be as good as that of a genie-aided LS estimator, which is informed, in advance, of the positions of the nonzero coefficients.

Unfortunately, the LASSO estimator *cannot* satisfy simultaneously both conditions. It has been shown, [152–154] that:

- For support consistency, the regularization parameter  $\lambda := \lambda_N$  should be time varying such as

$$\lim_{N \rightarrow \infty} \frac{\lambda_N}{\sqrt{N}} = \infty, \quad \lim_{N \rightarrow \infty} \frac{\lambda_N}{N} = 0.$$

That is,  $\lambda_N$  must grow faster than  $\sqrt{N}$ , but slower than  $N$ .

- For  $\sqrt{N}$ -consistency,  $\lambda_N$  must grow as

$$\lim_{N \rightarrow \infty} \frac{\lambda_N}{\sqrt{N}} = 0,$$

i.e., it grows slower than  $\sqrt{N}$ .

The previous two conditions are conflicting and the LASSO estimator cannot comply with the two oracle conditions simultaneously. The proofs of the previous two points are somewhat technical and are not given here. The interested reader can obtain them from the previously given references. However, before we proceed, it is instructive to see why the regularization parameter has to grow slower than  $N$ , in any case. Without being too rigorous mathematically, recall that the LASSO solution comes from Eq. (23.13). This can be written as

$$\mathbf{0} \in -\frac{2}{N} \sum_{n=1}^N \mathbf{x}_n y_n + \frac{2}{N} \left( \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \right) \boldsymbol{\theta} + \frac{\lambda_N}{N} \partial \|\boldsymbol{\theta}\|_1, \quad (23.66)$$

where we have divided by  $N$  both sides. Taking the limit as  $N \rightarrow \infty$ , if  $\lambda_N/N \rightarrow 0$ , then we are left with the first two terms; this is exactly what we would have if the unregularized LS had been chosen as the cost function. In this case, the solution asymptotically converges<sup>4</sup> (under some general assumptions, which are assumed to hold true, here) to the true parameter vector; that is, we have strong consistency, e.g., [150].

### 1.23.13.2 The adaptive norm-weighted LASSO

There are two ways to get out of the previously stated conflict. One is to replace the  $\ell_1$  norm with a nonconvex function and this can lead to an estimator that satisfies the oracle properties simultaneously [153]. The other is to modify the  $\ell_1$  norm by replacing it with a weighted version. Recall that the weighted  $\ell_1$  norm was discussed in Section 1.23.12, as a means to assist the optimization procedure to unveil the sparse solution. Here the notion of weighted  $\ell_1$  norm comes as a necessity imposed by our willingness to satisfy the oracle properties. This gives rise to the *adaptive time-and-norm-weighted LASSO* (TNWL) cost estimator defined as<sup>5</sup>

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathcal{R}^l} \left\{ \sum_{j=1}^n \beta^{n-j} \left( y_j - \mathbf{x}_j^T \boldsymbol{\theta} \right)^2 + \lambda_n \sum_{i=1}^l w_i(n) |\theta_i| \right\}, \quad (23.67)$$

---

<sup>4</sup>Recall that this convergence is with probability 1.

<sup>5</sup>To emphasize that the number of training points is now increasing, we have used  $n$  in place of  $N$ . Capital  $N$  was previously used to denote a fixed number of points.

where  $\beta \leq 1$  is used as the forgetting factor to allow for tracking slow variations. The time varying weighting sequences is denoted as  $w_i(n)$ . There are different options. In [154] and under a stationary environment with  $\beta = 1$ , it is shown that if

$$w_i(n) = \frac{1}{|\theta_i^{\text{est}}|^\gamma},$$

where  $\theta_i^{\text{est}}$  is the estimate of the  $i$ th component obtained by *any*  $\sqrt{n}$ -consistent estimator, such as the unregularized LS, then for specific choices of  $\lambda_n$  and  $\gamma$  the estimator satisfies the oracle properties simultaneously. The main reasoning behind the weighted  $\ell_1$  norm is that as time goes by, and the  $\sqrt{n}$ -consistent estimator provides better and better estimates, then the weights corresponding to indices outside the true support (zero values) are inflated and those corresponding to the true support converge to a finite value. This helps the algorithm, simultaneously, to locate the support and obtain unbiased (asymptotically) estimates of the large coefficients.

Another choice for the weighing sequence is related to the so called *Smoothly Clipped Absolute Deviation* (SCAD) [153, 155]. This is defined as

$$w_i(n) = \chi_{(0, \mu_n)}(|\theta_i^{\text{est}}|) + \frac{(\alpha \mu_n - |\theta_i^{\text{est}}|)_+}{(\alpha - 1)\mu_n} \chi_{(\mu_n, \infty)}(|\theta_i^{\text{est}}|),$$

where  $\chi(\cdot)$  stands for the characteristic function,  $\mu_n = \lambda_n/n$ , and  $\alpha > 2$ . Basically, this corresponds to a quadratic spline function. It turns out, [155], that if  $\lambda_n$  is chosen to grow faster than  $\sqrt{n}$  and slower than  $n$ , the adaptive LASSO, with  $\beta = 1$  satisfies both oracle conditions, simultaneously.

A time adaptive scheme for solving the TNWL LASSO was presented in [156]. The cost function of the adaptive LASSO in (23.67) can be written as

$$J(\boldsymbol{\theta}) = \boldsymbol{\theta}^T R_n \boldsymbol{\theta} - \mathbf{r}_n^T \boldsymbol{\theta} + \lambda_n \|\boldsymbol{\theta}\|_{1,w(n)},$$

where

$$R_n := \sum_{j=1}^n \beta^{n-j} \mathbf{x}_j \mathbf{x}_j^T, \quad \mathbf{r}_n := \sum_{j=1}^n \beta^{n-j} y_j \mathbf{x}_j,$$

and  $\|\boldsymbol{\theta}\|_{1,w(n)}$  is the weighted  $\ell_1$  norm. It is straightforward to see, that

$$\mathbf{R}_n = \beta \mathbf{R}_{n-1} + \mathbf{x}_n \mathbf{x}_n^T, \quad \mathbf{r}_n = \beta \mathbf{r}_{n-1} + y_n \mathbf{x}_n.$$

The complexity for both of the previous updates, for matrices of a general structure, amounts to  $\mathcal{O}(l^2)$  multiply/add operations. One alternative is to update  $R_n$  and  $\mathbf{r}_n$  and then solve a convex optimization task for each time instant,  $n$ , using any standard algorithm. However, this is not appropriate for real time applications, due to its excessive computational cost. In [156], a time recursive version of a coordinate descent algorithm has been developed. As we have seen in Section 1.23.11.2, coordinate descent algorithms update one component at each recursive step. In [156], recursive steps are associated with time updates, as it is always the case with the time-recursive algorithms. As each new training pair  $(y_n, \mathbf{x}_n)$  is received, a single component of the unknown vector is updated. Hence, at each time instant,

a scalar optimization task has to be solved and its solution is given in closed form, which results in a simple soft thresholding operation. One of the drawbacks of the coordinate techniques is that each coefficient is updated every  $l$  time instants, which, for large values of  $l$ , can slow down convergence. Variants of the basic scheme that cope with this drawback are also addressed in [156], referred to as online cyclic coordinate descent time weighted Lasso (OCCD-TWL). If the weighted norm is to be used in place of the  $\ell_1$ , a RLS is run in parallel to provide the necessary weights; the resulting scheme is referred as (OCCD-TNWL). The complexity of the scheme is of the order of  $\mathcal{O}(l^2)$ . Computational savings are possible, if the input sequence is a time series and fast schemes for the updates of  $R_n$  and the RLS can then be exploited. However, if an RLS-type algorithm is used in parallel, the convergence of the overall scheme may be slowed down, since the RLS-type algorithm has to converge first, in order to provide reliable estimates for the weights, as pointed out before.

### 1.23.13.3 Adaptive CoSaMP algorithm (AdCoSaMP)

In [157], an adaptive version of the CoSaMP algorithm, which was presented in Section 1.23.11.1.3, was proposed. Iteration steps,  $i$ , now coincide with time updates,  $n$ , and the LS solver in Step 3c of the general CSMP scheme is replaced by an LMS one.

Let us focus first on the quantity  $X^T e^{(i-1)}$  in Step 3a of the CSMP scheme, which is used to compute the support at iteration  $i$ . In the adaptive setting and at (iteration) time  $n$ , this quantity is now “rephrased” as

$$X^T e(n-1) = \sum_{j=1}^{n-1} x_j e(j).$$

In order to make the algorithm flexible to adapt to variations of the environment, as the time index,  $n$ , increases, the previous correlation sum is modified to

$$p(n) := \sum_{j=1}^{n-1} \beta^{n-1-j} x_j e(j) = \beta p(n-1) + x_{n-1} e(n-1).$$

The LS task, constrained on the active columns that correspond to the indices in the support  $S$  in Step 3c, is performed in an adaptive rationale by involving the basic LMS recursions, i.e.,

$$\begin{aligned} \tilde{e}(n) &:= y_n - x_{n|S}^T \tilde{\theta}_{|S}(n-1), \\ \tilde{\theta}_{|S}(n) &:= \tilde{\theta}_{|S}(n-1) + \mu x_{n|S} \tilde{e}(n), \end{aligned}$$

where  $\tilde{\theta}_{|S}(\cdot)$  and  $x_{n|S}$  denote the respective subvectors corresponding to the indices in the support  $S$ . The resulting algorithm is given as follows:

#### Algorithm 4 (The AdCoSaMP Scheme).

1. Select the value of  $t = 2k$ .
2. Initialize the algorithm:  $\theta(1) = \mathbf{0}, \tilde{\theta}(1) = \mathbf{0}, p(1) = \mathbf{0}, e(1) = y_1$ .
3. Choose  $\mu$  and  $\beta$ .

4. For  $n = 2, 3, \dots$ , execute the following steps:

- a.  $\mathbf{p}(n) = \beta\mathbf{p}(n-1) + \mathbf{x}_{n-1}e(n-1)$ .
- b. Obtain the current support:

$$S = \text{supp}\{\boldsymbol{\theta}(n-1)\} \cup \left\{ \begin{array}{l} \text{indices of the } t \text{ largest} \\ \text{in magnitude components of } \mathbf{p}(n) \end{array} \right\}.$$

- c. Perform the LMS update:

$$\begin{aligned} \tilde{e}(n) &= y_n - \mathbf{x}_{n|S}^T \tilde{\boldsymbol{\theta}}_{|S}(n-1), \\ \tilde{\boldsymbol{\theta}}_{|S}(n) &= \tilde{\boldsymbol{\theta}}_{|S}(n-1) + \mu \mathbf{x}_{n|S} \tilde{e}(n). \end{aligned}$$

- d. Obtain the set  $S_k$  of the indices of the  $k$  largest components of  $\tilde{\boldsymbol{\theta}}_{|S}(n)$ .
- e. Obtain  $\boldsymbol{\theta}(n)$  such that:

$$\boldsymbol{\theta}_{|S_k}(n) = \tilde{\boldsymbol{\theta}}_{|S_k}, \quad \text{and} \quad \boldsymbol{\theta}_{|S_k^c}(n) = \mathbf{0},$$

where  $S_k^c$  is the complement set of  $S_k$ .

- f. Update the error:  $e(n) = y_n - \mathbf{x}_n^T \boldsymbol{\theta}(n)$ .

In place of the standard LMS, its normalized version can alternatively be adopted. Note that Step 4e is directly related to the hard thresholding operation.

In [157], it is shown that if the sensing matrix, which is now time dependent and keeps increasing in size, satisfies a condition similar to RIP, for each time instant, called *Exponentially Weighted Isometry Property* (ERIP), which depends on  $\beta$ , then the algorithm asymptotically satisfies an error bound, which is similar to the one that has been derived for CoSaMP in [89], plus an extra term that is due to the excess Mean Square Error, which is the price paid by replacing the LS solver by the LMS.

#### 1.23.13.4 Sparse adaptive parallel projection onto convex sets method (SpAPSM)

The APSM family of algorithms is one among the most powerful techniques for adaptive learning [149]. A major advantage of this algorithmic family is that one can readily incorporate convex constraints. The rationale behind APSM is that since our data are known to be generated by a regression model, then the unknown vector could be estimated by finding a point in the intersection of a sequence of hyperslabs, that are defined by the data points, i.e.,  $S_n[\epsilon] := \{\boldsymbol{\theta} \in \mathcal{R}^l : |y_n - \mathbf{x}_n^T \boldsymbol{\theta}| \leq \epsilon\}$ . Such a model is most natural when the noise is bounded, (which, after all, it is the case in any practical application). In case the noise is assumed unbounded, a choice of  $\epsilon$  of the order say,  $\sigma$ , can guarantee, with high probability, that the unknown solution lies inside these hyperslabs.

The APSM family builds upon the elegant philosophy that runs across the classical projections onto convex sets (POCS) theory. Recall that the basic rationale behind POCS is that starting from an arbitrary point in the space and sequentially projecting onto a *finite number* of convex sets then the sequence of projections converges, in some sense, into the intersection of all these sets, assuming this is not empty. The theory was extended to embrace the online processing setting in [158–160]. In contrast to

the classical POCS theory, here the number of the involved convex sets is *infinite*. It turns out that, under certain general conditions, a sequence of projections over all these sets also converges to a point in their intersection.

To fit the theory into our needs, the place of the aforementioned convex sets is taken by the hyperslabs, which are formed by the received training data, as mentioned before. Thus, the resulting algorithms involves (metric) projections onto these hyperslabs (see Appendix). However, when dealing with sparse vectors, there is an extra projection associated with the convex set formed by the  $\ell_1$  ball; that is,  $\|\boldsymbol{\theta}\|_1 \leq \rho$  (see, also, the LASSO formulation (23.14)). Hence, this task fits nicely in the APSM rationale and the basic recursion can be readily written, without much thought or derivation, as follows; for any arbitrarily chosen initial point  $\boldsymbol{\theta}(0)$ , define  $\forall n$ ,

$$\boldsymbol{\theta}(n) := P_{B_{\ell_1}[\delta]} \left( \boldsymbol{\theta}(n-1) + \mu_n \left( \frac{1}{q} \sum_{i=n-q+1}^n P_{S_i[\epsilon]}(\boldsymbol{\theta}(n-1)) - \boldsymbol{\theta}(n-1) \right) \right),$$

where  $P_{S_i[\epsilon]}$  is the metric projection onto the hyperslab  $S_i[\epsilon]$  (see Appendix). Note, that in the previous recursion we have used  $q$ , instead of one, hyperslabs whose metric projections are averaged out at time  $n$ . It turns out that such an averaging improves convergence significantly. Parameter  $\mu_n$  is an extrapolation parameter, which takes values in the interval  $(0, 2\mathcal{M}_n)$ , where

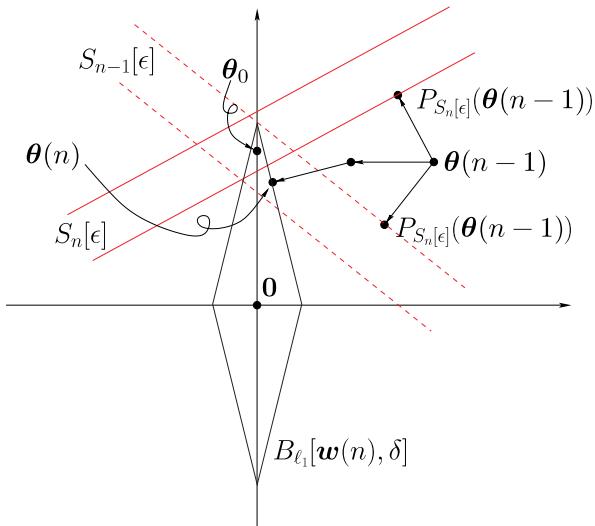
$$\mathcal{M}_n := \begin{cases} \frac{\sum_{i=n-q+1}^n \omega_i^{(n)} \|P_{S_i[\epsilon]}(\boldsymbol{\theta}(n-1)) - \boldsymbol{\theta}(n-1)\|^2}{\left\| \sum_{i=n-q+1}^n \omega_i^{(n)} P_{S_i[\epsilon]}(\boldsymbol{\theta}(n-1)) - \boldsymbol{\theta}(n-1) \right\|^2}, \\ \quad \text{if } \left\| \sum_{i=n-q+1}^n \omega_i^{(n)} P_{S_i[\epsilon]}(\boldsymbol{\theta}(n-1)) - \boldsymbol{\theta}(n-1) \right\| \neq 0, \\ 1, \quad \text{otherwise,} \end{cases} \quad (23.68)$$

and  $P_{B_{\ell_1}[\rho]}(\cdot)$  is the projection operator onto the  $\ell_1$  ball  $B_{\ell_1}[\rho] := \{\boldsymbol{\theta} \in \mathcal{R}^l : \|\boldsymbol{\theta}\|_1 \leq \rho\}$ , since the solution is constrained to live within this ball. Note, that the previous recursion is analogous to the iterative soft thresholding shrinkage algorithm in the batch processing case, (23.53). There, we saw that the only difference that the sparsity imposes on an iteration, with respect to its unconstrained counterpart, is an extra soft thresholding. This is exactly the case here. The term in the parenthesis is the iteration for the unconstrained task. Moreover, as it has been shown in [161], projection on the  $\ell_1$  ball is equivalent to a soft thresholding operation. It can be shown that the previous iteration converges arbitrarily close to a point in the intersection

$$B_{\ell_1}[\delta] \cap \bigcap_{n \geq n_0} S_n[\epsilon],$$

for some finite value of  $n_0$  [149, 158–160]. In [162, 163] the weighted  $\ell_1$  ball has been used to improve convergence as well as the tracking speed of the algorithm, when the environment is time varying. The weights were adopted in accordance to what was discussed in Section 1.23.12, i.e.,

$$w_i(n) := \frac{1}{|\theta_i(n-1)| + \epsilon_n}, \quad \forall i \in \{1, 2, \dots, l\},$$

**FIGURE 23.22**

Geometric illustration of the update steps involved in the SpAPSM algorithm, for the case of  $q = 2$ . The update at time  $n + 1$  is obtained by first convexly combining the projections onto the current and previously formed hyperslabs,  $S_n[\epsilon]$ ,  $S_{n-1}[\epsilon]$  and then projecting onto the weighted  $\ell_1$  ball. This brings the update closer to the target solution  $\theta_*$ .

where  $(\epsilon_n)_{n \geq 0}$  is a sequence (can be also constant) of small numbers to avoid division by zero. The basic time iteration becomes as follows; for any arbitrarily chosen initial point  $\theta(0)$ , define  $\forall n$ ,

$$\theta(n) := P_{B_{\ell_1}[\mathbf{w}(n), \rho]} \left( \theta(n-1) + \mu_n \left( \sum_{i=n-q+1}^n \omega_i^{(n)} P_{S_i[\epsilon]}(\theta(n-1)) - \theta(n-1) \right) \right), \quad (23.69)$$

where  $\mu_n \in (0, 2\mathcal{M}_n)$  and  $\mathcal{M}_n$  is given in (23.68). Figure 23.22 illustrates the associated geometry of the basic iteration in  $\mathbb{R}^2$  and for the case of  $q = 2$ . It comprises two parallel projections on the hyperslabs followed by one projection onto the weighted  $\ell_1$  ball. In [162], it is shown that a good bound for the weighted  $\ell_1$  norm is the sparsity level  $k$  of the target vector, which is assumed to be known and it is a user-defined parameter. In [162], it is shown that asymptotically, and under some general assumptions, this algorithmic scheme converges arbitrarily close to the intersection of the hyperslabs with the weighted  $\ell_1$  balls, i.e.,

$$\bigcap_{n \geq n_0} \left( P_{B_{\ell_1}[\mathbf{w}(n), \rho]} \cap S_j[\epsilon] \right),$$

for some non-negative integer  $n_0$ . It has to be pointed out that, in the case of weighted  $\ell_1$  norms, the constraint is *time varying* and the convergence analysis is not covered by the standard analysis used for APSM, and had to be extended to this more general case. The complexity of the algorithm amounts to  $\mathcal{O}(ql)$ . The larger the  $q$  the faster the convergence rate, at the expense of higher complexity. In [163],

in order to reduce the dependence of the complexity on  $q$ , the notion of the *sub-dimensional* projection is introduced, where projections onto the  $q$  hyperslabs could be restricted along the directions of the most significant coefficients, of the currently available estimates. The dependence on  $q$  now becomes  $\mathcal{O}(qk_n)$  where  $k_n$  is the sparsity level of the currently available estimate, which, after a few steps of the algorithm, gets much lower than  $l$ . The total complexity amounts to  $\mathcal{O}(l) + \mathcal{O}(qk_n)$ , per iteration step. This allows the use of large values of  $q$ , which drives the algorithm to a performance close to that of the adaptive weighted LASSO, at only a small extra computational cost.

#### 1.23.13.4.1 Projection onto the weighted $\ell_1$ ball

Projecting onto an  $\ell_1$  ball is equivalent to a soft thresholding operation. Projection onto the weighted  $\ell_1$  norm results to a slight variation of the soft thresholding, with different threshold values per component. In the sequel, we give the iteration steps for the more general case of the weighted  $\ell_1$  ball. The proof is a bit technical and lengthy and it will not be given here. It was derived, for the first time, via purely geometric arguments, and without the use of the classical Lagrange multipliers, in [162]. Lagrange multipliers have been used instead in [161], for the case of the  $\ell_1$  ball.

Given a point outside the ball,  $\boldsymbol{\theta} \in \mathcal{R}^l \setminus B_{\ell_1}[\mathbf{w}, \rho]$ , then its projection onto the weighted  $\ell_1$  ball is the point  $P_{B_{\ell_1}[\mathbf{w}, \rho]}(\boldsymbol{\theta}) \in B_{\ell_1}[\mathbf{w}, \rho] := \{z \in \mathcal{R}^l : \sum_{i=1}^l w_i |z_i| \leq \rho\}$ , that lies closest to  $\boldsymbol{\theta}$  in the Euclidean sense. If  $\boldsymbol{\theta}$  lies within the ball then it coincides with its projection. Given the weights and the value of  $\rho$ , the following iterations provide the projection.

**Algorithm 5 (Projection onto the weighted  $\ell_1$  ball  $B_{\ell_1}[\mathbf{w}, \rho]$ ).**

1. Form the vector  $[|\theta_1|/w_1, \dots, |\theta_l|/w_l]^T \in \mathcal{R}^l$ .
2. Sort the previous vector in a non-ascending order, so that  $|\theta_{\tau(1)}|/w_{\tau(1)} \geq \dots \geq |\theta_{\tau(l)}|/w_{\tau(l)}$ . The notation  $\tau$  stands for the permutation, which is implicitly defined by the sorting operation. Keep in memory the inverse  $\tau^{-1}$ , which is the index of the position of the element in the original vector.
3.  $r_1 := l$ .
4. Let  $m = 1$ . While  $m \leq l$ , do:
  - a.  $m_* := m$ .
  - b. Find the maximum  $j_*$  among those  $j \in \{1, 2, \dots, r_m\}$  such that  $\frac{|\theta_{\tau(j)}|}{w_{\tau(j)}} > \frac{\sum_{i=1}^{r_m} w_{\tau(i)} |\theta_{\tau(i)}| - \rho}{\sum_{i=1}^{r_m} w_{\tau(i)}^2}$ .
  - c. If  $j_* = r_m$  then break the loop.
  - d. Otherwise set  $r_{m+1} := j_*$ .
  - e. Increase  $m$  by 1 and go back to Step 4a.
5. Form the vector  $\hat{\mathbf{p}} \in \mathcal{R}^{r_{m_*}}$  whose  $j$ th component,  $j = 1, \dots, r_{m_*}$ , is given by

$$\hat{p}_j := |\theta_{\tau(j)}| - \frac{\sum_{i=1}^{r_{m_*}} w_{\tau(i)} |\theta_{\tau(i)}| - \rho}{\sum_{i=1}^{r_{m_*}} w_{\tau(i)}^2} w_{\tau(j)}.$$

6. Use the inverse mapping  $\tau^{-1}$  to insert the element  $\hat{p}_j$  into the  $\tau^{-1}(j)$  position of the  $l$ -dimensional vector  $\mathbf{p}$ ,  $\forall j \in \{1, 2, \dots, r_{m_*}\}$ , and fill in the rest with zeros.
7. The desired projection is  $P_{B_{\ell_1}[\mathbf{w}, \rho]}(\boldsymbol{\theta}) = [\text{sgn}(\theta_1) p_1, \dots, \text{sgn}(\theta_l) p_l]^T$ .

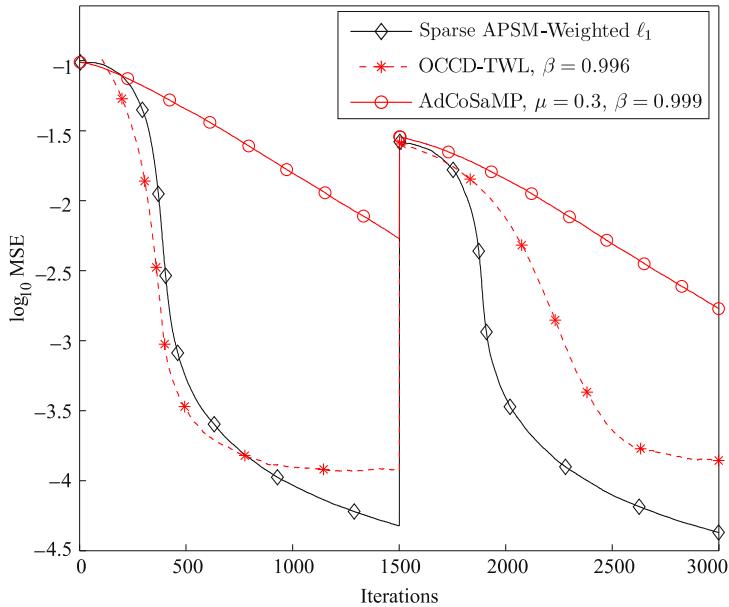
**Remarks 12.** Projections onto both  $\ell_1$  and weighted  $\ell_1$  balls impose convex sparsity inducing constraints via properly performed soft thresholding operations. More recent advances within the SpAPSM framework [164], allow the substitution of  $P_{B_{\ell_1}[\rho]}$  and  $P_{B_{\ell_1}[w, \rho]}$  with a *generalized thresholding*, built around the notions of SCAD, non-negative garrote, as well as a number of thresholding functions corresponding to the non-convex,  $\ell_p$ ,  $p < 1$  penalties. Moreover, it is shown that such generalized thresholding operators (GT) are nonlinear mappings with their fixed point set being a union of subspaces, i.e., the non-convex object which lies at the heart of any sparsity-promoting technique. Such schemes are very useful for low values of  $q$ , where one can improve upon the performance obtained by the LMS-based AdCoSAMP, at comparable complexity levels.

**Example 7 (Time varying signal).** In this example, the performance curves of the most typical online algorithms, mentioned before, are studied in the context of a time varying environment. A typical simulation setup, which is commonly adopted by the adaptive filtering community, in order to study the tracking agility of an algorithm, is that of an unknown vector which undergoes an abrupt change after a number of observations. Here, we consider a signal,  $s$ , with a sparse wavelet representation, i.e.,  $s = \Phi\theta$ , where  $\Phi$  is the corresponding transformation matrix. In particular, we set  $l = 1024$  with 100 nonzero wavelet coefficients. After 1500 measurements (observations), ten arbitrarily picked wavelet coefficients change their values to new ones selected uniformly at random from the interval  $[-11]$ . Note that this may affect the sparsity level of the signal, and we can now end with up to 110 nonzero coefficients. A total of  $N = 3000$  sensing vectors are used, which result from the wavelet transform of the input vectors  $x_n \in \mathcal{R}^l$ ,  $n = 1, 2, \dots, 3000$ , having elements drawn from  $\mathcal{N}(0, 1)$ . In this way, the adaptive algorithms do not estimate the signal itself, but its sparse wavelet representation,  $\theta$ . The observations are corrupted by additive white Gaussian noise of variance  $\sigma_n^2 = 0.1$ . Regarding SpAPSM, the extrapolation parameter  $\mu_n$  is set equal to  $1.8 \times \mathcal{M}_n$ , the hyperslabs parameter  $\epsilon$  was set equal to  $1.3\sigma_n$  and  $q = 390$ . The parameters for all algorithms were selected in order to optimize their performance. Since the sparsity level of the signal may change (from  $k = 100$  up to  $k = 110$ ) and since in practice it is not possible to know in advance the exact value of  $k$ , we feed the algorithms with an overestimate,  $\hat{k}$ , of the true sparsity value and in particular we used  $\hat{k} = 150$  (i.e., 50% overestimation up to the 1500th iteration).

The results are shown in Figure 23.23. Note the enhanced performance obtained via the SpAPSM algorithm. However, it has to be pointed out that the complexity of the AdCoSAMP is much lower compared to the other two algorithms, for the choice of  $q = 390$  for the SpAPSM. The interesting observation is that SpAPSM achieves a better performance compared to OCCD-TWL, albeit at significantly lower complexity. If on the other hand complexity is of major concern, as it has already been pointed out in 12, use of SpAPSM offers the flexibility to use GT operators, which lead to improved performance for small values of  $q$  at complexity comparable to that of LMS-based sparsity promoting algorithms [165].

### 1.23.14 Learning sparse analysis models

All our discussion, so far, has been exhausted in the terrain of signals which are either sparse themselves or they can be sparsely represented in terms of the atoms of a dictionary in a synthesis model, as

**FIGURE 23.23**

MSE learning curves for AdCoSAMP, SpAPSM and OCCD-TWL for the simulation example discussed in 7. The vertical axis shows the  $\log_{10}$  of the Mean Squares Error, i.e.,  $\log_{10} \left( \frac{1}{2} \|\mathbf{s} - \Phi\theta(n)\|_2^2 \right)$  and the horizontal shows the time index. At time  $n = 1500$ , the system undergoes a sudden change.

introduced in (23.23), i.e.,

$$\mathbf{s} = \sum_{i \in \mathcal{I}} \theta_i \psi_i.$$

As a matter of fact, most of the research activity over the last decade or so has been focused on the synthesis model. This may be partly due to the fact that the synthesis modeling path may provide a more intuitively appealing structure to describe the generation of the signal in terms of the elements (atoms) of a dictionary. Recall from Section 1.23.10 that the sparsity assumption was imposed on  $\theta$  in the synthesis model and the corresponding optimization task was formulated in (23.45) and (23.46) for the exact and noisy cases, respectively.

However, this is not the only way to attack the task of sparsity modeling. Very early in this paper, in Section 1.23.5, we referred to the analysis model,

$$\mathbf{S} = \Phi^H \mathbf{s}$$

and pointed out that in a number of real life applications the resulting transform  $\mathbf{S}$  is sparse. To be fair, for such an experimental evidence, the most orthodox way to deal with the underlying model sparsity would be to consider  $\|\Phi^H \mathbf{s}\|_0$ . Thus, if one wants to estimate  $\mathbf{s}$ , a very natural way would be to cast the

related optimization task as

$$\begin{aligned} \min_s \quad & \left\| \Phi^H s \right\|_0, \\ \text{s.t. } & y = Xs, \quad \text{or} \quad \|y - Xs\|_2^2 \leq \epsilon, \end{aligned} \quad (23.70)$$

depending on whether the measurements via a sensing matrix,  $X$ , are exact or noisy. Strictly speaking, the total variation minimization approach, which was used in Example 6, falls under this analysis model formulation umbrella, since what is minimized is the  $\ell_1$  norm of the gradient transform of the image.

The optimization tasks in either of the two formulations given in (23.70) build around the assumption that the signal of interest has *sparse analysis representation*. The obvious question that is now raised is whether the optimization tasks in (23.70) and their counterparts in (23.45) or (23.46) are any different. One of the first efforts to shed light in this problem was in [166]. There, it was pointed out that the two tasks, although related, yet they are in general different. Moreover, their comparative performance depends on the specific problem at hand. However, it is fair to say that this is a new field of research and more definite conclusions are currently being shaped. An easy answer can be obtained for the case where the involved dictionary corresponds to an orthonormal transformation matrix, e.g., DFT. In this case, we already know that the analysis and synthesis matrices are related as

$$\Phi = \Psi = \Psi^{-H},$$

which leads to an equivalence between the two previously stated formulations. Indeed, for such a transform we have

$$\underbrace{S = \Phi^H s}_{\text{Analysis}} \Leftrightarrow \underbrace{s = \Phi S}_{\text{Synthesis}}.$$

Using the last formula into the (23.70), the tasks in (23.45) or (23.46) are readily obtained by replacing  $\theta$  by  $s$ . However, this reasoning cannot be carried out to the case of overcomplete dictionaries; it is for these cases, where the two optimization tasks may lead to different solutions.

The previous discussion, concerning the comparative performance between the synthesis or analysis-based sparse representations, is not only of a “philosophical” value. It turns out that, often in practice, the nature of certain overcomplete dictionaries does not permit the use of the synthesis based formulation. These are the cases where the columns of the overcomplete dictionary exhibit high degree of dependence; that is, the coherence of the matrix, as defined in Section 1.23.7.1, has large values. Typical examples of such overcomplete dictionaries are the Gabor frames, the curvelet frames and the oversampled DFT. The use of such dictionaries lead to enhanced performance in a number of applications, e.g., [167, 168]. Take as an example the case of our familiar DFT transform. This transform provides a representation of our signal samples in terms of sampled exponential sinusoids, whose frequencies are multiples of  $\frac{2\pi}{lT}$ , where  $T$  is the sampling frequency and  $lT$  is the length of our signal segment  $s$ ; that is,

$$s := \begin{bmatrix} s(0) \\ s(T) \\ \vdots \\ s((l-1)T) \end{bmatrix} = \sum_{i=0}^{l-1} S_i \psi_i, \quad (23.71)$$

where  $S_i$  are the DFT coefficients and  $\psi_i$  is the sampled sinusoid with frequency equal to  $\frac{2\pi}{T}i$ , i.e.,

$$\psi_i = \begin{bmatrix} 1 \\ \exp(-j\frac{2\pi}{T}iT) \\ \vdots \\ \exp(-j\frac{2\pi}{T}i(l-1)T) \end{bmatrix}. \quad (23.72)$$

However, this is not necessarily the most efficient representation. For example, it is highly unlikely that a signal comprises only frequencies which are multiples of the basic one; only such signals can result in a sparse representation using the DFT basis. Most probably, in general, there will be frequencies lying in between the frequency samples of the DFT basis, which result in non-sparse representations. Using these extra frequencies, a much better representation of the frequency content of the signal can be obtained. However, in such a dictionary the atoms are no more linearly independent and the coherence of the respective (dictionary) matrix increases.

Once a dictionary exhibits high coherence, then there is no way of finding a sensing matrix,  $X$ , so that  $X\Psi$  to obey the RIP. Recall that at the heart of the sparsity-aware learning lies the concept of stable embedding, that allows the recovery of a vector/signal after projecting it in a lower dimensional space; this is what all the available conditions, e.g., RIP, guarantee. However, no stable embedding is possible with highly coherent dictionaries. Take as an extreme example the case where the first and second atoms are identical. Then no sensing matrix  $X$  can achieve a signal recovery that distinguishes the vector  $[1, 0, \dots, 0]^T$  from  $[0, 1, 0, \dots, 0]^T$ . Can then one conclude that for highly coherent overcomplete dictionaries compressed sensing techniques are not possible? Fortunately, the answer to this is negative. After all, our goal in compressed sensing has always been the recovery of the signal  $s = \Psi\theta$  and not the identification of the sparse vector  $\theta$  in the synthesis model representation. The latter was just a means to an end. While the unique recovery of  $\theta$  cannot be guaranteed for highly coherent dictionaries, this does not necessarily cause any problems for the recovery of  $s$ , using a small set of measurement samples. The escape route will come by considering the analysis model formulation. However, prior to this treatment, it will be of no harm to refresh our basics concerning the theory of *frames* and recall some key definitions.

### 1.23.14.1 Some hints from the theory of frames

In order to remain in the same framework as the one already adopted for this paper and comply with the notation previously used, we will adhere to the real data case, although everything we are going to say is readily extended to the complex case, by replacing transposition with its Hermitian counterpart.

A frame in a vector space<sup>6</sup>  $V \subseteq \mathcal{R}^l$  is a generalization of the notion of a basis. Recall from our linear algebra basics that a *basis* is a set of vectors  $\psi_i, i \in \mathcal{I}$ , with the following two properties: (a)  $V = \text{span}\{\psi_i : i \in \mathcal{I}\}$ , where the cardinality  $\text{card}(\mathcal{I}) = l$  and (b)  $\psi_i, i \in \mathcal{I}$ , are mutually independent. If, in addition,  $\langle \psi_i, \psi_j \rangle = \delta_{i,j}$  then the basis is known as orthonormal. If we now relax the second condition and allow  $l < \text{card}(\mathcal{I}) := p$ , we introduce redundancy in the signal representations,

---

<sup>6</sup>We constrain our discussion in this section to finite dimensional Euclidean spaces. The theory of frames has been developed for general Hilbert spaces.

which, as it has already been mentioned, can offer a number of advantages in a wide range of applications. However, once redundancy is introduced we lose uniqueness in the signal representation

$$\mathbf{s} = \sum_{i \in \mathcal{I}} \theta_i \boldsymbol{\psi}_i, \quad (23.73)$$

due to the dependency among the vectors  $\boldsymbol{\psi}_i$ . The question that is now raised is whether there is a simple and systematic way to compute the coefficients  $\theta_i$  in the previous expansion.

**Definition 4.** The set  $\boldsymbol{\psi}_i, i \in \mathcal{I}$ , which spans a vector space,  $V$ , is called a *frame* if there exist positive real numbers,  $A$  and  $B$ , such that for any nonzero  $\mathbf{s} \in V$ ,

$$0 < A \|\mathbf{s}\|_2^2 \leq \sum_{i \in \mathcal{I}} |\langle \boldsymbol{\psi}_i, \mathbf{s} \rangle|^2 \leq B \|\mathbf{s}\|_2^2, \quad (23.74)$$

where  $A$  and  $B$  are known as the bounds of the frame.

Note that if  $\boldsymbol{\psi}_i, i \in \mathcal{I}$ , comprise an orthonormal basis, then  $A = B = 1$  and (23.74) is the celebrated Parseval's theorem. Thus, (23.74) can be considered as a generalization of Parseval's theorem. Looking at it more carefully, we notice that this is a stability condition that closely resembles our familiar RIP condition in (23.36). Indeed, the upper bound guarantees that the expansion never diverges (this applies to infinite dimensional spaces) and the lower bound guarantees that no nonzero vector,  $\|\mathbf{s}\| \neq 0$ , will ever become zero after projecting it along the atoms of the frame. To look at it from a slightly different perspective, form the dictionary matrix

$$\Psi = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \dots, \boldsymbol{\psi}_p],$$

where we used  $p$  to denote the cardinality of  $\mathcal{I}$ . Then, the lower bound in (23.74) guarantees that  $\mathbf{s}$  can be reconstructed from its transform samples  $\Psi^T \mathbf{s}$ ; note that in such a case, if  $s_1 \neq s_2$ , then their respective transform values will be different.

It can be shown that if condition (23.74) is valid, then there exists another set of vectors,  $\tilde{\boldsymbol{\psi}}_i, i \in \mathcal{I}$ , known as the *dual frame*, with the following elegant property

$$\mathbf{s} = \sum_{i \in \mathcal{I}} \langle \tilde{\boldsymbol{\psi}}_i, \mathbf{s} \rangle \boldsymbol{\psi}_i = \sum_{i \in \mathcal{I}} \langle \boldsymbol{\psi}_i, \mathbf{s} \rangle \tilde{\boldsymbol{\psi}}_i, \quad \forall \mathbf{s} \in V. \quad (23.75)$$

Once a dual frame is available, the coefficients in the expansion of a vector in terms of the atoms of a frame are easily obtained. If we form the matrix  $\tilde{\Psi}$  of the dual frame vectors, then it is easily checked out that since condition (23.75) is true for any  $\mathbf{s}$ , it implies that

$$\tilde{\Psi} \Psi^T = \Psi \tilde{\Psi}^T = I, \quad (23.76)$$

where  $I$  is the  $l \times l$  identity matrix. Note that all of us have used the property in (23.75), possibly in a disguised form, many times in our professional life. Indeed, consider the simple case of two independent vectors in the two-dimensional space (in order to make things simple). Then, (23.73) becomes

$$\mathbf{s} = \theta_1 \boldsymbol{\psi}_1 + \theta_2 \boldsymbol{\psi}_2 = \Psi \boldsymbol{\theta}.$$

Solving for the unknown  $\theta$  is nothing but the solution of a linear set of equations; note that the involved matrix  $\Psi$  is invertible. Let us rephrase a bit our familiar solution

$$\theta = \Psi^{-1}s := \tilde{\Psi}s := \begin{bmatrix} \tilde{\psi}_1^T \\ \tilde{\psi}_2^T \end{bmatrix} s, \quad (23.77)$$

where  $\tilde{\psi}_i^T$ ,  $i = 1, 2$ , are the rows of the inverse matrix. Using now the previous notation, it is readily seen that

$$s = \langle \tilde{\psi}_1, s \rangle \psi_1 + \langle \tilde{\psi}_2, s \rangle \psi_2.$$

Moreover, note that in this special case of independent vectors, the respective definitions imply

$$\begin{bmatrix} \tilde{\psi}_1^T \\ \tilde{\psi}_2^T \end{bmatrix} [\psi_1, \psi_2] = I,$$

and the dual frame is not only unique but it also fulfills the *biorthogonality* condition, i.e.,

$$\langle \tilde{\psi}_i, \psi_j \rangle = \delta_{i,j}. \quad (23.78)$$

In the case of a general frame, the dual frames are neither biorthogonal nor uniquely defined. The latter can also be verified by the condition (23.76) that defines the respective matrices.  $\Psi^T$  is a rectangular tall matrix and its left inverse is not unique. There is, however, a uniquely defined dual frame, known as the *canonical* dual frame, given as

$$\tilde{\psi}_i := (\Psi\Psi^T)^{-1}\psi_i, \quad \text{or} \quad \tilde{\Psi} := (\Psi\Psi^T)^{-1}\Psi. \quad (23.79)$$

Another family of frames of special type are the so-called *tight frames*. For tight frames, the two bounds in (23.74) are equal, i.e.,  $A = B$ . Thus, once a tight frame is available, we can normalize each vector in the frame as

$$\psi_i \mapsto \frac{1}{\sqrt{A}}\psi_i,$$

which then results to the so-called *Parseval tight frame*; the condition (23.74) now becomes similar in appearance with our familiar Parseval's theorem for orthonormal bases

$$\sum_{i \in \mathcal{I}} |\langle \psi_i, s \rangle|^2 = \|s\|_2^2. \quad (23.80)$$

Moreover, it can be shown that a Parseval tight frame coincides with its canonical dual frame (that is, it is self dual) and we can write

$$s = \sum_{i \in \mathcal{I}} \langle \psi_i, s \rangle \psi_i,$$

or in matrix form

$$\tilde{\Psi} = \Psi, \quad (23.81)$$

which is similar with what we know for orthonormal bases; however in this case, orthogonality does not hold.

We will conclude this subsection with a simple example of a Parseval (tight) frame, known as the *Mercedes Benz (MB)*,

$$\Psi = \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \end{bmatrix}.$$

One can easily check that all the properties of a Parseval tight frame are fulfilled. If constructing a frame, especially in high dimensional spaces, may sound a bit difficult, the following theorem (due to Naimark, see, e.g., [169]) offers a systematic way for such constructions.

**Theorem 6.** *A set  $\{\psi_i\}_{i \in \mathcal{I}}$  in a Hilbert space  $\mathcal{H}_s$  is a Parseval tight frame, if and only if it can be obtained via orthogonal projection,  $P_{\mathcal{H}_s} : \mathcal{H} \rightarrow \mathcal{H}_s$ , of an orthonormal basis  $\{e_i\}_{i \in \mathcal{I}}$  in a larger Hilbert space  $\mathcal{H}$ , such that  $\mathcal{H}_s \subset \mathcal{H}$ .*

To verify the theorem, check that the MB frame is obtained by orthogonally projecting the three-dimensional orthonormal basis

$$e_1 = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad e_2 = \begin{bmatrix} \sqrt{\frac{2}{3}} \\ -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \end{bmatrix}, \quad e_3 = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix},$$

using the projection matrix

$$P_{\mathcal{H}_s} := \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

Observe that the effect of the projection

$$P_{\mathcal{H}_s}[e_1, e_2, e_3] = \Psi^T,$$

is to set  $e_3$  to the zero vector.

Frames were introduced by Duffin and Schaeffer in their study on nonharmonic Fourier series in 1952 [170] and they remained rather obscured till they were used in the context of wavelet theory, e.g., [171]. The interested reader can obtain the proofs of what has been said in this section from these references. An introductory review with a lot of engineering flavor can be found in [172], where the major references in the field are given.

### 1.23.14.2 Compressed sensing for signals sparse in coherent dictionaries

Our goal in this subsection is to establish conditions that guarantee recovery of a signal vector, which accepts a sparse representation in a redundant and coherent dictionary, using a small number of signal-related measurements. Let the dictionary at hand be a tight frame,  $\Psi$ . Then, our signal vector is written as

$$\mathbf{s} = \Psi\boldsymbol{\theta}, \quad (23.82)$$

where  $\boldsymbol{\theta}$  is assumed to be  $k$ -sparse. Recalling the properties of a tight frame, as they were summarized in the previous subsection, the coefficients in the expansion (23.82) can be written as  $\langle \psi_i, \mathbf{s} \rangle$ , and the respective vector as

$$\boldsymbol{\theta} = \Psi^T \mathbf{s},$$

since a tight frame is self dual. Then, the analysis counterpart of the synthesis formulation in (23.46) can be cast as

$$\begin{aligned} \min_{\mathbf{s}} \quad & \left\| \Psi^T \mathbf{s} \right\|_1, \\ \text{s.t.} \quad & \|\mathbf{y} - X\mathbf{s}\|_2^2 \leq \epsilon. \end{aligned} \quad (23.83)$$

The goal now is to investigate the accuracy of the recovered solution to this convex optimization task. It turns out that similar strong theorems are also valid for this problem as with the case of the synthesis formulation, which was studied earlier.

**Definition 5.** Let  $\Sigma_k$  be the union of all subspaces spanned by all subsets of  $k$  columns of  $\Psi$ . A sensing matrix,  $X$ , obeys the restricted isometry property adapted to  $\Psi$ , ( $\Psi$ -RIP) with  $\delta_k$ , if

$$(1 - \delta_k) \|\mathbf{s}\|_2^2 \leq \|X\mathbf{s}\|_2^2 \leq (1 + \delta_k) \|\mathbf{s}\|_2^2, \quad (23.84)$$

for all  $\mathbf{s} \in \Sigma_k$ .

The union of subspaces,  $\Sigma_k$ , is the image under  $\Psi$  of all  $k$ -sparse vectors. This is the difference with the RIP definition given in Section 1.23.8.2. All the random matrices discussed earlier in this paper can be shown to satisfy this form of RIP, with overwhelming probability, provided the number of measurements,  $N$ , is at least of the order of  $k \ln(p/k)$ . We are now ready to establish the main theorem concerning our  $\ell_1$  minimization task.

**Theorem 7.** Let  $\Psi$  be an arbitrary tight frame and  $X$  a sensing matrix that satisfies the  $\Psi$ -RIP with  $\delta_{2k} \leq 0.08$ , for some positive  $k$ . Then the solution,  $\mathbf{s}_*$ , of the minimization task in (23.83) satisfies the following property

$$\|\mathbf{s} - \mathbf{s}_*\|_2 \leq C_0 k^{-\frac{1}{2}} \left\| \Psi^T \mathbf{s} - (\Psi^T \mathbf{s})_k \right\|_1 + C_1 \sqrt{\epsilon}, \quad (23.85)$$

where  $C_0, C_1$  are constants depending on  $\delta_{2k}$ ,  $(\Psi^T \mathbf{s})_k$  denotes the best  $k$ -sparse approximation of  $\Psi^T \mathbf{s}$ ; i.e., it results by setting all but the  $k$  largest in magnitude components of  $\Psi^T \mathbf{s}$  equal to zero.

The bound in (23.85) is the counterpart of that given in (23.43). In other words, the previous theorem states that if  $\Psi^T \mathbf{s}$  decays rapidly, then  $\mathbf{s}$  can be reconstructed from just a few (compared to the signal length  $l$ ) measurements. The theorem was first given in [173] and it is the first time that such a theorem provides results for the sparse analysis model formulation in a general context.

### 1.23.14.3 Cosparsity

In [174], the task of sparse analysis modeling was approached via an alternative route, employing the tools which were developed in [137, 175] for treating general union-of-subspaces models. This complementary point of view will also unveil different aspects of the problem by contributing to its deeper understanding. We have done it before, where the notions of spark, coherence and RIP were all mobilized to shed light from different corners to the sparse synthesis modeling task.

In the sparse synthesis formulation, one searches for a solution in a union of subspaces, which are formed by all possible combinations of  $k$  columns of the dictionary,  $\Psi$ . Our signal vector lies in one of these subspaces; the one which is spanned by the columns of  $\Psi$  whose indices lie in the support set (Section 1.23.11.1). In the sparse analysis approach things get different. The kick off point is the sparsity of the transform  $S := \Phi^T s$ , where  $\Phi$  defines the transformation matrix or analysis operator. Since  $S$  is assumed to be sparse, there exists an index set  $\mathcal{I}$  such that  $\forall i \in \mathcal{I}, S_i = 0$ . In other words,  $\forall i \in \mathcal{I}, \langle \phi_i^T s \rangle := \langle \phi_i, s \rangle = 0$ , where  $\phi_i$  stands for the  $i$ th column of  $\Phi$ . Hence, the subspace in which  $s$  lives is the orthogonal complement of the subspace formed by those columns of  $\Phi$ , which correspond to a zero in the transform vector  $S$ . Assume, now, that  $\text{card}(\mathcal{I}) = C_o$ . The signal,  $s$ , can be identified by searching on the *orthogonal complements* of the subspaces formed by all possible combinations of  $C_o$  columns of  $\Phi$ , i.e.,

$$\langle \phi_i, s \rangle = 0, \quad \forall i \in \mathcal{I}.$$

The difference between the synthesis and analysis problems is illustrated in Figure 23.24. To facilitate the theoretical treatment of this new setting, the notion of *cosparsity* was introduced in [174].

**Definition 6.** The cosparsity of a signal  $s \in \mathbb{R}^l$  with respect to a  $p \times l$  matrix  $\Phi^T$  is defined as

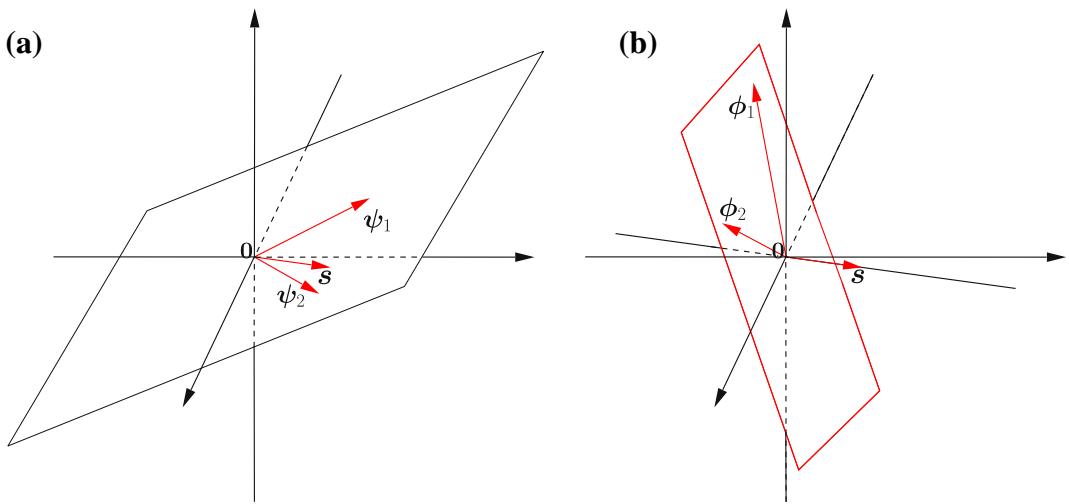
$$C_o := p - \left\| \Phi^T s \right\|_0. \quad (23.86)$$

In words, the cosparsity is the number of zeros in the obtained transform vector  $S = \Phi^T s$ ; in contrast, the sparsity measures the number of the nonzero elements of the respective sparse vector. If one assumes that  $\Phi$  has “full spark,”<sup>7</sup> i.e.,  $l + 1$ , then any  $l$  of the columns of  $\Phi$ , and thus any  $l$  rows of  $\Phi^T$  are guaranteed to be independent. This indicates that for such matrices, the maximum value that cosparsity can take is equal to  $C_o = l - 1$ . Otherwise, the existence of  $l$  zeros will necessarily correspond to a zero signal vector. Higher cosparsity levels are possible, by relaxing the full spark requirement.

Let now the cosparsity of our signal with respect to a matrix  $\Phi^T$  be  $C_o$ . Then, in order to dig out the signal from the subspace in which is hidden, one must form all possible combinations of  $C_o$  columns of  $\Phi$  and search in their orthogonal complements. In case that  $\Phi$  is full rank, we have seen previously that  $C_o < l$ , and hence any set of  $C_o$  columns of  $\Phi$  are linearly independent. In other words, the dimension of the span of those columns is  $C_o$ . As a result, the dimensionality of the orthogonal complement, into which we search for  $s$ , is  $l - C_o$ .

We have by now accumulated enough information to elaborate a bit more on the statement made before, concerning the different nature of the synthesis and analysis tasks. Let us consider a synthesis task using an  $l \times p$  dictionary and let  $k$  be the sparsity level in the corresponding expansion of a signal in terms of this dictionary. The dimensionality of the subspaces in which the solution is sought is  $k$

<sup>7</sup>Recall by Definition 2 that  $\text{spark}(\Phi)$  is defined for an  $l \times p$  matrix  $\Phi$  with  $p \geq l$  and of full rank.

**FIGURE 23.24**

Searching for a sparse vector  $s$ . (a) In the synthesis model, the sparse vector lies in subspaces formed by combinations of  $k$  (in this case  $k = 2$ ) columns of the dictionary  $\Psi$ . (b) In the analysis model, the sparse vector lies in the orthogonal compliment of the subspace formed by  $C_o$  (in this case  $C_o = 2$ ) columns of the transformation matrix  $\Phi$ .

( $k$  is assumed to be less than the spark of the respective matrix). Let us keep the same dimensionality for the subspaces in which we are going to search for a solution in an analysis task. Hence, in this case  $C_o = l - k$  (assuming a full spark matrix). Also, for the sake of comparison, assume that the analysis matrix is  $p \times l$ . Solving the synthesis task, one has to search  $\binom{p}{k}$  subspaces, while solving the analysis

task one has to search for  $\binom{p}{C_o = l - k}$  subspaces. These are two different numbers; assuming that  $k \ll l$  and also that  $l < p/2$ , which are natural assumptions for overcomplete dictionaries, then the latter of the two numbers is much larger than the former one (use your computer to play with some typical values). In other words, there are much more analysis than synthesis low-dimensional subspaces to be searched for. The large number of low-dimensional subspaces makes the algorithmic recovery of a solution from the analysis model a tougher task, [174]. However, it might reveal a much stronger descriptive power of the analysis model compared to the synthesis one.

Another interesting aspect that highlights the difference between the two approaches is the following. Assume that the synthesis and analysis matrices are related as  $\Phi = \Psi$ , as it was the case for tight frames. Under this assumption,  $\Phi^T s$  provides a set of coefficients for the synthesis expansion in terms of the atoms of  $\Phi = \Psi$ . Moreover, if  $\|\Phi^T s\|_0 = k$ , then the  $\Phi^T s$  is a possible  $k$ -sparse solution for the synthesis model. However, there is no guarantee that this is the sparsest one.

It is now the time to investigate whether conditions that guarantee uniqueness of the solution for the sparse analysis formulation can be derived. The answer is affirmative and it has been established in [174], for the case of exact measurements.

**Lemma 5.** Let  $\Phi$  be a transformation matrix of full spark. Then, for almost all  $N \times l$  sensing matrices and for  $N > 2(l - C_o)$ , the equation

$$\mathbf{y} = \mathbf{X}\mathbf{s},$$

has at most one solution with cosparcity at least  $C_o$ .

The above lemma guarantees the uniqueness of the solution, if one exists, of the following optimization

$$\begin{aligned} \min_{\mathbf{s}} \quad & \|\Phi^T \mathbf{s}\|_0 \\ \text{s.t. } & \mathbf{y} = \mathbf{X}\mathbf{s}. \end{aligned} \tag{23.87}$$

However, solving the previous  $\ell_0$  minimization task is a difficult one and we know that its synthesis counterpart has been shown to be NP-hard, in general. Its relaxed convex relative is the  $\ell_1$  minimization

$$\begin{aligned} \min_{\mathbf{s}} \quad & \|\Phi^T \mathbf{s}\|_1 \\ \text{s.t. } & \mathbf{y} = \mathbf{X}\mathbf{s}. \end{aligned} \tag{23.88}$$

In [174], conditions are derived that guarantee the equivalence of the  $\ell_0$  and  $\ell_1$  tasks, in (23.87) and (23.88), respectively; this is done in a way similar to that for the sparse synthesis modeling. Also, in [174], a greedy algorithm inspired by the Orthogonal Matching Pursuit, discussed in Section 1.23.11.1, has been derived. Other algorithms that solve the  $\ell_1$  optimization in the analysis modeling framework can be found in, e.g., [176–178]. NESTA can also be used for the analysis formulation.

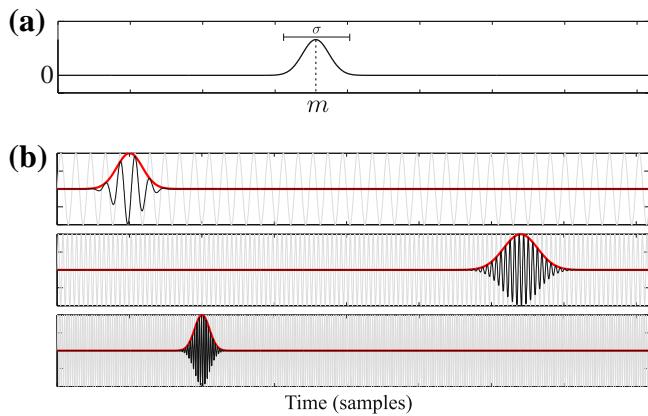
## 1.23.15 A case study: time-frequency analysis

The goal of this section is to demonstrate how all the previously stated theoretical findings can be exploited in the context of a real application. Sparse modeling has been applied to almost everything. So, picking up a typical application would not be easy. We preferred to focus on a less “publicized” application; that of analyzing echolocation signals emitted by bats. However, the analysis will take place within the framework of time-frequency representation, which is one of the research areas that significantly inspired the evolution of compressed sensing theory. Time-Frequency analysis of signals has been the field of intense research for a number of decades, and it is one of the most powerful signal processing tools. Typical applications include speech processing, sonar sounding, communications, biological signals, EEG processing, to name but a few, see, e.g., [179, 180].

### 1.23.15.1 Gabor transform and frames

It is not our intention to present the theory behind the Gabor transform. Our goal is to outline some basic related notions and use it as a vehicle for the less familiar reader so that (a) to better understand how redundant dictionaries are used and (b) get more acquainted with their potential performance benefits.

The Gabor transform was introduced in the middle 1940s by Dennis Gabor (1900–1979), who was a Hungarian-British engineer. His most notable scientific achievement was the invention of holography, for which he won the Nobel prize for Physics in 1971. The discrete version of the Gabor transform can

**FIGURE 23.25**

(a) The Gaussian window with spreading factor  $\sigma$  centered at time instant  $m$ . (b) Pulses obtained by windowing three different sinusoids with Gaussian windows of different spread and applied at different time instants.

be seen as a special case of the Short Time Fourier Transform (STFT), e.g., [180, 181]. In the standard DFT transform, the full length of a time sequence, comprising  $l$  samples, is used all in “one-go” in order to compute the corresponding frequency content. However, the latter can be time varying, so the DFT will provide an average information, which cannot be of much use. The Gabor transform (and the STFT in general) introduces time localization via the use of a window function, which slides along the signal segment in time, and at each time instant focusses on a different part of the signal; this is a way that allows one to follow the slow time variations, which take place in the frequency domain. The time localization in the context of the Gabor transform is achieved via a Gaussian window function, i.e.,

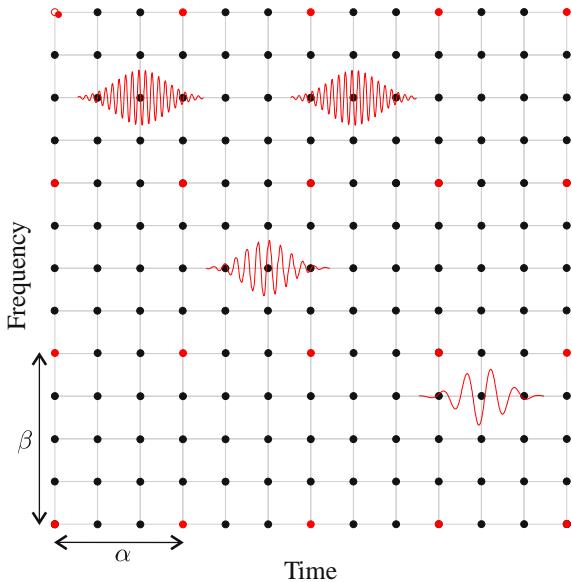
$$g(n) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{n^2}{2\sigma^2}\right). \quad (23.89)$$

Figure 23.25a shows the Gaussian window,  $g(n - m)$ , centered at time instant  $m$ . The choice of the window spreading factor,  $\sigma$ , will be discussed later on.

Let us now construct the atoms of the Gabor dictionary. Recall that in the case of the signal representation in terms of the DFT in (23.71), each frequency is represented only once, by the corresponding sampled sinusoid, (23.72). In the Gabor transform, each frequency appears  $l$  times; the corresponding sampled sinusoid is multiplied by the Gaussian window sequence, each time shifted by one sample. Thus, at the  $i$ th frequency bin, we have  $l$  atoms,  $g^{(m,i)}$ ,  $m = 0, 1, \dots, l - 1$ , with elements given by

$$g^{(m,i)}(n) = g(n - m)\psi_i(n), \quad n, m, i = 0, 1, \dots, l - 1, \quad (23.90)$$

where  $\psi_i(n)$  is the  $n$ th element of the vector  $\psi_i$  in (23.72). This results to an overcomplete dictionary comprising  $l^2$  atoms in the  $l$ -dimensional space. Figure 23.25b illustrates the effect of multiplying different sinusoids with Gaussian pulses of different spread and at different time delays. Figure 23.26

**FIGURE 23.26**

Each atom of the Gabor dictionary corresponds to a node in the time-frequency grid. That is, it is a sampled windowed sinusoid whose frequency and location in time are given by the coordinates of the respective node. In practice, this grid may be subsampled by factors  $\alpha$  and  $\beta$  for the two axes respectively, in order to reduce the number of the involved atoms.

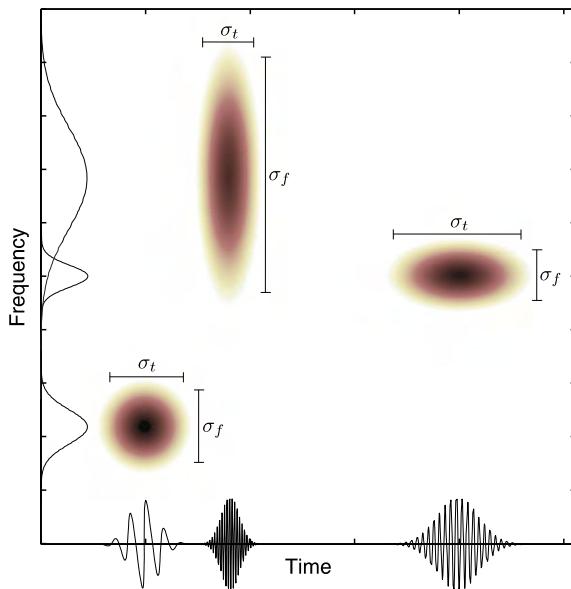
is a graphical interpretation of the atoms involved in the Gabor dictionary. Each node,  $(m, i)$ , in this time-frequency plot, corresponds to an atom of frequency equal to  $\frac{2\pi}{lT} i$  and delay equal to  $m$ .

Note that the windowing of a signal of finite duration inevitably introduces boundary effects, especially when the delay  $m$  gets close to the time segment edges, 0 and  $l - 1$ . A solution to it, that facilitates the theoretical analysis, is to use a modulo  $l$  arithmetic to wrap around at the edge points (this is equivalent with extending the signal periodically), see, e.g., [182].

Once the atoms have been defined, they can be stacked one next to the other to form the columns of the  $l \times l^2$  Gabor dictionary,  $G$ . It can be shown that the Gabor dictionary is a tight frame, [183].

### 1.23.15.2 Time-frequency resolution

By the definition of the Gabor dictionary, it is readily understood that the choice of the window spread, as measured by  $\sigma$ , must be a critical factor, since it controls the localization in time. As it is known from our Fourier transform basics, when the pulse becomes short, in order to increase the time resolution, its corresponding frequency content spreads out, and vice versa. From Heisenberg's principle, we know that we can never achieve high time and frequency resolution, simultaneously; one is gained at the expense of the other. It is here where the Gaussian shape in the Gabor transform is justified. It can be shown that the Gaussian window gives the optimal trade-off between time and frequency resolution, [180, 181].



**FIGURE 23.27**

The shorter the width of the pulsed (windowed) sinusoid is in time the wider the spread of its frequency content around the frequency of the sinusoid. The Gaussian-like curves along the frequency axis indicate the energy spread in frequency of the respective pulses. The values of  $\sigma_t$  and  $\sigma_f$  indicate the spread in time and frequency, respectively.

The time-frequency resolution trade-off is demonstrated in Figure 23.27, where three sinusoids are shown windowed with different pulse durations. The diagram shows the corresponding spread in the time-frequency plot. The value of  $\sigma_t$  indicates the time spread and  $\sigma_f$  the spread of the respective frequency content around the basic frequency of each sinusoid.

### 1.23.15.3 Gabor frames

In practice,  $l^2$  can take large values and it is desirable to see whether one can reduce the number of the involved atoms, without sacrificing the frame-related properties. This can be achieved by an appropriate subsampling, as this is illustrated in Figure 23.26. We only keep the atoms that correspond to the red nodes. That is, we subsample by keeping every  $\alpha$  nodes in time and every  $\beta$  nodes in frequency in order to form the dictionary, i.e.,

$$G_{(\alpha,\beta)} = \{\mathbf{g}^{(m\alpha,i\beta)}\}, \quad m = 0, 1, \dots, \frac{l}{\alpha} - 1, \quad i = 0, 1, \dots, \frac{l}{\beta} - 1,$$

where  $\alpha$  and  $\beta$  are divisors of  $l$ . Then, it can be shown, e.g., ([180]), that if  $\alpha\beta < l$  the resulting dictionary retains its frame properties. Once  $G_{(\alpha,\beta)}$  is obtained, the canonical dual frame is readily available via (23.79) (adjusted for complex data), from which the corresponding set of expansion coefficients,  $\theta$ , results.

### 1.23.15.4 Time-frequency analysis of echolocation signals emitted by bats

Bats are using echolocation for navigation (flying around at night), for prey detection (small insects) and for prey approaching and catching; each bat adaptively changes the shape and frequency content of its calls in order to better serve the previous tasks. Echolocation is used in a similar way for sonars. Bats emit calls as they fly, and “listen” to the returning echoes in order to build up a sonic map of their surroundings. In this way, bats can infer on the distance, the size of obstacles as well as of other flying creatures/insects. Moreover, all bats emit special types of calls, called social calls, which are used for socializing, flirting, etc. The fundamental characteristics of the echolocation calls, as for example, the frequency range and the average time duration, differ from species to species since, thanks to evolution, bats have adapted their calls in order to get best suited to the environment in which a species operates.

Time-Frequency analysis of echolocation calls provides information about the species (species identification) as well as of the specific task and behavior of the bats in certain environments. Moreover, the bat-biosonar system is studied in order humans to learn more about nature and be inspired for subsequent advances in applications such as sonar navigation systems, radars, medical ultrasonic devices, etc.

Figure 23.28a shows a case of a recorded echolocation signal from bats. Zooming at two different parts of the signal, we can observe that the frequency is changing with time. In Figure 23.28b, the DFT of the signal is shown, but there is no much information that can be drawn from it, except that the signal is compressible in the frequency domain; most of the activity takes place within a short range of frequencies.

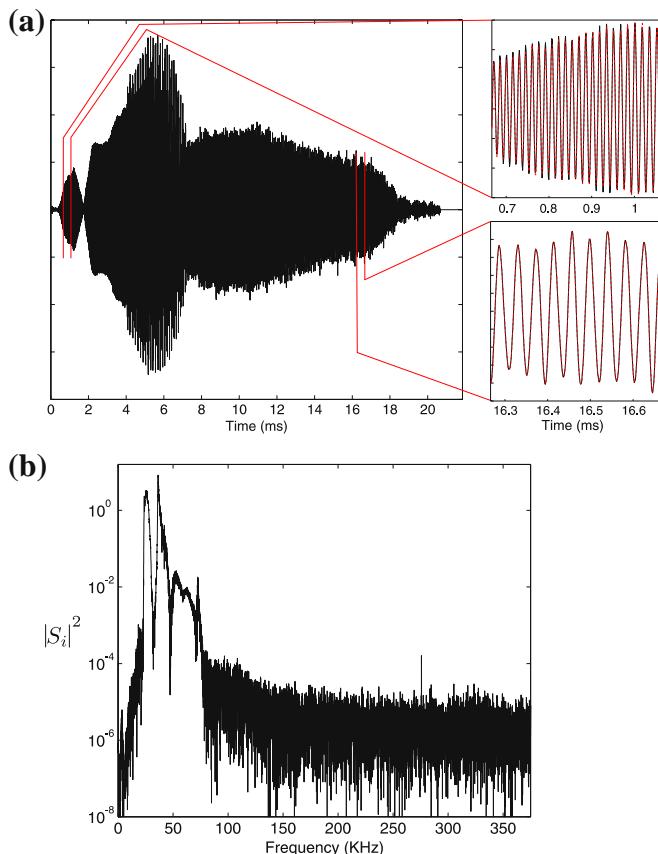
Our echolocation signal was a recording of total length  $T = 21.845$  ms, [184]. Samples were taken at the sampling frequency  $f_s = 750$  kHz, which results in a total of  $l = 16384$  samples. Although the signal itself is not sparse in the time domain, we will take advantage of the fact that it is sparse in a transformed domain. We will assume that the signal is sparse in its expansion in terms of the Gabor dictionary.

Our goal in this example is to demonstrate that one does not really need all 16384 samples to perform time-frequency analysis; all the processing can be carried out by using a reduced number of measurements, by exploiting the theory of compressed sensing. To form the measurements vector,  $\mathbf{y}$ , the number of measurements was chosen to be  $N = 2048$ . This amounts to a reduction of eight times with respect to the number of available samples. The measurements vector was formed as

$$\mathbf{y} = \mathbf{X}\mathbf{s},$$

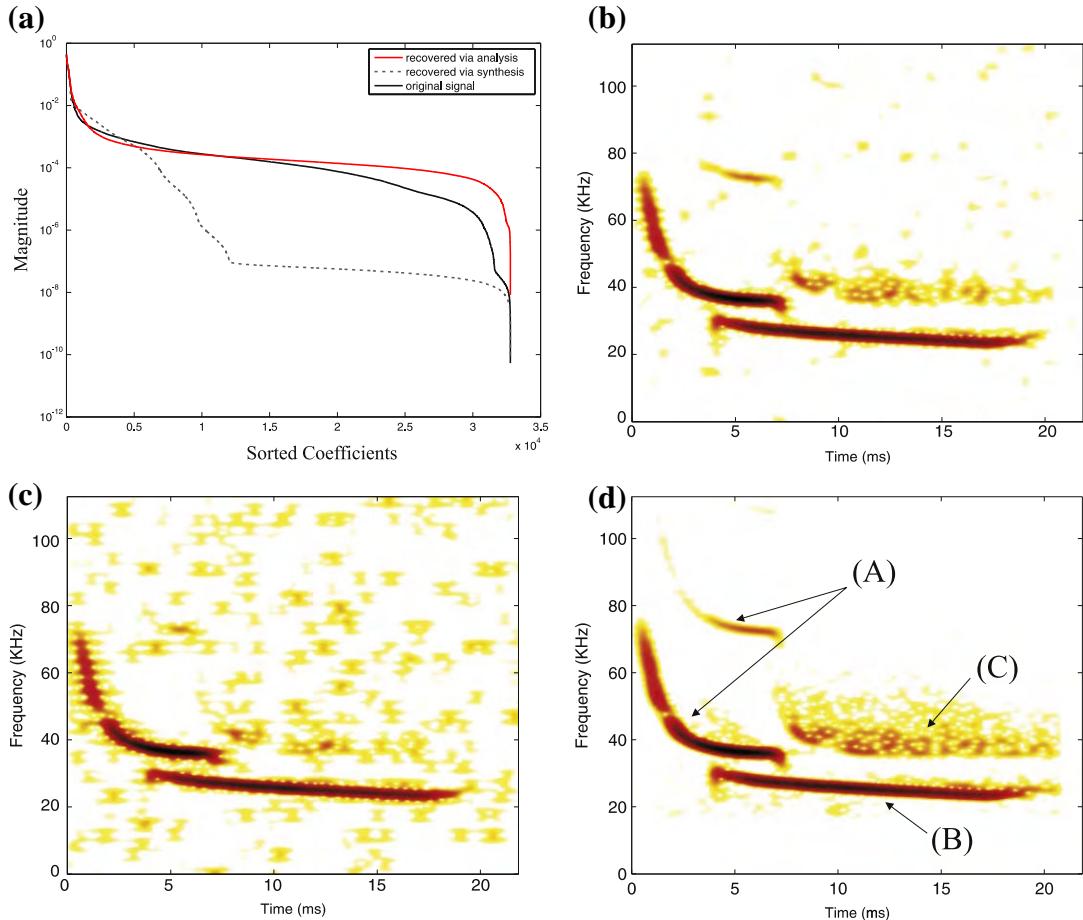
where  $\mathbf{X}$  is a  $N \times l$  sensing matrix comprising  $\pm 1$  generated in a random way. This means that once we obtain  $\mathbf{y}$ , one does not need to store the original samples any more, leading to a saving in memory. Ideally, one could have obtained the reduced number of measurements by sampling directly the analog signal at sub-Nyquist rates, as it has already been discussed at the end of Section 1.23.10. Another goal is to use both the analysis and synthesis models and demonstrate their difference.

Three different spectrograms were computed. Two of them, shown in Figures 23.29b and 23.29c, correspond to the reconstructed signals obtained by the analysis, (23.88), and the synthesis, (23.44), formulations, respectively. In both cases, the NESTA algorithm was used and the  $G_{(128,64)}$  frame was employed. Note that the latter dictionary is redundant by a factor of 2. The spectrograms are

**FIGURE 23.28**

(a) The recorded echolocation signal. The frequency of the signal is time varying and this is indicated by focussing on two different parts of the signal. (b) Plot of the energy of the DFT transform coefficients,  $S_i$ . Observe that most of the frequency activity takes place within a short frequency range.

the result of plotting the time-frequency grid and coloring each node  $(t, i)$  according to the energy  $|\theta|^2$  of the coefficient associated with the respective atom in the Gabor dictionary. The full Gabor transform applied on the reconstructed signals to obtain the spectrograms, in order to get a better coverage of the time-frequency grid. The scale is logarithmic and the darker areas correspond to larger values. The spectrogram of the original signal obtained via the full Gabor transform is shown in Figure 23.29d. It is evident, that the analysis model resulted in a more clear spectrogram, which resembles the original one better. When the frame  $G_{(64,32)}$  is employed, which is a highly redundant Gabor dictionary comprising  $8l$  atoms, then the analysis model results in a recovered signal whose spectrogram is visually indistinguishable from the original one in Figure 23.29d.

**FIGURE 23.29**

(a) Plot of the magnitude of the coefficients, sorted in decreasing order, in the expansion in terms of the  $G_{(128,64)}$  Gabor frame. The results correspond to the analysis and synthesis model formulations. The third curve corresponds to the case of analyzing the original vector signal directly, by projecting it on the dual frame.  
 (b) The spectrogram from the analysis. (c) The spectrogram from the synthesis formulations, respectively.  
 (d) The spectrogram corresponding to  $G_{(64,32)}$  frame using the analysis formulation. For all cases, the number of measurements used was one eighth of the total number of signal samples. A, B, and C indicate different parts of the signal, as explained in the text.

Figure 23.29a is the plot of the magnitude of the corresponding Gabor transform coefficients, sorted in decreasing values. The synthesis model provides a sparser representation, in the sense that the coefficients decrease much faster. The third curve is the one that results if we multiply the dual frame matrix  $\tilde{G}_{(128,64)}$  directly with the vector of the original signal samples and it is shown for comparison reasons.

To conclude, the curious reader may wonder what do these curves in Figure 23.29d mean after all. The call denoted by (A) belongs to a Pipistrellus pipistrellus (!) and the call denoted by (B) is either a social call or belongs to a different species. The (C) is the return echo from the signal (A). The large spread in time of (C) indicates a highly reflective environment, [184].

## 1.23.16 From sparse vectors to low rank matrices: a highlight

In this section, we move beyond sparse vectors and our goal is to investigate if and how notions related to sparsity can be generalized to matrices. We will see that such a generalization builds upon linear algebra tools and notions related to SVD decomposition, low rank approximation and dimensionality reduction. Our goal is to simply highlight the basic concepts and definitions without delving into a deeper treatment. Our aim is to make the reader alert of the problems and their potential for applications.

### 1.23.16.1 Matrix completion

Consider a signal vector  $s \in \mathcal{R}^l$ , where only  $N$  of its components are observed and the rest are unknown. This is equivalent with sensing  $s$  via a sensing matrix  $X$  having its  $N$  rows picked uniformly at random from the standard (canonical) basis  $\Phi = I$ , where  $I$  is the  $l \times l$  identity matrix. The question which is now posed is whether it is possible to recover the missing components of  $s$  based on these  $N$  components. From the theory presented, so far, we know that one can recover all the components of  $s$ , provided that  $s$  is sparse in some basis or dictionary,  $\Psi$ , which exhibits low mutual coherence with  $\Phi = I$ , and  $N$  is large enough, as it has been pointed out in Section 1.23.10.

Inspired by the theoretical advances in Compressed Sensing, a question similar in flavor and with a prominent impact regarding practical applications was posed in [185]. Given a  $l_1 \times l_2$  matrix  $M$ , assume that only  $N << l_1 l_2$  among its entries are known. The question now is whether one is able to recover the exact full matrix. This problem is widely known as *matrix completion* [185]. The answer, although it might come as a surprise, is “yes” with high probability, provided that (a) the matrix is *well structured*, (b) it has a *low rank*,  $r << l$ , where  $l = \min(l_1, l_2)$ , and (c) that  $N$  is large enough. Intuitively, this is plausible because a low rank matrix is fully described in terms of a number of parameters (degrees of freedom), which is much smaller than its total number of entries. These parameters are revealed via its Singular Value Decomposition (SVD)

$$M = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T = U \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{bmatrix} V^T, \quad (23.91)$$

where  $r$  is the rank of the matrix,  $\mathbf{u}_i \in \mathcal{R}^{l_1}$  and  $\mathbf{v}_i \in \mathcal{R}^{l_2}$ ,  $i = 1, 2, \dots, r$ , are the left and right orthonormal singular vectors, spanning the column and row spaces of  $M$  respectively,  $\sigma_i$ ,  $i = 1, 2, \dots, r$ , are the corresponding singular values and  $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r]$ ,  $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r]$ .

Let  $\sigma_M$  denote the vector containing all the singular values of  $M$ , i.e.,  $\sigma_M = [\sigma_1, \sigma_2, \dots, \sigma_l]^T$ , then  $\text{rank}(M) := \|\sigma_M\|_0$ . Counting the parameters associated with the singular values and vectors in (23.91)

turns out that the number of degrees of freedom of a rank  $r$  matrix is equal to  $d_M = r(l_1 + l_2) - r^2$  [186]. When  $r$  is small,  $d_M$  is much smaller than  $l$ .

Let us denote with  $\Omega$  the set of  $N$  pairs of indexes,  $(i, j)$ ,  $i = 1, 2, \dots, l_1$ ,  $j = 1, 2, \dots, l_2$ , of the locations of the known entries of  $M$ , which have been sampled uniformly at random. Adopting the main reasoning followed so far, one would attempt to recover  $M$  based on the following rank minimization problem

$$\begin{aligned} & \min_{\hat{M} \in \mathcal{R}^{l_1 \times l_2}} \|\sigma_{\hat{M}}\|_0 \\ & \text{s.t. } \hat{M}_{i,j} = M_{i,j}, \quad (i, j) \in \Omega. \end{aligned} \quad (23.92)$$

It turns out that, assuming that there exist a unique low-rank matrix having the specific known entries, then (23.92) leads to the exact solution [185]. However, compared to the case of sparse vectors, in the matrix completion problem the uniqueness issue gets much more involved. The following issues play a crucial part concerning the uniqueness of the task in (23.92).

1. If the number of known entries is lower than the degrees of freedom, i.e.,  $N < d_M$ , then there is no way to recover the missing entries whatsoever, since there is an infinite number of low rank matrices consistent with the  $N$  observed entries.
2. Even if  $N \geq d_M$ , uniqueness is still not guaranteed. It is required that the  $N$  elements with indices in  $\Omega$  are such that at least one entry per column and one entry per row is observed. Otherwise, even a rank-1 matrix, i.e.,  $M = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$ , is not possible to be recovered. This becomes clear with a simple example. Assume that  $M$  is a rank-1 matrix and that no entry in the first column as well as in the last row is observed. Then, since for this case  $M(i, j) = \sigma_1 u_{1i} v_{1j}$ , it is clear that no information concerning the first component of  $\mathbf{v}_1$  as well as the last component of  $\mathbf{u}_1$  is available; hence these singular vector components are impossible to be recovered, regardless which method is used. As a consequence, the matrix can not be completed. On the other hand, if the elements of  $\Omega$  are picked at random and  $N$  is large enough, one can only hope that  $\Omega$  is such that to comply with the previous requirement; i.e., at least one entry per row and column is observed, with high probability. It turns out that this problem resembles the famous in probability theory theorem known as the *coupon collector's* problem. According to this, at least  $N = C_0 l \ln l$  entries are needed, where  $C_0$  is a constant [187]. This is the information theoretic limit for exact matrix completion [186] of any low-rank matrix.
3. Even if points (1) and (2) before are fulfilled, still uniqueness is not guaranteed. In fact, not every low rank matrix is liable to exact completion, regardless of the number and the positions of the observed entries. We will demonstrate that with the aid of an example. Let one of the singular vectors be sparse. Assume, without loss of generality, that the third left singular vector,  $\mathbf{u}_3$ , is sparse with sparsity level  $k = 1$  and also that its nonzero component is the first one, i.e.,  $u_{31} \neq 0$ . The rest of  $\mathbf{u}_i$  and all  $\mathbf{v}_i$  are assumed to be dense. Let us return to the SVD for a while, and specifically to the leftmost formula given in (23.91). Observe that the matrix  $M$  is written as the sum of  $r, l_1 \times l_2$  matrices  $\sigma_i \mathbf{u}_i \mathbf{v}_i^T, i = 1, \dots, r$ . Thus, in this specific case where  $\mathbf{u}_3$  is  $k = 1$  sparse, the matrix  $\sigma_3 \mathbf{u}_3 \mathbf{v}_3^T$  has zeros everywhere except from its first row. In other words, the information that  $\sigma_3 \mathbf{u}_3 \mathbf{v}_3^T$  brings to the formation of  $M$  is concentrated to its first row only. This argument can also be viewed from another perspective; the entries of  $M$  obtained from any row but the first one, do not provide

any useful information with respect to the values of the free parameters  $\sigma_3, \mathbf{u}_3, \mathbf{v}_3$ . As a result, in this case, unless if one incorporates extra information about the sparse nature of the singular vector, the entries from the first row that are missed are not recoverable, since the number of parameters concerning this row is larger than the available number of data.

Intuitively, when a matrix has dense singular vectors is better rendered for exact completion, since each one among the observed entries carries information associated with all the  $d_M$  parameters that fully describe it. To this end, a number of conditions, which evaluate the suitability of the singular vectors, have been established. The simplest one is given next [185]:

$$\|\mathbf{u}_i\|_\infty \leq \sqrt{\frac{\mu_B}{l_1}}, \quad \|\mathbf{v}_i\|_\infty \leq \sqrt{\frac{\mu_B}{l_2}}, \quad i = 1, \dots, r, \quad (23.93)$$

where  $\mu_B$  is a bound parameter. In fact,  $\mu_B$  is a measure of the coherence<sup>8</sup> of matrix  $U$  (and similarly of  $V$ ), (vis-à-vis the standard basis), defined as follows:

$$\mu(U) := \frac{l_1}{r} \max_{1 \leq i \leq l_1} \|\mathbf{P}_U \mathbf{e}_i\|^2, \quad (23.94)$$

where  $\mathbf{P}_U$  defines the orthogonal projection to subspace  $U$  and  $\mathbf{e}_i$  is the  $i$ th vector of the canonical basis. It is easy to show that  $\|\mathbf{P}_U \mathbf{e}_i\|^2 = \|U^T \mathbf{e}_i\|^2$ . In essence, coherence is an index quantifying the extent to which the singular vectors are correlated with the standard basis,  $\mathbf{e}_i, i = 1, 2, \dots, l$ . The smaller the  $\mu_B$  is the less “spiky” the singular vectors are likely to be, and the corresponding matrix is better suited for exact completion. Indeed, assuming for simplicity a square matrix  $M$ , i.e.,  $l_1 = l_2 = l$ , then if *any one* among the singular vectors is sparse having a single nonzero component only, then, taking into account that  $\mathbf{u}_i^T \mathbf{u}_i = \mathbf{v}_i^T \mathbf{v}_i = 1$ , this value will have magnitude equal to one and the bound parameter will take its largest value possible, i.e.,  $\mu_B = l$ . On the other hand, the smaller value that  $\mu_B$  can get is 1, something that occurs when the components of *all* the singular vectors assume the same value (in magnitude). Note that in this case, due to the normalization, this common component value has magnitude  $\frac{1}{l}$ . Tighter bounds to the matrix coherence result via the more elaborate incoherence property [185, 188] and the strong incoherence property [186]. In all cases, the larger the bound parameter is the larger the number of known entries, which is required in order to guarantee uniqueness, becomes.

In Section 1.23.16.3, the aspects of uniqueness will be discussed in the context of a real life application.

The problem described in (23.92) is of limited practical interest since it is an NP-hard task. Thus, following the Compressed Sensing paradigm, it is replaced by a *convexly* relaxed counterpart of it, i.e.,

$$\begin{aligned} & \min_{\hat{M} \in \mathcal{R}^{l_1 \times l_2}} \|\boldsymbol{\sigma}_{\hat{M}}\|_1 \\ & \text{s.t. } \hat{M}_{i,j} = M_{i,j}, \quad (i, j) \in \Omega, \end{aligned} \quad (23.95)$$

---

<sup>8</sup>This is a quantity different than the mutual-coherence already discussed in Section 1.23.7.1.

where  $\|\sigma_{\hat{M}}\|_1$ , i.e., the sum of the singular values, is referred to as *nuclear norm* of the matrix  $\hat{M}$ , often denoted as  $\|\hat{M}\|_*$ . The nuclear norm minimization was proposed in [189] as a convex approximation of rank minimization, which can be cast as a semidefinite program.

**Theorem 8 ([186], Corollary 1.5).** *Let  $M$  be a  $l_1 \times l_2$  matrix of rank  $r$ , which is a constant much smaller than  $l = \max(l_1, l_2)$ , obeying (23.93). Suppose that we observe  $N$  entries of  $M$  with locations sampled uniformly at random. Then there is a positive constant  $C$  such that if*

$$N \geq C\mu_B^4 l \ln^2 l, \quad (23.96)$$

*then  $\hat{M}$  is the unique solution to (23.95) with probability at least  $1 - l^{-3}$ .*

There might be an ambiguity on how small the rank should be in order for the corresponding matrix to be characterized as “low rank.” More rigorously, a matrix is said to be of low rank if  $r = \mathcal{O}(1)$ , which means that  $r$  is a constant with no dependence (not even logarithmic), on  $l$ . Matrix completion is also possible for more general rank cases where instead of the mild coherence property of (23.93), the incoherence and the strong incoherence properties [185, 186, 188, 190] are mobilized in order to get similar theoretical guarantees. The detailed exposition of these alternatives is out of the scope of this paper. In fact, Theorem 8 embodies the essence of the matrix completion task: with high probability, nuclear-norm minimization recovers all the entries of a low rank matrix  $M$  with no error. More importantly, the number of entries  $N$ , which the convexly relaxed problem requires, is only by a logarithmic factor larger than the information theoretic limit; recall that the latter is equal to  $C_0 l \ln l$ . Moreover, similarly to Compressed Sensing, robust matrix completion in the present of noise is also possible as long as the request  $\hat{M}_{i,j} = M_{i,j}$  in (23.92) and (23.95) is replaced by  $\|\hat{M}_{i,j} - M_{i,j}\|_2 \leq \epsilon$  [191]. Furthermore, the notion of matrix completion has also been extended to tensors [192, 193].

### 1.23.16.2 Robust PCA

The developments on matrix completion theory led, more recently, to the formulation and solution of another problem of high significance. To this end, the notation  $\|M\|_1$ , i.e., the  $\ell_1$  norm of a matrix, is introduced and it is defined as the sum of the absolute values of its entries, i.e.,  $\|M\|_1 = \sum_{i=1}^{l_1} \sum_{j=1}^{l_2} |M_{i,j}|$ . In other words, it acts on the matrix as if this were a long vector. Assume that  $M$  is expressed as the sum of a low rank matrix,  $L$ , and a sparse matrix,  $S$ , i.e.,  $M = L + S$ . The following convex minimization problem [194–196], usually referred to as *principal component pursuit* (PCP),

$$\begin{aligned} \min_{\hat{L} \in \mathcal{R}^{l_1 \times l_2}, \hat{S} \in \mathcal{R}^{l_1 \times l_2}} \quad & \|\sigma_{\hat{L}}\|_1 + \lambda \|\hat{S}\|_1 \\ \text{s.t. } \hat{L} + \hat{S} = M, \end{aligned} \quad (23.97)$$

is shown to recover both  $L$  and  $S$  according to the following theorem [194]:

**Theorem 9.** *The PCP recovers both  $L$  and  $S$  with probability at least  $1 - cl_1^{-10}$ , where  $c$  is a constant, provided that:*

1. the support set  $\Omega$  of  $S$  is uniformly distributed among all sets of cardinality  $N$ ,
2. the number,  $k$ , of nonzero entries of  $S$  is relatively small, [197], i.e.,  $k \leq \rho l_1 l_2$ , where  $\rho$  is a sufficiently small positive constant,
3.  $L$  obeys the incoherence property, with bound  $\mu_c$ , which is simultaneously larger than  $\mu(U)$ ,  $\mu(V)$  and  $\frac{l_1 l_2}{r} \|U V^T\|_\infty^2$
4. the regularization parameter,  $\lambda$ , is constant with value  $\lambda = \frac{1}{\sqrt{l_2}}$ ,
5.  $\text{rank}(L) \leq C \frac{l_2}{\mu_c \ln^2 l_1}$ , with  $C$  being a constant.

In other words, based on *all* the entries of a matrix  $M$ , which is known that is the sum of two unknown matrices  $L$  and  $S$ , with the first one being of low rank matrix and the second being sparse, then PCP recovers exactly, with probability almost 1, both  $L$  and  $S$ , irrespective of how large the magnitude of the entries of  $S$  are, provided that *both  $r$  and  $k$  are sufficiently small*.

The applicability of the previous task is very broad. For example, PCP can be employed in order to find a low rank approximation of  $M$ . It is well known that the task of low rank approximation is closely related to the dimensionality reduction task, where the columns of  $M$  are expressed in terms of the  $r$  (principal components) columns of  $U$ , e.g., Chapter 6, [1]. However, in contrast to the standard SVD or PCA approach, PCP is robust and insensitive to the presence of outliers, since these are naturally modeled, via the presence of  $S$ . For this reason, the above task is widely known as *robust PCA via nuclear norm minimization*. (More classical PCA techniques are known to be sensitive to outliers and a number of alternative approaches have in the past been proposed towards its robustification, e.g., [198–202]).

When PCP serves as a robust PCA approach, the matrix of interest is  $L$  and  $S$  accounts for the outliers. However, PCP provides estimates for both  $L$  and  $S$ . As it will be discussed in the next subsection, state-of-the-art applications are well accommodated when the focus of interest is turned into the sparse matrix  $S$  itself.

### Remarks 13.

- Just as  $\ell_1$ -minimization is the tightest convex relaxation of the combinatorial  $\ell_0$ -minimization problem in compressed sensing, the nuclear-norm minimization is the tightest convex relaxation of the NP-hard rank minimization problem; i.e., the nuclear ball  $\{M \in \mathcal{R}^{l_1 \times l_2} : \|M\|_* \leq 1\}$  is the convex hull of the set of rank-one matrices with spectral norm bounded by one. Besides the Nuclear norm, other heuristics have also been proposed, such as the log-det heuristic [203] and the max-norm [204].
- The nuclear norm, as a rank minimization approach, is the generalization of the trace-related cost, which is often used in the control community for the rank minimization of positive semidefinite matrices [205]. Indeed, when the matrix is symmetric and positive semidefinite, the nuclear norm of  $M$  is the sum of the eigenvalues and thus it is equal to the trace of  $M$ . Such problems arise when, for example, the rank minimization task refers to covariance matrices and positive semidefinite Toeplitz or Hankel matrices (see, e.g., [203]).
- Both matrix completion (23.95) and PCP (23.97) can be formulated as semidefinite programs and are solved via mobilizing interior-point methods. However, whenever the size of a matrix becomes large (e.g.,  $100 \times 100$ ), these methods are deemed to fail in practice due to excessive power and memory requirements. As a result, there is an increasing interest, which has propelled intensive research efforts, for the development of efficient methods to solve (23.95), (23.97) or related approximations,

which scale well with large matrices. Many of these methods revolve around the philosophy of the iterative soft and hard thresholding techniques, as discussed in previous sections. However, in the current low rank approximation setting, it is the singular values of the estimated matrix which are thresholded. As a result, in each iteration, the estimated matrix, after thresholding its singular values, tends to be of lower rank. The thresholding of the singular values is either imposed, such as in the case of the singular value thresholding (SVT) algorithm [206] or it results as a solution of the regularized versions of (23.95) and (23.97) (see, e.g., [207–211]). Moreover, algorithms inspired by greedy methods such as CoSaMP, have also been proposed [212, 213].

- Further developments on PCP led to improved versions [197] allowing for exact recovery even though the number of nonzero entries of  $S$  approaches  $l_1 l_2$  arbitrarily close, provided that the sign pattern of  $S$  is random. Furthermore, even full columns are allowed to be corrupted [214, 215]. Moreover, fusions of PCP with matrix completion and Compressed Sensing are possible, in the sense that only a subset of the entries of  $M$  is available and/or linear measurements of the matrix in a Compressed Sensing fashion can be used instead of matrix entries (see, e.g., [213, 216]). Moreover, stable versions of PCP dealing with noise have also been investigated (see, e.g., [217]).

### 1.23.16.3 Applications of matrix completion and PCP

The number of applications in which these techniques are involved is ever increasing and their extensive presentation is out of the scope of this paper. Next, some key applications will be selectively discussed since they reveal the potential of these methods and at the same time will assist the reader for a better understanding of the underline notions.

#### 1.23.16.3.1 Matrix completion

A typical application, where the matrix completion problem arises, is in the collaborative filtering task [218], which is essential for building up successful recommender systems. Let us consider that a group of individuals provide their ratings concerning products, which they have enjoyed. Then a matrix with ratings can be filled where each row indexes a different individual and the columns index the products. As a popular example take the case where the products are different movies. Inevitably, the associated matrix will be partially filled, since it is not common that all customers have watched all the movies and submit ratings for all of them. Matrix completion comes to provide an answer, potentially in the affirmative, to the following question. Can we predict the ratings that the users would give to films that they have not seen yet? This is the task of a recommender system in order to encourage users to watch movies, which are likely to be of their preference. The exact objective of competition for the famous Netflix prize (<http://www.netflixprize.com/>) was the development of such a recommender system.

The aforementioned problem provides a good opportunity to build up our intuition about the matrix completion task. First, an individual's preferences or taste on movies are typically governed by a small number of factors, such as the gender, the actors they play in it, the continent of origin, etc. As a result, a matrix fully filled with ratings is expected to be low rank. Moreover, it is clear that each user need to have at least one movie rated in order to have any hope to fill out his/her ratings across all movies. The same is true for each movie. This requirement complies with the second requirement in Section 1.23.16.1, concerning uniqueness, i.e., one needs to know at least one entry per row and column. Finally, imagine a single user who rates movies with criteria that are completely different to those used by the

rest of the users. He/She could, for example, provide ratings at random or depending on, let us say, the first letter of the movie title. Such a scenario complies with the third point concerning the uniqueness in the matrix completion problem, as previously discussed. Unless all the ratings of the specific user are known, the matrix cannot get fully completed.

In the previous application, the matrix of interest can be characterized as approximately low rank. In other cases, such as in sensor network localization [219], the rank of the matrix assumes an exact value. The goal of localization is to assign geographic coordinates to each node in the sensor network based on a square matrix, which contains the pairwise distances between the nodes [220, 221]. It turns out that this matrix is of very low rank, e.g., two or three, depending on whether the sensors are placed in the 2D or 3D space. As a result, matrix completion is possible using a limited number of distance measurements. The number of distance measurements is reduced either intentionally, in order to save power and/or due to the presence of irregularities and obstacles in the deployment area, which renders the communication among nodes impossible. Other applications of matrix completion includes system identification [222], recovering structure from motion [223] and multi-task learning [224].

### 1.23.16.3.2 Robust PCA/PCP

In the collaborative filtering task, robust PCA offers an extra attribute compared to matrix completion, which can be proved very crucial in practice. The users are allowed to even tamper with some of the ratings without affecting the estimation of the low rank matrix. This seems to be the case whenever the rating process involves many individuals in an environment, which is not strictly controlled, since some of them occasionally are expected to provide ratings in an ad hoc, or even malicious manner.

One of the first applications of PCP was in video surveillance systems [194] and the main idea behind it appeared to be popular and extendable to a number of computer vision applications. Take the example of a camera recording a sequence of frames consisting of a merely static background and a foreground with few moving objects, e.g., vehicles and/or individuals. A common task in surveillance video is to extract from the background the foreground, in order, for example, to detect any activity or to proceed with further processing such as face recognition. Suppose the successive frames are converted to vectors in lexicographic order and then are placed as columns in a matrix  $M$ . Due to the background, even though this may slightly vary due to changes in illumination, successive columns are expected to be highly correlated. As a result, the background can be modeled as an approximately low rank matrix  $L$ . On the other hand, the objects on the foreground, appear as “anomalies” concerning a fraction of pixels of each frame, i.e., a limited number of entries in each column of  $M$ . Moreover, due to the motion of the foreground objects, the positions of these anomalies are likely to change from one column of  $M$  to the next. Therefore, they can be modeled as a sparse matrix  $S$ . Note that in this application, the matrix of interest is the sparse matrix rather than the low rank one.

---

## 1.23.17 Conclusions

In this paper, we provided an overview of the major theoretical advances as well as the main trends in algorithmic developments in the area of sparsity-aware learning and compressed sensing. Both batch processing and online processing techniques were considered. A case study in the context of time-frequency analysis of signals was also presented. Our intent is to update the review from time to time, since this is a very hot research area with a momentum and speed that is sometimes difficult to follow up.

## Appendix

The stage of our discussion in this Appendix is the real Euclidean space  $\mathcal{R}^l$ , where  $l$  is a positive integer. Although all of the following arguments hold true even in the case where the  $\mathcal{R}^l$  is substituted by the much more general Hilbert space setting, we confine ourselves here, for the sake of simplicity, to the Euclidean space. Henceforth, the space  $\mathcal{R}^l$  is considered to be equipped with an inner product, which, in the present context, is denoted by  $\langle \theta_1, \theta_2 \rangle, \forall \theta_1, \theta_2 \in \mathcal{R}^l$ . A standard example of such an inner product is the classical vector/dot one, defined by  $\langle \theta_1, \theta_2 \rangle := \theta_1^T \theta_2, \forall \theta_1, \theta_2 \in \mathcal{R}^l$ , where the superscript  $(\cdot)^T$  stands for vector transposition. Another example of an inner product for the space  $\mathcal{R}^l$  is the following *weighted* one;  $\langle \theta_1, \theta_2 \rangle := \theta_1^T W \theta_2, \forall \theta_1, \theta_2 \in \mathcal{R}^l$ , where  $W \in \mathcal{R}^{l \times l}$  is any user-defined positive definite matrix. In order not to spare the generality of the following discussion, we let  $\langle \cdot, \cdot \rangle$  stand for any user-defined inner product on the linear space  $\mathcal{R}^l$ . Given such an inner product, the associated norm is induced according to the following rule:  $\|\cdot\| := \sqrt{\langle \cdot, \cdot \rangle}$ . Excellent resources for a deeper study on the extremely rich subject of convex analysis are [225–228].

We start, now, with few notions of fundamental importance to convex analysis.

### A.1 Closed convex sets and metric projection mappings

**Definition 7 (Convex set, convex function).** A non-empty subset  $C$  of  $\mathcal{R}^l$  is called *convex* if  $\forall \theta_1, \theta_2 \in \mathcal{R}^l$ , and  $\forall \lambda \in [0, 1]$ , the following holds true:  $\lambda \theta_1 + (1 - \lambda) \theta_2 \in C$ . Moreover, a function  $\mathcal{L} : \mathcal{R}^l \rightarrow \mathcal{R}$  is called *convex* if  $\forall \theta_1, \theta_2 \in \mathcal{R}^l$ , and  $\forall \lambda \in [0, 1]$ ,  $\mathcal{L}(\lambda \theta_1 + (1 - \lambda) \theta_2) \leq \lambda \mathcal{L}(\theta_1) + (1 - \lambda) \mathcal{L}(\theta_2)$ . The function  $\mathcal{L}$  is called *strictly convex* if  $\forall \lambda \in (0, 1)$  and  $\forall \theta_1, \theta_2 \in \mathcal{R}^l$ , such that  $\theta_1 \neq \theta_2$ , we have  $\mathcal{L}(\lambda \theta_1 + (1 - \lambda) \theta_2) < \lambda \mathcal{L}(\theta_1) + (1 - \lambda) \mathcal{L}(\theta_2)$ .

The *epigraph* of a function  $\mathcal{L}$  is defined as the set

$$\text{epi}(\mathcal{L}) := \{(\theta, r) \in \mathcal{R}^l \times \mathcal{R} : \mathcal{L}(\theta) \leq r\}.$$

In other words, the epigraph of  $\mathcal{L}$  is the set of all points of  $\mathcal{R}^l \times \mathcal{R}$  which belong to and lie above the graph of  $\mathcal{L}$ . Notice, also, by the definition of convexity, that  $\mathcal{L}$  is convex if and only if  $\text{epi}(\mathcal{L})$  is convex.

Given a real number  $\xi$ , the *lower level set* of  $\mathcal{L}$  at height  $\xi$  is defined as the set

$$\text{lev}_{\leq \xi}(\mathcal{L}) := \{\theta \in \mathcal{R}^l : \mathcal{L}(\theta) \leq \xi\}.$$

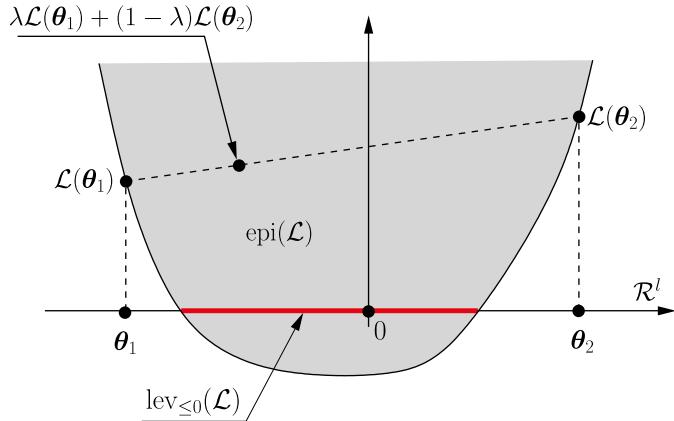
For the geometry behind the previous definitions, see Figure 23.30.

**Definition 8 (The metric projection mapping).** Given a non-empty *closed* convex set  $C \subset \mathcal{R}^l$ , the *metric projection mapping onto*  $C$  is defined as the operator that maps to each  $\theta \in \mathcal{R}^l$  the *unique*  $P_C(\theta) \in C$  such that

$$\|\theta - P_C(\theta)\| = d(\theta, C).$$

In other words, the point  $P_C(\theta)$  is the unique minimizer of the function  $\|\theta - x\|, x \in C$ . Obviously, in the case where  $\theta \in C$ , then  $P_C(\theta) = \theta$ .

As an example, the metric projection mapping onto the *hyperslab* is given next.

**FIGURE 23.30**

A convex function  $\mathcal{L}$ , its epigraph, and the lower level set of  $\mathcal{L}$  at height 0.

**Example 8 (Hyperslab).** A hyperslab  $S[\epsilon]$  is the closed convex subset of  $\mathcal{R}^l$  which is defined as

$$S[\epsilon] := \{x \in \mathcal{R}^l : |\langle a, x \rangle - c| \leq \epsilon\},$$

for some nonzero  $a \in \mathcal{R}^l$  and some  $c \in \mathcal{R}$ . The projection mapping  $P_{S[\epsilon]}$  onto  $S[\epsilon]$  is given as follows:

$$P_{S[\epsilon]}(\theta) = \theta - \begin{cases} \frac{\langle a, \theta \rangle - c - \epsilon}{\|a\|^2} a, & \text{if } \langle a, \theta \rangle > c + \epsilon, \\ 0, & \text{if } |\langle a, \theta \rangle - c| \leq \epsilon, \\ \frac{\langle a, \theta \rangle - c + \epsilon}{\|a\|^2} a, & \text{if } \langle a, \theta \rangle < c - \epsilon. \end{cases} \quad (23.98)$$

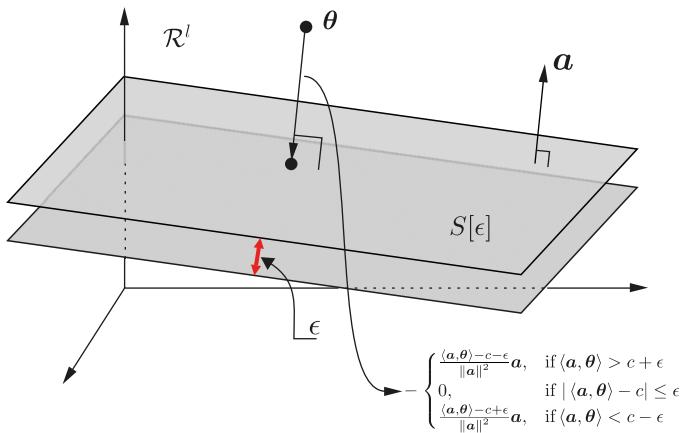
For the related geometry, see Figure 23.31. Notice that  $a$  stands for the *normal vector* defining the hyperplanes associated with the hyperslab.

## A.2 The subgradient

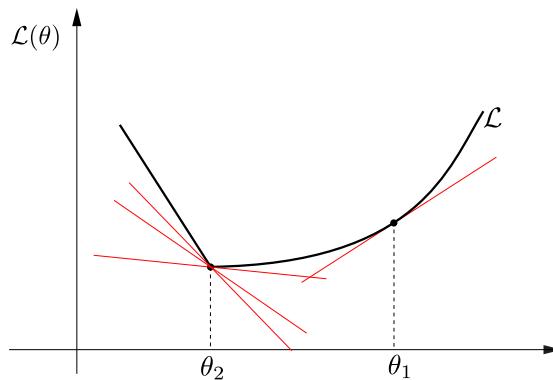
**Definition 9 (Subgradient, Subdifferential).** Given a convex function  $\mathcal{L}$ , defined on  $\mathcal{R}^l$ , and a point  $\theta \in \mathcal{R}^l$ , the subgradient of  $\mathcal{L}$  at  $\theta$  is defined as any vector,  $h$ , such that

$$\langle h, x - \theta \rangle + \mathcal{L}(\theta) \leq \mathcal{L}(x), \quad \forall x \in \mathcal{R}^l. \quad (23.99)$$

If the function  $\mathcal{L}$  is differentiable at  $\theta$  then the subgradient coincides with the (unique) gradient. As it is the case for the gradient, a subgradient defines a hyperplane. This hyperplane “supports” the epigraph of  $\mathcal{L}$ ; that is, the epigraph is on the one side of this hyperplane (see Figure 23.32). At  $\theta_1$ , the convex function is differentiable and there is only one subgradient, which coincides with the gradient. Thus at this point, there is a simple hyperplane that supports the epigraph. At  $\theta_2$ , the function is not differentiable.

**FIGURE 23.31**

A hyperslab and its associated projection mapping.

**FIGURE 23.32**

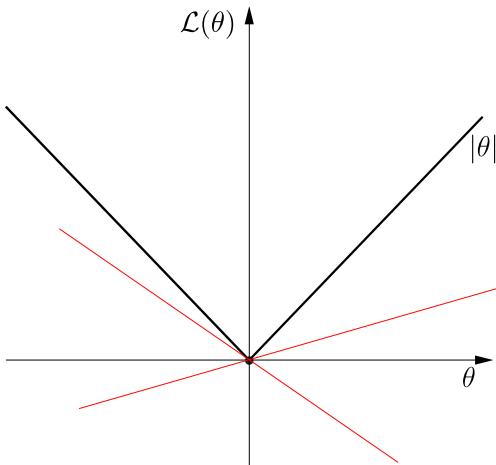
The graph of  $\mathcal{L}$  and supporting hyperplanes generated by the subgradients at points,  $\theta_1, \theta_2$ , where the function is differentiable and non-differentiable, respectively.

Hence there is an infinity of subgradients that define hyperplanes that support the epigraph. The set of all subgradients at a point  $\theta$  is known as the subdifferential and is denoted as  $\partial\mathcal{L}$ , i.e.,

$$\partial\mathcal{L}(\theta) := \{\mathbf{h} \in \mathbb{R}^l : \langle \mathbf{h}, \mathbf{x} - \theta \rangle + \mathcal{L}(\mathbf{x}) \leq \mathcal{L}(\theta), \quad \forall \mathbf{x} \in \mathbb{R}^l\}.$$

Next, the subdifferential of  $\mathcal{L}(\theta) := |\theta|$ ,  $\theta \in \mathbb{R}$  is given:

$$\partial\mathcal{L}(\theta) = \begin{cases} [-1, 1], & \text{if } \theta = 0, \\ \text{sgn}(\theta), & \text{if } \theta \neq 0, \end{cases}$$

**FIGURE 23.33**

The graph of the function  $|\cdot|$ , and the supporting hyperplanes generated by the subgradients of  $|\cdot|$  at  $\theta = 0$ .

where  $\text{sgn}(\cdot)$  stands for the sign of a real number. For the geometry associated to this cost function see Figure 23.33.

*Relevant theory:* Signal processing theory

See this Volume, [Chapter 10](#) Frames

See this Volume, [Chapter 11](#) Parametric Estimation

See this Volume, [Chapter 12](#) Adaptive Filters

## References

- [1] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth ed., Academic Press, 2009.
- [2] J.F. Claerbout, F. Muir, Robust modeling with erratic data, *Geophysics* 38 (5) (1973) 826–844.
- [3] H.L. Taylor, S.C. Banks, J.F. McCoy, Deconvolution with the  $\ell_1$  norm, *Geophysics* 44 (1) (1979) 39–52.
- [4] D.L. Donoho, B.F. Logan, Signal recovery and the large sieve, *SIAM J. Appl. Math.* 52 (2) (1992) 577–591.
- [5] F. Santosa, W.W. Symes, Linear inversion of band limited reflection seismograms, *SIAM J. Sci. Comput.* 7 (4) (1986) 1307–1330.
- [6] R. Tibshirani, Regression shrinkage and selection via the LASSO, *J. Roy. Statist. Soc. B* 58 (1) (1996) 267–288.
- [7] S. Chen, D.L. Donoho, M. Saunders, Atomic decomposition by basis pursuit, *SIAM J. Sci. Comput.* 20 (1) (1998) 33–61.
- [8] A.M. Bruckstein, D.L. Donoho, M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM Rev.* 51 (1) (2009) 34–81.
- [9] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
- [10] Y. Ye, Interior Point Methods: Theory and Analysis, Wiley, New York, 1997.

- [11] A. Antoniadis, Wavelet methods in statistics: some recent developments and their applications, *Statist. Surv.* 1 (2007) 16–55.
- [12] J. Arenas-Garcia, A.R. Figueiras-Vidal, Adaptive combination of proportionate filters for sparse echo cancellation, *IEEE Trans. Audio Speech Lang. Process.* 17 (6) (2009) 1087–1098.
- [13] J. Benesty, T. Gansler, D.R. Morgan, M.M. Sondhi, S.L. Gay, *Advances in Network and Acoustic Echo Cancellation*, Springer-Verlag, Berlin, 2001.
- [14] P.A. Naylor, J. Cui, M. Brookes, Adaptive algorithms for sparse echo cancellation, *Signal Process.* 86 (2004) 1182–1192.
- [15] S. Haykin, *Adaptive Filter Theory*, third ed., Prentice-Hall, NJ, 1996.
- [16] A.H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley & Sons, New Jersey, 2003.
- [17] S. Ariyavivatkul, N.R. Sollenberger, L.J. Greenstein, Tap-selectable decision feedback equalization, *IEEE Trans. Commun.* 45 (12) (1997) 1498–1500.
- [18] S.F. Cotter, B.D. Rao, Matching pursuit based decision-feedback equalizers, in: *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Istanbul, Turkey, 2000.
- [19] M. Ghosh, Blind decision feedback equalization for terrestrial television receivers, *Proc. IEEE* 86 (10) (1998) 2070–2081.
- [20] A. Rondogiannis, K. Berberidis, Efficient decision feedback equalization for sparse wireless channels, *IEEE Trans. Wireless Commun.* 2 (3) (2003) 570–581.
- [21] D. Eiwen, G. Taubock, F. Hlawatsch, H.G. Feichtinger, Group sparsity methods for compressive channel estimation in doubly dispersive multicarrier systems, in: *Proceedings IEEE SPAWC*, Marrakech, Morocco, June 2010.
- [22] D. Eiwen, G. Taubock, F. Hlawatsch, H. Rauhut, N. Czink, Multichannel-compressive estimation of doubly selective channels in MIMO-OFDM systems: exploiting and enhancing joint sparsity, in: *Proceedings International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas, 2010.
- [23] W.U. Bajwa, J. Haupt, A.M. Sayeed, R. Nowak, Compressed channel sensing: a new approach to estimating sparse multipath channels, *Proc. IEEE* 98 (6) (2010) 1058–1076.
- [24] S. Mallat, S. Zhang, Matching pursuit in a time-frequency dictionary, *IEEE Trans. Signal Process.* 41 (1993) 3397–3415.
- [25] R.R. Coifman, M.V. Wickerhauser, Entropy-based algorithms for best basis selection, *IEEE Trans. Inform. Theory* 38 (2) (1992) 713–718.
- [26] Q. Qiu, V.M. Patel, P. Turaga, R. Chellappa, Domain adaptive dictionary learning, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, Florene, Italy, 2012.
- [27] R. Rubinstein, A. Bruckstein, M. Elad, Dictionaries for sparse representation modeling, *Proc. IEEE* 98 (6) (2010) 1045–1057.
- [28] I. Tosić, P. Frossard, Dictionary Learning, *IEEE Signal Process. Mag.* 28 (2) (2011) 27–38.
- [29] M. Yaghoobi, L. Daudet, M. Davies, Parametric dictionary design for sparse coding, *IEEE Trans. Signal Process.* 57 (12) (2009) 4800–4810.
- [30] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010.
- [31] S. Wright, R. Nowak, M. Figueiredo, Sparse reconstruction by separable approximation, *IEEE Trans. Signal Process.* 57 (7) (2009) 2479–2493.
- [32] A. Maleki, L. Anitori, Z. Yang, R. Baraniuk, Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP), *IEEE Trans. Inform. Theory* (2011). [arXiv:1108.0477](https://arxiv.org/abs/1108.0477).
- [33] I. Daubechies, Time-frequency localization operators: a geometric phase space approach, *IEEE Trans. Inform. Theory* 34 (4) (1988) 605–612.
- [34] B.K. Natarajan, Sparse approximate solutions to linear systems, *SIAM J. Comput.* 24 (1995) 227–234.

- [35] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.
- [36] A.M. Pinkus, On  $\ell_1$ -approximation, Cambridge Tracts in Mathematics, vol. 93, Cambridge University Press, 1989.
- [37] D.L. Donoho, M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via  $\ell_1$  minimization, in: Proceedings of National Academy of Sciences, 2003, pp. 2197–2202.
- [38] I.F. Gorodnitsky, B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm, IEEE Trans. Signal Process. 45 (3) (1997) 600–614.
- [39] L.R. Welch, Lower bounds on the maximum cross correlation of signals, IEEE Trans. Inform. Theory 20 (3) (1974) 397–399.
- [40] D.L. Donoho, X. Huo, Uncertainty principles and ideal atomic decomposition, IEEE Trans. Inform. Theory 47 (7) (2001) 2845–2862.
- [41] R. Gribonval, M. Nielsen, Sparse decompositions in unions of bases, IEEE Trans. Inform. Theory 49 (12) (2003) 3320–3325.
- [42] M. Elad, A.M. Bruckstein, A generalized uncertainty principle and sparse representations in pairs of basis, IEEE Trans. Inform. Theory 48 (9) (2002) 2558–2567.
- [43] D.L. Donoho, J.T. Precise undersampling theorems, Proc. IEEE, 98 (6) (2010) 913–924.
- [44] E.J. Candès, T. Tao, Decoding by linear programming, IEEE Trans. Inform. Theory 51 (12) (2005) 4203–4215.
- [45] R.G. Baraniuk, M. Davenport, R. DeVore, M.B. Wakin, A simple proof of the restricted isometry property for random matrices, Constr. Approx. 28 (2008) 253–263.
- [46] E.J. Candès, J. Romberg, Practical signal recovery from random projections, in: Proceedings of the SPIE 17th Annual Symposium on Electronic Imaging, Bellingham, WA, 2005.
- [47] E.J. Candès, J. Romberg, T. Tao, Stable recovery from incomplete and inaccurate measurements, Commun. Pure Appl. Math. 59 (8) (2006) 1207–1223.
- [48] E.J. Candès, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
- [49] T.T. Cai, G. Xu, J. Zhang, On recovery of sparse signals via  $\ell_1$  minimization, IEEE Trans. Inform. Theory 55 (7) (2009) 3388–3397.
- [50] S. Mendelson, A. Pajor, N. Tomczak-Jaegermann, Uniform uncertainty principle for Bernoulli and subgaussian ensembles, Constr. Approx. 28 (2008) 277–289.
- [51] M. Rudelson, R. Vershynin, On sparse reconstruction from Fourier and Gaussian measurements, Commun. Pure Appl. Math. 61 (8) (2008) 1025–1045.
- [52] E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete Fourier information, IEEE Trans. Inform. Theory 52 (2) (2006) 489–509.
- [53] J. Haupt, W.U. Bajwa, G. Raz, R. Nowak, Toeplitz compressed sensing matrices with applications to sparse channel estimation, IEEE Trans. Inform. Theory 56 (11) (2010) 5862–5875.
- [54] M.F. Duarte, R.G. Baraniuk, Kronecker Compressive Sensing, IEEE Trans. Image Process. 21 (2) (2012) 494–504.
- [55] Y. Rivenson, A. Stern, Compressed imaging with a separable sensing operator, IEEE Signal Process. Lett. 16 (6) (2009) 449–452.
- [56] D.L. Donoho, J. Tanner, Counting faces of randomly projected polytopes when the projection radically lowers dimension, Technical Report 2006–11, Stanford University, 2006.
- [57] P. Bickel, Y. Ritov, A. Tsybakov, Simultaneous analysis of LASSO and Dantzig selector, Ann. Statist. 37 (4) (2009) 1705–1732.
- [58] A. Cohen, W. Dahmen, R. DeVore, Compressed sensing and best k-term approximation, J. Am. Math. Soc. 22 (1) (2009) 211–231.

- [59] G. Tang, A. Nehorai, Performance analysis of sparse recovery based on constrained minimal singular values, *IEEE Trans. Signal Process.* 59 (12) (2011) 5734–5745.
- [60] E. Candès, T. Tao, Near optimal signal recovery from random projections: universal encoding strategies, *IEEE Trans. Inform. Theory* 52 (12) (2006) 5406–5425.
- [61] J.A. Tropp, J.N. Laska, M.F. Duarte, J.K. Romberg, G. Baraniuk, Beyond Nyquist: efficient sampling of sparse bandlimited signals, *IEEE Trans. Inform. Theory* 56 (1) (2010) 520–544.
- [62] D. Takhar, V. Bansal, M. Wakin, M. Duarte, D. Baron, K.F. Kelly, R.G. Baraniuk, A compressed sensing camera: new theory and an implementation using digital micromirrors, in: *Proceedings on Computational Imaging (SPIE)*, San Jose, CA, 2006.
- [63] R. Baraniuk, V. Cevher, M. Wakin, Low-dimensional models for dimensionality reduction and signal recovery: a geometric perspective, *Proc. IEEE* 98 (6) (2010) 959–971.
- [64] Y.M. Lu, M.N. Do, Sampling signals from a union of subspaces, *IEEE Signal Process. Mag.* 25 (2) (2008) 41–47.
- [65] D. Achlioptas, Database-friendly random projections, in: *Proceedings of the Symposium on Principles of Database Systems (PODS)*, ACM Press, 2001, pp. 274–281.
- [66] A. Blum, Random projection, margins, kernels and feature selection, in: *Lecture Notes on Computer Science (LNCS)*, 2006, pp. 52–68.
- [67] Dasgupta S. Experiments with random projections, in: *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Morgan-Kaufmann, San Francisco, CA, USA, 2000, pp. 143–151.
- [68] R. Calderbank, S. Jeafarpour, R. Schapire, Compressed learning: universal spars dimensionality reduction and learning in the measurement domain, Technical Report, Rice University, 2009.
- [69] R. Baraniuk, M. Wakin, Random projections of smooth manifolds, *Found. Comput. Math.* 9 (1) (2009) 51–77.
- [70] M. Wakin, Manifold-based signal recovery and parameter estimation from compressive measurements, 2008. preprint: <http://arxiv.org/abs/1002.1247>.
- [71] M. Unser, Sampling: 50 years after Shannon, *Proc. IEEE* 88 (4) (2000) 569–587.
- [72] Y.-P. Lin, P.P. Vaidyanathan, Periodically nonuniform sampling of bandpass signals, *IEEE Trans. Circ. Syst. II* 45 (3) (1998) 340–351.
- [73] R.G. Vaughan, N.L. Scott, D.R. White, The theory of bandpass sampling, *IEEE Trans. Signal Process.* 39 (9) (1991) 1973–1984.
- [74] R. Venkataramani, Y. Bresler, Perfect reconstruction formulas and bounds on aliasing error in sub-Nyquist nonuniform sampling of multiband signals, *IEEE Trans. Inform. Theory* 46 (6) (2000) 2173–2183.
- [75] M. Mishali, Y.C. Eldar, A. Elron, Xampling: analog data compression, in: *Proceedings Data Compression Conference*, Snowbird, Utah, USA, 2010.
- [76] Z. Tian, G.B. Giannakis, Compressed sensing for wideband cognitive radios, in: *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 1357–1360.
- [77] Z. Yu, S. Hoyos, B.M. Sadler, Mixed-signal parallel compressed sensing and reception for cognitive radio, in: *Proceedings IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 3861–3864.
- [78] S. Kirolos, J.N. Laska, M.B. Wakin, M.F. Duarte, D. Baron, T. Ragheb, Y. Massoud, R.G. Baraniuk, Analog to information conversion via random demodulation, in: *Proceedings of the IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, Dallas, USA, 2006, pp. 71–74.
- [79] M. Mishali, Y.C. Eldar, A. Elron, Xampling: signal acquisition and processing in union of subspaces, *IEEE Trans. Signal Process.* 59 (10) (2011) 4719–4734.
- [80] F. Chen, A.P. Chandrakasan, V.M. Stojanovic, Design and analysis of hardware efficient compressed sensing architectures for compression in wireless sensors, *IEEE Trans. Solid State Circ.* 47 (3) (2012) 744–756.

- [81] P. Maechler, N. Felber, H. Kaeslin, A. Burg, Hardware-efficient random sampling of Fourier-sparse signals, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2012.
- [82] P.L. Dragotti, M. Vetterli, T. Blu, Sampling moments and reconstructing signals of finite rate of innovation: Shannon meets Strang-Fix, *IEEE Trans. Signal Process.* 55 (5) (2007) 1741–1757.
- [83] T. Blu, P.L. Dragotti, M. Vetterli, P. Marziliano, L. Coulot, Sparse sampling of signal innovations, *IEEE Signal Process. Mag.* 25 (2) (2008) 31–40.
- [84] M. Vetterli, P. Marziliano, T. Blu, Sampling signals with finite rate of innovation, *IEEE Trans. Signal Process.* 50 (6) (2002) 1417–1428.
- [85] E. Matusiak, Y.C. Eldar, Sub-Nyquist sampling of short pulses, *IEEE Trans. Signal Process.* 60 (3) (2012) 1134–1148.
- [86] M.F. Duarte, Y. Eldar, Structured compressed sensing: from theory to applications, *IEEE Trans. Signal Process.* 59 (9) (2011) 4053–4085.
- [87] M. Mishali, Y.C. Eldar, Sub-Nyquist sampling, *IEEE Signal Process. Mag.* 28 (6) (2011) 98–124.
- [88] A.C. Gilbert, S. Muthukrishnan, M.J. Strauss, Improved time bounds for near-optimal sparse Fourier representation via sampling, in: Proceedings of SPIE (Wavelets XI), San Diego, CA, 2005.
- [89] D. Needell, J.A. Tropp, COSAMP: iterative signal recovery from incomplete and inaccurate samples, *Appl. Comput. Harmon. Anal.* 26 (3) (2009) 301–321.
- [90] V.N. Temlyakov, Nonlinear methods of approximation, *Foun. Comput. Math.* 3 (1) (2003) 33–107.
- [91] R.A. DeVore, V.N. Temlyakov, Some remarks on greedy algorithms, *Adv. Comput. Math.* 5 (1996) 173–187.
- [92] M.A. Davenport, M.B. Wakin, Analysis of orthogonal matching pursuit using the restricted isometry property, *IEEE Trans. Inform. Theory* 56 (9) (2010) 4395–4401.
- [93] J.A. Tropp, Greed is good, *IEEE Trans. Inform. Theory* 50 (2004) 2231–2242.
- [94] T. Zhang, Sparse recovery with orthogonal matching pursuit under RIP, *IEEE Trans. Inform. Theory* 57 (9) (2011) 6215–6221.
- [95] B. Efron, T. Hastie, I.M. Johnstone, R. Tibshirani, Least angle regression, *Ann. Statist.* 32 (2004) 407–499.
- [96] Y. Tsai, Sparse solution of underdetermined linear systems: algorithms and applications, PhD Thesis, Stanford University, 2007.
- [97] M.R. Osborne, B. Presnell, B.A. Turlach, A new approach to variable selection in least squares problems, *IMA J. Numer. Anal.* 20 (2000) 389–403.
- [98] M.S. Asif, J. Romberg, Dynamic updating for  $\ell_1$  minimization, *IEEE J. Sel. Top. Signal Process.* 4 (2) (2010) 421–434.
- [99] D.M. Malioutov, M. Cetin, A.S. Willsky, Homotopy continuation for sparse signal representation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2005, pp. 733–736.
- [100] M.D. Plumbley, Geometry and homotopy for L1 sparse representation, in: Proceedings of the International Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS), Rennes, France, 2005.
- [101] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, *IEEE Trans. Inform. Theory* 55 (5) (2009) 2230–2249.
- [102] D.L. Donoho, I.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika* 81 (3) (1994) 425–455.
- [103] J.C. Hoch, A.S. Stern, D.L. Donoho, I.M. Johnstone, Maximum entropy reconstruction of complex (phase sensitive) spectra, *J. Magn. Reson.* 86 (2) (1990) 236–246.
- [104] P.A. Jansson, Deconvolution: Applications in Spectroscopy, Academic Press, New York, 1984.
- [105] N.G. Kingsbury, T.H. Reeves, Overcomplete image coding using iterative projection-based noise shaping, in: Proceedings IEEE International Conference on Image Processing (ICIP), 2002, pp. 597–600.
- [106] P.L. Combettes, V.R. Wajs, Signal recovery by proximal forward-backward splitting, *SIAM J. Multiscale Model. Simul.* 4 (4) (2005) 1168–1200.

- [107] I. Daubechies, M. Defrise, C. De-Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Commun. Pure Appl. Math.* 57 (11) (2004) 1413–1457.
- [108] M. Elad, B. Matalon, M. Zibulevsky, Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization, *Appl. Comput. Harmon. Anal.* 23 (2007) 346–367.
- [109] M.A. Figueiredo, R.D. Nowak, An EM algorithm for wavelet-based image restoration, *IEEE Trans. Image Process.* 12 (8) (2003) 906–916.
- [110] L. Hageman, D. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.
- [111] A. Beck, M. Teboulle, A fast iterative shrinkage algorithm for linear inverse problems, *SIAM J. Imag. Sci.* 2 (1) (2009) 183–202.
- [112] R.T. Rockafellar, Monotone operators and the proximal point algorithms, *SIAM J. Control Optim.* 14 (5) (1976) 877–898.
- [113] T. Hale, W. Yin, Y. Zhang, A fixed-point continuation method for  $l_1$  regularized minimization with applications to compressed sensing, Technical Report TR07-07, Department of Computational and Applied Mathematics, Rice University, 2007.
- [114] Y.E. Nesterov, A method for solving the convex programming problem with convergence rate  $O(1/k^2)$ , *Dokl. Akad. Nauk SSSR* 269 (1983) 543–547 (in Russian).
- [115] T. Blumensath, M.E. Davies, Iterative hard thresholding for compressed sensing, *Appl. Comput. Harmon. Anal.* 27 (3) (2009) 265–274.
- [116] T. Blumensath, M.E. Davies, Normalized iterative hard thresholding: guaranteed stability and performance, *IEEE Sel. Top. Signal Process.* 4 (2) (2010) 298–309.
- [117] Z.Q. Luo, P. Tseng, On the convergence of the coordinate descent method for convex differentiable minimization, *J. Optim. Theory Appl.* 72 (1) (1992) 7–35.
- [118] M. Zibulevsky, M. Elad, L1-L2 optimization in signal processing, *IEEE Signal Process. Mag.* 27 (3) (2010) 76–88.
- [119] A. Maleki, D.L. Donoho, Optimally tuned iterative reconstruction algorithms for compressed sensing, *IEEE J. Sel. Top. Signal Process.* 4 (2) (2010) 330–341.
- [120] S. Foucart, Hard Thresholding pursuit: an algorithm for compressive sensing, *SIAM J. Numer. Anal.* 49 (6) (2011) 2543–2563.
- [121] P.L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, in: *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer-Verlag, 2011.
- [122] S. Becker, J. Bobin, E.J. Candès, NESTA: a fast and accurate first-order method for sparse recovery, *SIAM J. Imag. Sci.* 4 (1) (2011) 1–39.
- [123] E. van den Berg, M.P. Friedlander, Probing the pareto frontier for the basis pursuit solutions, *SIAM J. Sci. Comput.* 31 (2) (2008) 890–912.
- [124] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale  $\ell_1$ -regularized Least Squares, *IEEE J. Sel. Top. Signal Process.* 1 (4) (2007) 606–617.
- [125] I. Daubechies, R. DeVore, M. Fornasier, C.S. Güntürk, Iteratively reweighted least squares minimization for sparse recovery, *Commun. Pure Appl. Math.* 63 (1) (2010) 1–38.
- [126] W. Yin, S. Osher, D. Goldfarb, J. Darbon, Bregman iterative algorithms for  $\ell_1$ -minimization with applications to compressed sensing, *SIAM J. Imag. Sci.* 1 (1) (2008) 143–168.
- [127] D.L. Donoho, J. Tanner, Neiborliness of randomly-projected simplices in high dimensions, in: *Proceedings on National Academy of Sciences*, 2005, pp. 9446–9451.
- [128] D.L. Donoho, J.T. Tanner, Counting the faces of randomly projected hypercubes and orthants, with applications, *Discrete Comput. Geom.* 43(3) (2010) 522–541.
- [129] R.G. Baraniuk, V. Cevher, M.F. Duarte, C. Hegde, Model-based compressive sensing, *IEEE Trans. Inform. Theory* 56 (4) (2010) 1982–2001.

- [130] F. Parvaresh, H. Vikalo, S. Misra, B. Hassibi, Recovering sparse signals using sparse measurement matrices in compressed DNA microarrays, *IEEE J. Sel. Top. Signal Process.* 2 (3) (2008) 275–285.
- [131] P.J. Garrigues, B. Olshausen, Learning horizontal connections in a sparse coding model of natural images, in: *Advances in Neural Information Processing Systems (NIPS)*, 2008.
- [132] S. Bakin, Adaptive regression and model selection in data mining problems, PhD Thesis, Australian National University, 1999.
- [133] J. Friedman, T. Hastie, R. Tibshirani, A note on the group LASSO and a sparse group LASSO, 2010.
- [134] G. Obozinski, B. Taskar, M. Jordan, Multi-task feature selection, Department of Statistics, Technical Report, University of California, Berkeley, 2006.
- [135] M. Yuan, Y. Lin, Model selection and estimation in regression with grouped variables, *J. Roy. Statist. Soc. Ser. B* 68 (1) (2007) 49–67.
- [136] Y.C. Eldar, P. Kuppinger, H. Bolcskei, Block-sparse signals: uncertainty relations and efficient recovery, *IEEE Trans. Signal Process.* 58 (6) (2010) 3042–3054.
- [137] T. Blumensath, M.E. Davies, Sampling theorems for signals from the union of finite-dimensional linear subspaces, *IEEE Trans. Inform. Theory* 55 (4) (2009) 1872–1882.
- [138] V. Cevher, P. Indyk, C. Hegde, R.G. Baraniuk, Recovery of Clustered Sparse Signals from Compressive Measurements, in: *International Conference on Sampling Theory and Applications (SAMPTA)*, Marseille, France, 2009.
- [139] V. Cevher, P. Indyk, L. Carin, R.G. Baraniuk, Sparse signal recovery and acquisition with graphical models, *IEEE Signal Process. Mag.* 27 (6) (2010) 92–103.
- [140] P. Sprechmann, I. Ramirez, G. Sapiro, Y.C. Eldar, CHiLasso: a collaborative hierarchical sparse modeling framework, *IEEE Trans. Signal Process.* 59 (9) (2011) 4183–4198.
- [141] E.J. Candès, M.B. Wakin, S.P. Boyd, Enhancing sparsity by reweighted  $\ell_1$  minimization, *J. Fourier Anal. Appl.* 14 (5) (2008) 877–905.
- [142] E.J. Candès, T. Tao, The Dantzig selector: statistical estimation when  $p$  is much larger than  $n$ , *Ann. Statist.* 35 (6) (2007) 2313–2351.
- [143] M.S. Asif, J. Romberg, On the LASSO and Dantzig selector equivalence, in: *Conference on Information Sciences and Systems (CISS)*, Princeton, NJ, March 2010.
- [144] L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, *Phys. Nonlinear Phenom.* 60 (1–4) (1992) 259–268.
- [145] T. Goldstein, S. Osher, The split Bregman algorithm for  $\ell_1$  regularized problems, *SIAM J. Imag. Sci.* 2 (2) (2009) 323–343.
- [146] J. Yang, Y. Zhang, W. Yin, A fast alternating direction method for TV  $\ell_1$  -  $\ell_2$  signal reconstruction from partial Fourier data, *IEEE Trans. Sel. Top. Signal Process.* 4 (2) (2010) 288–297.
- [147] D. Needell, R. Ward, Stable image reconstruction using total variation minimization, 2012.
- [148] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, *J. Mach. Learn. Res.* 10 (2009) 777–801.
- [149] S. Theodoridis, K. Slavakis, I. Yamada, Adaptive Learning in a World of Projections, *IEEE Signal Process. Mag.* 28 (1) (2011) 97–123.
- [150] S. Kay, *Statistical Signal Processing*, Prentice Hall, 1993.
- [151] D. Donoho, I. Johnstone, G. Kerkyacharian, D. Picard, Wavelet shrinkage: asymptopia? *J. Roy. Statist. Soc. Ser. B* 57 (1995) 301–337.
- [152] K. Knight, W. Fu, Asymptotics for the LASSO-type estimators, *Ann. Statist.* 28 (5) (2000) 1356–1378.
- [153] J. Fan, R. Li, Variable Selection via nonconcave penalized likelihood and its oracle properties, *J. Am. Statist. Assoc.* 96 (456) (2001) 1348–1360.
- [154] H. Zou, The adaptive LASSO and its oracle properties, *J. Am. Statist. Assoc.* 101 (2006) 1418–1429.

- [155] H. Zou, R. Li, One-step sparse estimates in nonconcave penalized likelihood models, *Ann. Statist.* 36 (4) (2008) 1509–1533.
- [156] D. Angelosante, J.A. Bazerque, G.B. Giannakis, Online adaptive estimation of sparse signals: where RLS meets the  $\ell_1$ -norm, *IEEE Trans. Signal Process.* 58 (7) (2010) 3436–3447.
- [157] G. Mileounis, B. Babadi, N. Kalouptsidis, V. Tarokh, An adaptive greedy algorithm with application to nonlinear communications, *IEEE Trans Signal Process.* 58 (6) (2010) 2998–3007.
- [158] N. Ogura, I. Yamada, Non-strictly convex minimization over the fixed point set of the asymptotically shrinking nonexpansive mapping, *Numer. Func. Anal. Optim.* 23 (2002) 113–137.
- [159] K. Slavakis, I. Yamada, N. Ogura, The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings, *Numer. Func. Anal. Optim.* 27 (7 and 8) (2006) 905–930.
- [160] I. Yamada, N. Ogura, Hybrid steepest descent method for variational inequality problem over the fixed point set of certain quasi-nonexpansive mappings, *Numer. Func. Anal. Optim.* 25 (7 and 8) (2004) 619–655.
- [161] J. Duchi, S.S. Shwartz, Y. Singer, T. Chandra, Efficient projections onto the  $\ell_1$ -ball for learning in high dimensions, in: *Proceedings of the International Conference on Machine Learning (ICML)*, 2008, pp. 272–279.
- [162] Y. Kopsinis, K. Slavakis, S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted  $\ell_1$  balls, *IEEE Trans. Signal Process.* 59 (3) (2011) 936–952.
- [163] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Reduced complexity online sparse signal reconstruction using projections onto weighted  $\ell_1$  balls, in: *17th International Conference on Digital Signal Processing (DSP 2011)*, July 2011, pp. 1–8.
- [164] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, Generalized thresholding sparsity-aware algorithm for low complexity online learning, in: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, March 2012, pp. 3277–3280.
- [165] Y. Kopsinis, K. Slavakis, S. Theodoridis, S. McLaughlin, in: *ISCASS 2013*, Submitted for publication.
- [166] M. Elad, P. Milanfar, R. Rubinstein, Analysis versus synthesis in signal priors, *Inv. Prob.* 23 (2007) 947–968.
- [167] J.L. Starck, E.J. Cès, D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. Image Process.* 11 (6) (2002) 670–684.
- [168] J.L. Starck, J. Fadili, F. Murtagh, The undecimated wavelet decomposition and its reconstruction, *IEEE Trans. Signal Process.* 16 (2) (2007) 297–309.
- [169] D. Han, D.R. Larson, *Frames, Bases and Group Representations*, American Mathematical Society, Providence, RI, 2000.
- [170] R.J. Duffin, A.C. Schaeffer, A class of nonharmonic Fourier series, *Trans. Am. Math. Soc.* 72 (1952) 341–366.
- [171] I. Daubechies, A. Grossman, Y. Meyer, Painless nonorthogonal expansions, *J. Math. Phys.* 27 (1986) 1271–1283.
- [172] J. Kovacevic, A. Chebira, Life beyond bases: the advent of frames, *IEEE Signal Process. Mag.* 24 (4) (2007) 86–104.
- [173] E.J. Candès, Y.C. Eldar, D. Needell, P. Randall, Compressed sensing with coherent and redundant dictionaries, *Appl. Comput. Harmon. Anal.* 31 (1) (2011) 59–73.
- [174] S. Nam, M.E. Davies, M. Elad, R. Gribonval, The cosparse analysis model and algorithms, *Appl. Comput. Harmon. Anal.* (in press).
- [175] Y.M. Lu, M.N. Do, A theory for sampling signals from a union of subspaces, *IEEE Trans. Signal Process.* 56 (6) (2008) 2334–2345.
- [176] J.F. Cai, S. Osher, Z. Shen, Split Bregman methods and frame based image restoration, *Multiscale Model. Simul.* 8 (2) (2009) 337–369.
- [177] M. Elad, J.L. Starck, P. Querre, D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA), *Appl. Comput. Harmon. Anal.* 19 (2005) 340–358.

- [178] I.W. Selesnick, M.A.T. Figueiredo, Signal restoration with overcomplete wavelet transforms: comparison of analysis and synthesis priors, in: Proceedings of SPIE, 2009.
- [179] B. Boashash, Time Frequency Analysis, Elsevier, 2003.
- [180] P. Flandrin Time-Frequency/Time-Scale Analysis, Academic Press, 1999.
- [181] S.A. Mallat, Wavelet Tour of Signal Processing: The Sparse Way, third ed., Academic Press, 2008.
- [182] T. Strohmer, Gabor Analysis and Algorithms: Theory and Applications, Birkhauser, Boston, MA, 1998, pp. 267–294 (Chapter Numerical algorithms for discrete Gabor expansions).
- [183] M. Zibulevsky, Y.Y. Zeevi, Frame analysis of the discrete Gabor scheme, *IEEE Trans. Signal Process.* 42 (4) (1994) 942–945.
- [184] Y. Kopsinis, E. Abouantios, D.E. Waters, S. McLaughlin, Time-frequency and advanced frequency estimation techniques for the investigation of bat echolocation calls, *J. Acoust. Soc. Am.* 127 (2) (2010) 1124–1134.
- [185] E. Candès, B. Recht, Exact matrix completion via convex optimization, *Found. Comput. Math.* 9 (6) (2009) 717–772.
- [186] E. Candès, T. Tao, The power of convex relaxation: near-optimal matrix completion, *IEEE Trans. Inform. Theory* 56 (5) (2010) 2053–2080.
- [187] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
- [188] B. Recht, A simpler approach to matrix completion, *J. Mach. Learn. Res.* 12 (2011) 3413–3430.
- [189] M. Fazel, H. Hindi, S.P. Boyd, A rank minimization heuristic with application to minimum order system approximation, in: Proceedings of the American Control Conference 2001, 2001, pp. 4734–4739.
- [190] D. Gross, Recovering low-rank matrices from few coefficients in any basis, *IEEE Trans. Inform. Theory* 57 (3) (2011) 1548–1566.
- [191] E.J. Candès, Y. Plan, Matrix completion with noise, *Proc. IEEE* 98 (6) (2010) 925–936.
- [192] S. Gandy, B. Recht, I. Yamada, Tensor completion and low-n-rank tensor recovery via convex optimization, *Inv. Prob.* 27 (2) (2011).
- [193] M. Signoretto, R. Van de Plas, B. De Moor, J. Suykens, Tensor versus matrix completion: a comparison with application to spectral data, *IEEE Signal Process. Lett.* 18 (7) (2011) 403–406.
- [194] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? *J. ACM*, 58 (3) (2011).
- [195] V. Chandrasekaran, S. Sanghavi, P.A. Parrilo, A.S. Willsky, Rank-sparsity incoherence for matrix decomposition, *SIAM J. Optim.* 21 (2) (2011) 572–596.
- [196] J. Wright, Y. Peng, Y. Ma, A. Ganesh, S. Rao, Robust principal component analysis: exact recovery of corrupted low-rank matrices by convex optimization, in: Neural Information Processing Systems (NIPS), 2009.
- [197] A. Ganesh, J. Wright, X. Li, E.J. Candès, Y. Ma, Dense error correction for low-rank matrices via principal component pursuit, in: IEEE International Symposium on Information Theory Proceedings (ISIT 2010) , 2010, pp. 1513–1517.
- [198] M. Hubert, S. Engelen, Robust PCA and classification in biosciences, *Bioinformatics* 20 (11) (2004) 1728–1736.
- [199] M. Hubert, P.J. Rousseeuw, K. Vanden Branden, ROBPCA: a new approach to robust principal component analysis, *Technometrics* 47 (1) (2005) 64–79.
- [200] J. Karhunen, J. Joutsensalo, Generalizations of principal component analysis, optimization problems, and neural networks, *Neural Netw.* 8 (4) (1995) 549–562.
- [201] F. De la Torre, M.J. Black, A framework for robust subspace learning, *Int. J. Comput. Vis.* 54 (1) (2003) 117–142.
- [202] L. Xu, A. Yuille, Robust principal component analysis by self-organizing rules based on statistical physics approach, *IEEE Trans. Neural Netw.* 6 (1) (1995) 131–143.
- [203] M. Fazel, H. Hindi, S. Boyd, Rank minimization and applications in system theory, in: Proceedings of the American Control Conference 2004, 2004, pp. 3273–3278.

- [204] R. Foygel, N. Srebro, Concentration-based guarantees for low-rank matrix reconstruction, in: 24th Annual Conference on Learning Theory (COLT), 2011.
- [205] M. Mesbahi, G. Papavassilopoulos, On the rank minimization problem over a positive semidefinite linear matrix inequality, *IEEE Trans. Automat. Control* 42 (2) (1997) 239–243.
- [206] J.-F. Cai, E.J. Candès, Z. Shen, A Singular Value Thresholding Algorithm for Matrix Completion, *SIAM J. Optim.* 20 (4) (2010) 1956.
- [207] C. Chen, B. He, X. Yuan, Matrix completion via an alternating direction method, *IMA J. Numer. Anal.* (2011).
- [208] A. Ganesh, Z. Lin, J. Wright, L. Wu, M. Chen, Y. Ma, Fast algorithms for recovering a corrupted low-rank matrix, in: 3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 2009), December 2009, pp. 213–216.
- [209] Z. Lin, M. Chen, Y. Ma, The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices, September 2010. [arXiv:1009.5055](https://arxiv.org/abs/1009.5055).
- [210] K.C. Toh, S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, *Pac. J. Optim.* 6 (2010) 615–640.
- [211] X.M. Yuan, J.F. Yang, Sparse and low-rank matrix decomposition via alternating direction methods, *Pac. J. Optim.* (2009).
- [212] K. Lee, Y. Bresler, ADMiRA: atomic decomposition for minimum rank approximation, *IEEE Trans. Inform. Theory* 56 (9) (2010) 4402–4416.
- [213] A.E. Waters, A.C. Sankaranarayanan, R.G. Baraniuk, SpaRCS: recovering low-rank and sparse matrices from compressive measurements, in: Proceedings of Advances in Neural Processing Systems (NIPS), Granada, Spain, 2011.
- [214] M. McCoy, A.J. Tropp, Two proposals for robust PCA using semidefinite programming, *Electron. J. Statist.* 5 (2011) 1123–1160, Mathematical Reviews number (MathSciNet): MR2836771.
- [215] H. Xu, C. Caramanis, S. Sanghavi, Robust PCA via outlier pursuit, *IEEE Trans. Inform. Theory* 58 (5) (2012) 3047–3064.
- [216] J. Wright, A. Ganesh, K. Min, Y. Ma, Compressive Principal Component Pursuit, February 2012. [arXiv:1202.4596](https://arxiv.org/abs/1202.4596).
- [217] Z. Zhou, X. Li, J. Wright, E. Candès, Y. Ma, Stable principal component pursuit, in: IEEE International Symposium on Information Theory Proceedings (ISIT 2010), 2010, pp. 1518–1522.
- [218] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, *Adv. Artif. Intell.* 2009 (2009) 1–19.
- [219] G. Mao, B. Fidan, B.D. Anderson, Wireless sensor network localization techniques, *Comput. Netw.* 51 (10) (2007) 2529–2553.
- [220] P. Biswas, T.-c. Liang, T.-c. Wang, Y. Ye, Semidefinite programming based algorithms for sensor network localization, *ACM Trans. Sens. Netw.* 2 (2006) 2006.
- [221] A. Montanari, S. Oh, On positioning via distributed matrix completion, in: IEEE Sensor Array and Multi-channel Signal Processing Workshop (SAM 2010), October 2010, pp. 197–200.
- [222] Z. Liu, L. Vandenberghe, Interior-point method for nuclear norm approximation with application to system identification, *SIAM J. Matrix Anal. Appl.* 31 (3) (2010) 1235–1256.
- [223] P. Chen, D. Suter, Recovering the missing components in a large noisy low-rank matrix: application to SFM, *IEEE Trans. Pattern Anal. Mach. Intell.* 26 (8) (2004) 1051–1063.
- [224] A. Argyriou, T. Evgeniou, M. Pontil, Multi-task feature learning, in: Advances in Neural Information Processing Systems 19, MIT Press, 2007.
- [225] H.H. Bauschke, P.L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Springer, 2011.

- [226] J. Hiriart-Urruty, C. Lemaréchal, Fundamentals of Convex Analysis, Springer-Verlag, Grundlehren Text Editions Series, 2001.
- [227] R.T. Rockafellar, R.J.-B. Wets, Variational Analysis, Springer, Berlin, 2004.
- [228] T.R. Rockafellar, Convex Analysis, Princeton University Press, Princeton, NJ, 1970.

# Information Based Learning

# 24

**José C. Príncipe\***, **Badong Chen†** and **Luis G. Sanchez Giraldo\***

\**Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA*

†*Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, P.R. China*

Traditional methods in machine learning and adaptive signal processing mainly exploit second order signal statistics (covariance,  $L_2$  distance, correlation functions, etc.). The emphasis on second order statistics as the choice of optimality criterion is due to the computational simplicity and the optimality under linear and Gaussian assumptions. Although second order statistics are still prevalent today in machine learning community and provide successful engineering solutions to most practical problems, it has become evident that this approach can be improved, especially when data possess non-Gaussian distributions (either fat tails or finite range). Recent studies suggest that machine learning problems that cannot be successfully solved using second order statistics benefit greatly from the use of the information theoretic performance measures. This information based learning has been named information theoretic learning (ITL), and it utilizes information concepts or descriptors from information theory to adapt the learning machines (linear or nonlinear) in both supervised and unsupervised paradigms. In this chapter, we present a brief but comprehensive introduction on general ITL methods to implement learning algorithms with information theoretic criteria. Some basic information theoretic descriptors (or information measures) are introduced, and a unifying learning framework based on these descriptors is presented. We also provide nonparametric kernel estimators for information measures and express them in reproducing kernel Hilbert space (RKHS). The information particle interaction model for learning is also presented. Finally, several application examples are provided, including TDNN training, classification, clustering, image retrieval, and independent component analysis (ICA).

---

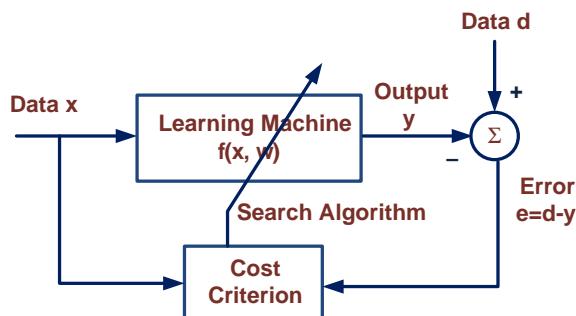
## 1.24.1 Introduction

The concept of learning was first applied to biological organisms and describes the process of revising behavior or understanding through acquisition of new experience from the environment. In the areas of machine learning, learning means to build a model from data with the goal of extracting useful structure contained in the data. The learning process is in essence a procedure of information processing, decreasing data redundancy in the presence of uncertainty and encoding the data into a model. Therefore, learning theory is intrinsically related to information theory, which was first conceptualized by Claude Shannon in the mathematical design of optimal communication systems [1,2]. A general statistical description of learning processes was given in [3], where learning is defined as a process in which the system's subjective entropy or, equivalently, its missing information decreases in time. In [4],

the mathematical concept of information was brought to biologically plausible information processing. The principle of maximum information preservation (Infomax) was applied in self-organization [5]. Further, a unifying framework for learning based on information theoretic criteria has been presented in [6,7].

To extract information from data, we have to build a model, which basically summarizes the process of data generation. In the framework of statistical machine learning, the learning machine goal can be thought of as approximating the a posteriori distribution of the targets given a set of examples (training data). If the joint or conditional probability density functions (PDFs) of the data source can be obtained, the data modeling can be directly implemented, by Bayesian reasoning. Examples of this approach include Bayesian filtering, hidden Markov modeling, and so on. Another approach for statistical learning, which is much simpler computationally, deals with the construction of scalar descriptors of the PDFs, the so called Bayes risks [8]. These descriptors, combined with a parametric data model, can be used to measure the similarity or dependency between the model output and the data. They are often used as cost functions for optimizing the learning systems' parameters. Figure 24.1 shows a general scheme of supervised machine learning. In this figure the cost criterion is, in general, an evaluator of the error's PDF. The goal of the learning is to search the parameter  $\omega$  of the model (learning machine)  $f(x, \omega)$ , such that the discrepancy (measured by the cost criterion) between the model output  $y$  and the desired (teacher) signal  $d$  is minimized. In this scenario, the process of finding parameters from the data is called learning.

The most widely used descriptors of the PDFs are the statistical moments. The combination of the second order moments (variance, correlation, etc.) and the Gaussian assumption in general leads to mathematically convenient and analytically tractable optimal solutions. Familiar examples are the mean square error (MSE) in least-squares linear regression and output variance in principal components analysis (PCA). The second order statistics as a cost criterion, however, may be a poor descriptor of optimality especially in nonlinear and non-Gaussian (e.g., heavy-tail or finite range distributions) situations. Thus selecting a new criterion beyond second order statistics is attractive in both signal processing and machine learning domains. Recent studies indicate that the information theoretic descriptors of entropy and dissimilarity (divergence and mutual information) do not suffer from the limitation of Gaussian assumption, and can improve performance in many realistic scenarios. Combined with nonparametric estimators of entropy and divergence, one can easily develop a variety of new learning algorithms.



**FIGURE 24.1**

Model building through supervised machine learning.

In this chapter, we will provide a brief but comprehensive picture of *information theoretic learning* (ITL), which describes machine learning under a general framework of information theory, and designs new learning costs based on information theoretic descriptors like entropy and mutual information.

## 1.24.2 Information theoretic descriptors

Let's start with a brief introduction of some basic information theoretic descriptors. The concept of information is so rich that a single information measure (or information descriptors) may never be able to quantify information properly. Here we focus mainly on Claude Shannon and Alfred Renyi's definitions of entropy, divergence and mutual information (a special case of divergence).

### 1.24.2.1 Entropy

Given a discrete random variable  $X$  with probability mass function  $P\{X = x_k\} = p_k, k = 1, \dots, n$ , Shannon's entropy is defined by [1, 2]

$$H_S(X) = \sum_{k=1}^n p_k I(p_k), \quad (24.1)$$

where  $I(p_k) = -\log p_k$  is Hartley's amount of information associated with the discrete event  $x_k$  with probability  $p_k$ . This information measure was originally devised by Claude Shannon in 1948 to study the amount of information in a transmitted message. Since learning systems deal, in general, with continuous random variables, we are more interested in information measure of a continuous random variable. Given a continuous random variable  $X$  with PDF  $p(x), x \in C$ , Shannon's differential entropy is defined as

$$H_S(X) = - \int_C p(x) \log p(x) dx. \quad (24.2)$$

Shannon entropy measures the average information (or uncertainty) contained in the probability distribution, and can also be used to measure many other concepts such as diversity, similarity, disorder, randomness, and so on.

A well-known generalization of Shannon entropy is Renyi entropy named after Alfred Renyi [9]. For a continuous random variable  $X$ , order-  $\alpha$  Renyi entropy is defined by ( $\alpha > 0, \alpha \neq 1$ )

$$H_\alpha(X) = \frac{1}{1-\alpha} \log V_\alpha(X), \quad (24.3)$$

where  $V_\alpha(X)$  is the order-  $\alpha$  information potential (IP) [7, 10, 11], given by

$$V_\alpha(X) = \int_C p^\alpha(x) dx = E[p^{\alpha-1}(X)]. \quad (24.4)$$

When  $\alpha = 2$ , we have  $H_2(X) = -\log V_2(X)$ . By L'Hoptital's rule one can easily show that as  $\alpha \rightarrow 1$ , Renyi entropy converges to Shannon entropy. The information potential  $V_\alpha(X)$  can be interpreted as the  $\alpha$ -power of the  $\alpha$ -norm in the PDF space. It was proposed as an optimality criterion for learning

systems (adaptive filters, neural networks, etc.) in [11]. The main reasons are: (1) due to the monotonic property of the logarithm function, optimizing the information potential will be equivalent to optimizing Renyi entropy; (2) the information potential, especially the quadratic information potential ( $\alpha = 2$ ), can easily be estimated from sample data by kernel density estimation method (also referred to as Parzen windowing approach [12, 13]); (3) in general the learning seeks extrema (either minimum or maximum) of the cost function, independently to its actual value, so the dependence on estimation error is decreased.

There is an important optimization principle of entropy, i.e., the maximum entropy (MaxEnt) principle enunciated by Jaynes [14, 15]. According to MaxEnt, among all the distributions that satisfy certain constraints, one should choose the distribution that maximizes the entropy. MaxEnt is a powerful and widely accepted method for statistical inference with incomplete knowledge of the probability distribution.

### 1.24.2.2 Divergence

Information divergence measures the discrepancy between two distributions. The Kullback-Leibler (KL) divergence (or Shannon's relative entropy) between two PDFs  $p_1(x)$  and  $p_2(x)$  is

$$D_{\text{KL}}(p_1 \| p_2) = \int_C p_1(x) \log \frac{p_1(x)}{p_2(x)} dx. \quad (24.5)$$

In statistics, the KL divergence  $D_{\text{KL}}(p_1 \| p_2)$  is a measure of the equivocation of assuming that the distribution is  $p_2$  when the true distribution is  $p_1$ . One can easily show  $D_{\text{KL}}(p_1 \| p_2) \geq 0$ , with equality if and only if  $p_1(x)$  and  $p_2(x)$  are identical almost everywhere. Note that the KL divergence is a measure of the “distance” between the two PDFs, but in general we have  $D_{\text{KL}}(p_1 \| p_2) \neq D_{\text{KL}}(p_2 \| p_1)$ , so it does not obey all the properties of a distance.

Renyi's order-  $\alpha$  divergence between  $p_1(x)$  and  $p_2(x)$  is defined by [16]

$$D_\alpha(p_1 \| p_2) = \frac{1}{\alpha - 1} \log \int_C p_1(x) \left( \frac{p_1(x)}{p_2(x)} \right)^{\alpha-1} dx. \quad (24.6)$$

When  $\alpha \rightarrow 1$ , we have  $\lim_{\alpha \rightarrow 1} D_\alpha(p_1 \| p_2) = D_{\text{KL}}(p_1 \| p_2)$ . In [17], a new Renyi's divergence, called the relative  $\alpha$ -Renyi entropy between  $p_1(x)$  and  $p_2(x)$  is defined as

$$D_{R_\alpha}(p_1 \| p_2) = \log \frac{\left( \int p_2^{\alpha-1}(x) p_1(x) dx \right)^{\frac{1}{(1-\alpha)}} \left( \int p_2^\alpha(x) dx \right)^{\frac{1}{\alpha}}}{\left( \int p_1^\alpha(x) dx \right)^{\frac{1}{\alpha(1-\alpha)}}}. \quad (24.7)$$

Note that this definition is more robust than Renyi's original definition in (24.6) since the denominator in the argument of the log contains an integral.

In machine learning, to simplify the computation of the above divergences, some quadratic divergences are frequently used. These quadratic divergences involve only a simple quadratic form of PDFs. The Euclidean distance (ED) in probability spaces and Cauchy-Schwarz (CS) divergence are popular,

which are defined respectively as [7]

$$D_{\text{ED}}(p_1 \| p_2) = \int_C (p_1(x) - p_2(x))^2 dx, \quad (24.8)$$

$$D_{\text{CS}}(p_1 \| p_2) = -\log \frac{\left(\int_C p_1(x)p_2(x)dx\right)^2}{\int_C p_1^2(x)dx \int_C p_2^2(x)dx}. \quad (24.9)$$

It is worth noting that  $D_{\text{ED}}$  can be expressed in terms of quadratic information potential (QIP):

$$D_{\text{ED}}(p_1 \| p_2) = V_2(p_1) + V_2(p_2) - 2V_2(p_1; p_2) \quad (24.10)$$

where  $V_2(p_1; p_2) \triangleq \int p_1(x)p_2(x)dx$  is the cross information potential (CIP). Further,  $D_{\text{CS}}$  can be rewritten in terms of Renyi's quadratic entropy:

$$D_{\text{CS}}(p_1 \| p_2) = 2H_2(p_1; p_2) - H_2(p_1) - H_2(p_2), \quad (24.11)$$

where  $H_2(p_1; p_2) \triangleq -\log \int p_1(x)p_2(x)dx$  is Renyi's quadratic cross-entropy.

### 1.24.2.3 Mutual information

Mutual information measures the degree of dependence between two random variables, and it is actually a special case of divergence. Let  $X_1$  and  $X_2$  be two random variables with joint PDF  $p_{X_1 X_2}(x_1, x_2)$  and marginal PDFs  $p_{X_1}(x_1)$  and  $p_{X_2}(x_2)$ , the mutual information between  $X_1$  and  $X_2$  is defined as the KL divergence between  $p_{X_1 X_2}(x_1, x_2)$  and  $p_{X_1}(x_1)p_{X_2}(x_2)$ , that is

$$\begin{aligned} I(X_1, X_2) &= D_{\text{KL}}(p_{X_1 X_2} \| p_{X_1} p_{X_2}) \\ &= \int_C p_{X_1 X_2}(x_1, x_2) \log \frac{p_{X_1 X_2}(x_1, x_2)}{p_{X_1}(x_1)p_{X_2}(x_2)} dx_1 dx_2. \end{aligned} \quad (24.12)$$

Clearly, mutual information  $I(X_1, X_2)$  is symmetric and always greater than zero, which vanishes if and only if  $X_1$  and  $X_2$  are statistically independent.

Mutual information (24.12) can alternatively be expressed as

$$I(X_1, X_2) = H_S(X_1) - H_S(X_1|X_2), \quad (24.13)$$

where  $H_S(X_1|X_2)$  is the conditional Shannon entropy of  $X_1$  given  $X_2$ , defined as

$$H_S(X_1|X_2) = - \int_C p_{X_1 X_2}(x_1, x_2) \log p_{X_1|X_2}(x_1|x_2) dx_1 dx_2, \quad (24.14)$$

where  $p_{X_1|X_2}(x_1|x_2)$  denotes the conditional PDF of  $X_1$  given  $X_2$ . As one can see from (24.13), mutual information quantifies the reduction of uncertainty in  $X_1$  after observing  $X_2$ .

In a similar way one can also define Renyi's order-  $\alpha$  mutual information  $I_\alpha(X_1, X_2)$ , Euclidean distance mutual information  $I_{\text{ED}}(X_1, X_2)$ , and Cauchy-Schwartz mutual information  $I_{\text{CS}}(X_1, X_2)$ , that is

$$\begin{cases} I_\alpha(X_1, X_2) = D_\alpha(p_{X_1 X_2} \| p_{X_1} p_{X_2}), \\ I_{\text{ED}}(X_1, X_2) = D_{\text{ED}}(p_{X_1 X_2} \| p_{X_1} p_{X_2}), \\ I_{\text{CS}}(X_1, X_2) = D_{\text{CS}}(p_{X_1 X_2} \| p_{X_1} p_{X_2}). \end{cases} \quad (24.15)$$

The previous information measures are all defined based on PDFs (for continuous random variables). Some researchers also propose to define information measures using distribution functions or survival functions. For example, the cumulative residual entropy (CRE) of random variable  $X$  is defined by [18]

$$\mathcal{E}(X) = - \int_{\mathbb{R}_+} \bar{F}_{|X|}(x) \log \bar{F}_{|X|}(x) dx, \quad (24.16)$$

where  $\bar{F}_{|X|}(x) = P(|X| > x)$  is the survival function of  $|X|$ . In a recent paper [19], we define the order  $\alpha$  survival information potential (SIP) as ( $\alpha > 0$ )

$$S_\alpha(X) = \int_{\mathbb{R}_+} \bar{F}_{|X|}^\alpha(x) dx. \quad (24.17)$$

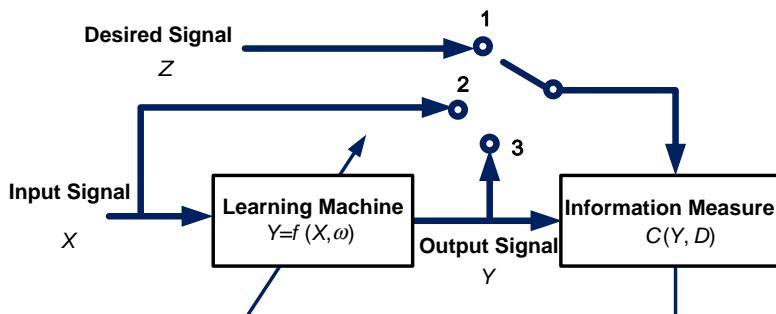
The survival information potential (24.16) is just defined by replacing the PDF with the survival function (of an absolute value transformation of  $X$ ) in the original information potential (24.4). This new definition seems more natural and reasonable, because the survival function is more regular and general than the PDF.

### 1.24.3 Unifying information theoretic framework for machine learning

Information theoretic methodology provides a unifying framework for both supervised and unsupervised machine learning [7]. Figure 24.2 shows a unifying view of learning based on information theoretic cost criteria. In Figure 24.2, the cost  $C(Y, D)$  denotes generally the information measure (divergence or mutual information) between  $Y$  and  $D$ , where  $Y$  is the output signal of the model (learning machine), and  $D$  depends on which position the switch is in. Information theoretic learning (ITL) is then to adjust the parameters  $\omega$  such that the cost  $C(Y, D)$  is optimized (minimized or maximized).

#### 1.24.3.1 Switch in position 1

When the switch is in position 1, the cost includes the model output  $Y$  and an external desired signal  $Z$ . Then the learning is supervised, and the goal is to make the output signal and the desired signal



**FIGURE 24.2**

Information theoretic learning framework.

as “close” as possible. In this case the learning can be categorized into two categories: (1) filtering (or regression) and classification; (2) feature extraction.

#### 1.24.3.1.1 Filtering and classification

In tradition filtering and classification, the cost function is in general the mean square error (the squares error loss) or misclassification error rate (the 0–1 loss). In ITL framework, the problem can be formulated as minimizing the divergence or maximizing the mutual information between output  $Y$  and the desired response  $Z$ , that is

$$\min_{\omega} D_{\text{KL}}(Y \| Z) = \int p_Y(x) \log \frac{p_Y(x)}{p_Z(x)} dx, \quad (24.18)$$

$$\max_{\omega} I(Y, Z) = \max_{\omega} \{H_S(Y) - H_S(Y|Z)\}. \quad (24.19)$$

The above optimization criteria are based purely on PDFs of the output and the desired signal; therefore we don't require the same number of desired and output data to calculate the errors.

For filtering and classification problems, another important ITL criterion is the *Minimum Error Entropy* (MEE) criterion [20–23], which aims at minimizing the entropy of the error between the output and the desired responses:

$$\min_{\omega} H_S(e) = \min_{\omega} \left\{ - \int p(e) \log p(e) de \right\}, \quad (24.20)$$

where  $e = Z - Y$  is the error. As entropy measures the average uncertainty or dispersion of a random variable, its minimization makes the error concentrated. Different from conventional Bayesian risks, like MSE, the “loss function” in MEE is  $-\log p(\cdot)$ , which is directly related to the error's PDF, transforming nonlinearly the error by its own PDF. If using Renyi's entropy as the cost, the optimization problem in (24.20) becomes

$$\begin{aligned} \min_{\omega} H_{\alpha}(e) &= \min_{\omega} \left\{ \frac{1}{1-\alpha} \log V_{\alpha}(e) \right\} \\ &\Leftrightarrow \begin{cases} \min_{\omega} V_{\alpha}(e) & \text{if } \alpha < 1, \\ \max_{\omega} V_{\alpha}(e) & \text{if } \alpha > 1, \end{cases} \end{aligned} \quad (24.21)$$

where  $V_{\alpha}(e)$  denotes error's information potential,  $V_{\alpha}(e) = \int p^{\alpha}(e) de$ .

#### 1.24.3.1.2 Feature extraction

In machine learning, when the input data are too large and the dimensionality is very high, it is necessary to transform nonlinearly the input data into a reduced representation set of features. Feature extraction (or feature selection) involves reducing the amount of resources required to describe a large set of data accurately. The feature set will extract the relevant information from the input in order to perform the desired task using the reduced representation instead of the full size input. Suppose the desired signal is the class label, then an intuitive cost for feature extraction should be some measure of “relevance” between the projection outputs (features) and the labels. In ITL, this problem can be solved

by maximizing the mutual information between the output  $Y$  and the label  $C$ :

$$\max_{\omega} I(Y, C) = \max_{\omega} \{H_S(Y) - H_S(Y|C)\}. \quad (24.22)$$

There are many other mutual information based criteria for feature extraction. A summary of these criteria can be found in [24], where a unifying framework for information theoretic feature selection has been presented. The use of mutual information criterion for feature extraction can be justified using Fano's inequality, according to which one should maximize the mutual information to improve the lower bound on the achievable probability of error [25].

### 1.24.3.2 Switch in position 2

When the switch is in position 2, the learning is in essence unsupervised because there is no external signal besides the input and output signals. In this situation, the well-known optimization principle is the *Maximum Information Transfer*, which aims to maximize the mutual information between the original input data and the output of the system:

$$\max_{\omega} I(Y, X) = \max_{\omega} \int p_{XY}(y, x) \log \frac{p_{XY}(y, x)}{p_Y(y)p_X(x)} dy dx. \quad (24.23)$$

This principle is also known as the principle of maximum information preservation (Infomax) described by Linsker, which is a widely accepted optimization principle for neural networks, self-organization, and other information processing systems [5]. Infomax is also related to an algorithm for independent component analysis (ICA) described by Bell and Sejnowski in 1995 [26].

Another information optimization principle for unsupervised learning (clustering, principal curves, vector quantization, etc.) is the *Principle of Relevant Information* (PRI), which can be formulated as [27–29]

$$\min_{\omega} \{H_S(Y) + \lambda D_{KL}(Y||X)\}, \quad (24.24)$$

where  $X$  denotes the original data set,  $Y$  is the compressed version of the original data, and  $\lambda$  is the trade-off parameter. The basic idea of PRI is to minimize the data redundancy (entropy term) while preserving the similarity to the original data (divergence term). Using Renyi's definition of entropy and divergence, (24.24) can be reformulated as

$$\min_{\omega} \{H_{\alpha}(Y) + \lambda D_{\alpha}(Y||X)\}. \quad (24.25)$$

To simplify the computation, one often uses Renyi's quadratic entropy to measure the redundancy, and Cauchy-Schwarz divergence to measure the distortion, which yields

$$\min_{\omega} \{H_2(Y) + \lambda D_{CS}(Y||X)\}. \quad (24.26)$$

### 1.24.3.3 Switch in position 3

When the switch is in position 3, the only source of data is the model output, which in this case is in general assumed multidimensional. Typical examples of this case include independent component analysis (ICA), clustering, output entropy optimization, and so on:

*Independent component analysis:* ICA is an unsupervised technique aiming to reduce the redundancy between components of the system output [30]. Given a nonlinear multiple-input-multiple-output (MIMO) system  $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\omega})$ , the nonlinear ICA usually optimizes the parameter vector  $\boldsymbol{\omega}$  such that the mutual information between the components of  $\mathbf{y}$  is minimized:

$$\min_{\boldsymbol{\omega}} I(\mathbf{y}) = \int p(y_1 \cdots y_m) \log \frac{p(y_1 \cdots y_m)}{\prod_{d=1}^m p(y_d)} dy. \quad (24.27)$$

*Clustering:* Clustering (or clustering analysis) is a common technique for statistical data analysis used in machine learning, pattern recognition, bioinformatics, etc. The goal of clustering is to divide the input data into groups (called clusters) so that the objects in the same cluster are more “similar” to each other than to those in other clusters, and different clusters are defined as compactly and distinctly as possible. Information theoretic measures like entropy and divergence are frequently used as an optimization criterion for clustering. For example, in the case of two clusters, one can use the symmetrized KL divergence as the cost:

$$\max_{\boldsymbol{\omega}} D_{\text{KL}}(p_1(\mathbf{y}) \| p_2(\mathbf{y})) + D_{\text{KL}}(p_2(\mathbf{y}) \| p_1(\mathbf{y})). \quad (24.28)$$

*Output entropy optimization:* If the switch is in position 3, one can also optimize (minimize or maximize) the entropy at system output (usually subject to some constraint on the weight norm or nonlinear topology) so as to capture the underlying structure in high dimensional data:

$$\begin{cases} \min_{\boldsymbol{\omega}} \text{ or } \max_{\boldsymbol{\omega}} H(\mathbf{y}) = - \int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}, \\ \text{s.t. } E[h_i(\mathbf{y})] = \alpha_i, \quad i = 1, \dots, d. \end{cases} \quad (24.29)$$

This optimization formulation is useful in blind equalization, nonlinear PCA, ICA and novelty filtering [30,31].

#### 1.24.3.4 Switch simultaneously in position 1 and 2

In Figure 24.2 the switch can be simultaneously in position 1 and 2. In this case, the cost has access to input data  $X$ , output data  $Y$ , and the desired or reference data  $Z$ . A well-known example is the *Information Bottleneck* (IB) method, introduced by Naftali Tishby et al. [32,33]. Given a random variable  $X$  and an observed relevant variable  $Z$ , and assuming that the joint distribution between  $X$  and  $Z$  is known, the IB method aims to compress  $X$  and try to find the best tradeoff between: (1) the minimization of mutual information between  $X$  and its compressed version  $Y$ ; and (2) the maximization of mutual information between  $Y$  and the relevant variable  $Z$ . Then the IB method minimizes the following cost:

$$\min_{\boldsymbol{\omega}} I(X, Y) - \beta I(Y, Z), \quad (24.30)$$

where  $\beta$  is the trade-off parameter. The basic idea in IB is to find a reduced representation of  $X$  while preserving the information of  $X$  with respect to another variable  $Z$ .

### 1.24.4 Nonparametric information estimators

To implement the previous ITL learning methods, one should evaluate the information theoretic costs, such as entropy and divergence. In practice, the data distributions are usually unknown, and the analytical evaluation of these measures is not possible. Thus we have to estimate ITL costs using sample data. There are two simple ways: (1) one way is to estimate the underlying distributions based on available samples, and plug the estimated distributions directly into the cost functions to obtain the information estimators (the so called “*plug-in estimators*”); and (2) another way is to substitute the estimated distributions into the *sample mean approximation* of the information measures (approximating the expected values by their sample mean). In the literature there are three possible techniques for estimating the PDF of a random variable based on its sample data: parametric, semi-parametric, and nonparametric. Accordingly, there are also parametric, semi-parametric, and nonparametric approaches for estimating information theoretic quantities. We refer the reader to [34] for a review of entropy estimation methods.

In the following, we focus only on the nonparametric kernel approach (Parzen windowing approach) that has been widely and successfully applied in ITL learning. Further, we mainly discuss entropy estimators. The estimation of other information measures is similar.

Given a set of independent and identically distributed (i.i.d.) samples  $\{x_1, \dots, x_N\}$  drawn from  $p(x)$ , the kernel density estimation (KDE) for  $p(x)$ , assuming a fixed-size kernel function  $\kappa_\sigma(\cdot)$  for simplicity, is given by [12, 13]

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x - x_i). \quad (24.31)$$

In general the kernel function satisfies  $\kappa_\sigma(x) \geq 0$ , and  $\int \kappa_\sigma(x) dx = 1$ , and hence  $\hat{p}(x)$  is still a PDF. Further, to make the estimator smooth, the kernel function is usually continuous and differentiable (and preferably symmetric and unimodal). The most widely used kernel function is the Gaussian function:

$$\kappa_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right). \quad (24.32)$$

The kernel size (or bandwidth) for the Gaussian kernel can be optimized using maximum likelihood (ML) principle, or selected according to rules-of-thumb, such as Silverman’s rule [13].

For fixed kernel size  $\sigma$ , we have  $\lim_{N \rightarrow \infty} \hat{p}(x) = p(x) * \kappa_\sigma(x)$  (here, “ $*$ ” denotes the convolution operator). Using a suitable annealing rate for the kernel size, the KDE can be asymptotically unbiased and consistent. Specifically, if  $\lim_{N \rightarrow \infty} \sigma(N) = 0$ , and  $\lim_{N \rightarrow \infty} N\sigma(N) = \infty$ , then  $\lim_{N \rightarrow \infty} \hat{p}(x) = p(x)$  in probability [35].

Substitute (24.31) in Shannon’s entropy expression (24.2), we get the following plug-in estimator:

$$\widehat{H}_S(X) = -\frac{1}{N} \int \sum_{i=1}^N \kappa_\sigma(x - x_i) \log \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x - x_i) \right) dx. \quad (24.33)$$

To avoid the evaluation of integral, one can alternatively substitute (24.31) in the sample mean estimator of Shannon entropy and obtain

$$\hat{H}_S(X) = -\frac{1}{N} \sum_{j=1}^N \log \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right). \quad (24.34)$$

Similarly, substituting (24.31) in Renyi's entropy expression, we have

$$\hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \int \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x - x_i) \right)^\alpha dx. \quad (24.35)$$

Further, the sample mean estimator of Renyi's entropy will be

$$\begin{aligned} \hat{H}_\alpha(X) &= \frac{1}{1-\alpha} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \\ &= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1}. \end{aligned} \quad (24.36)$$

In (24.35), if  $\alpha = 2$  and the kernel function is Gaussian function, we can derive

$$\begin{aligned} \hat{H}_2(X) &= -\log \int_{-\infty}^{\infty} \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x - x_i) \right)^2 dx \\ &= -\log \frac{1}{N^2} \int_{-\infty}^{\infty} \left( \sum_{i=1}^N \sum_{j=1}^N \kappa_\sigma(x - x_j) \kappa_\sigma(x - x_i) \right)^2 dx \\ &= -\log \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \int_{-\infty}^{\infty} \kappa_\sigma(x - x_j) \kappa_\sigma(x - x_i) dx \\ &= -\log \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_j - x_i) \right). \end{aligned} \quad (24.37)$$

This result comes from the fact that the integral of the product of two Gaussian functions can be exactly evaluated as the value of the Gaussian function computed at the difference of the arguments and whose variance is the sum of the variances of the two original Gaussian functions. As one can see, the plug-in estimator (24.37) is identical to the sample mean estimator of quadratic Renyi's entropy but with kernel size  $\sqrt{2}\sigma$  instead of  $\sigma$ . The argument of the logarithm in (24.37) is the estimator of quadratic information potential (QIP):

$$\hat{V}_2(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_j - x_i). \quad (24.38)$$

Based on the estimator of QIP, one can easily estimate other quadratic information measures ( $D_{ED}$ ,  $D_{CS}$ ,  $I_{ED}$ ,  $I_{CS}$ ). For example, the Cauchy-Schwarz divergence (24.9) can be estimated as

$$\widehat{D}_{CS}(p_1 \| p_2) = \log \frac{\widehat{V}_{p_1} \widehat{V}_{p_2}}{\widehat{V}_c^2}, \quad (24.39)$$

where

$$\widehat{V}_{p_1} = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_1(j) - x_1(i)), \quad (24.40)$$

$$\widehat{V}_{p_2} = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_2(j) - x_2(i)), \quad (24.41)$$

$$\widehat{V}_c = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_1(j) - x_2(i)), \quad (24.42)$$

where  $\{x_1(i)\}_{i=1}^N$  and  $\{x_2(i)\}_{i=1}^N$  are samples from  $p_1$  and  $p_2$ , respectively,  $\widehat{V}_c$  is the cross information potential estimator.

There are some nice properties of the nonparametric information estimators. For example, the entropy estimators possess the same location of the extremum (maximum or minimum) as the actual entropy [10]. The following theorem shows that under a certain condition, the minimum value of entropy estimators occurs when sample data are related to a  $\delta$  distribution.

**Theorem 1.** *If the kernel function  $\kappa_\sigma(\xi)$  achieves its maximum value at  $\xi = 0$ , then the minimum value of the entropy estimator in (24.36) is achieved when all sample data are equal to each other, that is  $x_1 = \dots = x_N = c$ .*

The above result can be easily proved. Actually, for  $\alpha > 1$ , we have

$$\sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \leq \sum_{j=1}^N \left( \sum_{i=1}^N \max_x \kappa_\sigma(x) \right)^{\alpha-1} = N^\alpha \kappa_\sigma^{\alpha-1}(0). \quad (24.43)$$

And hence

$$\widehat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \geq -\log \kappa_\sigma(0). \quad (24.44)$$

When  $x_1 = \dots = x_N = c$ , the equality will hold. The proof for the case  $\alpha < 1$  is similar.

One can further prove that if the kernel function is continuous, differentiable, symmetric, and unimodal, then the global minimum in the above theorem will be smooth, that is, it has a zero gradient and a positive semi-definite Hessian matrix [10]. In adaptive learning using numerical optimization techniques, it is crucial that the global optimum be a smooth point in the weight space with zero-gradient.

## 1.24.5 Reproducing kernel Hilbert space framework for ITL

ITL learning is closely related to kernel methods and Reproducing Kernel Hilbert Space (RKHS) [36–38]. In this section, we present a RKHS framework for ITL. In fact, we can uniquely determine a RKHS using the symmetric non-negative definite kernel function defined as the *cross information potential* (CIP).

### 1.24.5.1 RKHS for ITL

Let  $\mathcal{E}$  be the set of all square integrable one-dimensional PDFs over the real numbers; that is,  $p_i(x) \in \mathcal{E}, \forall i \in \mathcal{I}$ , where  $\int p_i(x)^2 dx < \infty$  and  $\mathcal{I}$  is an index set. Here, for simplicity, we focus on one dimensional PDFs, the extension to multi-dimensions is straightforward. Then one can form a linear manifold:  $\{\sum_{i \in \mathcal{K}} \alpha_i p_i(x)\}$  for any  $\mathcal{K} \subseteq \mathcal{I}$ , and  $\alpha_i \in \mathbb{R}$ . Let  $L_2(\mathcal{E})$  be the Hilbert space that consists of all linear combinations of PDFs and their limit points, and is equipped with inner product:

$$\langle f_i(x), f_j(x) \rangle_{L_2} = \int f_i(x) f_j(x) dx \quad \forall f_i, f_j \in \mathcal{E}. \quad (24.45)$$

Note that the above inner product is exactly the CIP between  $f_i$  and  $f_j$ . The Hilbert space  $L_2(\mathcal{E})$  is not a RKHS since the inner product is not reproducing in  $L_2(\mathcal{E})$ ; that is, the evaluation of any element in  $L_2(\mathcal{E})$  cannot be reproduced via the inner product between two functions in  $L_2(\mathcal{E})$ . However, one can use the inner product in (45) to induce a RKHS that is congruent with  $L_2(\mathcal{E})$ .

Let  $v(f_i, f_j)$  be a bivariate function on the set  $\mathcal{E}$ , defined as

$$v(f_i, f_j) = \int f_i(x) f_j(x) dx \quad \forall f_i, f_j \in \mathcal{E}. \quad (24.46)$$

It can be easily verified that the function  $v(f_i, f_j)$  is symmetric and non-negative definite, and is a Mercer kernel function. According to the Moore-Aronszajn theorem [39,40], there is a unique RKHS, denoted by  $\mathcal{H}_v$ , associated with  $v(f_i, f_j)$ . One can construct the RKHS  $\mathcal{H}_v$  bottom-up. By Mercer's theorem [41], function  $v(f_i, f_j)$  has eigen-decomposition:

$$v(f_i, f_j) = \sum_{k=1}^{\infty} \lambda_k \psi_k(f_i) \psi_k(f_j), \quad (24.47)$$

where  $\{\psi_k(f_i), k = 1, 2, \dots\}$  and  $\{\lambda_k, k = 1, 2, \dots\}$  are sequences of eigenfunctions and corresponding eigenvalues of  $v(f_i, f_j)$ , respectively. The above series converges absolutely and uniformly on  $\mathcal{E} \times \mathcal{E}$ . We then define a space  $\mathcal{H}_v$  consisting of all functionals  $G(\cdot)$  whose evaluation for any given PDF  $f_i \in \mathcal{E}$  is

$$G(f_i) = \sum_{k=1}^{\infty} \lambda_k a_k \psi_k(f_i), \quad (24.48)$$

where the sequence  $\{a_k, k = 1, 2, \dots\}$  satisfies  $\sum_{k=1}^{\infty} \lambda_k a_k^2 < \infty$ . The inner product of two functionals in  $\mathcal{H}_v$  is defined by

$$\langle G, F \rangle_{\mathcal{H}_v} = \sum_{k=1}^{\infty} \lambda_k a_k b_k, \quad (24.49)$$

where  $F(f_i) = \sum_{k=1}^{\infty} \lambda_k b_k \psi_k(f_i)$ ,  $\sum_{k=1}^{\infty} \lambda_k b_k^2 < \infty$ . Now it can be verified that the space  $\mathcal{H}_v$  induced by the kernel function  $v(f_i, f_j)$  is indeed a RKHS, and this kernel function is a reproducing kernel because it satisfies:

1.  $\forall f_i \in \mathcal{E}$ ,  $v(f_i, f_j)$  as a functional of  $f_j$  belongs to  $\mathcal{H}_v$ ,
2.  $\forall G \in \mathcal{H}_v$ ,  $\langle G, v(f_i, .) \rangle_{\mathcal{H}_v} = G(f_i)$  (the so called reproducing property).

By the reproducing property, one can rewrite the kernel function  $v(f_i, f_j)$  as the inner product:

$$v(f_i, f_j) = \langle v(f_i, .), v(f_j, .) \rangle_{\mathcal{H}_v}, \quad v(f_i, .) : f_i \mapsto \sqrt{\lambda_i} \psi_i(f_i), \quad k = 1, 2, \dots \quad (24.50)$$

The reproducing kernel nonlinearly maps the original PDF  $f_i$  into the RKHS  $\mathcal{H}_v$ .

It is worth noting that, although the RKHS  $\mathcal{H}_v$  and the Hilbert space  $L_2(\mathcal{E})$  are much different, they are congruent with each other. Actually one can prove that there exists a one-to-one congruence mapping  $\Psi$  from  $\mathcal{H}_v$  to  $L_2(\mathcal{E})$  such that  $\Psi(v(f_i, .)) = f_i$  [37].

### 1.24.5.2 ITL cost functions in RKHS

The ITL costs can be reformulated in RKHS  $\mathcal{H}_v$ . First of all, the cross information potential (CIP) can be expressed as the inner product in  $\mathcal{H}_v$ :

$$\int p_1(x) p_2(x) dx = \langle v(p_1, .), v(p_2, .) \rangle_{\mathcal{H}_v}. \quad (24.51)$$

The inner product quantifies similarity between two functionals and this agrees with the definition of cross information potential. Then the quadratic information potential (QIP) can also be specified as the inner product between a functional and itself:

$$\int p^2(x) dx = \langle v(p, .), v(p, .) \rangle_{\mathcal{H}_v} = \|v(p, .)\|_{\mathcal{H}_v}^2. \quad (24.52)$$

Therefore, maximizing QIP in ITL turns out to be maximization of the norm square in  $\mathcal{H}_v$ . Using (24.52), we can rewrite Renyi's quadratic entropy as

$$H_2(X) = -\log \|v(p, .)\|_{\mathcal{H}_v}^2. \quad (24.53)$$

Based on the reformulations of CIP (24.51) and QIP (24.52) one can easily rewrite other quadratic information measures in terms of operations on functional in  $\mathcal{H}_v$ . For example, the Cauchy-Schwarz divergence measure can be expressed as

$$D_{CS}(p_1 \| p_2) = -2 \log \left( \frac{\langle v(p_1, .), v(p_2, .) \rangle_{\mathcal{H}_v}}{\sqrt{\|v(p_1, .)\|_{\mathcal{H}_v}^2 \|v(p_2, .)\|_{\mathcal{H}_v}^2}} \right) = -2 \log (\cos \theta), \quad (24.54)$$

where  $\theta$  is the angle in  $\mathcal{H}_v$  between two functionals  $v(p_1, .)$  and  $v(p_2, .)$ . Therefore the RKHS  $\mathcal{H}_v$  provides an insightful geometric interpretation for CS-divergence; that is, the argument of the log of CS-divergence truly measures the separation (angle) between two functional vectors in  $\mathcal{H}_v$ .

### 1.24.5.3 Information estimator in RKHS

The estimators of information theoretic quantities can also be reinterpreted in RKHS (kernel space). Consider the nonparametric kernel estimator of quadratic information potential (QIP) in (24.38). The Gaussian kernel  $\kappa_{\sqrt{2}\sigma}(.)$  is a Mercer kernel and can be written as an inner product in a RKHS:

$$K_{\sqrt{2}\sigma}(x_i - x_j) = k(x_i - x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle_{\mathcal{H}_\kappa}, \quad (24.55)$$

where  $\varphi(x) = \kappa(x, .)$  defines the nonlinear mapping between input space and kernel space  $\mathcal{H}_\kappa$ . Hence the estimated QIP in (38) can be expressed in terms of an inner product in kernel space:

$$\begin{aligned} \widehat{V}_2(x) &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_j - x_i) \\ &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \langle \varphi(x_i), \varphi(x_j) \rangle_{\mathcal{H}_\kappa} \\ &= \left\langle \frac{1}{N} \sum_{j=1}^N \varphi(X_i), \frac{1}{N} \sum_{j=1}^N \varphi(X_i) \right\rangle_{\mathcal{H}_\kappa} \\ &= \mathbf{m}^T \mathbf{m} = \|\mathbf{m}\|^2. \end{aligned} \quad (24.56)$$

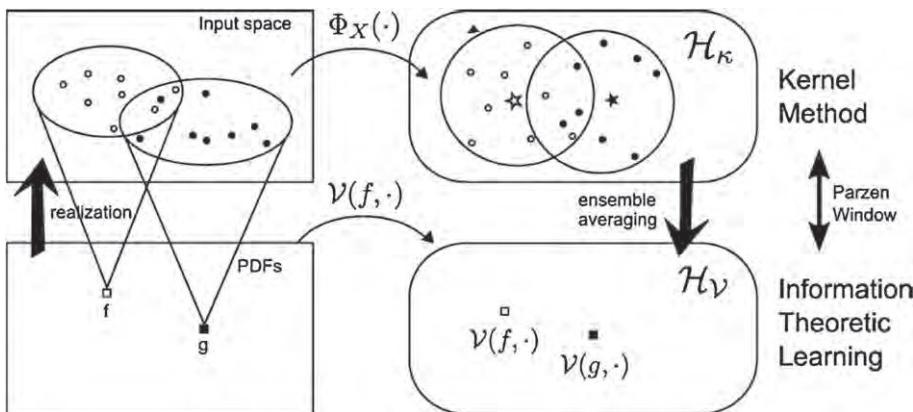
It turns out that the estimated QIP may be expressed as the squared norm of the mean vector of the data in kernel space. Other quadratic information estimators may also be expressed in terms of mean vectors in the kernel space. Actually one can express the estimated ED-divergence and CS-divergence as

$$\widehat{D}_{\text{ED}}(p_1 \| p_2) = \|\mathbf{m}_1\|^2 - 2\mathbf{m}_1^T \mathbf{m}_2 + \|\mathbf{m}_2\|^2 = \|\mathbf{m}_1 - \mathbf{m}_2\|^2, \quad (24.57)$$

$$\widehat{D}_{\text{ED}}(p_1 \| p_2) = -2 \log \left( \frac{\mathbf{m}_1^T \mathbf{m}_2}{\|\mathbf{m}_1\| \|\mathbf{m}_2\|} \right) = -2 \log (\cos(\langle \mathbf{m}_1, \mathbf{m}_2 \rangle)), \quad (24.58)$$

where  $\mathbf{m}_1$  and  $\mathbf{m}_2$  are kernel space mean vectors of the data points drawn from  $p_1(x)$  and  $p_2(x)$ , respectively. From (24.57) and (24.58), one can see the ED-divergence estimator measures the square of the norm of the difference vector between  $\mathbf{m}_1$  and  $\mathbf{m}_2$ , while the argument of the log of CS-divergence estimator measures the cosine of the angle between mean vectors.

Figure 24.3 illustrates the relationship between the two RKHS  $\mathcal{H}_v$  and  $\mathcal{H}_\kappa$ . The information quantities in  $\mathcal{H}_v$  can be estimated by the mean operator of the projected functionals in  $\mathcal{H}_\kappa$ , which was effectively derived with Parzen's nonparametric PDF estimator employed in ITL cost functions, and with a Mercer kernel (symmetric and non-negative definite kernel) as the Parzen window.

**FIGURE 24.3**

The relationship between  $\mathcal{H}_V$  and  $\mathcal{H}_\kappa$  (from [37]).

### 1.24.6 Information particle interaction for learning from samples

In training of a multilayer network such as multilayer perceptrons (MLPs), the output (or output error) is back-propagated through the layers to determine the gradient update rule for weight vector [42]. If the learning is supervised, the mean square error (MSE) is widely used as an adaptation cost. The simple LMS algorithm is a special case of back-propagation (BP) for single layer network with linear processing elements. The conventional BP algorithms can be readily extended to ITL costs. Under ITL costs (e.g., error entropy), the gradient of the weight vector can be calculated by the chain rule of partial derivatives as: (1) sensitivity of the output of the adaptive network with respective to its weights, and (2) sensitivity of the ITL cost on the values of the samples. With the kernel estimator in Renyi's entropy, the latter sensitivity has an interesting analogy with physical particles interacting on an *information potential field* [43,44]. This analogy has its roots in the link between Renyi's entropy and the norms of the PDF of the data. Actually, since the kernels in PDF estimation are positive functions that decay with the distance between samples, we can think that one kernel placed on a sample creates a potential field in the same place, just like physical particles create a gravity field in space. In our case, however, the law of interaction is dictated by the kernel shape. The potential field in the sample space is thus an approximation of the PDF that generates the data. In this context, samples can be named as *information particles* and they interact in the information potential field of the PDF through *information forces* [43].

The kernel estimator of quadratic information potential can alternatively be expressed as the average sum of interactions from each sample in the sample set, that is

$$\widehat{V}_2(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa_{\sqrt{2}\sigma}(x_j - x_i)$$

$$\begin{aligned}
&= \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_{\sqrt{2}\sigma}(x_j - x_i) \right) \\
&= \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \widehat{V}_2(x_j, x_i) \right) \\
&= \frac{1}{N} \sum_{j=1}^N \widehat{V}_2(x_j),
\end{aligned} \tag{24.59}$$

where  $\widehat{V}_2(x_j, x_i) = \kappa_{\sqrt{2}\sigma}(x_j - x_i)$ , and  $\widehat{V}_2(x_j) = \frac{1}{N} \sum_{i=1}^N \widehat{V}_2(x_j, x_i)$ . Here the term  $\widehat{V}_2(x_j, x_i)$  measures the contribution of sample  $x_i$  to the potential of  $x_j$ , and the quantity  $\widehat{V}_2(x_j)$  measures the potential field in the space location occupied by the sample  $x_j$  due to all the other samples. The information potential is thus just the average value of the information potential field of the sample set (hence the name).

The derivative of the information potential with respect to the position of sample  $x_j$  can be easily evaluated as

$$\frac{\partial}{\partial x_j} \widehat{V}_2(X) = \frac{1}{N} \sum_{j=1}^N \widehat{F}_2(x_j), \quad \widehat{F}_2(x_j) = \frac{1}{N} \sum_{i=1}^N \widehat{F}_2(x_j, x_i), \tag{24.60}$$

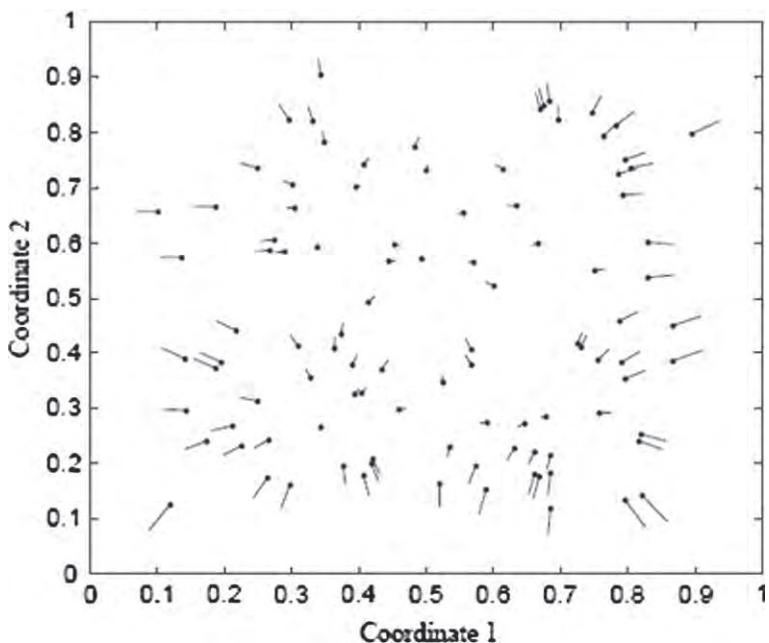
where  $\widehat{F}_2(x_j) = \frac{\partial}{\partial x_j} \widehat{V}_2(x_j)$ ,  $\widehat{F}_2(x_j, x_i) = \frac{\partial}{\partial x_j} \widehat{V}_2(x_j, x_i)$ . These two quantities are named the (*total*) *information force* acting on sample  $x_j$  and the *information force* on sample  $x_j$  due to sample  $x_i$ , respectively. Note that the concepts of information potential fields and information forces can be generalized to other information measures, such as quadratic divergence and quadratic mutual information [7].

In adaptive system training with ITL costs, the system parameters can be adapted by information forces created among the pair-wise interactions in the sample set. In supervised learning, if the error samples are generated by an MLP with weights  $\boldsymbol{\omega}$ , the gradient of the information potential with respect to these weights would be [45]

$$\frac{\partial}{\partial \boldsymbol{\omega}} \widehat{V}_2(e) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \widehat{F}_2(x_j, x_i) \left( \frac{\partial x_i}{\partial \boldsymbol{\omega}} - \frac{\partial x_j}{\partial \boldsymbol{\omega}} \right). \tag{24.61}$$

Thus, for the adaptation of an MLP, the principle of error back-propagation can be extended to information force back-propagation by simply substituting the injected error of the MSE with the information force  $\widehat{F}_2(x_j, x_i)$ .

Figure 24.4 shows a snapshot of a two-dimensional entropy maximization scenario, where the particles are bounded to within a unit square and interact under the quadratic force definition with Gaussian kernel choice. The objective is to maximize the entropy of the sample ensemble, and hence the forces become repulsive. Given a set of randomly spaced samples in the unit square, when the forces acting on each sample are evaluated, it becomes clear that the information particles are pushed by the other particles in order to move along the direction of maximal entropy. In Figure 24.4, the lines attached to each sample are vectors that display intensity and point to the direction of change.

**FIGURE 24.4**

A snapshot of the locations of the information particles and the instantaneous quadratic information forces acting on them to maximize the joint entropy in the two-dimensional unit square (from [43]).

## 1.24.7 Illustrative examples

### 1.24.7.1 MEE based adaptive system training

*Back Propagation for MEE:* The entropy of the error can be employed as a robust cost function to learn the parameters of function approximation. Below, we derive a back propagation procedure to adjust the parameters of a feedforward network based on the error entropy. In particular, we want to minimize the Renyi's second order entropy of the prediction error in a time series using a time delay neural network as predictor. A sliding window of size  $L$  is utilized to compute the estimation of the error information potential. Suppose we have an upcoming example at time  $t$ , the estimate of the information potential is given by,

$$\widehat{V}_E(t) = \sum_{i,j=t-L+1}^t \kappa_\sigma(\|\mathbf{e}(i) - \mathbf{e}(j)\|). \quad (24.62)$$

Although, (24.62) is conceptually the right cost, the computation of the gradient for multiple output functions is rather cumbersome. Therefore a modified version of this cost is introduced,

$$\widehat{V}_E(t) = \sum_{i,j=t-L+1}^t \sum_{k=1}^{M_p} \kappa_\sigma(\mathbf{e}_k(i) - \mathbf{e}_k(j)). \quad (24.63)$$

The gradient equations with respect to the network parameters of the output layer  $p$  are:

$$\frac{\partial \widehat{V}_E(t)}{\partial \mathbf{w}_{(p)_k}} = \frac{1}{\sigma^2 L^2} \sum_{j=t-L+1}^t \kappa_\sigma(e_k(i) - e_k(j))(e_k(i) - e_k(j)) \begin{bmatrix} f'_p \left( \mathbf{w}_k^{(p)T} \delta^{(p-1)}(i) \right) \delta^{(p-1)}(i) + \\ -f'_p \left( \mathbf{w}_k^{(p)T} \delta^{(p-1)}(j) \right) \delta^{(p-1)}(j) \end{bmatrix} \quad (24.64)$$

and for the hidden layer  $p-1$ ,

$$\begin{aligned} \frac{\partial \widehat{V}_E(t)}{\partial \mathbf{w}_l^{(p-1)}} &= \frac{1}{\sigma^2 L^2} \sum_{k=1}^{M_p} \sum_{j=t-L+1}^t \kappa'_\sigma(e_k(i) - e_k(j)) \\ &\times \begin{bmatrix} f'_p \left( \mathbf{w}_k^{(p)T} \delta^{(p-1)}(i) \right) w_{kl}^{(p)} f'_{p-1} \left( \mathbf{w}_l^{(p-1)T} \delta^{(p-2)}(i) \right) \delta^{(p-2)}(i) + \\ -f'_p \left( \mathbf{w}_k^{(p)T} \delta^{(p-1)}(j) \right) w_{kl}^{(p)} f'_{p-1} \left( \mathbf{w}_l^{(p-1)T} \delta^{(p-2)}(j) \right) \delta^{(p-2)}(j) \end{bmatrix}, \end{aligned} \quad (24.65)$$

where the matrices  $\mathbf{w}^{(i)}$  and  $f_i$  are the weights and activation function of the  $i$ th layer and  $\delta^{(i)}(t) = f_i(\mathbf{w}^{(i)} \delta^{(i-1)}(t))$  the outputs of the units at the same layer.

*Experiments:* In the following a feedforward neural network is trained to perform prediction of a nonlinear time series. The Mackey-Glass system is a chaotic system defined by

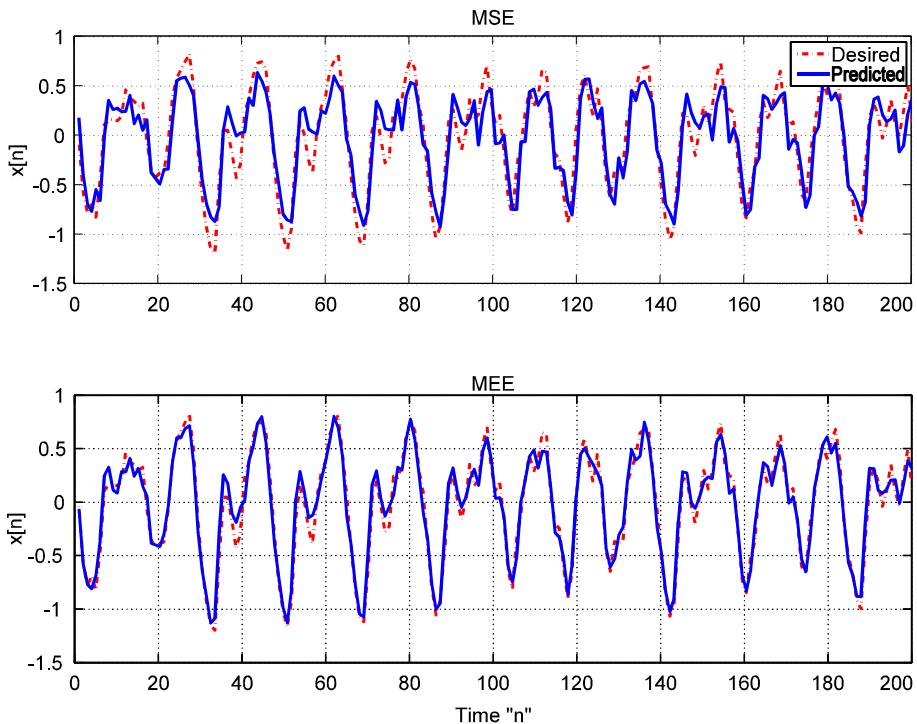
$$\dot{x}(t) = -0.1x(t) + \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}}. \quad (24.66)$$

In particular, we use  $\tau = 30$ , the number of delays for the time embedding of the series is 7, the generation of the samples follows [46]; a total of 5000 samples are generated. A TDNN with 10 units in the hidden layer and sigmoidal tangent activation function is trained using the first 500 samples from the series and in the case of MEE the sliding window has  $L = 50$ . The kernel size is kept fixed at  $\sigma = 0.2$ . The parameters are adjusted using momentum gradient with parameter  $\gamma = 0.5$  and the step size is  $\eta = 0.08$  for MSE and  $\eta = 0.02$  for MEE. Figure 24.5 shows the desired and predicted test signals for the two different cost functions.

The advantage of MEE can be observed from the distribution of the test error shown in Figure 24.6. Note that the test errors obtained from the network trained with MEE is more concentrated around zero than the one trained by MSE.

### 1.24.7.2 ED-divergence-based classification

In this case, we are given two or more distributions, for which, we are interested in basically 2 types of interactions: the particle interactions within a distribution, and the particle interactions among distributions. First, let us recall the definition of the so called Euclidean divergence between densities.

**FIGURE 24.5**

Predicted and desired outputs for TDNN trained with MSE and MEE.

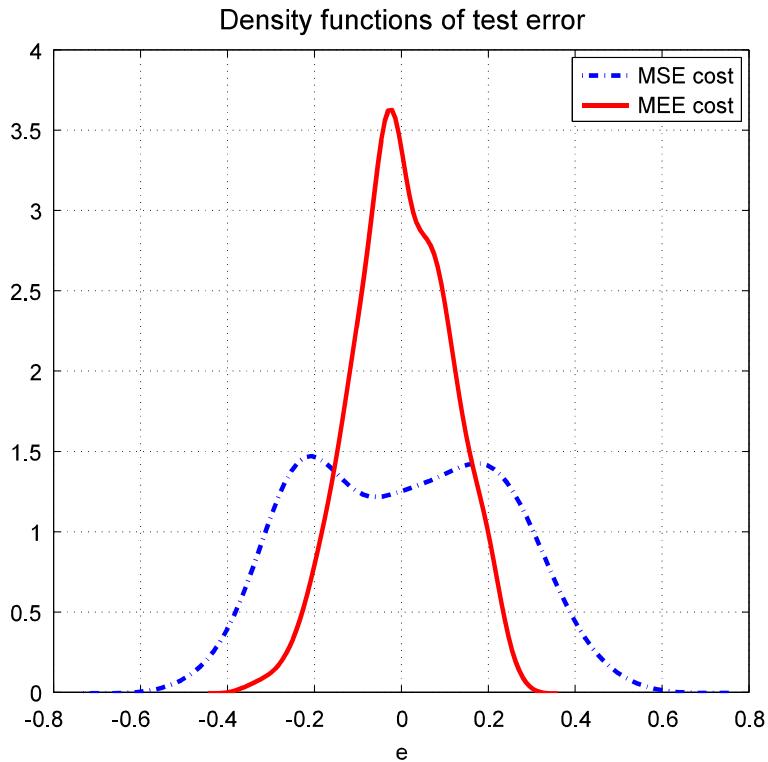
For random variables (vectors)  $X_1$  and  $X_2$  and their respective densities  $p(x)$  and  $q(x)$ . The Euclidean divergence is the overall squared difference between  $p$  and  $q$  along their common support  $X \subset \mathbb{R}^d$ , that is

$$D_{\text{ED}}(p\|q) = \int_X (p(x) - q(x))^2 dx. \quad (24.67)$$

The first condition that naturally arises from this definition is that  $p$  and  $q$  must be square integrable functions. Mutual information was originally conceived as a measure of common entropy associated with two variables. In Shannon's definition of mutual information leads to the more general concept of KL-divergence that can be utilized to compute mutual information as a dissimilarity between the joint and marginal densities (or probability mass functions). Extending this idea of dissimilarity we have the Quadratic mutual information based on the Euclidean divergence QMI<sub>ED</sub> for two random variables (or vectors)  $X_1$  and  $X_2$  with joint  $p(x_1, x_2)$  and marginals  $p_1(x)$  and  $p_2(x)$ , respectively:

$$\text{QMI}_{\text{ED}}(X_1; X_2) = D_{\text{ED}}(p(x_1, x_2)\|p_1(x_1)p_2(x_2)). \quad (24.68)$$

Notice this definition can be adapted for discrete random variables where densities are now probability mass functions and integrals become finite or countable summations.

**FIGURE 24.6**

Densities of the test errors for MSE and MEE after training.

The dataset is divided into two classes; therefore we have a second variable  $Y$  which is the class label. We can use the Euclidean divergence to compare the conditional density of  $X$  given the label  $Y$  that yields  $D_{ED}(p(x|Y=1)\|p(x|Y=2))$ . We will denote the data points that belong to class 1 as  $x_i^{(1)}$  and class 2 as  $x_i^{(2)}$ , the number of instances for class 1 is  $N_1$  and  $N_2$  for class 2. Let us construct a kernel matrix with all the data points available for the two classes in the following way:

$$\mathbf{K}_{\text{full}} = \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_2 \end{bmatrix}, \quad (24.69)$$

where  $\mathbf{K}_1$  corresponds to all pair-wise comparisons between points from class 1, likewise  $\mathbf{K}_2$  is defined for points in class 2 and the elements of  $\mathbf{K}_{12} = \mathbf{K}_{21}^T$  are given by  $K_{ij} = \kappa(x_i, x_j)$ . Having introduced this notation, let's compute the empirical estimate of the Euclidean divergence as:

$$\widehat{D}_{ED}(p(x|Y=1)\|p(x|Y=2)) = \frac{1}{N_1^2} \mathbf{1}^T \mathbf{K}_1 \mathbf{1} + \frac{1}{N_2^2} \mathbf{1}^T \mathbf{K}_2 \mathbf{1} - \frac{2}{N_1 N_2} \mathbf{1}^T \mathbf{K}_{12} \mathbf{1}. \quad (24.70)$$

The first two terms of the right-hand side are the information potential within class and the third term is the cross-information potential. Basically, we have two potential fields each one generated by the samples of the corresponding class. Figure 24.7 shows the information potential fields created by each one of the classes along with the forces experienced by each data point under that specific field.

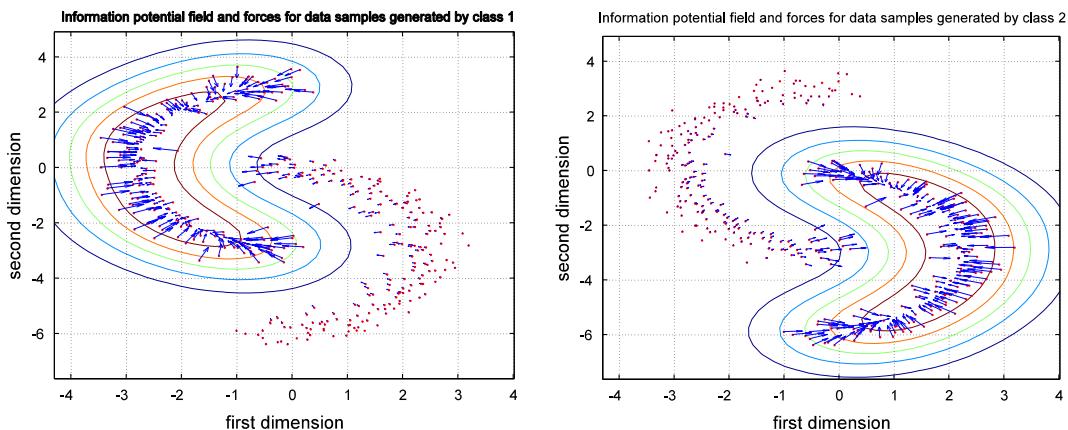
The total forces are presented in Figure 24.8 as well as the difference between the two potential fields generated by each class.

The first important observation is that the interaction of these two fields leads to the maximum likelihood classification rule. If we compute the difference between the information potentials of generated by class 1 and class 2, we are basically comparing the likelihood of a data point with respect to a conditional density given a particular class and the force experienced by the point will drive it to the class it is more likely to belong. Figure 24.9 shows the partition obtained by using the Gaussian kernel with  $\sigma = 1$ .

An interesting problem to use mutual information with the data set is the assessment of the strength of the relation between the class labels and the random vector describing the patterns that is  $\text{QMI}_{\text{ED}}(X; Y)$ . This problem involves the combination of a discrete random variable  $Y$  (the labels) and a real random vector  $X$ . The quantity of interest reduces to:

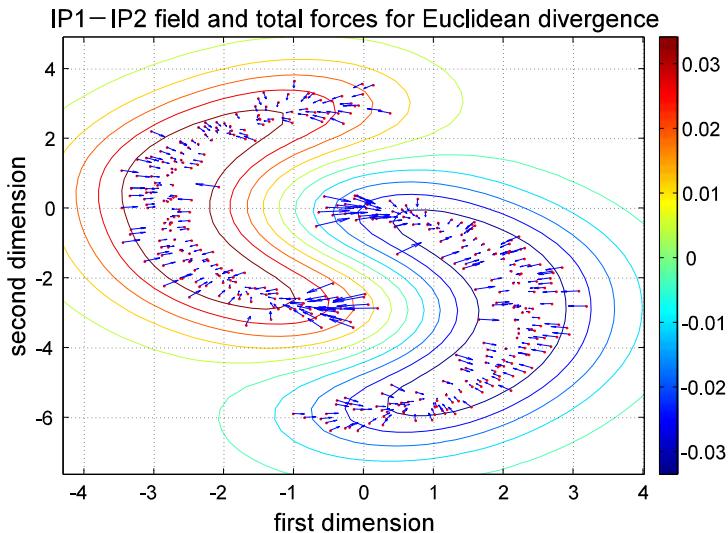
$$\text{QMI}_{\text{ED}}(X; Y) = P(Y = 1)^2 D_{\text{ED}}(p(x|Y = 1)\|p(x)) + P(Y = 2)^2 D_{\text{ED}}(p(x|Y = 2)\|p(x)). \quad (24.71)$$

In this case we are measuring the difference between the two classes by weighting the divergences between the PDF of the data when the class is not known and the density of the data given the class is known. If we relate this problem to classification we can think of it as an approximation to the MAP rule for Parzen density estimation. In our setting  $P(Y = 1) = \frac{N_1}{N_1+N_2}$  and  $P(Y = 2) = \frac{N_2}{N_1+N_2}$ . However, if we set  $N_1 = N_2$  the result from the particle interaction should be the same than the one using  $\bar{D}_{\text{ED}}(p(x|Y = 1)\|p(x|Y = 2))$ .



**FIGURE 24.7**

Information potential and forces created by each class.

**FIGURE 24.8**

$IP_1 - IP_2$  field and total information forces for the two class problem.

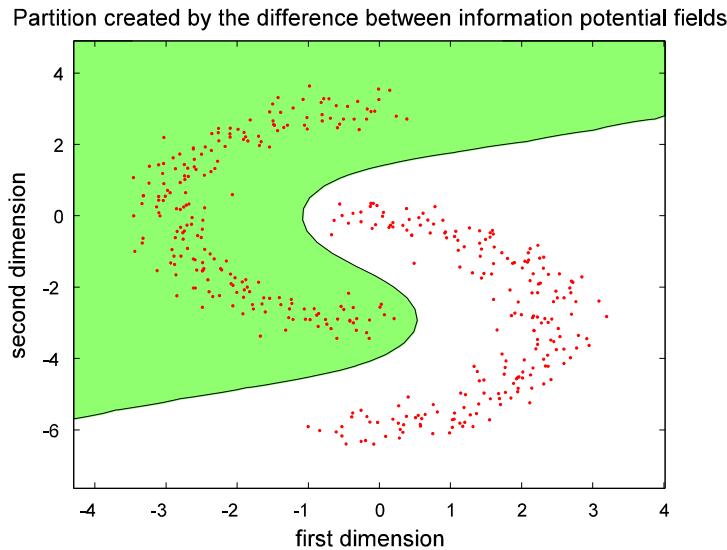
### 1.24.7.3 Information cut for clustering

*Cauchy-Schwarz objective function:* Clustering objective functions encourage the compactness of the clusters. In information theoretic clustering each cluster is represented by a density that should be as far apart as possible from the other densities (clusters). However, distances are defined between two distributions, and therefore it becomes necessary to find a suitable extension that considers more than 2 clusters. In [47], an extension based on Cauchy-Schwarz divergences is proposed. It uses the ratio between the sum of cross-information potentials between all pairs of clusters and the product of information potentials of each cluster. Consider the kernel matrix  $\mathbf{K}$  with all pairwise computations of the kernel function and the membership matrix  $\mathbf{M}$ , where each row contains a binary vector  $\mathbf{m}_i^T$  that indicates the membership of the  $i$ th point associated with the given row. The cost  $J$  associated with the membership matrix  $\mathbf{M}$  is given by:

$$J(\mathbf{M}) = \frac{\mathbf{1}^T \mathbf{K} \mathbf{1} - \sum \text{diag}(\mathbf{M}^T \mathbf{K} \mathbf{M})}{\prod \text{diag}(\mathbf{M}^T \mathbf{K} \mathbf{M})}, \quad (24.72)$$

where the sum and product are over the elements of the diagonal of  $(\mathbf{M}^T \mathbf{K} \mathbf{M})$ . To solve this problem using gradient descend techniques the membership vectors are allowed to take real values.

*Experiments:* Figure 24.10 present the resulting clusters from Cauchy-Schwarz criterion and  $k$ -means for two synthetic datasets. The matrix  $\mathbf{M}$  is initialized with random cluster assignments. As expected  $k$ -means perform poorly since it cannot identify non-convex clusters or cluster with different scales. It has to be noted that the even though these results are achieved the majority of times since they still depend on the initial assignments and the selection of proper kernel sizes.

**FIGURE 24.9**

Partition of the space created by the difference between information potentials due to each class. This is equivalent to the maximum likelihood rule for classification when using Parzen window estimation for the conditional densities given the class.

#### 1.24.7.4 Principle of relevant information

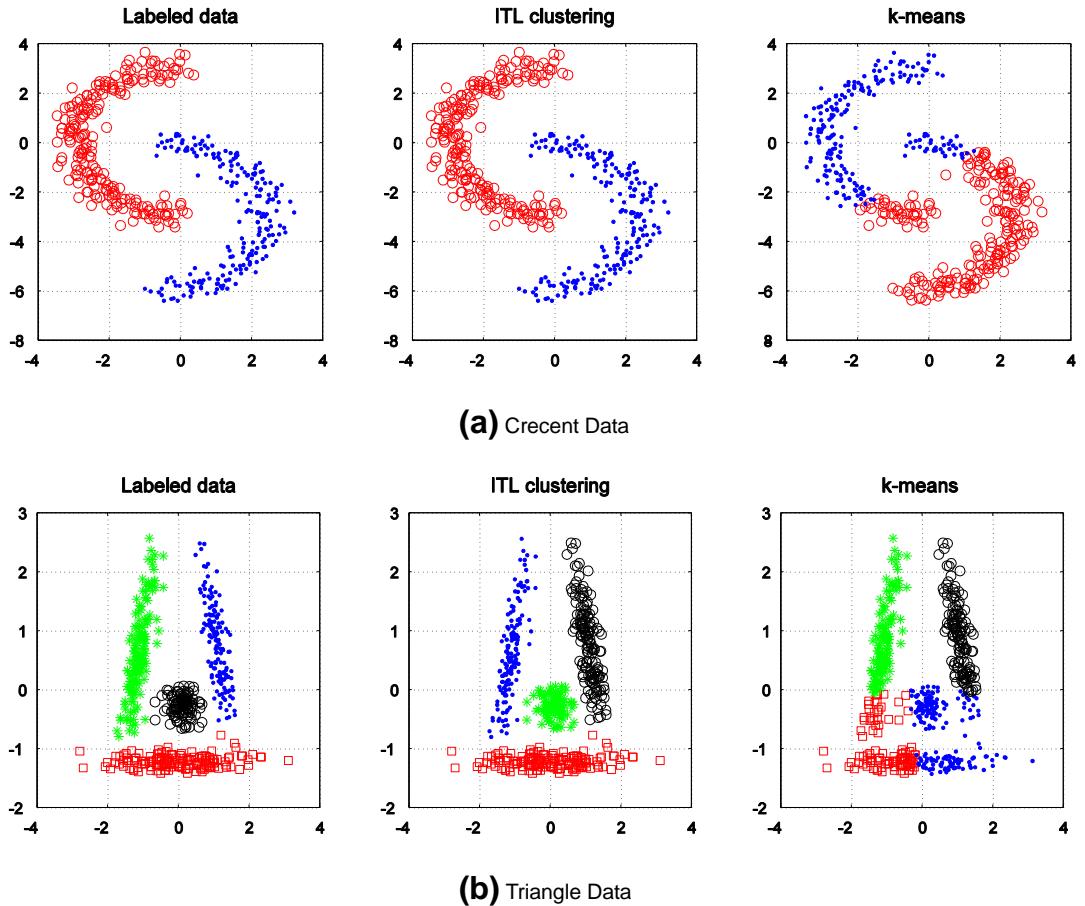
*The PRI as a Weighting Problem:* For the set  $F$  of probability density functions that are square integrable in  $X \subset \mathbb{R}^d$ , the cross-information potential (CIP) is defined as a bilinear form that maps densities  $f_i, f_j \in F$  to the real numbers through the integral,

$$V(f_i, f_j) = \int_X f_i(x) f_j(x) dx. \quad (24.73)$$

It is easy to see that for a basis of uniformly bounded, square integrable probability density functions,  $V(f_i, f_j)$  is a positive semidefinite function on the span  $\{f_i \in F\}$ . Now, consider the set  $G = \{g = \sum_{i=1}^m \alpha_i \kappa_\sigma(x_i, \cdot) | x_i \in \mathbb{R}^d, \sum_{i=1}^m \alpha_i = 1, \alpha_i \geq 0\}$ , where  $\kappa_\sigma$  is a “Parzen” type of kernel which is also square integrable, that is,  $\kappa_\sigma$  is symmetric, nonnegative, has bounded integral (can be normalized), belongs to  $L^2$ , and shift invariant with  $\sigma$  as the scale parameter. Clearly, for any  $g \in G$  we have  $\|g\|^2 \leq \|\kappa_\sigma(x, \cdot)\|^2$ . Hence,  $G$  is bounded. However, if  $X$  is non-compact our search space is also non-compact.

The objective function for the principle of relevant information can be written in terms of IP functionals. Using Parzen based estimation, we restrict the search problem to  $G \subset F$ . In this case, we have that Eq. (24.26) can be rewritten as:

$$J(f) = -\log V(f, f) - \lambda \log \frac{V(f, g)^2}{V(f, f)V(g, g)}. \quad (24.74)$$

**FIGURE 24.10**

Clustering results for synthetic datasets.

Straightforward manipulation of the terms yields an equivalent problem:

$$\arg \min_f J(f) = -(1 - \lambda) \log V(f, f) - 2\lambda \log V(f, g). \quad (24.75)$$

Two important aspects of the above objective are: the choice of the kernel and its size  $\sigma$ , which determines different scale of the analysis; and the trade-off parameter  $\lambda$ , which defines a set of regimes for the possible solutions to the problem. The only available information is contained in the sample  $S = \{x_i\}_{i=1}^N$ . An approximation of the target density  $g$  is then given by its weighted Parzen window estimator  $\hat{g}(x) = \sum_{i=1}^N \alpha_i \kappa(x_i, x)$ . In our experiments, we use  $\alpha_i = \frac{1}{N}$ . To enforce compactness in our search space, we

look for a solution  $f$  that has the same form of  $\hat{g}$ , that is

$$f(x) = \sum_{i=1}^N \beta_i \kappa(x_i, x), \quad (24.76)$$

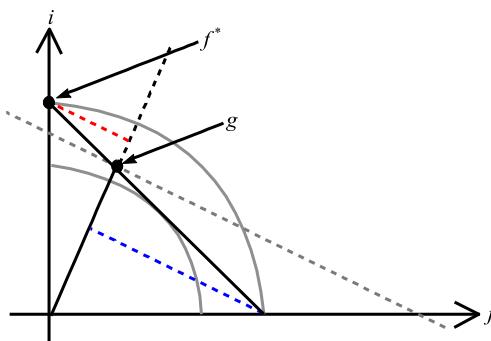
where  $\beta_i \geq 0$ ,  $\sum_{i=1}^N \beta_i = 1$ . By fixing  $\lambda$  and evaluating the information potential between each pair  $(x_i, x_j) \in S \times S$ , we can rewrite (24.69) in matrix notation as:

$$\begin{aligned} \min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) &= (\lambda - 1) \log \boldsymbol{\beta}^T \mathbf{V} \boldsymbol{\beta} - 2\lambda \log \boldsymbol{\beta}^T \mathbf{V} \boldsymbol{\alpha} \\ \text{s.t. } \boldsymbol{\beta} &\geq 0 \quad \text{and} \quad \boldsymbol{\beta}^T \mathbf{1} = 1, \end{aligned} \quad (24.77)$$

where  $V_{ij} = \int_X \kappa(x_i, x) \kappa(x_j, x) dx$ . It is also important to highlight that for values of  $\lambda \rightarrow 1$  the regularization term almost vanishes, and the maximum information gain (inner product term) can be related to the modes of  $\hat{g}$ , which are approximated by the sharpest functions available in  $G$ , which in our setting, are individual windows that satisfy convex combination constraints. Therefore, we can expect the modes of  $\hat{g}$  to be almost orthogonal components. When  $\lambda$  is very close to one, the cross-entropy term is dominant in the objective and the solution will lie in one of the corners of the simplex. Figure 24.11 depicts the geometrical elements that explain this phenomenon. The semicircles are the loci of minimum and maximum of the  $L^2$  norm that intersect with the simplex. The solution  $f^*$  with the largest norm is the one maximizing the dot product with the target distribution  $g$ .

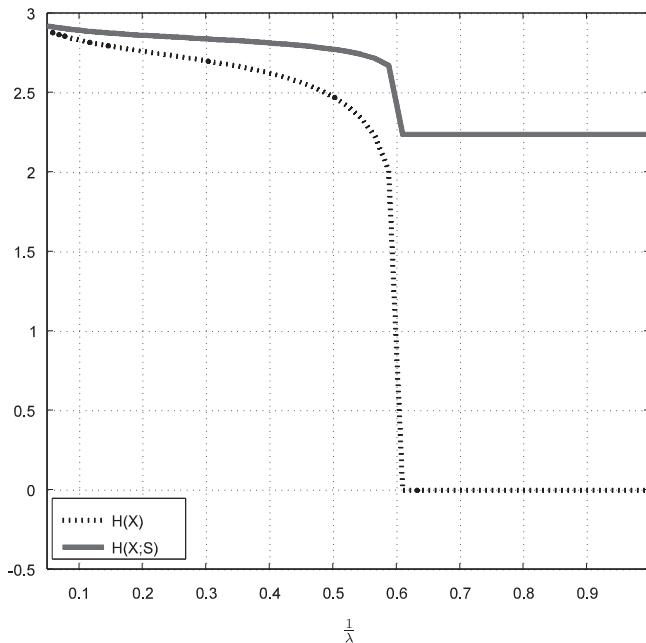
This corner phenomenon motivates the extraction of more than one solution. A reasonable approach is to deflate the target PDF  $\hat{g}$  and run the algorithm on the new function.

We perform an illustrative experiment on a synthetic data set that correspond to a Gaussian surrounded by a circular crown as depicted in Figure 24.13. Figure 24.12 shows different values of the entropy when the trade off parameter  $\lambda$  is varied between 1 and 20. The actual plot shows the inverse of  $\lambda$  to facilitate visualization.



**FIGURE 24.11**

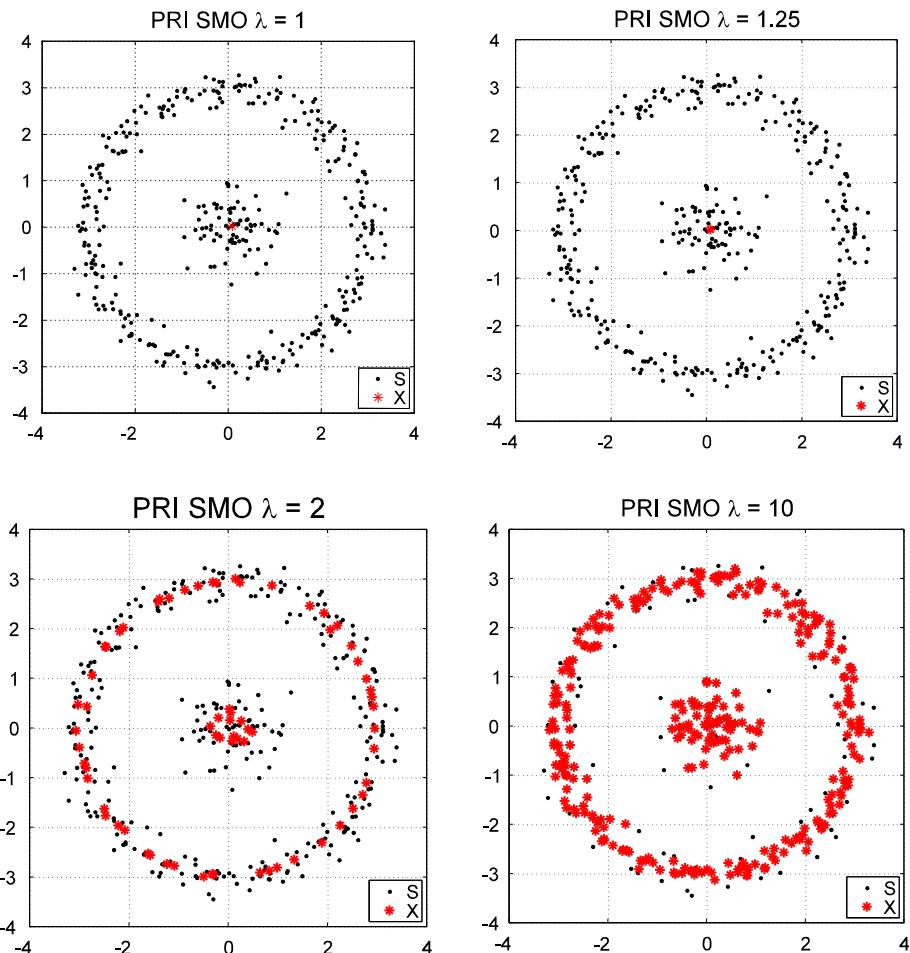
Effect of the PRI objective based on weights.

**FIGURE 24.12**

Entropy of the weighted sample and cross-entropy with respect to the original sample  $S$  for different values of  $\lambda$ . Notice that the values are plotted against  $\frac{1}{\lambda}$  for better visualization.

In this case, we can observe two regimes; for  $\lambda < 1.5$ , all the importance is given to one point, which corresponds to the largest peak on the Parzen density estimate from the observed data  $S$ . Once  $\lambda$  becomes larger than 1.5 more points become active, and the weights progressively equalize across all data points. Figure 24.13 display the resulting support vectors found after optimization for different values of  $\lambda$ .

*Image Retrieval with Partially Occluded Data MNIST:* We employ a subset of the MNIST database to perform experiments on pattern retrieval from partially occluded queries. The weighting scheme for the principle of relevant information is applied to learn the representative samples from the training data. Results are compared against kernel PCA and kernel entropy component analysis for different number of components and different values of  $\lambda$  in the case of PRI. The pattern retrieval application requires pre-imaging of the patterns from the feature space back to the input space to apply KPCA (KECA). We employ the method presented in [48] to compute the pre-images of the projected patterns in the KPCA and KECA subspaces (For more details on this problem see [49, 50]). The principle of relevant information requires a different approach since it does not provide an explicit projection method unlike KPCA and KECA. A simple pre-imaging algorithm can be based on the Nadaraya-Watson kernel regression [51]. In our experiments, we use the Gaussian kernel. The pre-imaging for PRI consists of the following steps:

**FIGURE 24.13**

Resulting support vectors for different values of  $\lambda$ .

- Compute the  $k$ -nearest neighbors on the set of support vectors (training points with nonzero weights) of the query pattern  $x$ .
- Reconstruct using the following equation:

$$x_{\text{rec}} = \frac{\sum_{i=1}^k \kappa_\sigma(x_i, x)x_i}{\sum_{i=1}^k \kappa_\sigma(x_i, x)}, \quad (24.78)$$

where the indexes for  $x_i$  are understood as the  $k$  nearest neighbors of  $x$ .

**Table 24.1** Results for the Image Retrieval from Partially Occluded Queries

Number of components	3	8	15	30	50	100
KPCA	0.59(0.03)	0.55(0.015)	0.53(0.007)	0.50(0.007)	0.48(0.008)	0.47(0.006)
KECA	0.64(0.018)	0.57(0.018)	0.51(0.022)	0.49(0.013)	0.49(0.009)	0.48(0.007)
PRI	0.58(0.013)	0.53(0.004)	0.51(0.007)	0.50(0.009)	0.49(0.01)	0.48(0.01)
$\lambda$	2	3	4	5	6	8
N support vectors	62.7	126	181	232	276	347

Note: The highlighted values indicate the smallest “normalized SNR” in each column.

**FIGURE 24.14**

Sample queries with missing information and their corresponding complete versions.

In this case, we want to see if we can effectively retrieve a digit by presenting an image with a missing portion. The model is trained with complete images and tested with images for which the lower half of the matrix has been replaced with zeros.

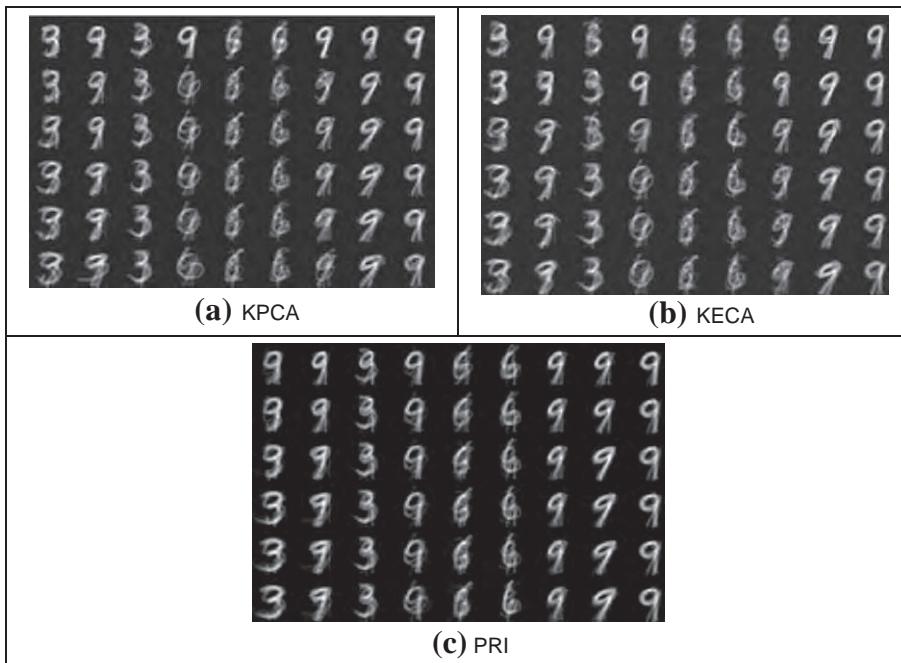
Table 24.1 shows the normalized SNR for the retrieved patterns using a KPCA KECA and the weighted PRI. The numbers correspond to averages from ten Monte Carlo simulations using 200 images per digit (total 600) for training and 50 incomplete images of each digit (150 total) for testing. We also present the average number of support vectors for the RKHS PRI. The standard deviations are also shown. The kernel size is 5, and the number of neighbors for pre-imaging is 10.

Notice that differences between closest results for all methods and also the best performances are within the range set by one standard deviation.

Figure 24.14 shows some of the exemplars that were used for testing along with the incomplete versions that were input to the algorithms. Figure 24.15a shows the reconstructed images for KPCA where each row corresponds to the number of components in Table 24.1. Likewise, Figure 24.15b displays the results for KECA, and Figure 24.15c shows the results of PRI approach for the different  $\lambda$  values in Table 24.1.

### 1.24.7.5 Independent component analysis

*Minimum Renyi's Mutual Information:* The MRMI algorithm is motivated by minimization of mutual dependence between the estimated sources in the context of non-parametric estimation [52,53]. The basic idea is to use an alternative definition of entropy that facilitates the application of non-parametric density estimation techniques such as Parzen windowing.



## FIGURE 24.15

Sample queries with missing information and their corresponding complete versions.

Renyi's mutual information is

$$I_\alpha(Y) = \frac{1}{\alpha - 1} \log \left( \int_{-\infty}^{\infty} \frac{f_Y(y)^\alpha}{\prod_{i=1}^d f_{Y_i}(y_i)^{\alpha-1}} dy \right). \quad (24.79)$$

Although this definition generalizes the concept of KL-divergence, properties such as recursion fail to hold since the formulation followed for Shannon's entropy yields

$$\sum_{i=1}^d H_\alpha(Y_i) - H_\alpha(Y) = \frac{1}{\alpha-1} \log \left( \frac{\int_{-\infty}^\infty f_Y(y)^\alpha dy}{\int_{-\infty}^\infty \prod_{i=1}^d f_{Y_i}(y_i)^{\alpha-1} dy} \right). \quad (24.80)$$

The expression in (24.80) is zero if the  $Y_i$ 's are independent. Unfortunately, zero value is not sufficient condition for independence. Assuming favorable condition on the distribution of the sources, we have that for an orthogonal rotation  $\mathbf{R}$  on a whitened version of the observed variables  $X_i$ , the joint entropy of the estimated sources  $Y = \mathbf{R}P\mathbf{X} = \mathbf{W}\mathbf{X}$  is invariant under orthonormal transformations. Thus minimization of (24.80) can be done only on the sum of the marginal entropies. The objective function is then

$$J(\mathbf{R}) = \sum_{i=1}^d H_\alpha(Y_i). \quad (24.81)$$

For  $\alpha = 2$ , Eq. (24.81) is estimated by using Parzen windowing with a Gaussian kernel; it is easy to note that the minimization of this cost will look for super-Gaussian distributions since sum of output marginal entropies is minimized. Therefore we should expect the algorithm to fail in the presence of sub-Gaussian sources. Nevertheless, switching the sign of some of the terms in the sum  $J(\mathbf{R})$  yields a search procedure that can converge to sub-Gaussian distributions. Therefore an algorithm obtained by assigning individual signs to the marginal entropies  $H_\alpha((\mathbf{R}Y)_i)$  can find sub and super Gaussian sources.

Performance of the algorithms is measured using signal-to-distortion ratio (SDR):

$$\text{SDR} = -10 \log \left[ \frac{1}{d(d-1)} \sum_{i=1}^d \left( \frac{\sum_{j=1}^d |B_{ij}|}{\max_j |B_{ij}|} - 1 \right) \right], \quad (24.82)$$

where  $\mathbf{B} = \mathbf{WA}$ , being  $\mathbf{A}$  the known mixing matrix.

*Synthetic Data:* The aim of the artificial data is to check the validity of our analysis of the expected properties for the algorithms. For this purpose we create a set of artificial signals described as follows:

$$\begin{aligned} x_s(t) &= \sin \left( \frac{2\pi}{T_s} t \right), \\ x_w(t) &= \text{saw} \left( \frac{2\pi}{T_w} t + \theta \right), \\ x_e(t) &= \frac{1}{5} \text{sign}(x_w(t)) (\exp(5x_w(t)) - 1). \end{aligned} \quad (24.83)$$

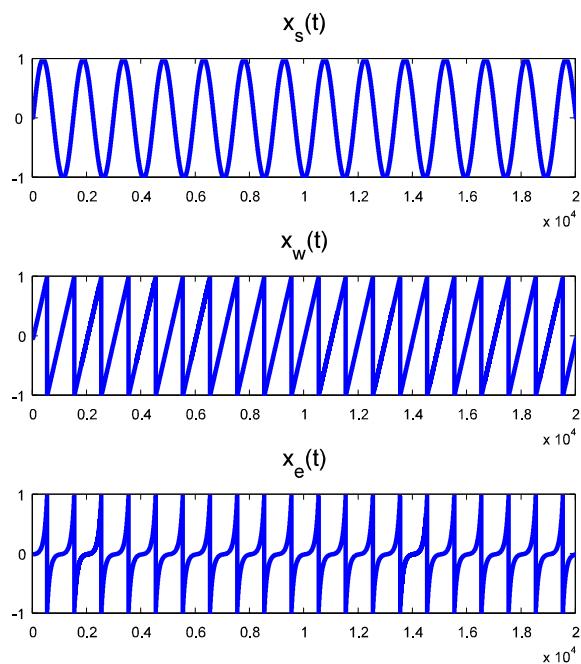
In this case  $x_s(t)$  and  $x_w(t)$  are sub-Gaussian, and  $x_e(t)$  is super-Gaussian. We create two sets of sources  $\mathbf{x}_a(t) = (x_s(t) \ x_w(t))^T$  (both sub-Gaussian) and  $\mathbf{x}_b(t) = (x_s(t) \ x_e(t))^T$  (mixed type sub- and super-Gaussian). Figure 24.16 displays the three simulated sources.

We only want to evaluate the properties of the estimated sources; therefore, we run the update throughout the whole signal using on-line learning mode. Figure 24.17 display the estimated sources using infomax for observed  $\mathbf{x}_a(t)$  and  $\mathbf{x}_b(t)$ , respectively; the corresponding SDRs are 0.8414 and 2.72. As expected the algorithm failed to find the correct de-mixing since the activation function cannot model sub-Gaussian distributions and so prefers a solution for which the estimated sources are apparently super-Gaussian.

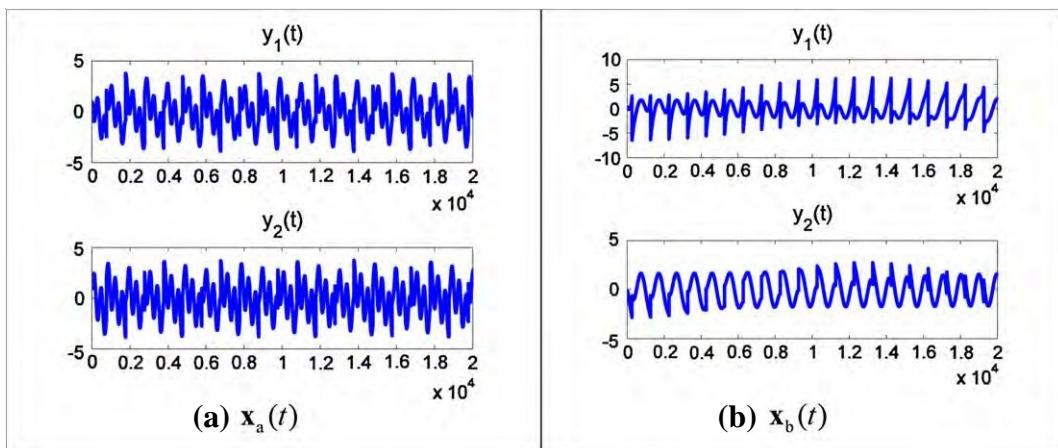
For the MRMI criterion, given that we have a priori knowledge of the source distribution, we switch the sign of the objective function for both sources for the observed  $\mathbf{x}_a(t)$ . The estimated density is calculated taking a uniform random sub-sample of 500 instances from the entire observed signal. The sign of each marginal output entropy (maximum or minimum) is chosen according to the sign of the deviation from the kurtosis for a normal random variable. The objective function from Eq. (24.81) can be modified as follows:

$$J(\mathbf{R}) = - \sum_{i=1}^d \text{sign}(\text{kurt}(Y_i) - 3) H_\alpha(Y_i). \quad (24.84)$$

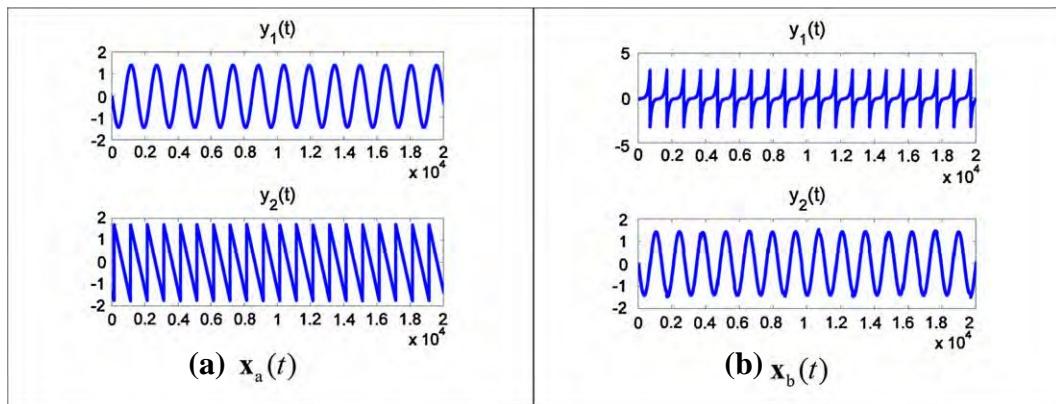
Figure 24.18 shows the extracted sources by using the modified version of MRMI using kurtosis. The SDRs are 22.2165 for  $\mathbf{x}_a(t)$  and  $\mathbf{x}_b(t)$ . Notice that in both cases the modified version succeeded in finding the independent components.

**FIGURE 24.16**

Sub-Gaussian and Super-Gaussian artificial sources.

**FIGURE 24.17**

Estimated sources for infomax.

**FIGURE 24.18**

Estimated sources for modified MRMI.

**Table 24.2** SDR Behavior of 20 Simulations of Infomax in On-Line Mode for Different Training Sample Sizes

SDR	Sub-sample size		
	100	1000	10000
Average	7.8	8.01	8.71
Std Deviation	3.39	4.12	4.26

**Table 24.3** SDR Behavior of 20 Simulations of MRMI for Different Training Sample Sizes and 300 epochs

SDR	Sub-sample size		
	50	100	500
Average	13.92	14.56	14.77
Std Deviation	1.39	0.67	0.439

*Mixture of Audio Signals:* The problem is to separate an interest sound from the background activity (noise). To see the stability and accuracy of the estimation based on the size of the training data we randomly sub-sample the whole signal in different proportions. Table 24.2 displays the average SDR of 20 simulations of infomax along with the standard deviation on sub-sample sizes of 100, 1000, and 10,000 instances. The number of epochs is inversely proportional to the sub-sample size providing the same number of iterations (200,000); the stepsize is  $\eta = 0.3$ . A different procedure is followed for

MRMI. Sample sizes are 50, 100, and 500 and the number of epochs is fixed to 300. Table 24.3 displays the average SDR of 20 simulations of MRMI and their corresponding standard deviations. Kernel is fixed to  $\sigma = 0.5$  since data is whitened, the stepsize is  $\eta = 0.5$ .

### 1.24.8 Conclusions and future trends

Traditional approaches to machine learning and signal processing in general aim to optimize the second order statistics. When data is non-Gaussian, a more appropriate approach would be to capture higher order statistics or information content of signals rather than simply their energy. Especially, information theoretic quantities as optimality criteria attract ever-increasing attention to this end, since they provide a natural framework where information content of available data can be assessed. Combining information theoretic descriptors (such as entropy, divergence, and mutual information) and kernel density estimation, one can obtain a general, nonparametric, and sample based methodology to learn arbitrary nonlinear systems.

In this chapter, we briefly reviewed the general methodology to implement machine learning and adaptation algorithm with information theoretic criteria, which synergistically integrates the general framework of information theory in the design of new cost functions for machine learning and adaptive signal processing. Information based learning not only affects our understanding of optimal signal processing, but also influence the way we approach machine learning, data compression, and adaptation. With smooth nonparametric information estimators, ITL methodology allows simple gradient based learning rule to be constructed for learning nonlinear topologies. In most situations, the performance of ITL learning algorithms compare favorably with existing methods developed under second order statistics criteria. For illustration purpose, we also include in this chapter several application examples.

Information-based learning is still one of the freshest ideas floating around, which has not yet become prevalent today. There are many issues that need to be handled in the next stage. Examples are: (1) how to choose the proper parameters such as the order of the information potential or the kernel size in entropy estimator; (2) how to simplify the computational complexity especially in batch mode learning; (3) how to efficiently implement the algorithms in the hardware. If these issues have been properly resolved, information based learning will become more practical and powerful, and it will find a widespread acceptance in machine learning community.

*Relevant Theory:* Signal Processing Theory, Machine Learning and Statistical Signal Processing

See this Volume, [Chapter 12](#) Adaptive Filters

See this Volume, [Chapter 16](#) Kernel Methods and SVMs

See this Volume, [Chapter 17](#) Online Learning in Reproducing Kernel Hilbert Spaces

See [Vol. 3](#), [Chapter 4](#) Bayesian Computational Methods in Signal Processing

---

## References

- [1] T. Cover, J. Thomas, Element of Information Theory, Wiley, New York, 1991.
- [2] C. Shannon, W. Weaver, The Mathematical Theory of Communication, University of Illinois Press, Urbana, 1949.
- [3] E. Pfaffelhuber, Learning and information theory, Int. J. Neurosci. 3 (1972) 83–88.

- [4] H. Barlow, Unsupervised learning, *Neural Comput.* 1 (1989) 295–311.
- [5] R. Linsker, Self-organization in a perceptual network, *IEEE Comput.* 21 (1988) 105–117.
- [6] J.C. Principe, D. Xu, Q. Zhao, J.W. Fisher III, Learning from examples with information theoretic criteria, *J. VLSI Signal Process.* 26 (2000) 61–77.
- [7] J.C. Principe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*, Springer, 2010.
- [8] J.O. Berger, *Statistical Decision Theory and Bayesian Analysis*, second ed., Springer-Verlag, New York, 1985.
- [9] A. Renyi (Ed.), *Selected Papers of Alfred Renyi*, vol. 2, Akademia Kiado, Budapest, 1976.
- [10] D. Erdogmus, *Information Theoretic Learning: Renyi's Entropy and Its Applications to Adaptive Systems Training*, University of Florida, Gainesville, Ph. D. Dissertation, 2002.
- [11] D. Erdogmus, J.C. Principe, Generalized information potential for adaptive systems training, *IEEE Trans. Neural Networks* 13 (5) (2002) 1035–1044.
- [12] E. Parzen, On the estimation of a probability density function and the mode, *Ann. Math. Stat.* 33 (1962) 1065–1067.
- [13] B. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.
- [14] E. Jaynes, Information theory and statistical mechanics, *Phys. Rev.* 106 (1957) 620–630.
- [15] E. Jaynes, *Probability Theory, the Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.
- [16] A. Renyi, *Probability Theory*, University Amsterdam, North-Holland, 1970.
- [17] E. Lutwak, D. Yang, G. Zhang, Cramer-Rao and moment-entropy inequalities for Renyi entropy and generalized Fisher information, *IEEE Trans. Inform. Theory* 51 (2) (2005) 473–479.
- [18] M. Rao, Y. Chen, B.C. Venuri, F. Wang, Cumulative residual entropy: a new measure of information, *IEEE Trans. Inform. Theory* 50 (6) (2004) 1220–1228.
- [19] B. Chen, P. Zhu, J.C. Principe, Survival information potential: a new criterion for adaptive system training, *IEEE Trans. Signal Process.* 60 (3) (2012) 1184–1194.
- [20] H.L. Weidemann, E.B. Stear, Entropy analysis of estimating systems, *IEEE Trans. Inform. Theory* 16 (1970) 264–270.
- [21] P. Kalata, R. Priemer, Linear prediction, filtering and smoothing: an information theoretic approach, *Inform. Sci.* 17 (1979) 1–14.
- [22] D. Erdogmus, J.C. Principe, An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems, *IEEE Trans. Signal Process.* 50 (7) (2002) 1780–1786.
- [23] J. Santos, L. Alexandre, J. Sa, The error entropy minimization algorithm for neural network classification, in: A Lofti (Ed.), *Int. Conf. Recent Advances in Soft Computing*, 2004, pp. 92–97.
- [24] G. Brown, A. Pocock, M.-J. Zhao, M. Lujan, Conditional likelihood maximization: a unifying framework for information theoretic feature selection, *J. Mach. Learning Res.* 13 (2012) 27–66.
- [25] R. Fano, *Transmission of Information*, MIT Press, 1961.
- [26] A.J. Bell, T.J. Sejnowski, An information maximization approach to blind separation and blind deconvolution, *Neural Comput.* 7 (6) (1995) 1129–1159.
- [27] S. Rao, *Unsupervised Learning: An Information Theoretic Learning Approach*, University of Florida, Gainesville, Ph. D. Dissertation, 2008.
- [28] L. Sanchez Giraldo, J.C. Principe, An efficient rank-deficient computation of the principle of relevant information, in: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2011)*, pp. 2176–2179.
- [29] L. Sanchez Giraldo, J.C. Principe, A reproducing kernel Hilbert space formulation of the principle of relevant information, in: *IEEE International Workshop on Machine Learning for, Signal Processing (MLSP)*, 2011.
- [30] A. Hyvarinen, J. Karhunen, E. Oja, *Independent Component Analysis*, Wiley, New York, 2001.
- [31] S. Haykin (Ed.), *Blind Deconvolution*, Prentice-Hall, Upper Saddle River, NJ, 1994.
- [32] N. Tishby, F. Pereira, W. Bialek, The information bottleneck method, in: *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, 1999, pp. 368–377.

- [33] N. Tishby, N. Slonim, Data clustering by markovian relaxation and the information bottleneck method, in: Advances in Neural Information Processing Systems, vol.13, Denver, 2000, pp. 640–646.
- [34] J. Beirlant, E.J. Dudewicz, L. Gyorfi, E.C. van der Meulen, Nonparametric entropy estimation: an overview, Int. J. Math. Stat. Sci. 6 (1) (1997) 17–39.
- [35] L. Devroye, G. Lugosi, Combinatorial Methods in Density Estimation, Springer, New York, 2001.
- [36] J.-W. Xu, Nonlinear Signal Processing Based on Reproducing Kernel Hilbert Space, University of Florida, Gainesville, Ph. D. Dissertation, 2008.
- [37] J.-W. Xu, A. Paiva, I. Park, J.C. Principe, A reproducing kernel Hilbert space framework for information-theoretic learning, IEEE Trans. Signal Process. 56 (12) (2008) 5891–5902.
- [38] R. Jenssen, D. Erdogmus, J.C. Principe, T. Eltoft, Some equivalence between kernel and information theoretic methods, J. VLSI Signal Process. 45 (2006) 49–65.
- [39] E. Moore, On properly positive Hermitian matrices, Bull. Amer. Math. Soc. 23 (59) (1916) 66–67.
- [40] N. Aronszajn, The theory of reproducing kernels and their applications, Cambridge Phil. Soc. Proc. 39 (1943) 133–153.
- [41] J. Mercer, Functions of positive and negative type, and their connection with the theory of integral equations, Phil. Trans. Roy. Soc. Lond. 209 (1909) 415–446.
- [42] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, NJ, 1999.
- [43] D. Erdogmus, K. Hild, J.C. Principe, Beyond second order statistics for learning: a pairwise interaction model for entropy estimation, J. Nat. Comput. 1 (1) (2003) 85–108.
- [44] J.C. Principe, D. Xu, J. Fisher, Information theoretic learning, in: S. Haykin (Ed.), Unsupervised Adaptive Filtering, Wiley, New York, 2000, pp. 265–319.
- [45] D. Erdogmus, J.C. Principe, From linear adaptive filtering to nonlinear information processing, IEEE Signal Process. Mag. 23 (2006) 14–33.
- [46] W. Liu, Il Park, J.C. Principe, An information theoretic approach of designing sparse kernel adaptive filters, IEEE Trans. Neural Networks 20 (2009) 1950–1961.
- [47] R. Jenssen, D. Erdogmus, II K. Hild, J.C. Principe, T. Eltoft, Information cut for clustering using a gradient descent approach, Pattern Recog. 40 (2006) 796–806.
- [48] J.T. Kwok, I.W. Tsang, The preimage problem in kernel methods, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 408–415.
- [49] S. Mika, G. Ratch, J. Weston, B. Schölkopf, K.-R. Müller, Fisher discriminant analysis with kernels, in: IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing, August, 1999, pp. 41–48.
- [50] R. Jenssen, T. Eltoft, M. Girolami, D. Erdogmus, Kernel maximum entropy data transformation and an enhanced spectral clustering algorithm, in: NIPS, 2006, pp. 633–640.
- [51] E.A. Nadaraya, On estimating regression, Theory Probab. Appl. 9 (1) (1964) 141–142.
- [52] K.E. Hild II, D. Erdogmus, J.C. Principe, Blind source separation using Renyi's mutual information, IEEE Signal Process. Lett. 8 (6) (2001) 174–176.
- [53] D. Erdogmus, K.E. Hild II, J.C. Principe, Blind source separation using Renyi's  $\alpha$ -marginal entropies, Neurocomputing 49 (2002) 25–38.

# A Tutorial on Model Selection

# 25

Enes Makalic\*, Daniel Francis Schmidt\* and Abd-Krim Seghouane†

\*Centre for M.E.G.A. Epidemiology, The University of Melbourne, Australia

†Department of Electrical and Electronic Engineering, The University of Melbourne, Australia

## 1.25.1 Introduction

A common and widespread problem in science and engineering is the determination of a suitable model to describe or characterize an experimental data set. This determination consists of two tasks: (i) the choice of an appropriate model structure, and (ii) the estimation of the associated model parameters. Given the structure or dimension of the model, the task of parameter estimation is generally done by maximum likelihood or least squares procedures. The choice of the dimension of a model is often facilitated by the use of a model selection criterion. The key idea underlying most model selection criteria is the parsimony principle which says that there should be a tradeoff between data fit and complexity. This chapter examines three broad approaches to model selection commonly applied in the literature: (i) methods that attempt to estimate the distance between the fitted model and the unknown distribution that generated the data (see Section 1.25.2); (ii) Bayesian approaches which use the posterior distribution of the parameters to make probabilistic statements about the plausibility of the fitted models (see Section 1.25.3), and (iii) information theoretic model selection procedures that measure the quality of the fitted models by how well these models compress the data (see Section 1.25.4).

Formally, we observe a sample of  $n$  data points  $\mathbf{y} = (y_1, \dots, y_n)'$  from an unknown probability distribution  $p^*(\mathbf{y})$ . The model selection problem is to learn a good approximation to  $p^*(\cdot)$  using only the observed data  $\mathbf{y}$ . This problem is intractable unless some assumptions are made about the unknown distribution  $p^*(\cdot)$ . The assumption made in this chapter is that  $p^*(\cdot)$  can be adequately approximated by one of the distributions from a set of parametric models  $p(\mathbf{y}|\boldsymbol{\theta}; \gamma)$  where  $\gamma \in \Gamma \subseteq \mathbb{N}$  defines the model structure and the parameter vector  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_k)' \in \Theta_\gamma \subset \mathbb{R}^k$  indexes a particular distribution from the model  $\gamma$ . The dimensionality of  $\boldsymbol{\theta}$  will always be clear from the context. As an example, if we are interested in inferring the order of a polynomial in a linear regression problem, the set  $\Gamma$  may include first-order, second-order and third-order polynomials. The parameter vector  $\boldsymbol{\theta}$  then represents the coefficients for a particular polynomial; for example, the intercept and linear term for the first-order model. The model selection problem is to infer an appropriate model structure  $\gamma$  and parameter vector  $\boldsymbol{\theta}$  that provide a good approximation to the data generating distribution  $p^*(\cdot)$ .

This chapter largely focuses on the inference of the model structure  $\gamma$ . A common approach to estimating the parameters  $\boldsymbol{\theta}$  for a given model  $\gamma$  is the method of maximum likelihood. Here, the

parameter vector is chosen such that the probability of the observed data  $\mathbf{y}$  is maximized, that is

$$\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma) = \arg \max_{\boldsymbol{\theta} \in \Theta_\gamma} p(\mathbf{y}|\boldsymbol{\theta}; \gamma) \quad (25.1)$$

where  $p(\mathbf{y}|\boldsymbol{\theta}; \gamma)$  denotes the likelihood of the data under the distribution indexed by  $\boldsymbol{\theta}$ . This is a powerful approach to parameter estimation that possesses many attractive statistical properties [1]. However, it is not without its flaws, and as part of this chapter we will examine an alternative method of parameter estimation that should sometimes be preferred to maximum likelihood (see Section 1.25.4.2). The statistical theory underlying model selection is often abstract and difficult to understand in isolation, and it is helpful to consider these concepts within the context of a practical example. To this end, we have chosen to illustrate the application of the model selection criteria discussed in this chapter to the important and commonly used linear regression model.

### 1.25.1.1 Nested and non-nested sets of candidate models

The structure of the set of candidate models  $\Gamma$  can have a significant effect on the performance of model selection criteria. There are two general types of candidate model sets: *nested* sets of candidate models, and *non-nested* sets of candidate models. Nested sets of candidate models have special properties that make model selection easier and this structure is often assumed in the derivation of model selection criteria, such as in the case of the distance based methods of Section 1.25.2. Let us partition the model set  $\Gamma = \{\Gamma_0, \Gamma_1, \Gamma_2, \dots\}$ , where the subsets  $\Gamma_k$  are the set of all candidate models with  $k$  free parameters. A candidate set  $\Gamma$  is considered nested if: (1) there is only one candidate model with  $k$  parameters ( $|\Gamma_k| = 1$  for all  $k$ ), and (2) models  $\gamma \in \Gamma_k$  with  $k$  parameters can exactly represent all distributions indexed by models with less than  $k$  parameters. That is, if  $\gamma_k \in \Gamma$  is a model with  $k$  free parameters, and model  $\gamma_l \in \Gamma$  is a model with  $l$  free parameters, where  $l < k$ , then

$$\forall \boldsymbol{\theta}_l \in \Theta_l, \exists \boldsymbol{\theta}_k \in \Theta_k : \forall \mathbf{y}, \forall p(\mathbf{y}|\boldsymbol{\theta}_l; \gamma_l) = p(\mathbf{y}|\boldsymbol{\theta}_k; \gamma_k).$$

A canonical example of a nested model structure is the polynomial order selection problem. Here, a second-order model can exactly represent any first-order model by setting the quadratic term to zero. Similarly, a third-order model can exactly represent any first-order or second-order model by setting the appropriate parameters to zero, and so on. In the non-nested case, there is not necessarily only a single model with  $k$  parameters, and there is no requirement that models with more free parameters can exactly represent all simpler models. This can introduce difficulties for some model selection criteria, and it is thus important to be aware of the structure of the set  $\Gamma$  before choosing a particular criterion to apply. The concept of nested and non-nested model sets will be made clearer in the following linear regression example.

### 1.25.1.2 Example: the linear model

Linear regression is of great importance in engineering and signal processing as it appears in a wide range of problems such as smoothing a signal with polynomials, estimating the number of sinusoids in noisy data and modeling linear dynamic systems. The linear regression model for explaining data  $\mathbf{y}$  assumes that the means of the observations are a linear combination of  $q$  ( $\geq 0$ ) measured variables, or

covariates, that is,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad (25.2)$$

where  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_q)$  is the *full* design matrix,  $\boldsymbol{\beta} \in \mathbb{R}^q$  are the parameter coefficients and  $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_n)'$  are independent and identically distributed Gaussian variates  $\varepsilon_i \sim N(0, \tau)$ . Model selection in the linear regression context arises because it is common to assume that only a subset of the covariates is associated with the observations; that is, only a subset of the parameter vector is non-zero. Let  $\gamma \subseteq \{1, \dots, q\}$  denote an index set determining which of the covariates comprise the design submatrix  $\mathbf{X}_\gamma$ ; the set of all candidate subsets is denoted by  $\Gamma$ . The linear model indexed by  $\gamma$  is then

$$\mathbf{y} = \mathbf{X}_\gamma \boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (25.3)$$

where  $\boldsymbol{\beta}$  is the parameter vector of size  $|\gamma|$  and

$$\mathbf{X}_\gamma = (\mathbf{x}_{\gamma_1}, \dots, \mathbf{x}_{\gamma_k}).$$

The total number of unknown parameters to be estimated for model  $\gamma$ , including the noise variance parameter, is  $k = |\gamma| + 1$ . The negative log-likelihood for the data  $\mathbf{y}$  is

$$-\log p(\mathbf{y} | \mathbf{X}_\gamma, \boldsymbol{\beta}, \tau; \gamma) = \frac{n}{2} \log 2\pi + \frac{n}{2} \log \tau + \frac{1}{2\tau} (\mathbf{y} - \mathbf{X}_\gamma \boldsymbol{\beta})' (\mathbf{y} - \mathbf{X}_\gamma \boldsymbol{\beta}). \quad (25.4)$$

The maximum likelihood estimates for the coefficient  $\boldsymbol{\beta}$  and the noise variance  $\tau$  are

$$\hat{\boldsymbol{\beta}}(\mathbf{y}; \gamma) = \left( \mathbf{X}'_\gamma \mathbf{X}_\gamma \right)^{-1} \mathbf{X}'_\gamma \mathbf{y}, \quad (25.5)$$

$$\hat{\tau}(\mathbf{y}; \gamma) = \frac{1}{n} (\mathbf{y} - \mathbf{X}_\gamma \hat{\boldsymbol{\beta}}(\mathbf{y}; \gamma))' (\mathbf{y} - \mathbf{X}_\gamma \hat{\boldsymbol{\beta}}(\mathbf{y}; \gamma)), \quad (25.6)$$

which coincide with the estimates obtained by minimising the sum of squared errors (the least squares estimates). The negative log-likelihood evaluated at the maximum likelihood estimates is given by

$$-\log p(\mathbf{y} | \mathbf{X}_\gamma, \hat{\boldsymbol{\beta}}(\mathbf{y}; \gamma), \hat{\tau}(\mathbf{y}; \gamma); \gamma) = \frac{n}{2} \log 2\pi + \frac{n}{2} \log \hat{\tau}(\mathbf{y}; \gamma) + \frac{n}{2}. \quad (25.7)$$

The structure of the set  $\Gamma$  in the context of the linear regression model depends on the nature and interpretation of the covariates. In the case that the covariates are polynomials of increasing order, or sinusoids of increasing frequency, it is often convenient to assume a nested structure on  $\Gamma$ . For example, let us assume that the  $q = 4$  covariates are polynomials such that  $\mathbf{x}_1$  is the zero-order term (constant),  $\mathbf{x}_2$  the linear term,  $\mathbf{x}_3$  the quadratic term and  $\mathbf{x}_4$  the cubic term. To use a model selection criterion to select an appropriate order of polynomial, we can form the set of candidate models  $\Gamma$  as

$$\Gamma = \{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\},$$

where

$$\Gamma_1 = \{1\}, \quad \Gamma_2 = \{1, 2\}, \quad \Gamma_3 = \{1, 2, 3\}, \quad \Gamma_4 = \{1, 2, 3, 4\}.$$

From this structure it is obvious that there is only one model with  $k$  free parameters, for all  $k$ , and that models with more parameters can exactly represent all models with less parameters by setting

the appropriate coefficients to zero. In contrast, consider the situation in which the covariates have no natural ordering, or that we do not wish to impose an ordering; for example, in the case of polynomial regression, we may wish to determine which, if any, of the individual polynomial terms are associated with the observations. In this case there is no clear way of imposing a nested structure, as we cannot *a priori* rule out any particular combination of covariates being important. This is usually called the *all-subsets* regression problem, and the candidate model set then has the following structure

$$\begin{aligned}\Gamma_0 &= \{\emptyset\}, \\ \Gamma_1 &= \{\{1\}, \{2\}, \{3\}, \{4\}\}, \\ \Gamma_2 &= \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}, \\ \Gamma_3 &= \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}\}, \\ \Gamma_4 &= \{1, 2, 3, 4\},\end{aligned}$$

where  $\emptyset$  denotes the empty set, that is, none of the covariates are to be used. It is clear that models with more parameters can not necessarily represent all models with less parameters; for example, the model  $\{1, 3\}$  cannot represent the model  $\{2\}$  as it is lacking covariate  $x_2$ . Further, there is now no longer just a single model with  $k$  free parameters; in the above example, there are four models with one free parameter, and six models with two free parameters. In general, if there are  $q$  covariates in total, the number of models with  $k$  free parameters is given by  $\binom{q}{k}$ , which may be substantially greater than one for moderate  $q$ .

Throughout the remainder of this chapter, the linear model example will be used to demonstrate the practical application of the model selection criteria that will be discussed.

## 1.25.2 Minimum distance estimation criteria

Intuitively, model selection criteria can be derived by quantifying “how close” the probability density of the fitted model is to the unknown, generating distribution. Many measures of distance between two probability distributions exist in the literature. Due to several theoretical properties, the Kullback-Leibler divergence [2], and its variants, is perhaps the most frequently used information theoretic measure of distance.

As the distribution  $p^*(\cdot)$  that generated the observations  $\mathbf{y}$  is unknown, the basic idea underlying the minimum distance estimation criteria is to construct an *estimate* of how close the fitted distributions are to the truth. In general, these estimators are constructed to satisfy the property of unbiasedness, at least for large sample sizes  $n$ . In particular, the distance-based methods we examine are the well known Akaike information criterion (AIC) (see Section 1.25.2.1) and symmetric Kullback information criterion (KIC) (see Section 1.25.2.2), as well as several corrected variants that improve the performance of these criteria when the sample size is small.

### 1.25.2.1 The Akaike information criterion

Recall the model selection problem as introduced in Section 1.25.1. Suppose a collection of  $n$  observations  $\mathbf{y} = (y_1, \dots, y_n)'$  has been sampled from an unknown distribution  $P^*(\mathbf{y})$  having density function

$p^*(\mathbf{y})$ . Estimation of  $p^*(\mathbf{y})$  is done within a set of nested candidate models  $\Gamma \subseteq \mathbb{N}$ , characterized by probability densities  $p(\mathbf{y}|\boldsymbol{\theta}; \gamma)$ ,  $\gamma \in \Gamma$ , where  $\boldsymbol{\theta} \in \Theta_\gamma \subseteq \mathbb{R}^k$  and  $k = \dim(\Theta_\gamma)$  is the number of free parameters possessed by model  $\gamma$ . Let  $\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma)$  denote the vector of estimated parameters obtained by maximizing the likelihood  $p(\mathbf{y}|\boldsymbol{\theta}; \gamma)$  over  $\Theta_\gamma$ , that is,

$$\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma) = \arg \max_{\boldsymbol{\theta} \in \Theta_\gamma} \{p(\mathbf{y}|\boldsymbol{\theta}; \gamma)\}$$

and let  $p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma)$  denote the corresponding fitted model. To simplify notation, we introduce the shorthand notation  $\hat{\boldsymbol{\theta}}_\gamma \equiv \hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma)$  to denote the maximum likelihood estimator, and  $p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma) \equiv p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma)$  to denote the maximized likelihood for model  $\gamma$ .

To determine which candidate density model best approximates the true unknown model  $p^*(\mathbf{y})$ , we require a measure which provides a suitable reflection of the separation between  $p^*(\mathbf{y})$  and an approximating model  $p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma)$ . The Kullback-Leibler divergence, also known as the cross-entropy, is one such measure. For the two probability densities  $p^* \equiv p^*(\mathbf{x})$  and  $p_{\boldsymbol{\theta}_\gamma} \equiv p(\mathbf{x}|\boldsymbol{\theta}; \gamma)$ , the Kullback-Leibler divergence,  $\Delta_n(\cdot)$ , between  $p^*(\mathbf{x})$  and  $p(\mathbf{x}|\boldsymbol{\theta}; \gamma)$  with respect to  $p^*(\mathbf{x})$  is defined as

$$\begin{aligned} 2\Delta_n(p^*, p_{\boldsymbol{\theta}_\gamma}) &= E_{\mathbf{x} \sim p^*} \left[ 2 \log \frac{p^*(\mathbf{x})}{p(\mathbf{x}|\boldsymbol{\theta}; \gamma)} \right] \\ &= E_{\mathbf{x} \sim p^*} [-2 \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)] - E_{\mathbf{x} \sim p^*} [-2 \log p^*(\mathbf{x})] \\ &= d_n(p^*, p_{\boldsymbol{\theta}_\gamma}) - d_n(p^*, p^*), \end{aligned}$$

where

$$d_n(p^*, p_{\boldsymbol{\theta}_\gamma}) = E_{\mathbf{x} \sim p^*} [-2 \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)], \quad (25.8)$$

and  $E_{\mathbf{x} \sim p^*}[\cdot]$  represents the expectation with respect to  $p^*(\mathbf{x})$ , that is,  $\mathbf{x} \sim p^*(\cdot)$ . Ideally, the selection of the best model from the set of candidate models  $\Gamma$  would be done by choosing the model with the minimum KL divergence from the data generating distribution  $p^*(\cdot)$ , that is

$$\hat{\gamma} = \arg \min_{\gamma \in \Gamma} \left\{ \Delta_n \left( p^*, p_{\hat{\boldsymbol{\theta}}_\gamma} \right) \right\},$$

where  $\hat{\boldsymbol{\theta}}_\gamma$  is the maximum likelihood estimator of the parameters of model  $\gamma$  based on the observed data  $\mathbf{y}$ . Since  $d_n(p^*, p^*)$  does not depend on the fitted model  $\hat{\boldsymbol{\theta}}_\gamma$ , any ranking of the candidate models according to  $\Delta_n(p^*, p_{\hat{\boldsymbol{\theta}}_\gamma})$  is equivalent to ranking the models according to  $d_n(p^*, p_{\hat{\boldsymbol{\theta}}_\gamma})$ . Unfortunately, evaluating  $d_n(p^*, p_\gamma)$  is not possible since doing so requires the knowledge of  $p^*(\cdot)$  which is the aim of model selection in the first place.

As noted by Takeuchi [3], twice the negative maximized log-likelihood,  $-2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma)$ , acts as a downwardly biased estimator of  $d_n(p^*, p_{\hat{\boldsymbol{\theta}}_\gamma})$  and is therefore not suitable for model comparison by itself. An unbiased estimate of the KL divergence is clearly of interest and could be used as a tool for model selection. It can be shown that the bias in estimation is given by

$$E_{\mathbf{y} \sim p^*} \left[ E_{\mathbf{x} \sim p^*} \left[ -2 \log p(\mathbf{x}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma) \right] \right] - E_{\mathbf{y} \sim p^*} \left[ -2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma) \right], \quad (25.9)$$

which, under suitable regularity conditions can be asymptotically estimated by [4,5]

$$2\text{Tr}(\boldsymbol{\Omega}^{-1}(\boldsymbol{\theta}_0; \gamma)\boldsymbol{\Sigma}(\boldsymbol{\theta}_0; \gamma)), \quad (25.10)$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\theta}_0; \gamma) = -E_{\mathbf{x} \sim p^*} \left[ \frac{\partial^2 \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \right] \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}, \quad (25.11)$$

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}_0; \gamma) = E_{\mathbf{x} \sim p^*} \left[ \left( \frac{\partial \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta}} \right) \left( \frac{\partial \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta}} \right)' \right] \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}, \quad (25.12)$$

and  $\text{Tr}(\cdot)$  denotes the trace of a matrix. The parameter vector  $\boldsymbol{\theta}_0$  defined by

$$\boldsymbol{\theta}_0 = \arg \min_{\boldsymbol{\theta} \in \Theta_\gamma} \{ \Delta_n(p^*, p_{\boldsymbol{\theta}}) \} \quad (25.13)$$

indexes the distribution in the model  $\gamma$  that is closest to the data generating distribution  $p^*(\cdot)$  in terms of KL divergence. Unfortunately, the parameter vector  $\boldsymbol{\theta}_0$  is unknown and so one cannot compute the required bias adjustment. However, Takeuchi [3] noted that under suitable regularity conditions, the maximum likelihood estimate can be used in place of  $\boldsymbol{\theta}_0$  to derive an approximation to (25.10), leading to the Takeuchi Information Criterion (TIC)

$$\text{TIC}(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma) + 2\text{Tr}(\boldsymbol{\Omega}^{-1}(\mathbf{y}; \gamma)\boldsymbol{\Sigma}(\mathbf{y}; \gamma)), \quad (25.14)$$

where

$$\boldsymbol{\Omega}(\mathbf{y}; \gamma) = - \frac{\partial^2 \log p(\mathbf{y}|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_\gamma}, \quad (25.15)$$

$$\boldsymbol{\Sigma}(\mathbf{y}; \gamma) = \sum_{i=1}^n \left( \frac{\partial \log p(y_i|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_\gamma} \right) \left( \frac{\partial \log p(y_i|\boldsymbol{\theta}; \gamma)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_\gamma} \right)' \quad (25.16)$$

are empirical estimates of the matrices (25.11) and (25.12), respectively. To use the TIC for model selection, we choose the model  $\gamma \in \Gamma$  with the smallest TIC score, that is

$$\hat{\gamma}_{\text{TIC}}(\mathbf{y}) = \arg \min_{\gamma \in \Gamma} \{ \text{TIC}(\gamma; \mathbf{y}) \}.$$

The model with the smallest TIC score corresponds to the model we believe to be the closest in terms of KL divergence to the data generating distribution  $p^*(\cdot)$ . Under suitable regularity conditions, the TIC is an unbiased estimate of the KL divergence between the fitted model  $p_{\hat{\boldsymbol{\theta}}_\gamma}$  and the data generating model  $p^*(\cdot)$ , that is

$$E_{\mathbf{y} \sim p^*} [\text{TIC}(\gamma; \mathbf{y})] = E_{\mathbf{y} \sim p^*} [d_n(p^*, p_{\hat{\boldsymbol{\theta}}_\gamma})] + o(1),$$

where  $o(1)$  is a quantity that vanishes as  $n \rightarrow \infty$ . The empirical matrices (25.15) and (25.16) are a good approximation to (25.11) and (25.12) only if the sample size  $n$  is very large. As this is often not the case in practice, the TIC is rarely used.

The dependence of the TIC on the empirical matrices can be avoided if one assumes that the data generating distribution  $p^*(\cdot)$  is a distribution contained in the model  $\gamma$ . In this case, the matrices (25.11) and (25.12) coincide. Thus,  $\text{Tr}(\boldsymbol{\Omega}^{-1}(\boldsymbol{\theta}_0; \gamma) \boldsymbol{\Sigma}(\boldsymbol{\theta}_0; \gamma))$  reduces to the number of parameters  $k$ , and we obtain the widely known and extensively used Akaike's Information Criterion (AIC) [6]

$$\text{AIC}(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y} | \hat{\boldsymbol{\theta}}_\gamma) + 2k. \quad (25.17)$$

An interesting way to view the AIC (25.17) is as a penalized likelihood criterion. For a given model  $\gamma$ , the negative maximized log-likelihood measures how well the model fits the data, while the “ $2k$ ” penalty measures the complexity of the model. Clearly, the complexity is an increasing function of the number of free parameters, and this acts to naturally balance the complexity of a model against its fit. Practically, this means that a complex model must fit the data well to be preferred to a simpler model with a similar quality of fit.

Similar to the TIC, the AIC is an asymptotically unbiased estimator of the KL divergence between the generating distribution  $p^*(\cdot)$  and the fitted distribution  $p_{\hat{\boldsymbol{\theta}}_\gamma}$ , that is

$$E_{\mathbf{y} \sim p^*} [\text{AIC}(\gamma; \mathbf{y})] = E_{\mathbf{y} \sim p^*} \left[ d_n(p^*, p_{\hat{\boldsymbol{\theta}}_\gamma}) \right] + o(1).$$

When the sample size is large and the number of free parameters is small relative to the sample size, the  $o(1)$  term is approximately zero, and the AIC offers an excellent estimate of the Kullback-Leibler divergence. However, in the case that  $n$  is small, or  $k$  is large relative to  $n$ , the  $o(1)$  term may be quite large, and the AIC does not adequately correct the bias, making it unsuitable for model selection.

To address this issue, Hurvich and Tsai [7] proposed a small sample correction to the AIC in the linear regression setting. This small sample AIC, referred to as  $\text{AIC}_c$ , is given by

$$\text{AIC}_c(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y}, \hat{\boldsymbol{\theta}}_\gamma) + \frac{2kn}{n - k - 1} \quad (25.18)$$

for a model  $\gamma$  with  $k$  free parameters, and has subsequently been applied to models other than linear regressions. From (25.18) it is clear that the penalty term is substantially larger than  $2k$  when the sample size  $n$  is not significantly larger than  $k$ , and that as  $n \rightarrow \infty$  the  $\text{AIC}_c$  is equivalent to the regular AIC (25.17). Practically, a large body of empirical evidence strongly suggests that the  $\text{AIC}_c$  performs significantly better as a model selection tool than the AIC, largely irrespective of the problem. The approach taken by Hurvich and Tsai to derive their  $\text{AIC}_c$  criterion is not the only way of deriving small sample minimum distance estimators based on the KL divergence; for example, the work in [8] derives alternative AIC-like criteria based on variants of the KL divergence, while [9] offers an alternative small sample criterion obtained through bootstrap approximations.

For the reader interested in the theoretical details and complete derivations of the AIC, including all regularity conditions, we highly recommend the excellent text by Linhart and Zucchini [5]. We also recommend the text by McQuarrie and Tsai [10] for a more practical treatment of AIC and  $\text{AIC}_c$ .

### 1.25.2.2 The Kullback information criterion

Since the Kullback-Leibler divergence is an asymmetric measure, an alternative directed divergence can be obtained by reversing the roles of the two models in the definition of the measure. A undirected

measure of model dissimilarity can be obtained from the sum of the two directed divergences. This measure is known as Kullback's symmetric divergence, or  $J$ -divergence [11]. Since the  $J$ -divergence combines information about model dissimilarity through two distinct measures, it functions as a gauge of model disparity which is arguably more sensitive than either of its individual components. The  $J$ -divergence,  $J_n(\cdot)$ , between the true generating distribution  $p^*(\cdot)$  and a distribution  $p_{\theta_\gamma}$  is given by

$$\begin{aligned} 2J_n(p^*, p_{\theta_\gamma}) &= 2\Delta_n(p^*, p_{\theta_\gamma}) + 2\Delta_n(p_{\theta_\gamma}, p^*) \\ &= d_n(p^*, p_{\theta_\gamma}) - d_n(p^*, p^*) + d_n(p_{\theta_\gamma}, p^*) - d_n(p_{\theta_\gamma}, p_{\theta_\gamma}), \end{aligned}$$

where

$$d_n(p_{\theta_\gamma}, p_{\theta_\gamma}) = \mathbb{E}_{\mathbf{x} \sim p_{\theta_\gamma}} [-2 \log p(\mathbf{x}|\boldsymbol{\theta}; \gamma)], \quad (25.19)$$

$$d_n(p_{\theta_\gamma}, p^*) = \mathbb{E}_{\mathbf{x} \sim p_{\theta_\gamma}} [-2 \log p^*(\mathbf{x})]. \quad (25.20)$$

As in Section (1.25.2.1), we note that the term  $d_n(p^*, p^*)$  does not depend on the fitted model  $\boldsymbol{\theta}_\gamma$  and may be dropped when ranking models. The quantity

$$K_n(p^*, p_\gamma) = d_n(p^*, p_{\theta_\gamma}) + d_n(p_{\theta_\gamma}, p^*) - d_n(p_{\theta_\gamma}, p_{\theta_\gamma})$$

may then be used as a surrogate for the  $J$ -divergence, and leads to an appealing measure of dissimilarity between the generating distribution and the fitted candidate model. In a similar fashion to the AIC, twice the negative maximized log-likelihood has been shown to be a downwardly biased estimate of the  $J$ -divergence. Cavanaugh [12] derives an estimate of this bias, and uses this correction to define a model selection criterion called KIC (symmetric Kullback information criterion), which is given by

$$\text{KIC}(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma) + 3k. \quad (25.21)$$

Comparing (25.21) to the AIC (25.17), we see that the KIC complexity penalty is slightly greater. This implies that the KIC tends to prefer simpler models (that is, those with less parameters) than those selected by minimising the AIC. Under suitable regularity conditions, the KIC satisfies

$$\mathbb{E}_{\mathbf{y} \sim p^*} [\text{KIC}(\gamma; \mathbf{y})] = \mathbb{E}_{\mathbf{y} \sim p^*} [K_n(p^*, p_{\theta_\gamma})] + o(1).$$

Empirically, the KIC has been shown to outperform AIC in large sample linear regression and autoregressive model selection, and tends to lead to less overfitting than AIC. Similarly to the AIC, when the sample size  $n$  is small the bias correction term  $3k$  is too small, and the KIC performs poorly as a model selection tool. Seghouane and Bekara [13] derived a corrected KIC, called  $\text{KIC}_c$ , in the context of linear regression that is suitable for use when the sample size is small relative to the total number of parameters  $k$ . The  $\text{KIC}_c$  is

$$\text{KIC}_c(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma) + \frac{2kn}{n-k-1} - n\psi\left(\frac{n-k-1}{2}\right) + n \log \frac{n}{2}, \quad (25.22)$$

where  $\psi(\cdot)$  denotes the digamma function. The digamma function can be difficult to compute, and approximating the digamma function by a second-order Taylor series expansion yields the so called AKIC<sub>c</sub>, which is given by

$$\text{AKIC}_c(\gamma; \mathbf{y}) = -2 \log p(\mathbf{y}|\hat{\boldsymbol{\theta}}_\gamma) + \frac{k(3n-k-1)}{n-k-1} + \frac{k-1}{n-k-1}. \quad (25.23)$$

The second term in (25.22) appears as the complexity penalty in the  $AIC_c$  (25.18), which is not surprising given that the  $J$ -divergence is a sum of two KL divergence functions. Empirically, the  $KIC_c$  has been shown to outperform the KIC as a model selection tool when the sample size is small.

### 1.25.2.3 Example application: linear regression

We now demonstrate the application of the minimum distance estimation criteria to the problem of linear regression introduced in Section 1.25.1.2. Recall that in this setting,  $\gamma$  specifies which covariates from the full design matrix  $\mathbf{X}$  should be used in the fitted model. The total number of parameters is therefore  $k = |\gamma| + 1$ , where the additional parameter accounts for the fact that we must estimate the noise variance. The various model selection criteria are listed below:

$$AIC(\gamma; \mathbf{y}) = n \log(2\pi\hat{\tau}(\mathbf{y}; \gamma)) + n + 2k, \quad (25.24)$$

$$AIC_c(\gamma; \mathbf{y}) = n \log(2\pi\hat{\tau}(\mathbf{y}; \gamma)) + n + \frac{2kn}{n - k - 1}, \quad (25.25)$$

$$KIC(\gamma; \mathbf{y}) = n \log(2\pi\hat{\tau}(\mathbf{y}; \gamma)) + n + 3k, \quad (25.26)$$

$$KIC_c(\gamma; \mathbf{y}) = n \log(2\pi\hat{\tau}(\mathbf{y}; \gamma)) + n + \frac{2kn}{n - k - 1} - n\psi\left(\frac{n - k - 1}{2}\right). \quad (25.27)$$

It is important to stress that all these criteria are derived within the context of a nested set of candidate models  $\Gamma$  (see Section 1.25.1.1). In the next section we will briefly examine the problems that can arise if these criteria are used in situations where the candidates are not nested.

### 1.25.2.4 Consistency and efficiency

Let  $\gamma^*$  denote the model that generated the observed data  $\mathbf{y}$  and assume that  $\gamma^*$  is a member of the set of candidate models  $\Gamma$  which is fixed and does not grow with sample size  $n$ ; recall that the model  $\gamma^*$  contains the data generating distribution  $p^*(\cdot)$  as a particular distribution. Furthermore, of all the models that contain  $p^*(\cdot)$ , the model  $\gamma^*$  has the least number of free parameters. A model selection criterion is consistent if the probability of selecting the true model  $\gamma^*$  approaches one as the sample size  $n \rightarrow \infty$ . None of the distance based criteria examined in this chapter are consistent model selection procedures. This means that even for extremely large sample sizes, there is a non-zero probability that these distance based criteria will overfit and select a model with more free parameters than  $\gamma^*$ . Consequently, if the aim of the experiment is to discover the true model, the aforementioned distance based criteria are inappropriate. For example, in the linear regression setting, if the primary aim of the model selection exercise is to determine only those covariates that are associated with the observations, one should not use the AIC or KIC (and their variants).

In contrast, if the true model is of infinite order, which in many settings may be deemed more realistic in practice, then the true model lies outside the class of candidate models. In this case, we cannot hope to discover the true model, and instead would like our model selection criterion to at least provide good predictions about future data arising from the same source. A criterion that minimizes the one-step mean squared error between the fitted model and the generating distribution  $p^*(\cdot)$  with high probability as  $n \rightarrow \infty$  is termed *efficient*. The AIC and KIC, and their corrected variants are all asymptotically efficient [4]. This suggests that for large sample sizes, model selection by distance based criteria will lead to adequate prediction errors even if the true model lies outside the set of candidate models  $\Gamma$ .

### 1.25.2.5 The AIC and KIC for non-nested sets of candidate models

The derivations of the AIC and KIC, and their variants, depend on the assumption that the candidate model set  $\Gamma$  is nested. We conclude this section by briefly examining the behavior of these criteria when they are used to select a model from a non-nested candidate model set. First, consider the case that  $\Gamma$  forms a nested set of models, and let  $\gamma^* \in \Gamma$  be the true model, that is, the model that generated the data; this means that  $\gamma^*$  is the model in  $\Gamma$  with the least number of free parameters, say  $k^*$ , that contains the data generating distribution  $p^*(\cdot)$  as a particular distribution. If  $n$  is sufficiently large, the probability that the model selected by minimising the AIC will have  $k^* + 1$  parameters is at least 16%, with the equivalent probability for KIC being approximately 8% [10].

Consider now the case in which  $\Gamma$  forms a non-nested set of candidate models. A recent paper by Schmidt and Makalic [14] has demonstrated that in this case, the AIC is a downwardly biased estimator of the Kullback-Leibler divergence even asymptotically in  $n$ , and this downward bias is determined by the size of the sets  $\Gamma_k$ . This implies, that in the case of all-subsets regression, the probability of overfitting is an increasing function of the number of covariates under consideration,  $q$ , and this probability cannot be reduced even by increasing the sample size.

As an example of how great this probability may be, we consider an all-subsets regression in which the generating distribution is the null model; that is, none of the measured covariates  $\mathbf{x}_i$  are associated with the observations  $\mathbf{y}$ . Even if there is only as few as  $q = 10$  covariates available for selection, the probability of erroneously preferring a non-null model to the null model using the AIC is approximately 81%, and by the time  $q = 50$ , this probability rises to 99.9%. Although [14] examines only the AIC, similar arguments follow easily for the KIC. It is therefore recommended that while both criteria are appropriate for nested model selection, neither of these distance based criteria should be used for model selection in a non-nested situation, such as the all-subsets regression problem.

### 1.25.2.6 Applications

Minimum distance estimation criteria have been widely applied in the literature, and we present below an inexhaustive list of successful applications:

- univariate linear regression models [7,13],
- multivariate linear regression models with arbitrary noise covariance matrices [15,16],
- autoregressive model selection [7,12,17],
- selection of smoothing parameters in nonparametric regression [18],
- signal denoising [19,20],
- selection of the size of a multilayer perceptron network [21].
- model selection in the presence of missing data [22].

For details of many of these applications the reader is referred to [10].

---

## 1.25.3 Bayesian approaches to model selection

Section 1.25.2 has described model selection criteria that are designed to explicitly minimise the distance between the estimated models and the true model that generated the observed data  $\mathbf{y}$ . An alternative

approach, based on Bayesian statistics [23], is now discussed. The Bayesian approach to statistical analysis differs from the “classical” approach through the introduction of a *prior distribution* that is used to quantify an experimenter’s *a priori* (that is, before the observation of data) beliefs about the source of the observed data. A central quantity in Bayesian analysis is the so-called *posterior distribution*; given a prior distribution,  $\pi_{\theta}(\theta|\gamma)$ , over the parameter space  $\theta \in \Theta_{\gamma}$ , and the likelihood function,  $p(\mathbf{y}|\theta, \gamma)$ , the posterior distribution of the parameters given the data may be found by applying Bayes’ theorem, yielding

$$p(\theta|\mathbf{y}, \gamma) = \frac{p(\mathbf{y}|\theta, \gamma)\pi_{\theta}(\theta|\gamma)}{p(\mathbf{y}|\gamma)}, \quad (25.28)$$

where

$$p(\mathbf{y}|\gamma) = \int_{\Theta_{\gamma}} p(\mathbf{y}|\theta, \gamma)\pi_{\theta}(\theta|\gamma)d\theta \quad (25.29)$$

is known as the marginal probability of  $\mathbf{y}$  for model  $\gamma$ , and is a normalization term required to render (25.28) a proper distribution. In general, the specification of the prior distribution may itself depend on further parameters, usually called *hyperparameters*, but for the purposes of the present discussion this possibility will not be considered. The main strength of the Bayesian paradigm is that it allows uncertainty about models and parameters to be defined directly in terms of probabilities; there is no need to appeal to more abstruse concepts, such as the “confidence interval” of classical statistics. However, there is in general no free lunch in statistics, and this strength comes at a price: the specification and origin of the prior distribution. There are essentially two main schools of thought on this topic: (i) subjective Bayesianism, where prior distributions are viewed as serious expressions of subjective belief about the process that generated the data, and (ii) objective Bayesianism [24], where one attempts to express prior ignorance through the use of uninformative, objective distributions, such as the Jeffreys’ prior [11], reference priors [25] and intrinsic priors [26]. The specification of the prior distribution is the basis of most criticism leveled at the Bayesian school, and an extensive discussion on the merits and drawbacks of Bayesian procedures, and the relative merits of subjective and objective approaches, is beyond the scope of this tutorial (there are many interesting discussions on this topic in the literature, see for example, [27, 28]). However, in this section of the tutorial, an approach to the problem of prior distribution specification based on asymptotic arguments will be presented.

While (25.28) specifies a probability distribution over the parameter space  $\Theta_{\gamma}$ , conditional on the observed data, it gives no indication of the likelihood of the model  $\gamma$  given the observed data. A common approach is based on the marginal distribution (25.29). If a further prior distribution,  $\pi_{\gamma}(\gamma)$ , over the set of candidate models  $\gamma \in \Gamma$  is available, one may apply Bayes’ theorem to form a posterior distribution over the models themselves, given by

$$p(\gamma|\mathbf{y}) = \frac{p(\mathbf{y}|\gamma)\pi_{\gamma}(\gamma)}{\sum_{\gamma' \in \Gamma} p(\mathbf{y}|\gamma')\pi_{\gamma'}(\gamma')}. \quad (25.30)$$

Model selection then proceeds by choosing the model  $\hat{\gamma}(\mathbf{y})$  that maximizes the probability (25.30), that is

$$\hat{\gamma}(\mathbf{y}) = \arg \max_{\gamma \in \Gamma} \{p(\mathbf{y}|\gamma)\pi_{\gamma}(\gamma)\}, \quad (25.31)$$

where the normalizing constant may be safely ignored. An interesting consequence of the posterior distribution (25.30) is that the posterior-odds in favor of some model,  $\gamma_1$ , over another model  $\gamma_0$ , usually

called the Bayes factor [29, 30], can be found from the ratio of posterior probabilities, that is,

$$\text{BF}(\gamma_1, \gamma_0) = \frac{p(\mathbf{y}|\gamma_1)\pi_\gamma(\gamma_1)}{p(\mathbf{y}|\gamma_0)\pi_\gamma(\gamma_0)}. \quad (25.32)$$

This allows for a straightforward and highly interpretable quantification of the uncertainty in the choice of any particular model. The most obvious weakness in the Bayesian approach, ignoring the origin of the prior distributions, is computational. As a general rule, the integral in the definition of the marginal distribution (25.29), does not admit a closed-form solution and one must resort to numerical approaches or approximations. The next section will discuss a criterion based on asymptotic arguments that circumvents both the requirement to specify an appropriate prior distribution as well as the issue of integration in the computation of the marginal distribution.

### 1.25.3.1 The Bayesian information criterion (BIC)

The Bayesian Information Criterion (BIC), also sometimes known as the Schwarz Information Criterion (SIC), was proposed by Schwarz [31]. However, the resulting criterion is not unique and was also derived from information theoretic considerations by Rissanen [32], as well as being implicit in the earlier work of Wallace and Boulton [33]; the information theoretic arguments for model selection are discussed in detail in the next section of this chapter. The BIC is based on the Laplace approximation for high dimensional integration [34]. Essentially, under certain regularity conditions, the Laplace approximation works by replacing the distribution to be integrated by a suitable multivariate normal distribution, which results in a straightforward integral. Making the assumptions that the prior distribution is such that its effects are “swamped” by the evidence in the data, and that the maximum likelihood estimator converges on the posterior mode as  $n \rightarrow \infty$ , one may apply the Laplace approximation to (25.29) yielding

$$\pi_\gamma(\gamma) \int_{\Theta_\gamma} p(\mathbf{y}|\boldsymbol{\theta}, \gamma) \pi_\boldsymbol{\theta}(\boldsymbol{\theta}|\gamma) d\boldsymbol{\theta} \simeq \frac{(2\pi)^{k/2} p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma), \gamma) \pi_\boldsymbol{\theta}(\hat{\boldsymbol{\theta}}_\gamma(\mathbf{y}; \gamma)|\gamma) \pi_\gamma(\gamma)}{|\mathbf{J}_n(\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma)|^{1/2}}, \quad (25.33)$$

where  $\simeq$  denotes that the ratio of the left and right hand side approaches one as the sample size  $n \rightarrow \infty$ . In (25.33),  $k = \dim(\Theta_\gamma)$  is the total number of continuous, free parameters for model  $\gamma$ ,  $\hat{\boldsymbol{\theta}}_\gamma(\mathbf{y}; \gamma)$  is the maximum likelihood estimate, and  $\mathbf{J}(\cdot)$  is the  $(k \times k)$  expected Fisher information matrix for  $n$  data points, given by

$$\mathbf{J}_n(\boldsymbol{\theta}_0; \gamma) = -E_{\boldsymbol{\theta}_0} \left[ \frac{\partial^2 \log p(\mathbf{y}|\boldsymbol{\theta}, \gamma)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} \right]. \quad (25.34)$$

The technical conditions under which (25.33) holds are detailed in [31]; in general, if the maximum likelihood estimates are consistent, that is, they converge on the true, generating value of  $\boldsymbol{\theta}$  as  $n \rightarrow \infty$ , and also satisfy the central limit theorem, the approximation will be satisfactory, at least for large sample sizes. To find the BIC, begin by taking negative logarithms of the right hand side of (25.33)

$$-\log p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma), \gamma) + \frac{1}{2} \log |\mathbf{J}_n(\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma); \gamma)| - \log \pi_\boldsymbol{\theta}(\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma)|\gamma) - \frac{k}{2} \log 2\pi - \log \pi_\gamma(\gamma). \quad (25.35)$$

A crucial assumption made is that the Fisher information satisfies

$$\mathbf{J}_1(\boldsymbol{\theta}; \gamma) = \lim_{n \rightarrow \infty} \left\{ \frac{\mathbf{J}_n(\boldsymbol{\theta}; \gamma)}{n} \right\}, \quad (25.36)$$

where  $\mathbf{J}_1(\cdot)$  is the asymptotic *per sample* Fisher information matrix satisfying  $|\mathbf{J}_1(\cdot)| > 0$ , and is free of dependency on  $n$ . This assumption is met by a large range of models, including linear regression models, autoregressive moving-average (ARMA) models and generalized linear models (GLMs) to name a few. This allows the determinant of the Fisher information matrix for  $n$  samples to be rewritten as

$$\begin{aligned} |\mathbf{J}_n(\boldsymbol{\theta}; \gamma)| &\approx n^k \cdot |\mathbf{J}_1(\boldsymbol{\theta}; \gamma)|, \\ &= n^k \cdot O(1), \end{aligned}$$

where  $O(1)$  denotes a quantity that is constant in  $n$ . Using this result, (25.35) may be dramatically simplified by assuming that  $n$  is large and discarding terms that are  $O(1)$  with respect to the sample size, yielding the BIC

$$\text{BIC}(\gamma; \mathbf{y}) = -\log p(\mathbf{y}|\hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma), \gamma) + \frac{k}{2} \log n. \quad (25.37)$$

Model selection using BIC is then done by finding the  $\gamma \in \Gamma$  that minimises the BIC score (25.37), that is,

$$\hat{\gamma}_{\text{BIC}}(\mathbf{y}) = \arg \min_{\gamma \in \Gamma} \{\text{BIC}(\gamma; \mathbf{y})\}. \quad (25.38)$$

It is immediately obvious from (25.37) that a happy consequence of the approximations that are employed is the removal of the dependence on the prior distribution  $\pi_{\boldsymbol{\theta}}(\cdot)$ . However, as usual, this comes at a price. The BIC satisfies

$$-\log \int_{\Theta_{\gamma}} p(\mathbf{y}|\boldsymbol{\theta}, \gamma) \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma) d\boldsymbol{\theta} = \text{BIC}(\gamma; \mathbf{y}) + O(1), \quad (25.39)$$

where the  $O(1)$  term is only guaranteed to be negligible for large  $n$ . For any finite  $n$ , this term may be arbitrarily large, and may have considerable effect on the behavior of BIC as a model selection tool. Thus, in general, BIC is only appropriate for use when the sample size is quite large relative to the number of parameters  $k$ . However, despite this drawback, when used correctly the BIC has several pleasant, and important, properties that are now discussed. The important point to bear in mind is that like all approximations, it can only be employed with confidence in situations that meet the conditions under which the approximation was derived.

### 1.25.3.2 Properties of BIC

#### 1.25.3.2.1 Consistency of BIC

A particularly important, and defining, property of BIC is the consistency of  $\hat{\gamma}_{\text{BIC}}(\mathbf{y})$  as  $n \rightarrow \infty$ . Let  $\gamma^* \in \Gamma$  be the true model, that is, the model that generated the data; this means that  $\gamma^*$  is the model in  $\Gamma$  with the least number of free parameters that contains the data generating distribution  $p^*(\cdot)$  as a particular distribution. Further, let the set of all models  $\Gamma$  be independent of the sample size  $n$ ; this means that the number of candidate models that are being considered does not increase with increasing sample size. Then, if all models in  $\Gamma$  satisfy the regularity conditions under which BIC was derived, and in particular (25.36), the BIC estimate satisfies

$$\Pr(\hat{\gamma}_{\text{BIC}}(\mathbf{y}) = \gamma^*) \rightarrow 1 \quad (25.40)$$

as  $n \rightarrow \infty$  [35,36]. In words, this says that the BIC estimate of  $\gamma$  converges on the true, generating model with probability one in the limit as the sample size  $n \rightarrow \infty$ , and this property holds irrespective of whether  $\Gamma$  forms a nested or non-nested set of candidate models. Practically, this means that as the sample size grows the probability that the model selected by the minimising the BIC score overfits or underfits the true, generating model, tends to zero. This is an important property if the discovery of relevant parameters is of more interest than making good predictions, and is not shared by any of the distance-based criteria discussed in Section 1.25.2. An argument against the importance of model selection consistency stems from the fact that the statistical models under consideration are abstractions of reality, and that the true, generating model is not a member of the set  $\Gamma$ . Regardless of this criticism, empirical studies suggest that the BIC tends to perform well in comparison to asymptotically efficient criteria such as AIC and KIC when the underlying generating process may be described by a small number of strong effects [10], and thus occupies a useful niche in the gallery of model selection techniques.

### 1.25.3.2.2 Bayesian Interpretations of BIC

Due to the fact that the BIC is an approximation of the marginal probability of data  $\mathbf{y}$  (25.29), it admits the full range of Bayesian interpretations. For example, by computing the BIC scores for two models  $\gamma_0$  and  $\gamma_1$ , one may compute an approximate negative log-Bayes factor of the form

$$-\log \text{BF}(\gamma_1, \gamma_0) \approx \text{BIC}(\gamma_0; \mathbf{y}) - \text{BIC}(\gamma_1; \mathbf{y}).$$

Therefore, the exponential of the negative difference in BIC scores for  $\gamma_1$  and  $\gamma_0$  may be interpreted as an approximate posterior-odds in favor of  $\gamma_1$ . Further, the BIC scores may be used to perform model averaging. Selection of a single best model from a candidate set is known to be statistically unstable [37], particularly if the sample size is small and many models have similar criterion scores. Instability in this setting is defined as the variability in the choice of the single best model under minor perturbations of the observed sample. It is clear that if many models are given similar criterion scores, then changing a single data point in the sample can lead to significant changes in the ranking of the models. In contrast to model selection, model averaging aims to improve predictions by using a weighted mixture of all the estimated models, the weights being proportional to the strength of evidence for the particular model. By noting that the BIC score is approximately equal to the negative log-posterior probability of the model, a Bayesian predictive density for future data  $\mathbf{y}_1$ , conditional on an observed sample  $\mathbf{y}_0$ , may be found by marginalising out the model class indicator  $\gamma$ , that is,

$$p(\mathbf{y}_1 | \mathbf{y}_0) \approx \frac{\sum_{\gamma \in \Gamma} p(\mathbf{y}_1 | \hat{\theta}(\mathbf{y}_0; \gamma), \gamma) e^{-\text{BIC}(\gamma; \mathbf{y}_0)}}{\sum_{\gamma \in \Gamma} e^{-\text{BIC}(\gamma; \mathbf{y}_0)}}. \quad (25.41)$$

Although such a mixture often outperforms a single best model when used to make predictions about future data [38], there are several drawbacks to the model averaging approach. The first is that the resulting density lacks the property of parsimony, as no model is excluded from the mixture, and therefore all model parameters contribute to the mixture. In the case of linear regression, for example, this means the ability to decide whether particular covariates are “relevant” is lost. The second drawback is that the predictive density (25.41) is in general not of the same form as any its component models, which can lead to interpretability issues.

### 1.25.3.3 Example application: linear regression

We conclude our examination of the BIC with an example of an application to linear regression models. Recalling that in this context the model indicator  $\gamma \subseteq \{1, \dots, q\}$  specifies which covariates, if any, should be used from the full design matrix  $\mathbf{X} \in \mathbb{R}^{(n \times q)}$  to explain the observed samples  $\mathbf{y}$ . The BIC score for a particular covariate set  $\gamma \in \Gamma$  is given by

$$\text{BIC}(\gamma; \mathbf{y}) = \frac{n}{2} \log(2\pi\hat{\tau}(\mathbf{y}; \gamma)) + \frac{n}{2} + \left(\frac{k+1}{2}\right) \log n, \quad (25.42)$$

where  $\hat{\tau}(\cdot; \gamma)$  is the maximum likelihood estimator of the noise variance for model  $\gamma$ , and the  $(k+1)$  accounts for the fact that the variance must be estimated from the data along with the  $k$  regression parameters. The BIC score (25.42) contains the terms  $(1/2)(n + \log n)$  which are constant with respect to  $\gamma$  and may be omitted if the set of candidate models only consists of linear regressions; if the set  $\Gamma$  also contains models from alternative classes, such as artificial neural networks, then these terms are required to fully characterize the marginal probability in comparison to this alternative class of models. The consistency property of BIC is particularly useful in this setting, as it guarantees that if the data were generated from (25.42) then as  $n \rightarrow \infty$ , minimising the BIC score will recover the true model, and thus determine exactly which covariates are associated with  $\mathbf{y}$ .

Given the assumption of Gaussian noise, the Bayesian mixture distribution is a weighted mixture of Gaussian distributions and thus has a particularly simple conditional mean. Let  $\boldsymbol{\alpha}(\gamma)$  denote a  $(q \times 1)$  vector with entries

$$\alpha_i(\gamma) = \begin{cases} \hat{\beta}_{(j: \gamma_j=i)}(\mathbf{y}; \gamma) & i \in \gamma \\ 0 & \text{otherwise} \end{cases},$$

that is, the vector  $\boldsymbol{\alpha}(\gamma)$  contains the maximum likelihood estimates for the covariates in  $\gamma$ , and zeros for those covariates that are not in  $\gamma$ . Then, the conditional mean of the predictive density for future data with design matrix  $\mathbf{X}_1$ , conditional on an observed sample  $\mathbf{y}_0$ , is simply a linear combination of the  $\boldsymbol{\alpha}(\gamma)$

$$E[\mathbf{y}_1 | \mathbf{X}_1, \mathbf{y}_0] = \mathbf{X}_1 \left( \sum_{\gamma \in \Gamma} \boldsymbol{\alpha}(\gamma) e^{-\text{BIC}(\gamma; \mathbf{y}_0)} \right),$$

which is essentially a linear regression with a special coefficient vector. While this coefficient vector will be dense, in the sense that none of its entries will be exactly zero, it retains the high degree of interpretability that makes linear regression models so attractive.

### 1.25.3.4 Priors for the model structure $\gamma$

One of the primary assumptions made in the derivation of the BIC is that the effects of the prior distribution for the model,  $\pi_\gamma(\cdot)$ , can be safely neglected. However, in some cases the number of models contained in  $\Gamma$  is very large relative to the sample size, or may even grow with growing sample size, and this assumption may no longer be valid. An important example is that of regression models in the context of genetic datasets; in this case, the number of covariates is generally several orders of magnitude greater than the number of samples. In this case, it is possible to use a modified BIC of the form

$$\text{BIC}(\gamma; \mathbf{y}) = -\log p(\mathbf{y} | \hat{\theta}(\mathbf{y}; \gamma), \gamma) + \frac{k}{2} \log n - \log \pi_\gamma(\gamma), \quad (25.43)$$

which requires specification of the prior distribution  $\pi_\gamma(\cdot)$ . In the setting of regression models, there are two general flavors of prior distributions for  $\gamma$  depending on the structure represented by  $\Gamma$ . The first case is that of *nested* model selection. A nested model class has the important property that all models with  $k$  parameters contain all models with less than  $k$  parameters as special cases. A standard example of this is polynomial regression in which the model selection criterion must choose the order of the polynomial. A uniform prior over  $\Gamma$  is an appropriate, and standard, choice of prior for nested model classes. Use of such a prior in (25.43) inflates the BIC score by a term of  $\log |\Gamma|$ ; this term is constant for all  $\gamma$ , and may consequently be ignored when using the BIC scores to rank the models.

In contrast, the second general case, known as *all-subsets regression*, is less straightforward. In this setting, there are  $q$  covariates, and  $\Gamma$  contains all possible subsets of  $\{1, \dots, q\}$ . This implies that the model class is no longer nested. It is tempting to assume a uniform prior over the members of  $\Gamma$ , as this would appear to be an uninformative choice. Unfortunately, such a prior is heavily biased towards subsets containing approximately half of the covariates. To see this, note that  $\Gamma$  contains a total of  $\binom{q}{k}$  subsets that include  $k$  covariates. This number may be extremely large when  $k$  is close to  $q/2$ , even for moderate  $q$ , and thus these subsets will be given a large proportion of the prior probability. For example, if  $q = 20$  then  $|\Gamma| \approx 10^6$  and  $\binom{20}{10} \approx 1.8 \times 10^5$ . Practically, this means that subsets with  $k = 10$  covariates are given approximately one-fifth of the total prior probability; in contrast, subsets with a single covariate receive less than one percent of the prior probability. To address this issue, an alternative approach is to use a prior which assigns equal prior probability to each set of subsets of  $k$  covariates, for all  $k$ , that is,

$$-\log \pi_\gamma(\gamma) = \log \binom{q}{|\gamma|} + \log(q+1). \quad (25.44)$$

This prior (or ones very similar) also arise from both empirical Bayes [39] and information theoretic arguments [27, 40], and have been extensively studied by Scott and Berger [41]. This work has demonstrated that priors of the form (25.44) help Bayesian procedures overcome the difficulties associated with all-subsets model selection that adversely affect distance based criteria such as AIC and KIC [14].

In fact, Chen and Chen [42] have proposed a generalization of this prior as part of what they call the “extended BIC.” An important (specific) result of this work is the proof that using the prior (25.44) relaxes conditions required for the consistency of the BIC. Essentially, the total number of covariates may now be allowed to depend on the sample size  $n$  in the sense that  $q = O(n^\kappa)$ , where  $\kappa$  is a constant satisfying  $0 < \kappa < \infty$ , implying that  $|\Gamma|$  grows with increasing  $n$ . Then, under certain identifiability conditions on the complete design matrix  $\mathbf{X}$  detailed in [42], and assuming that  $\gamma^* \in \Gamma$ , the BIC given by (25.43) using the prior (25.44) almost surely selects the true model  $\gamma^*$  as  $n \rightarrow \infty$ .

### 1.25.3.5 Markov-Chain Monte-Carlo Bayesian methods

We conclude this section by briefly discussing several alternative approaches to Bayesian model selection based on random sampling from the posterior distribution,  $p(\theta|y)$ , which fall under the general umbrella of Markov Chain Monte Carlo (MCMC) methods [43]. The basic idea is to draw a sample of parameter values from the posterior distribution, using a technique such as the Metropolis-Hastings algorithm [44, 45] or the Gibbs sampler [46, 47]. These techniques are in general substantially more complex than the information criteria based approaches, both computationally and in terms of implementation, and

this complexity generally brings with it greater flexibility in the specification of prior distributions as well as less stringent operating assumptions.

The most obvious way to apply MCMC methods to Bayesian model selection is through direct evaluation of the marginal (25.29). Unfortunately, this is a very difficult problem in general, and the most obvious approach, the so called harmonic mean estimator, suffers from statistical instability and convergence problems and should be avoided. In the particular case that Gibbs sampling is possible, and that the posterior is unimodal, Chib [48] has provided an algorithm to compute the approximate marginal probability from posterior samples.

An alternative to attempting to directly compute the marginal probability is to view the model indicator  $\gamma$  as the parameter of interest and randomly sample from the posterior  $p(\gamma|\mathbf{y})$ . This allows for the space of models  $\Gamma$  to be explored by simulation, and approximate posterior probabilities for each of the models to be determined by the frequency at which a particular model appears in the sample. There have been a large number of papers published that discuss this type of approach, and important contributions include the reversible jump MCMC algorithm of Green [49], the stochastic search variable selection algorithm [50], the shotgun stochastic search algorithm [51], and an interesting paper from Casella and Moreno [28] that combines objective Bayesian priors with a stochastic search procedure for regression models.

Finally, there has been a large amount of recent interest in using regularisation and “sparsity inducing” priors to combine Bayesian model selection with parameter estimation. In this approach, special priors over the model parameters are chosen that concentrate prior probability on “sparse” parameter vectors, that is, parameter vectors in which some of the elements are exactly zero. These methods have been motivated by the Bayesian connections with non-Bayesian penalized regression procedures such as the non-negative garotte [52] and the LASSO [53]. A significant advantage of this approach is that one needs only to sample from, or maximize over, the posterior of a single model containing all parameters of interest, and there is no requirement to compute marginal probabilities or to explore discrete model spaces. This is a rapidly growing area of research, and some important contributions of note include the relevance vector machine [54], Bayesian artificial neural networks [55] and the Bayesian LASSO [56,57].

## 1.25.4 Model selection by compression

This section examines the minimum message length (MML) (see Section 1.25.4.1) and minimum description length (MDL) (see Section 1.25.4.6) principles of model selection. Unlike the aforementioned distance based criteria, the MML and MDL principles are based on information theory and data compression. Informally, given a dataset and a set of candidate models, both MML and MDL advocate choosing the model that yields the briefest encoding of a hypothetical message comprising the model and the data. This optimal model is the simplest explanation of the data amongst the set of competing models. In this fashion, MML and MDL treat codelengths of compressed data as a measure of model complexity and, therefore, a yardstick for evaluating the explanatory power of competing models. Since data compression is equivalent to statistical learning of regularity in the data, this is intuitively a sensible procedure. It is important to note that one does not need to encode the models or the data in order to do MML and MDL inference. All that is required is to be able to calculate codelengths which can then be used for model comparison. Before discussing how MML and MDL compute codelengths of models

and data, a brief discussion of information theory is necessary. For a detailed treatment of information theory, the interested reader is recommended [58].

To aid in understanding the fundamental information theory concepts, it is helpful to consider the following hypothetical scenario. There exists a sender who is transmitting a data set (message) to a receiver over a transmission channel. The receiver does not know the content of the message. The message,  $\mathcal{M}$ , is recorded using some alphabet  $\mathcal{X}$ , such as a subset of the English alphabet  $\mathcal{X} = \{a, b, c\}$ , and comprises a sequence of symbols formed from  $\mathcal{X}$ ; for example, the message using  $\mathcal{X} = \{a, b, c\}$  may be  $\mathcal{M} = \{bcc\}$ . The transmission channel is noiseless and does not modify transmissions in any fashion. Without loss of generality, we assume the coding alphabet used on the transmission channel is the binary alphabet  $\mathcal{Y} = \{0, 1\}$ . Prior to sending the message, the sender and the receiver agree on a suitable codebook which will be used to transmit all messages on this channel. The codebook is a mapping  $C : \mathcal{X} \rightarrow \mathcal{Y}^*$  from the source alphabet  $\mathcal{X}$  to the target alphabet  $\mathcal{Y}^*$ , where  $\mathcal{Y}^*$  is the set of all strings obtained by concatenating elements of  $\mathcal{Y}$ . Some possible codebooks are:

$$C_1 = \{a \rightarrow 0, b \rightarrow 10, c \rightarrow 11\}, \quad (25.45)$$

$$C_2 = \{a \rightarrow 00, b \rightarrow 01, c \rightarrow 10\}, \quad (25.46)$$

$$C_3 = \{a \rightarrow 0, b \rightarrow 01, c \rightarrow 001\}. \quad (25.47)$$

For example, if the codebook  $C_1$  (25.45) is used by the sender, the message  $\{bcc\}$  is mapped to the target code  $\{1011110\}$ . If the sender uses codebook  $C_2$  instead, the message is encoded as  $\{01101000\}$ . The task of the sender and receiver is to choose a codebook such that the transmitted message is uniquely decodable by the receiver and as brief as possible.

If the codebook  $C$  possess the prefix property, a message transmitted using  $C$  is uniquely decodable requiring no delimiters between successive symbols. The prefix property states that no codeword is a prefix of any other codeword. In the above example, codebooks  $C_1$  and  $C_2$  possess the prefix property, while codebook  $C_3$  does not as the code 0 for source symbol  $a$  is a prefix to both codes 01 and 001. Since there exist infinitely many prefix codes, the sender and the receiver may choose the prefix code that results in the briefest possible encoding of the message  $\mathcal{M}$ . Intuitively, symbols in the message that appear more frequently should be assigned smaller codewords, while symbols that are very rare can be given longer codewords in order to reduce the average length of the encoding. This optimization problem is now formally explored. Note, that in the rest of the section we only consider codebooks which are uniquely decodable and possess the prefix property.

Let  $X$  denote a discrete random variable defined over the support (source alphabet)  $\mathcal{X}$  with probability distribution function  $p_X(\cdot)$ . The random variable  $X$  represents the sender of the message. Furthermore, let  $l : \mathcal{X} \rightarrow \mathbb{R}_+$  denote the codelength function which gives the codeword length for each symbol in the source alphabet  $\mathcal{X}$ . For example,  $l(a) = 1$  bit for the codebook  $C_1$ , while  $l(a) = 2$  bits, for the codebook  $C_2$ . The average codelength per symbol from source alphabet  $\mathcal{X}$  with probability distribution function  $p_X(\cdot)$  is defined as

$$E[l(X)] = \sum_{x \in \mathcal{X}} p_X(x)l(x), \quad (25.48)$$

where  $E$  denotes the expected value of a random variable. We wish to choose the function  $l(\cdot)$  such that the codelength for data from the random variable  $X$  is on average the shortest possible; that is we seek

a function  $l(\cdot)$  that minimises

$$\arg \min_{l(x), x \in \mathcal{X}} \{E[l(X)]\}. \quad (25.49)$$

The solution to this optimization problem is

$$l(X) = \log 1/p_X(x), \quad (25.50)$$

which is the Shannon information of a random variable [59]. The unit of information is derived from the base of the logarithm; commonly binary and natural logarithms are used corresponding to the bit and the nit (nat) units respectively. In the rest of this section, all logarithms are assumed to be natural logarithms, unless stated otherwise.

The Shannon information agrees with the previous intuition that high probability symbols should be assigned shorter codewords, while low probability symbols should be given longer code words. At first, it may seem troubling that Shannon information allows for non-integer length codewords. After all, how does one encode and transmit symbols of length, say, 0.2 of a nit? However, this is not an issue in practice as there exist sophisticated coding algorithms, such as arithmetic codes [60], that allow for efficient encoding in the presence of non-integer codewords. It is important to re-iterate here that MML and MDL model selection does not require any actual encoding of data to be performed; all that is required is a function to compute codelengths for data and models. Model selection is then performed by computing codelengths for all candidate models and choosing the one with the smallest codelength as optimal.

The minimum possible average codelength per symbol for a discrete random variable  $X$  is known as the Shannon entropy of  $X$  and is defined as

$$H(X) = E[\log 1/p_X(X)] = \sum_{x \in \mathcal{X}} p_X(x) \log 1/p_X(x) \quad (25.51)$$

with the convention that  $0 \log 0 = 0$ ; since  $x \log x \rightarrow 0$  as  $x \rightarrow 0$ , this is reasonable. The entropy of a random variable is always non-negative,  $H(X) \geq 0$ , and is a measure of uncertainty in the random variable. The maximum entropy is attained when there is maximum uncertainty in  $X$ , that is, when  $X$  is uniformly distributed over the support  $\mathcal{X}$ . For example, the entropy of a random variable  $X$  with support  $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$  and probability distribution function

$$p_X(j) = \frac{1}{6} \quad (j = 1, \dots, 6) \quad (25.52)$$

is approximately 2.585 bits or 1.792 nits. As another example, the entropy of random variable  $X$  following a Geometric distribution ( $\mathcal{X} = \{0, 1, 2, \dots\}$ ) with distribution function

$$p_X(X|\lambda) = (1 - \lambda)^x \lambda \quad (0 < \lambda \leq 1) \quad (25.53)$$

is given by

$$H(X|\lambda) = -\frac{1}{\lambda} [(1 - \lambda) \log(1 - \lambda) + \lambda \log \lambda], \quad (25.54)$$

where  $H(X|\lambda) = 0$  for  $\lambda = 1$ , and  $H(X|\lambda) \rightarrow \infty$  as  $\lambda \rightarrow 0$ .

The entropy of a random variable is the minimum average codelength per symbol with the length of each codeword  $X$  given by  $l(X) = \log 1/p_X(\cdot)$ . By the law of large numbers, it can also be shown that

using codelengths of the form  $l(X)$  results in a code that is optimal not only on average, but also optimal for the actually generated sequences of data. It is of interest to quantify the inefficiency resulting from encoding symbols generated by  $p_X(\cdot)$  using an alternative distribution function  $q_X(X)$ , where  $p_X(\cdot) \neq q_X(\cdot)$  for at least one  $X \in \mathcal{X}$ . The extra number of nits (or bits) required per symbol on average if  $q_X(\cdot)$  is used instead of  $p_X(\cdot)$  is given by the Kullback-Leibler (KL) divergence (see Section 1.25.2). The KL divergence is always non-negative, and is only equal to zero when  $p_X(\cdot) = q_X(\cdot)$ , for all  $X \in \mathcal{X}$ . In words, the briefest encoding for data from source  $p_X(\cdot)$  is achieved using  $p_X(\cdot)$  as the coding distribution and any other distribution, say  $q_X(\cdot)$ , will necessarily result in messages with a longer average codelength.

Recall that both MML and MDL seek models that result in the shortest codelength of a hypothetical message comprising the observed data and the model. That is, both MML and MDL seek models that result in the best compression of the observed data. Formally, this is a sensible procedure since the set of compressible strings from any data source is in fact quite small in comparison to incompressible strings from the same source. This is quantified by the following *no hypercompression inequality* (p. 136 [61]). Given a sample of  $n$  data points  $\mathbf{y} = (y_1, \dots, y_n)$  generated from a probability distribution  $p_X(\cdot)$ , the probability of compressing  $\mathbf{y}$  by *any* code is

$$p(\log 1/p_X(y_1, \dots, y_n) \geq \log 1/q_X(y_1, \dots, y_n) + K) \leq 2^{-K}. \quad (25.55)$$

In words, the inequality states that the probability of compressing  $\mathbf{y}$  by  $K$  bits using any code is small and decreases exponentially with increasing  $K$ . Clearly, the sender and the receiver would like  $K$  to be as large as possible leading to the briefest encoding of the observed data. For any moderate  $K$ , it is highly unlikely for a random model to result in a compression of  $K$ -bits as the number of models that can compress the data in such a manner is vanishingly small with  $K$ . In the next section, we examine model selection with the minimum message principle which is based on information theory and data compression.

#### 1.25.4.1 Minimum message length (MML)

The Minimum Message Length (MML) principle for model selection was introduced by Wallace [27] and collaborators in the late 1960s [33], and has been continuously refined in the following years [62, 63]. The MML principle connects the notion of compression with statistical inference, and uses this connection to provide a unified framework for model selection and parameter estimation. Recalling the transmitter-receiver framework outlined in Section 1.25.4, consider the problem of efficiently transmitting some observed data  $\mathbf{y}$  from the transmitter to the receiver. Under the MML principle, the model that results in the shortest encoding of a decodable message comprising the model and data is considered optimal. For this message to be decodable, the receiver and the transmitter must agree on a common language prior to any data being transmitted. In the MML framework, the common knowledge is a set of parametric models  $\Gamma$ , each comprising a set of distributions  $p(\mathbf{y}|\boldsymbol{\theta}, \gamma)$  indexed by  $\boldsymbol{\theta}$ , and a prior distribution  $\pi(\boldsymbol{\theta}, \gamma) = \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma)\pi_{\gamma}(\gamma)$ . The use of a subjective prior distribution means that MML is an unashamedly Bayesian principle built on information theoretic foundations.

Given a model from  $\gamma$  and a prior distribution  $\pi_{\boldsymbol{\theta}}(\cdot)$  over  $\Theta_{\gamma}$ , we may define an (implicit) prior probability distribution,  $r(\mathbf{y}|\gamma)$ , on the  $n$ -dimensional data space  $\mathcal{Y}^n$ . This distribution  $r(\cdot)$ , called the marginal distribution of the data, characterizes the probability of observing any particular data set  $\mathbf{y}$  given our choice of prior beliefs reflected by  $\pi_{\boldsymbol{\theta}}(\cdot)$ , and plays an important role in conventional Bayesian

inference (see Section 1.25.3). The marginal distribution is

$$r(\mathbf{y}|\gamma) = \int_{\boldsymbol{\theta} \in \Theta_\gamma} p(\mathbf{y}|\boldsymbol{\theta}, \gamma) \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma) d\boldsymbol{\theta}. \quad (25.56)$$

The receiver and the transmitter both have knowledge of the prior density  $\pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma)$  and the likelihood function  $p(\mathbf{y}|\boldsymbol{\theta}, \gamma)$ , and therefore the marginal distribution. Under the assumption that our prior beliefs accurately reflect the unknown process that generated the data  $\mathbf{y}$ , the average length of the shortest message that communicates the data  $\mathbf{y}$  to the receiver using the model  $\gamma$  is the entropy of  $r(\mathbf{y}|\gamma)$ . The marginal distribution is then used to construct a codebook for the data resulting in a decodable message with codelength

$$I_0(\mathbf{y}) = -\log r(\mathbf{y}|\gamma) = -\log \int_{\boldsymbol{\theta} \in \Theta_\gamma} p(\mathbf{y}|\boldsymbol{\theta}, \gamma) \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma) d\boldsymbol{\theta}. \quad (25.57)$$

This is the most efficient encoding of the data using the model  $\gamma$  under the assumed likelihood function and prior density. While the marginal distribution is commonly used in Bayesian inference for model selection (as discussed in Section 1.25.3), the fact that the parameter vector  $\boldsymbol{\theta}$  is integrated out means that it may not be used to make inferences about the particular distribution in  $\Theta_\gamma$  that generated the data. That is, the message does not convey anything about the quality of any particular distribution in  $\Theta_\gamma$  in regards to encoding the observed data and cannot be used for point estimation. Following Wallace ([27], pp. 152–153), we shall refer to this code as the *non-explanation* code for the data.

In the MML framework, the message transmitting the observed data comprises two components, commonly named the *assertion* and the *detail*. The assertion is a codeword naming a particular distribution indexed by  $\boldsymbol{\theta} \in \Theta_\gamma$  from the model  $\gamma$ , rather than a range of possible distributions in  $\gamma$ . The detail is a codeword that states the data  $\mathbf{y}$  using the distribution  $\boldsymbol{\theta}$  that was named in the assertion. The total, joint codelength of the message is then

$$I(\mathbf{y}, \boldsymbol{\theta}, \gamma) = I(\boldsymbol{\theta}, \gamma) + I(\mathbf{y}|\boldsymbol{\theta}, \gamma), \quad (25.58)$$

where  $I(\boldsymbol{\theta}, \gamma)$  and  $I(\mathbf{y}|\boldsymbol{\theta}, \gamma)$  denote the length of the assertion and detail respectively. The length of the assertion measures the complexity of a particular distribution  $\boldsymbol{\theta}$  in model  $\gamma$ , while the detail length measures the quality of the fit of the distribution  $\boldsymbol{\theta}$  to the data  $\mathbf{y}$ . Thus, the total, joint codelength automatically captures the tradeoff between model complexity and model capacity. Intuitively, a complex model with many free parameters can be tuned to fit a large range of data sets, leading to a short detail length. However, since all parameters need to be transmitted to the receiver, the length of the assertion will be long. In contrast, a simple model with few free parameters requires a short assertion, but may not fit the data well, potentially resulting in a long detail. This highlights the first defining feature of the MML principle: *the measure of model complexity and model fit are both expressed in the same unit, namely the codelength*. The minimum message length principle advocates choosing the model  $\gamma$  and distribution  $\boldsymbol{\theta}$  that minimises this two-part message, that is,

$$\left\{ \hat{\boldsymbol{\theta}}(\mathbf{y}; \gamma), \hat{\gamma}_{\text{MML}}(\mathbf{y}) \right\} = \arg \min_{\gamma \in \Gamma, \boldsymbol{\theta} \in \Theta_\gamma} \{I(\mathbf{y}, \boldsymbol{\theta}, \gamma)\}.$$

This expression highlights the second defining feature of the MML principle: *minimization of the codelength is used to perform both selection of a model structure,  $\gamma \in \Gamma$ , as well as estimation of the*

model parameters,  $\theta \in \Theta$ . This is in contrast to both the distance based criteria of Section 1.25.2, and the Bayesian information criterion introduced in Section 1.25.3, which only select a structure  $\gamma$  and do not provide an explicit framework for parameter estimation. It remains to determine how one computes the codelength (25.58).

In general, while the set of candidate models  $\Gamma$  will usually be countable,<sup>1</sup> the parameter space  $\Theta_\gamma$  for a model  $\gamma$  is commonly continuous. The continuity of  $\Theta_\gamma$  creates a problem for the transmitter when designing a codebook for the members of  $\Theta_\gamma$  needed for the assertion, as valid codebooks can only be constructed from discrete distributions. This problem may be circumvented by quantizing the continuum  $\Theta_\gamma$  into a discrete, countable subset  $\bar{\Theta}_\gamma = \{\bar{\theta}_1, \bar{\theta}_2, \dots\} \subset \Theta_\gamma$ . Given a subset  $\bar{\Theta}_\gamma$  we can define a distribution over  $\bar{\Theta}_\gamma$ , say  $h(\cdot|\gamma)$ . This distribution implicitly defines a codelength for the members of  $\bar{\Theta}_\gamma$ . The assertion length for stating a particular  $\theta \in \bar{\Theta}_\gamma$  is then  $I(\theta, \gamma) = -\log h(\theta|\gamma)\pi_\gamma(\gamma)$ . The length of the detail, encoded using the distribution indexed by  $\theta \in \bar{\Theta}_\gamma$  is  $I(\mathbf{y}|\theta, \gamma) = -\log p(\mathbf{y}|\theta, \gamma)$ , which is the negative log-likelihood of the observed data  $\mathbf{y}$ . The question remains: how do we choose the quantisation  $\bar{\Theta}_\gamma$  and the distribution  $h(\cdot|\gamma)$ ? The minimum message length principle specifies that the quantisation  $\bar{\Theta}_\gamma$  and the distribution  $h(\cdot|\gamma)$  be chosen such that the *average* codelength of the resulting two-part message is minimum, the average being taken with respect to the marginal distribution. Formally, for a model  $\gamma$ , the MML two-part codebook solves the following minimization problem:

$$\min_{\bar{\Theta}, h} \left\{ \sum_{\mathbf{y} \in \mathcal{Y}^n} r(\mathbf{y}|\gamma) I(\mathbf{y}, \theta, \gamma) \right\}. \quad (25.59)$$

In the literature, this procedure is referred to as strict minimum message length (SMML) [62]. For computational reasons, the SMML procedure commonly described in the literature solves an equivalent minimization problem by partitioning the dataspace  $\mathcal{Y}^n$ , which implicitly defines the parameter space quantisation  $\bar{\Theta}_\gamma$  and the distribution  $h(\cdot|\gamma)$  (for a detailed exposition the reader is referred to Chapter 3 of [27]). Model selection and parameter estimation by SMML proceeds by choosing the model/parameter pair from  $\bar{\Theta}_\gamma$  and  $\Gamma$  that results in the shortest two-part message, given  $h(\cdot|\gamma)$ . Interestingly, this is equivalent to performing *maximum a posteriori* (MAP) estimation over the quantised parameter space  $\bar{\Theta}_\gamma$ , treating  $h(\cdot|\gamma)$  as a special ‘‘quantised prior’’ which is implicitly defined by both the quantisation  $\bar{\Theta}_\gamma \subset \Theta_\gamma$  and the original continuous prior distribution  $\pi_\theta(\cdot)$  over  $\Theta_\gamma$ .

Analysis of the SMML code shows that it is approximately  $(1/2)(\log(k\pi) - 1)$  nits longer than the non explanation code (25.57), where  $k$  is the dimensionality of  $\Theta_\gamma$  (pp. 237–238, [27]). This is not surprising given that the SMML code makes a statement about the particular distribution from  $\Theta_\gamma$  which was used to code the data, rather than using a mixture of distributions as in the non-explanation code. This apparent ‘‘inefficiency’’ in SMML allows the codelength measure to be used not only for model selection (as in the case of the non-explanation code), but also provides a theoretically sound basis for parameter estimation.

While the SMML procedure possesses many attractive theoretical properties (see for example, pp. 187–195 of [27] and the ‘‘false oracle’’ property discussed in [64]), the minimization problem (25.59) is in general NP-hard [65], and exact solutions are computationally tractable in only a few

---

<sup>1</sup>A set  $A$  is countable if its members can be put into a one-to-one correspondence with the natural numbers, that is,  $A \leftrightarrow \mathbb{N}$ .

simple cases. The difficulties arise primarily due to the fact that a complete quantisation of the continuous parameter space must be found, which is a non-trivial problem that does not scale well with increasing dimensionality of  $\mathcal{Y}^n$ . In the next section we shall examine an approximation to the two-part message length that exploits the fact that we are only interested in computing the *length* of the codes, rather than the codebooks. This allows us to circumvent the problematic issue of complete quantisation by using only local properties of the likelihood and prior distribution, resulting in an implementation of the MML principle that is applicable to many commonly used statistical models.

### 1.25.4.2 The Wallace-Freeman message length approximation (MML87)

To circumvent the problem of determining a complete quantisation of the parameter space, Wallace and Freeman [63] presented a formula that gives an approximate length of the two-message for a particular  $\boldsymbol{\theta} \in \Theta_\gamma$  and data set  $\mathbf{y}$ . Rather than finding an optimal quantisation for the entire parameter space  $\Theta_\gamma$ , the key idea underlying their approach is to find an optimal *local* quantisation for the particular  $\boldsymbol{\theta}$  named in the assertion. The important implication of this approach is that the quantisation, and therefore the resulting message length formula, depends only on the particular  $\boldsymbol{\theta}$  that is of interest. Under certain regularity conditions, the Wallace-Freeman approximation states that the joint codelength of a distribution  $\boldsymbol{\theta} \in \Theta_\gamma$  in model  $\gamma$ , and data  $\mathbf{y}$ , is

$$I_{87}(\mathbf{y}, \boldsymbol{\theta}, \gamma) = -\underbrace{\log \pi_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\gamma) \pi_\gamma(\gamma)}_{I_{87}(\boldsymbol{\theta}, \gamma)} + \underbrace{\frac{1}{2} \log |\mathbf{J}_n(\boldsymbol{\theta}; \gamma)| + \frac{k}{2} \log \kappa_k + \frac{k}{2} - \log p(\mathbf{y}|\boldsymbol{\theta}, \gamma)}_{I_{87}(\mathbf{y}|\boldsymbol{\theta}, \gamma)}, \quad (25.60)$$

where  $k = \dim(\Theta_\gamma)$  is the total number of continuous, free parameters for model  $\gamma$  and  $\mathbf{J}(\cdot)$  is the Fisher information matrix for  $n$  data points given by

$$\mathbf{J}_n(\boldsymbol{\theta}_0; \gamma) = -\mathbf{E}_{\boldsymbol{\theta}_0} \left[ \frac{\partial^2 \log p(\mathbf{y}|\boldsymbol{\theta}, \gamma)}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} \right]. \quad (25.61)$$

The quantity  $\kappa_k$  is the normalized second moment of an optimal quantising lattice in  $k$ -dimensions ([66]). Following ([27], p. 237), the need to determine  $\kappa_k$  for arbitrary dimension  $k$  is circumvented by use of the following approximation

$$\frac{k}{2} (\log \kappa_k + 1) \approx -\frac{k}{2} \log (2\pi) + \frac{1}{2} \log (k\pi) + \psi(1) = c(k),$$

where  $\psi(\cdot)$  is the digamma function, and  $\psi(1) \approx -0.5772$ . Model selection, and parameter estimation, is performed by choosing the model/parameter pair that minimises the joint codelength, that is,

$$\left\{ \hat{\gamma}_{87}(\mathbf{y}), \hat{\boldsymbol{\theta}}_{87}(\mathbf{y}; \hat{\gamma}_{87}(\mathbf{y})) \right\} = \arg \min_{\gamma \in \Gamma, \boldsymbol{\theta} \in \Theta_\gamma} \{ I_{87}(\mathbf{y}, \boldsymbol{\theta}, \gamma) \}. \quad (25.62)$$

Unlike the SMML codelength, the MML87 codelength (25.60) is a continuous function of the continuous model parameters  $\boldsymbol{\theta}$  which makes the minimization problem (25.62) considerably easier. Again, it is important to re-iterate that unlike the distance based criteria (AIC, KIC, etc.) and the BIC, minimization

of the MML87 formula yields both estimates of model structure as well as model parameters. Further, the MML87 parameter estimates possess attractive properties in comparison with estimates based on other principles such as maximum likelihood or maximum posterior mode.

An important issue remains: under what conditions is the MML87 formula valid? In brief, the MML87 codelength is generally valid if: (i) the maximum likelihood estimates associated with the likelihood function obey the central limit theorem everywhere in  $\Theta_\gamma$ , and (ii) the local curvature of the prior distribution  $\pi_\theta(\cdot)$  is negligible compared to the curvature of the negative log-likelihood everywhere in  $\Theta_\gamma$ . An important point to consider is how close the Wallace–Freeman approximate codelength is to the exact SMML codelength given by (25.59). If the regularity conditions under which the MML87 formula was derived are satisfied, then the MML87 codelength is, on average, practically indistinguishable from the exact SMML codelength (pp. 230–231, [27]).

In the particular case that the prior  $\pi_\theta(\theta|\alpha, \gamma)$  is *conjugate*<sup>2</sup> with the likelihood  $p(\mathbf{y}|\theta, \gamma)$ , and depends on some parameters  $\alpha$  (usually referred to as “hyperparameters”), Wallace (pp. 236–237, [27]) suggested a clever modification of the MML87 formula that makes it valid even if the curvature of the prior is significantly greater than the curvature of the likelihood. The idea is to first propose some imaginary “prior data”  $\mathbf{y}_\alpha$  whose properties depend only on the prior hyperparameters  $\alpha$ . It is then possible to view the prior  $\pi_\theta(\theta|\alpha, \gamma)$  as a *posterior* of the likelihood of this prior data  $\mathbf{y}_\alpha$  and some initial uninformative prior  $\pi_0(\theta)$  that does not depend on the hyperparameters  $\alpha$ . Formally, we seek the decomposition

$$\pi_\theta(\theta|\alpha, \gamma) \propto \pi_0(\theta)p(\mathbf{y}_\alpha|\theta, \gamma), \quad (25.63)$$

where  $p(\mathbf{y}_\alpha|\theta, \gamma)$  is the likelihood of  $m$  imaginary prior samples and  $\pi_0(\theta)$  is an uninformative prior. The new Fisher Information  $\mathbf{J}_{n+m}(\theta; \gamma)$  is then constructed from the new combined likelihood  $p(\mathbf{y}, \mathbf{y}_\alpha|\theta, \gamma) = p(\mathbf{y}|\theta)p(\mathbf{y}_\alpha|\theta, \gamma)$ , and the “curved prior” MML87 approximation is simply

$$I_{87}^*(\mathbf{y}, \theta, \gamma) = -\log \pi_\theta(\theta|\alpha, \gamma)\pi_\gamma(\gamma) + \frac{1}{2} \log |\mathbf{J}_{n+m}(\theta; \gamma)| - \log p(\mathbf{y}|\theta, \gamma) + c(k), \quad (25.64)$$

$$= -\log \pi_0(\theta)\pi_\gamma(\gamma) + \frac{1}{2} \log |\mathbf{J}_{n+m}(\theta; \gamma)| - \log p(\mathbf{y}, \mathbf{y}_\alpha|\theta, \gamma) + c(k). \quad (25.65)$$

This new Fisher information matrix has the interesting interpretation of being the asymptotic lower bound of the inverse of the variance of the maximum likelihood estimator using the combined data  $(\mathbf{y}, \mathbf{y}_\alpha)$  rather than for simply the observed data  $\mathbf{y}$ . It is even possible in this case to treat the hyperparameters as unknown, and use an extended message length formula to estimate both the model parameters  $\theta$  and the prior hyperparameters  $\alpha$  in this hierarchical structure [67, 68].

What is the advantage of using the MML87 estimates over the traditional maximum likelihood and MAP estimates? In addition to possessing important invariance and consistency properties that are discussed in the next sections, there is a large body of empirical evidence demonstrating the excellent performance of the MML87 estimator in comparison to traditional estimators such as maximum likelihood and the *maximum a posteriori* (MAP) estimator, particularly for small to moderate sample sizes. Examples include the normal distribution [27], factor analysis [69], estimation of multiple means [67], the von Mises and spherical von Mises-Fisher distribution [70, 71] (pp. 266–269, [27])

---

<sup>2</sup>The use of a conjugate prior ensures that the posterior distribution of the parameters is of the same mathematical form as the conjugate prior distribution.

and autoregressive moving average models [72]. In the next sections we will discuss some important properties of the Wallace-Freeman approximation. For a detailed treatment of the MML87 approximation, we refer the interested reader to Chapter 5 of [27].

#### 1.25.4.2.1 Model selection consistency

Recall from Section 1.25.3.2 that the BIC procedure yields a consistent estimate of the true, underlying model, say  $\gamma^* \in \Gamma$ , that generated the data. The estimate  $\hat{\gamma}_{87}$  is also a consistent estimate of  $\gamma^*$  as  $n \rightarrow \infty$  under certain conditions. Assuming that the Fisher information matrix satisfies

$$\mathbf{J}_1(\boldsymbol{\theta}; \gamma) = \lim_{n \rightarrow \infty} \left\{ \frac{\mathbf{J}_n(\boldsymbol{\theta}; \gamma)}{n} \right\}, \quad (25.66)$$

with  $|\mathbf{J}_1(\cdot)| > 0$ , the MML87 codelength approximation can be rewritten as

$$I_{87}(\mathbf{y}, \boldsymbol{\theta}, \gamma) = -\log p(\mathbf{y}|\boldsymbol{\theta}, \gamma) + \frac{k}{2} \log n + O(1). \quad (25.67)$$

This is clearly equivalent to the well known BIC discussed in Section 1.25.3.1, and thus the estimator  $\hat{\gamma}_{87}$  is consistent as  $n \rightarrow \infty$ . For a more rigorous and general proof, the reader is directed to [73].

#### 1.25.4.2.2 Invariance

There is in general no unique way to parameterise the distributions that comprise a statistical model. For example, the normal distribution is generally expressed in terms of mean and variance parameters, say  $(\mu, \tau)$ . However, the alternate parameterisations  $(\mu, \sqrt{\tau})$  and  $(e^{-\mu}, \tau)$  are equally valid and there is no reason, beyond ease of interpretation, to prefer any particular parameterisation. A sensible estimation procedure should be invariant to the parameterisation of the model; that is, it should give the same answer irrespective of how the inference question is framed. While the commonly used maximum likelihood estimator possesses the invariance property, both the *maximum a posteriori* estimator and posterior mean estimator based on the Bayesian posterior distribution  $p(\boldsymbol{\theta}|\mathbf{y}, \gamma)$  given by (25.28) are *not invariant*. The SMML and MML87 are *invariant* under one-to-one reparameterisations, and thus provide an important alternative Bayesian estimation procedure.

#### 1.25.4.2.3 Parameter estimation consistency

Consider the case that the data  $\mathbf{y}$  is generated by a particular distribution  $\boldsymbol{\theta}^*$  from the model  $\gamma$ , that is,  $\boldsymbol{\theta}^* \in \Theta_\gamma$ . A sequence of estimators,  $\hat{\boldsymbol{\theta}}_n$ , of parameter  $\boldsymbol{\theta}$  is consistent if and only if

$$\Pr \left\{ |\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}^*| \geq \varepsilon \right\} \rightarrow 0$$

for any  $\varepsilon > 0$  as  $n \rightarrow \infty$ . In words, this means that as we accumulate more and more data about the parameters the estimator converges to the true parameter value  $\boldsymbol{\theta}^*$ . If we consider the case that the dimensionality of  $\Theta_\gamma$  does not depend on  $n$ , and the Fisher information matrix satisfies (25.36) then the maximum likelihood estimator is consistent under suitable regularity conditions [74]. The consistency of the MML87 estimator in this particular case follows by noting that minimising the asymptotic message length (25.67) is equivalent to maximizing the likelihood; similar arguments can also be used to establish the consistency of the MAP and mean posterior estimators in this case.

The situation is strikingly different in the case that the dimensionality of the parameter space depends on the sample size. In this setting there is usually a small number of parameters of interest for which information grows with increasing sample size, and a large number of auxiliary “nuisance” parameters for which an increasing sample size brings no new information. Maximum likelihood estimation, as well as the MAP and mean posterior estimation, of the parameters of interest is in general inconsistent. In contrast, the MML87 estimator has been shown to provide consistent estimates in the presence of many nuisance parameters in several important statistical models, including the Neyman-Scott problem [75], factor analysis [69, 76] (also pp. 297–303 in [27]), multiple cutpoint estimation [77] and mixture modeling ([78] and pp. 275–297 in [27]). While a general proof of the consistency of the MML principle in the presence of nuisance parameters does not currently exist, *there have been no problems studied so far in which it has failed to yield consistent estimates*.

#### 1.25.4.2.4 Similarities with Bayesian inference

There are many similarities between the minimum message length principle and the more conventional Bayesian approaches such as those discussed in Section 1.25.3 (see [79] for a detailed discussion). The differences in codelengths (MML87 or otherwise) between two models may be used to compute an approximate Bayes factor, that is,

$$-\log \text{BF}(\gamma_1, \gamma_0) \approx I(\mathbf{y}, \boldsymbol{\theta}, \gamma_0) - I(\mathbf{y}, \boldsymbol{\theta}, \gamma_1).$$

The usual interpretations applied to regular Bayes factors apply equally well to the approximate Bayes factors determined by codelength differences. Further, the codelength measure may be used to perform model averaging over the set of candidate models  $\Gamma$  in a similar fashion to BIC, as discussed in Section 1.25.3.1. In this case, we use (25.41), replacing the BIC score with the codelength, and (ideally) replacing the maximum likelihood estimate with the appropriate minimum message length estimate.

#### 1.25.4.3 Other message length approximations

Recently, Schmidt [72, 80] introduced a new codelength approximation that can be used for models for which the application of the Wallace-Freeman approximation is problematic. The new approximation, called MML08, is more computationally tractable than the strict MML procedure, but requires evaluation of sometimes difficult integrals. The joint MML08 codelength of data  $\mathbf{y}$  and parameters  $\boldsymbol{\theta}$ , for model  $\gamma$ , is given by

$$\begin{aligned} I_{08}(\boldsymbol{\theta}, \mathbf{y}, \gamma) &= \underbrace{-\log \pi_\gamma(\gamma) \int_{\Omega_\theta} \pi_\theta(\boldsymbol{\phi}|\gamma) d\boldsymbol{\phi}}_{I_{08}(\boldsymbol{\theta}, \gamma)} \\ &\quad - \underbrace{\log p(\mathbf{y}|\boldsymbol{\theta}, \gamma) + \left( \frac{1}{\int_{\Omega_\theta} \pi(\boldsymbol{\phi}|\gamma) d\boldsymbol{\phi}} \right) \int_{\Omega_\theta} \pi_\theta(\boldsymbol{\phi}|\gamma) \Delta(\boldsymbol{\theta}||\boldsymbol{\phi}) d\boldsymbol{\phi}}_{I_{08}(\mathbf{y}|\boldsymbol{\theta}, \gamma)}, \end{aligned} \quad (25.68)$$

where  $\Delta(\cdot)$  is the Kullback-Leibler divergence and  $\Omega_\theta \subset \Theta_\gamma$  is chosen to minimise the codelength. The MML08 is a generalization of the previously proposed MMLD codelength approximation [81, 82]

and is invariant under transformations of the parameters. The MML08 approximation has been applied to the problem of order selection for autoregressive moving average models [72]. Efficient algorithms to compute approximate MMLD and MML08 codelengths are given in [72, 83, 84].

#### 1.25.4.4 Example: MML linear regression

We continue our running example by applying the minimum message length principle to the selection of covariates in the linear regression model. By choosing different prior distributions for the coefficients  $\beta$  we can arrive at many different MML criteria (for example, [85, 86] and pp. 270–272, [27]); however, provided the chosen prior distribution is not unreasonable all codelength criteria will perform approximately equivalently, particularly for large sample sizes. Two recent examples of MML linear regression criteria, called “MML<sub>u</sub>” and “MML<sub>g</sub>,” that do not require the specification of any subjective prior information are given in [68]. The MML<sub>u</sub> criterion exploits some properties of the linear model with Gaussian noise to derive a data driven, proper uniform prior for the regression coefficients. Recall that in the linear regression case, the model index  $\gamma$  specifies the covariates to be used from the full design matrix  $\mathbf{X}$ . For a given  $\gamma$ , the MML<sub>u</sub> codelength for a linear regression model is

$$\begin{aligned} & \left(\frac{n-k}{2}\right) \log 2\pi + \left(\frac{n-k}{2}\right) (\log \hat{\tau}_{87}(\mathbf{y}; \gamma) + 1) + \frac{k}{2} \log (\pi_\gamma' \mathbf{y}) \\ & - \log \Gamma\left(\frac{k}{2} + 1\right) + \frac{1}{2} \log (k+1) - \log \pi_\gamma(\gamma), \end{aligned} \quad (25.69)$$

where the MML<sub>u</sub> parameter estimates are given by

$$\hat{\tau}_{87}(\mathbf{y}; \gamma) = \left(\frac{1}{n-k}\right) (\mathbf{y}' \mathbf{y} - R(\gamma)), \quad (25.70)$$

$$\hat{\beta}_{87}(\mathbf{y}; \gamma) = \left(\mathbf{X}_\gamma' \mathbf{X}_\gamma\right)^{-1} \mathbf{X}_\gamma' \mathbf{y}. \quad (25.71)$$

Here,  $k = |\gamma|$  denotes the number of covariates in the regression model  $\gamma$ ,  $\pi_\gamma(\gamma)$  is a suitable prior for the model index (see Section 1.25.3.4 for a discussion of suitable priors), and

$$R(\gamma) = \hat{\beta}(\mathbf{y}; \gamma)' (\mathbf{X}_\gamma' \mathbf{X}_\gamma) \hat{\beta}(\mathbf{y}; \gamma) \quad (25.72)$$

is the fitted sum-of-squares for the least-squares estimates (25.5). Due to the use of a uniform prior on the regression coefficients, and the nature of the Fisher information matrix, the MML87 estimates of the regression coefficients coincide with the usual maximum likelihood estimates (25.5). In contrast to the maximum likelihood estimate of  $\tau$ , given by (25.6), the MML87 estimate is exactly unbiased even for finite sample sizes  $n$ .

The MML<sub>g</sub> criterion further demonstrates the differences in parameter estimation that are possible by using the MML principle. This criterion is based on a special hierarchical Gaussian prior, and uses some recent extensions to the MML87 codelength to estimate the parameters of the prior distribution in addition to the regression coefficients [67]. For a given  $\gamma$ , the MML<sub>g</sub> codelength for a linear regression model is

$$\left(\frac{n-k+2}{2}\right) (\log \hat{\tau}_{87}(\mathbf{y}; \gamma) + 1) + \left(\frac{k-2}{2}\right) \log \left(\frac{R(\gamma)}{\delta}\right) + \frac{\delta}{2} + \frac{1}{2} \log (n-k) k^2 - \log \pi_\gamma(\gamma), \quad (25.73)$$

where  $\delta = \max(1, k - 2)$ . This formula is applicable only when  $k > 0$ , and  $m(\gamma) > 0$ , where

$$m(\gamma) = \left( \frac{R(\gamma)}{\delta} - \hat{\tau}_{87}(\mathbf{y}; \gamma) \right)_+,$$

and  $(\cdot)_+ = \max(0, \cdot)$  is the positive-part function. The codelength for a “null” model ( $k = 0$ ) is given by

$$\binom{n}{2} (\log \hat{\tau}_{87}(\mathbf{y}; \gamma) + 1) + \frac{1}{2} \log(n-1) + \frac{1}{2}. \quad (25.74)$$

We note that (25.73) corrects a minor mistake in the simplified form of  $\text{MML}_g$  in [68], which erroneously excluded the additional “ $\delta/2$ ” term. The  $\text{MML}_g$  estimates of  $\beta$  and  $\tau$  are given by

$$\hat{\tau}_{87}(\mathbf{y}; \gamma) = \left( \frac{1}{n-k+2} \right) (\mathbf{y}'\mathbf{y} - R(\gamma)), \quad (25.75)$$

$$\hat{\beta}_{87}(\mathbf{y}; \gamma) = \left( \frac{m(\gamma)}{m(\gamma) + \hat{\tau}_{87}(\mathbf{y}; \gamma)} \right) (\mathbf{X}'_{\gamma} \mathbf{X}_{\gamma})^{-1} \mathbf{X}'_{\gamma} \mathbf{y}, \quad (25.76)$$

which are closely related to the unbiased estimate (25.70) of  $\tau$  used by the  $\text{MML}_u$  criterion. The  $\text{MML}_g$  estimates of the regression coefficients  $\beta$  are the least-squares estimates scaled by a quantity less than unity which is called a *shrinkage* factor [87]. This factor depends crucially on the estimated strength of the noise variance and signal strength, and if the noise variance is too large,  $m(\gamma) = 0$  and the coefficients are completely shrunk to zero (that is, none of the covariates are deemed associated with the observations). Further, it turns out that the  $\text{MML}_g$  shrinkage factor is optimal in the sense that for  $k \geq 3$ , the shrunken estimates of  $\beta$  will, on average, be better at predicting future data arising from the same source than the corresponding least squares estimates [88].

#### 1.25.4.5 Applications of MML

The MML principle has been applied to a large number statistical models. Below is an inexhaustive list of some of the many successful applications of MML; a more complete list may be found in [82].

- Mixture modeling [33, 78, 89].
- Decision trees/graphs [90–92].
- Casual Networks [93, 94].
- Artificial neural networks [95, 96].
- Linear Regression [68, 85, 86, 97].
- Autoregressive moving average (ARMA) models [72, 98, 99].
- Discrete time series [77, 100].
- Hierarchical Bayesian Models [67, 68].

Detailed derivations of the message length formulas for many of these models, along with discussions of their behavior, can also be found in Chapters 6 and 7 of [27].

#### 1.25.4.6 The minimum description length (MDL) principle

The minimum description length (MDL) principle [101, 102] was independently developed by Jorma Rissanen from the late 1970s [32, 103, 104], and embodies the same idea of statistical inference via data

compression as the MML principle. While there are many subtle, and not so subtle, differences between MML and MDL, the most important is the way in which they view prior distributions (for a comparison of early MDL and MML, see [105]). The MDL principle views prior distributions purely as devices with which to construct codelengths, and the bulk of the MDL research has been focused on finding ways in which the specification of subjective prior distributions may be avoided. In the MML principle, the codebook is constructed to minimise the average codelength of data drawn from the marginal distribution, which captures our prior beliefs about the data generating source. The MDL principle circumvents the requirement to explicitly specify prior beliefs, and instead appeals to the concept of minimising the worst-case behavior of the codebook. Formally, we require the notion of coding *regret*

$$R(\mathbf{y}) = I(\mathbf{y}, \gamma) + \log p(\mathbf{y}|\hat{\theta}(\mathbf{y}; \gamma), \gamma), \quad (25.77)$$

where  $I(\mathbf{y}, \gamma)$  is the codelength of the data  $\mathbf{y}$  using model  $\gamma$ , and  $\hat{\theta}(\mathbf{y}; \gamma) \in \Theta_\gamma$  is the maximum likelihood estimator. As the maximum likelihood estimator minimises the codelength of the data (without stating anything about the model), the negative log-likelihood evaluated at the maximum represents an ideal target codelength for the data. Unfortunately, this is only realizable if the sender and receiver have *a priori* knowledge of the maximum likelihood estimator, which in an inferential setting is clearly nonsensical.

Clearly, one would like to keep the regret as small as possible in some sense, to ensure that the chosen codebook is efficient. The MML approach minimises average excess codelength over the marginal distribution, which requires an explicit prior distribution. To avoid this requirement, Rissanen advocates choosing a codebook that minimises the maximum (worst-case) coding regret (25.77), that is finding the probability distribution  $f$  that solves the following problem:

$$\min_f \left\{ \max_{\mathbf{y} \in \mathcal{Y}^n} \left\{ -\log f(\mathbf{y}) + \log p(\mathbf{y}|\hat{\theta}(\mathbf{y}; \gamma), \gamma) \right\} \right\}. \quad (25.78)$$

The solution to this optimization problem is known as the normalized maximum likelihood (NML) [106, 107] (or Shtarkov [108]) distribution, and the resulting codelength is given by

$$I_{\text{NML}}(\mathbf{y}, \gamma) = -\log p(\mathbf{y}|\hat{\theta}(\mathbf{y}; \gamma), \gamma) + \log \int_{\mathbf{x} \in \mathcal{X}^n} p(\mathbf{x}|\hat{\theta}(\mathbf{x}; \gamma), \gamma) d\mathbf{x}. \quad (25.79)$$

In MDL parlance, the second term on the right hand side of (25.79) is called the *parametric complexity* of the model  $\gamma$ . The parametric complexity has several interesting interpretations: (i) it is the logarithm of the normalizing constant required to render the infeasible maximum likelihood codes realizable; (ii) it is the (constant) regret obtained by the normalized maximum likelihood codes with respect to the unattainable maximum likelihood codes, and (iii) it is the logarithm of the number of distinguishable distributions contained in the model  $\gamma$  at sample size  $n$  [109]. In addition to the explicit lack of prior distributions, the NML code (25.79) differs from the MML two-part codes in the sense that it does not explicitly depend on any particular distribution in the model  $\gamma$ . This means that like the marginal distribution discussed in Section 1.25.4.1, the NML codes cannot be used for point estimation of the model parameters  $\theta_\gamma$ .

Practically, the normalizing integral in (25.79) is difficult to compute for many models. Rissanen has derived an asymptotic approximation to the NML distribution which is applicable to many commonly used statistical models (including for example, linear models with Gaussian noise and autoregressive

moving average models) [106]. Under this approximation, the codelength formula is given by

$$I_{\text{ANML}}(\mathbf{y}, \gamma) = -\log p(\mathbf{y}|\hat{\theta}(\mathbf{y}; \gamma), \gamma) + \frac{k}{2} \log \left( \frac{n}{2\pi} \right) + \log \int_{\theta \in \Theta_\gamma} \sqrt{|\mathbf{J}_1(\theta; \gamma)|} d\theta, \quad (25.80)$$

where  $k$  is the number of free parameters for the model  $\gamma$ ,  $n$  is the sample size and  $\mathbf{J}_1(\cdot)$  is the per sample Fisher information matrix given by (25.36). The approximation (25.80) swaps an integral over the dataspace for an often simpler integral over the parameter space. For models that satisfy the regularity conditions discussed in [106], the approximate NML codelength satisfies

$$I_{\text{NML}}(\mathbf{y}, \gamma) = I_{\text{ANML}}(\mathbf{y}, \gamma) + o(1),$$

where  $o(1)$  denotes a term that disappears as  $n \rightarrow \infty$ .

The NML distribution is only one of the coding methods considered by Rissanen and co-workers in the MDL principle; the interested reader is referred to Grünwald's book [61] for a detailed discussion of the full spectrum of coding schemes developed within the context of MDL inference.

#### 1.25.4.7 Problems with divergent parametric complexity

The NML distribution offers a formula for computing codelengths that is free of the requirement to specify suitable prior distributions, and therefore appears to offer a universal approach to model selection by compression. Unfortunately, the formula (25.79) is not always applicable as the parametric complexity can diverge, even in commonly used models [110]; this also holds for the approximate parametric complexity in (25.80). To circumvent this problem, Rissanen [106] recommends bounding either the dataspace, for (25.79), or the parameter space, for (25.80), so that the parametric complexity is finite. In the case of linear regression models with Gaussian noise, Rissanen extends this idea even further by treating the hyperparameters that specify the bounding region as parameters, and re-applying the NML formula to arrive at a parameter-free criterion (see [111] for details). Other models with infinite parametric complexity which have been addressed in a similar manner include the Poisson and geometric distributions [110] and stationary autoregressive models [112].

Unfortunately, the choice of bounding region is in general arbitrary, and different choices will lead to different codelength formulae, and hence different behavior. In this sense, the choice of bounding region is akin to the selection of a prior distribution in MML; however, advocates of the Bayesian approach argue that selection of a prior density is significantly more transparent, and interpretable, than the choice of a bounding region.

#### 1.25.4.8 Sequential variants of MDL

One way to overcome the problems of infinite parametric complexity is to code the data sequentially. In this framework, one uses the first  $t$  datapoints,  $(y_1, \dots, y_t)$ , to construct a predictive model that is used to code  $y_{t+1}$ . This process may be repeated until the entire dataset  $(y_1, \dots, y_n)$  has been “transmitted”, and the resulting codelength may be used for model selection. This idea was first proposed as part of the predictive MDL procedure [101, 113].

More recently, the idea has been refined with the introduction of the sequentially normalized maximum likelihood (SNML) model [114], and the related conditional normalized maximum likelihood (CNML) model [115]. These exploit the fact that the parametric complexity, *conditional* on some observed data, is often finite, even if the unconditional parametric complexity diverges. Additionally,

the CNML distributions can be used to make predictions about future data arising from the same source in a similar manner to the Bayesian predictive distribution. A recent paper comparing SNML and CNML against Bayesian and MML approaches in the particular case of exponential distributions found that the SNML predictive distribution has favorable properties in comparison to the predictive distribution formed by “plugging in” the maximum likelihood estimates [116]. Further information on variants of the MDL principle is available in [61, 102].

#### 1.25.4.9 Relation to MML and Bayesian inference

It is of interest to compare the NML coding scheme to the codebooks advocated by the MML principle. This is most easily done by comparing the approximate NML codelength (25.80) to the Wallace-Freeman approximate codelength (25.60). Consider the so-called Jeffreys prior distribution:

$$\pi_{\theta}(\theta; \gamma) = \frac{\sqrt{|\mathbf{J}_n(\theta; \gamma)|}}{\int_{\alpha \in \Theta_{\gamma}} \sqrt{|\mathbf{J}_n(\alpha; \gamma)|} d\alpha}. \quad (25.81)$$

The normalizing term in the above equation is the approximate parametric complexity in (25.80). Substituting (25.81) into the Wallace-Freeman code (25.60) leads to the cancellation of the Fisher information term, and the resulting Wallace-Freeman codelength is within  $O(\log k)$  of the approximate NML codelength. This difference in codelengths is attributed to the fact that MML codes state a specific member of the set  $\Theta_{\gamma}$  which is used to transmit the data, while the NML distribution makes no such assertion. The close similarity between the MDL and MML codelengths was demonstrated in [116] in the context of coding data arising from exponential distributions. A further comparison of MDL, the NML distribution and MML can be found in [27], pp. 408–415.

The close equivalence of the approximate NML and Wallace-Freeman codes implies that like the BIC, the approximate NML criterion is a consistent estimator of the true model  $\gamma^*$  that generated the data, assuming  $\gamma^* \in \Gamma$ , and that the normalizing integral is finite. In fact, the earliest incarnation of MDL introduced in 1978 [32] was exactly equivalent to the Bayesian information criterion, which was also introduced in 1978. This fact has caused some confusion in the engineering community, where the terms MDL and BIC are often used interchangeably to mean the “ $k$ -on-two  $\log n$ ” criterion given by (25.37) (this issue and its implications are discussed, for example, in [117]).

We conclude this section with a brief discussion of the differences and similarities between the MML and MDL principles. This is done to attempt to clear up some of the confusion and misunderstandings surrounding these two principles. The MDL principle encompasses a range of compression schemes (such as the NML model, the Bayesian model, etc.), which are unified by the deeper concept of *universal models* for data compression. In recent times, the MDL community has focused on those universal models that attain minimax regret with respect to the maximum likelihood codes to avoid the specification of subjective prior distributions. The two-part codebooks used in the MML principle are also types of universal models, and so advocates of the MDL school often suggest that MML is subsumed in the MDL principle. However, as the above coincidence of Wallace-Freeman and NML demonstrates, the MDL universal models can be implemented in the MML framework by the appropriate choice of prior distribution, and thus the MML school tend to suggest that MDL is essentially a special case of the MML theory, particularly given the time-line of developments in the two camps.

Practically, the most important difference between the two principles is in the choice of two-part (for MML) versus one-part (for MDL) coding schemes. The two-part codes chosen by MML allow for

the explicit definition of new classes of parameter estimators, in addition to model selection criteria, which are not obtainable by the one-part MDL codes. As empirical, and theoretical evidence strongly supports the general improvement of the MML estimators over the maximum likelihood and Bayesian MAP estimators, this difference seems to be of perhaps the greatest importance.

#### 1.25.4.10 Example: MDL linear regression

An MDL criterion for the linear regression example is now examined. Given the importance of linear regression, there have been a range of MDL inspired criteria developed, for example, the predictive MDL criterion [101],  $g$ -MDL [118], Liang and Barron's approach [119], sequentially normalized least squares [120, 121] and normalized maximum likelihood based methods [40, 111, 122]. We choose to focus on the NML criterion introduced by Rissanen in 2000 [111] as it involves no user specified hyperparameters and does not depend on the ordering of the data, a problem which affects the predictive and sequential procedures. Recall that in the linear regression case, the model index  $\gamma$  specifies which covariates are to be used from the full design matrix  $\mathbf{X}$ . For a given  $\gamma$ , the NML codelength, up to constants, is

$$\left(\frac{n-k}{2}\right) \log \hat{\tau}(\mathbf{y}; \gamma) + \frac{k}{2} \log \left(\frac{\hat{R}(\gamma)}{n}\right) - \log \Gamma\left(\frac{n-k}{2}\right) - \log \Gamma\left(\frac{k}{2}\right) - \log \pi_\gamma(\gamma), \quad (25.82)$$

where  $\hat{\tau}(\mathbf{y}; \gamma)$  is the maximum likelihood estimate of  $\tau$  given by (25.6),

$$\hat{R}(\gamma) = \hat{\beta}(\mathbf{y}; \gamma)' (\mathbf{X}'_\gamma \mathbf{X}_\gamma) \hat{\beta}(\mathbf{y}; \gamma) \quad (25.83)$$

and  $\hat{\beta}(\mathbf{y}; \gamma)$  are the least-squares estimates of the model coefficients  $\beta$  for subset  $\gamma$  given by (25.5). As in the case of the BIC given by (25.43), and the MML linear regression criteria given by (25.69) and (25.73, 25.74), the codelength includes a prior  $\pi_\gamma(\cdot)$  required to state the subset  $\gamma$ , and suitable choices are discussed in Section 1.25.3.4.

Interestingly, Hansen and Yu [123] have shown that NML is either asymptotically consistent or asymptotically efficient depending on the nature of the data generating distribution. In this way, the NML criterion can be viewed as a “bridge” between the AIC-based criteria and the BIC. Given the close similarity between the NML and MML linear regression criteria [68], this property is expected to also hold for the MML linear regression criteria.

#### 1.25.4.11 Applications of MDL

Below is an incomplete list of successful applications of the MDL principle to commonly used statistical models:

- the multinomial distribution [124],
- causal models and Naïve Bayes classification [125, 126],
- variable bin-width histograms [127, 128],
- linear regression [111, 118, 122],
- signal denoising [40, 111],
- autoregressive models [112],
- statistical genetics [129, 130].

Of particular interest is the multinomial distribution as it is one of the few models for which the exact parametric complexity is finite without any requirement for bounding, and may be efficiently computed in polynomial time using the clever algorithm discussed in [124]. This algorithm has subsequently been used to compute codelengths in histogram estimation and other similar models.

## 1.25.5 Simulation

We conclude with a brief simulation demonstrating the practical application of the distance based, Bayesian and information theoretic model selection criteria discussed in this chapter. The simulation compares the  $AIC_c$  (25.25),  $KIC_c$  (25.27),  $BIC$  (25.42),  $MML_u$  (25.69),  $MML_g$  (25.73, 25.74) and  $NML$  (25.82) criteria on the problem of polynomial order selection with linear regression. Datasets of various sample sizes  $25 \leq n \leq 500$  were generated from the polynomial model

$$y^* = x^3 - 0.5x^2 - 5x - 1.5, \quad x \in [-3, 3] \quad (25.84)$$

with design points  $x$  uniformly generated from the compact set  $x \in [-3, 3]$  [13, 102]. The noise variance  $\tau$  was chosen to yield a signal-to-noise ratio of ten. Recalling the linear regression example from Section 1.25.1.2, the complete design matrix consisted of polynomial bases  $x^i$  for  $(i = 0, \dots, q)$ ,

$$\mathbf{X} = (\mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}^q). \quad (25.85)$$

For each generated data set, each of the six criteria were used to select a nested polynomial model up to the maximum polynomial of order  $q = 20$ . The simulation was repeated  $10^4$  times and the zero-order polynomial model was not considered. The criteria were compared on two important metrics: (i) order selection, and (ii) squared prediction error. The results are presented in Table 25.1.

The simulation is highly illustrative of the general behaviors of the model selection criteria under consideration. For small sample sizes ( $n \leq 50$ ), the  $KIC_c$  criterion performs adequately in terms of order selection, but like the  $AIC_c$ , this performance does not improve with increasing sample size. This is not surprising given that both  $AIC_c$  and  $KIC_c$  are not consistent model selection procedures. While  $BIC$  performs relatively poorly for the smallest sample sizes, its performance dramatically improves when the sample size is larger. The asymptotic approximation from which  $BIC$  is derived is inadequate for small sample sizes which explains its poor performance. In contrast, the  $MML$  and  $NML$  criteria perform well for all sample sizes in terms of both order selection and prediction error. In this particular example, the  $MML_u$  and  $NML$  criteria are virtually indistinguishable, while the  $MML_g$  criterion performs slightly better. For larger sample sizes ( $n \geq 125$ ) all four consistent criteria performed essentially the same.

This brief simulation serves to demonstrate the general behavior of the six criteria and should not be taken as indicative of their performance in other model selection problems. For a more detailed comparison of some recent regression methods, including  $MML_u$ ,  $MML_g$  and  $NML$ , see [122]. The  $MML_u$ ,  $MML_g$  and  $NML$  criteria were found to be amongst the best of the eight criteria considered in all experiments conducted by the authors.<sup>3</sup>

---

<sup>3</sup>Unfortunately, through no fault of the authors, the formulae for  $MML_g$  given in [122] is missing the  $\delta/2$  term that is present in formula (25.73) in this tutorial. However, this has not effected the experimental results as the simulation code uses the correct formula (see Section 1.25.4.4).

**Table 25.1** Polynomial Order Selected by the Criteria (Expressed as Percentages) and Squared Error in Estimated Coefficients

	Criterion	Sample size						
		25	50	75	100	125	150	500
$\hat{p} = p$	AIC <sub>c</sub>	86.2	79.5	76.9	76.0	74.7	74.5	74.0
	KIC <sub>c</sub>	93.4	90.8	89.8	89.8	89.3	89.2	88.9
	BIC	77.5	91.4	94.1	95.7	96.0	97.0	97.4
	MML <sub>u</sub>	93.5	96.4	97.2	97.7	97.7	98.5	99.2
	MML <sub>g</sub>	95.6	97.9	98.4	98.6	98.6	99.1	99.0
	NML	94.4	96.8	97.4	97.9	97.9	98.7	98.6
$\hat{p} > p$	AIC <sub>c</sub>	13.8	20.5	23.1	24.0	25.3	25.5	26.1
	KIC <sub>c</sub>	6.60	9.30	10.2	10.2	10.7	10.8	11.1
	BIC	22.5	8.58	5.94	4.30	4.03	2.96	2.60
	MML <sub>u</sub>	6.50	3.64	2.82	2.29	2.42	1.48	1.50
	MML <sub>g</sub>	4.36	2.14	1.57	1.44	1.43	0.94	1.01
	NML	5.62	3.16	2.57	2.07	2.13	1.30	1.39
Error	AIC <sub>c</sub>	0.52	0.25	0.17	0.13	0.10	0.08	0.06
	KIC <sub>c</sub>	0.47	0.22	0.14	0.11	0.09	0.07	0.05
	BIC	0.59	0.22	0.14	0.10	0.08	0.07	0.05
	MML <sub>u</sub>	0.47	0.21	0.13	0.10	0.08	0.06	0.05
	MML <sub>g</sub>	0.46	0.21	0.13	0.10	0.08	0.06	0.05
	NML	0.46	0.21	0.13	0.10	0.08	0.06	0.05

## References

- [1] E.L. Lehmann, G. Casella, Theory of Point Estimation, Springer Texts in Statistics, Springer, fourth ed., 2003.
- [2] S. Kullback, R.A. Leibler, Ann. Math. Stat. 22 (1951) 79–86.
- [3] K. Takeuchi, Suri-Kagaku, Math. Sci. 153 (1976) 12–18 (in Japanese).
- [4] R. Shibata, Statistical aspects of model selection, in: J. C. Willems (Ed.), From Data to Model, Springer, New York, 1989, pp. 215–240.
- [5] H. Linhart, W. Zucchini, Model Selection, Wiley, New York, 1986.
- [6] H. Akaike, IEEE Trans. Automat. Contr. 19 (1974) 716–723.
- [7] C.M. Hurvich, C.-L. Tsai, Biometrika 76 (1989) 297–307.
- [8] A.-K. Seghouane, S.-I. Amari, IEEE Trans. Neural Networks 18 (2007) 97–106.
- [9] A.-K. Seghouane, Signal Process. 90 (2010) 217–224.
- [10] A.D.R. McQuarrie, C.-L. Tsai, Regression and Time Series Model Selection, World Scientific, 1998.
- [11] H. Jeffreys, Proc. Roy. Soc. Lond. Ser. A Math. Phys. Sci. 186 (1946) 453–461.
- [12] J.E. Cavanaugh, Stat. Probab. Lett. 42 (1999) 333–343.

- [13] A.-K. Seghouane, M. Bekara, *IEEE Trans. Signal Process.* 52 (2004) 3314–3323.
- [14] D.F. Schmidt, E. Makalic, *Lect. Notes Artif. Int.* 6464 (2010) 223–232.
- [15] A.K. Seghouane, *Signal Process.* 86 (2006) 2074–2084.
- [16] A.-K. Seghouane, *IEEE Trans. Aero. Electron. Syst.* 47 (2011) 1154–1165.
- [17] A.K. Seghouane, *IEEE Trans. Circ. Syst. Part I* 53 (2006) 2327–2335.
- [18] C.M. Hurvich, J.S. Simonoff, C.-L. Tsai, *J. Roy. Stat. Soc. Ser. B* 60 (1998) 271–293.
- [19] M. Bekara, L. Knockaert, A.-K. Seghouane, G. Fleury, *Signal Process.* 86 (2006) 1400–1409.
- [20] C.M. Hurvich, C.-L. Tsai, *Biometrika* 85 (1998) 701–710.
- [21] N. Murata, S. Yoshizawa, S. Amari, *IEEE Trans. Neural Networks* 5 (1994) 865–872.
- [22] A.K. Seghouane, M. Bekara, G. Fleury, *Signal Process.* 85 (2005) 1405–1417.
- [23] C. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, Springer Texts in Statistics, Springer, 2001.
- [24] J. Berger, L. Pericchi, in: P. Lahiri (Ed.), *Model selection*, vol. 38 of Lecture Notes Monograph Series, Hayward, CA: pp. 135–207.
- [25] J.M. Bernardo, *J. Roy. Stat. Soc. Ser. B* 41 (1979) 113–147.
- [26] J. Berger, L. Pericchi, *J. Am. Stat. Assoc.* 91 (1996) 109–122.
- [27] C.S. Wallace, *Statistical and Inductive Inference by Minimum Message Length*, Information Science and Statistics, first ed., Springer, 2005.
- [28] G. Casella, E. Moreno, *J. Am. Stat. Assoc.* 101 (2006) 157–167.
- [29] H. Jeffreys, *The Theory of Probability*, third ed., Oxford University Press, 1961.
- [30] R.E. Kass, A.E. Raftery, *J. Am. Stat. Assoc.* 90 (1995) 773–795.
- [31] G. Schwarz, *The Ann. Stat.* 6 (1978) 461–464.
- [32] J. Rissanen, *Automatica* 14 (1978) 465–471.
- [33] C.S. Wallace, D.M. Boulton, *Comput. J.* 11 (1968) 185–194.
- [34] O.E. Barndorff-Nielsen, D.R. Cox, *Asymptotic Techniques for Use in Statistics*, Chapman & Hall: New York, 1989.
- [35] D.M.A. Haughton, *Ann. Stat.* 16 (1988) 342–355.
- [36] C.R. Rao, Y. Wu, *Biometrika* 76 (1989) 369–374.
- [37] L. Breiman, *Ann. Stat.* 24 (1996) 2350–2383.
- [38] D. Madigan, A.E. Raftery, *J. Am. Stat. Assoc.* 89 (1994) 1536–1546.
- [39] E.I. George, D.P. Foster, *Biometrika* 87 (2000) 731–747.
- [40] T. Roos, P. Myllymäki, J. Rissanen, *IEEE Trans. Signal Process.* 57 (2009) 3347–3360.
- [41] J.G. Scott, J.O. Berger, *Ann. Stat.* 38 (2010) 2587–2619.
- [42] J. Chen, Z. Chen, *Biometrika* 95 (2008) 759–771.
- [43] C.P. Robert, G. Casella, *Monte Carlo Statistical Methods*, Springer, 2004.
- [44] A. Metropolis, M. Rosenbluth, A. Rosenbluth, E. Teller, *J. Chem. Phys.* 21 (1953) 1087–1092.
- [45] W. Hastings, *Biometrika* 57 (1970) 97–109.
- [46] S. Geman, D. Geman, *IEEE Trans. Pattern Anal. Mach. Int.* 6 (1984) 721–741.
- [47] G. Casella, E.I. George, *Am. Stat.* 46 (1992) 167–174.
- [48] S. Chib, *J. Am. Stat. Assoc.* 90 (1995) 1313–1321.
- [49] P.J. Green, *Biometrika* 82 (1995) 711–732.
- [50] E.I. George, R.E. McCulloch, *J. Am. Stat. Assoc.* 88 (1993) 881–889.
- [51] C. Hans, A. Dobra, M. West, *J. Am. Stat. Assoc.* 102 (2007) 507–516.
- [52] L. Breiman, *Technometrics* 37 (1995) 373–384.
- [53] R. Tibshirani, *J. Roy. Stat. Soc. Ser. B* 58 (1996) 267–288.
- [54] M.E. Tipping, *J. Mach. Learn. Res.* 1 (2001) 211–244.
- [55] R.M. Neal, *Bayesian learning for neural networks*, Lecture Notes in Statistics, Springer Verlag, 1996.

- [56] T. Park, G. Casella, *J. Am. Stat. Assoc.* 103 (2008) 681–686.
- [57] C. Hans, *Biometrika* 96 (2009) 835–845.
- [58] T.M. Cover, J.A. Thomas, *Elements of Information Theory*, second ed., Wiley-Interscience, 2006.
- [59] C.E. Shannon, *Bell Syst. Tech. J.* 27 (1948) 379–423 and 623–656.
- [60] J. Rissanen, J.G.G. Langdon, *IEEE Trans. Info. Theory* IT-27 (1981) 12–23.
- [61] P.D. Grünwald, *The Minimum Description Length Principle, Adaptive Communication and Machine Learning*, The MIT Press, 2007.
- [62] C. Wallace, D. Boulton, *Class. Soc. Bull.* 3 (1975) 11–34.
- [63] C.S. Wallace, P.R. Freeman, *J. Roy. Stat. Soc. Ser. B* 49 (1987) 240–252.
- [64] C.S. Wallace, in: *Proceedings of the International Conference on Information, Statistics and Induction in Science*, World Scientific, 1996, pp. 304–316.
- [65] G.E. Farr, C.S. Wallace, *Comput. J.* 45 (2002) 285–292.
- [66] J.H. Conway, N.J.A. Sloane, *Sphere Packing, Lattices and Groups*, third ed., Springer-Verlag, 1998.
- [67] E. Makalic, D.F. Schmidt, *Stat. Probab. Lett.* 79 (2009) 1155–1161.
- [68] D. Schmidt, E. Makalic, in: *The 22nd Australasian Joint Conference on Artificial Intelligence*, Melbourne, Australia, pp. 312–321.
- [69] C.S. Wallace, P.R. Freeman, *J. Roy. Stat. Soc. Ser. B* 54 (1992) 195–209.
- [70] C.S. Wallace, D. Dowe, MML estimation of the von Mises concentration parameter, Technical Report, Department of Computer Science, Monash University, 1993.
- [71] D.L. Dowe, J. Oliver, C. Wallace, *Lect. Notes Artif. Int.* 1160 (1996) 213–227.
- [72] D.F. Schmidt, Minimum message length inference of autoregressive moving average models, Ph.D. Thesis, Clayton School of Information Technology, Monash University, 2008.
- [73] A.R. Barron, T.M. Cover, *IEEE Trans. Info. Theory* 37 (1991) 1034–1054.
- [74] L. LeCam, University of California Publications in Statistics, vol. 11 1953.
- [75] D.L. Dowe, C.S. Wallace, in: *Proceedings of 28th Symposium on the Interface, Computing Science and Statistics*, vol. 28, Sydney, Australia, pp. 614–618.
- [76] C.S. Wallace, in: *Proceedings of the 14th Biennial Statistical Conference*, Queensland, Australia, p. 144.
- [77] M. Viswanathan, C.S. Wallace, D.L. Dowe, K.B. Korb, *Lect. Notes Artif. Int.* 1747 (1999) 405–416.
- [78] C.S. Wallace, D.L. Dowe, *Stat. Comput.* 10 (2000) 73–83.
- [79] J.J. Oliver, R.A. Baxter, MML and Bayesianism: similarities and differences, Technical Report TR 206, Department of Computer Science, Monash University, 1994.
- [80] D.F. Schmidt, in: *Proceedings of the 5th Workshop on Information Theoretic Methods in Science and Engineering (WITMSE-11)*.
- [81] L.J. Fitzgibbon, D.L. Dowe, L. Allison, in: *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pp. 147–154.
- [82] D.L. Dowe, *Comput. J.* 51 (2008) 523–560.
- [83] E. Makalic, Minimum message length inference of artificial neural networks, Ph.D. Thesis, Clayton School of Information Technology, Monash University, 2007.
- [84] E. Makalic, L. Allison, in: *Proceedings of the 85th Solomonoff Memorial Conference*, Melbourne, Australia.
- [85] M. Viswanathan, C. Wallace, in: *Proceedings of the 7th Int. Workshop on Artif. Intelligence and Statistics*, Ft. Lauderdale, Florida, USA, pp. 169–177.
- [86] G.W. Rumantir, C.S. Wallace, in: *Proceedings of the 5th International Symposium on Intelligent Data Analysis (IDA 2003)*, vol. 2810, Springer-Verlag, Berlin, Germany, pp. 486–496.
- [87] W. James, C.M. Stein, in: *Proceedings of the 4th Berkeley Symposium*, vol. 1, University of California Press, pp. 361–379.
- [88] S.L. Sclove, *J. Am. Stat. Assoc.* 63 (1968) 596–606.

- [89] N. Bouguila, D. Ziou, *IEEE Trans. Knowl. Data Eng.* 18 (2006) 993–1009.
- [90] C.S. Wallace, J.D. Patrick, *Mach. Learn.* 11 (1993) 7–22.
- [91] P. Tan, D. Dowe, *Lect. Notes Artif. Int.* 2903 (2003) 269–281.
- [92] P. Tan, D. Dowe, *Lect. Notes Artif. Int.* 4293 (2006) 593–603.
- [93] C.S. Wallace, K.B. Korb, in: A. Gammerman (Ed.), *Causal Models and Intelligent Data Management*, Springer-Verlag, pp. 89–111.
- [94] J.W. Comley, D. Dowe, *Minimum Message Length and Generalized Bayesian Nets with Asymmetric Languages*, M.I.T. Press (MIT Press), pp. 265–294.
- [95] E. Makalic, L. Allison, D.L. Dowe, in: *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2003)*.
- [96] E. Makalic, L. Allison, A.P. Paplinski, in: *Proceedings of the 8th Brazilian Symposium on Neural Networks (SBRN 2004)*, Sao Luis, Maranhao, Brazil.
- [97] D.F. Schmidt, E. Makalic, Shrinkage and denoising by minimum message length, Technical Report 2008/230, Monash University, 2008.
- [98] L.J. Fitzgibbon, D.L. Dowe, F. Vahid, in: *Proceedings of the International Conference on Intelligent Sensing and Information Processing (ICISIP)*, pp. 439–444.
- [99] D.F. Schmidt, in: *Proceedings of the 85th Solomonoff Memorial Conference*, Melbourne, Australia.
- [100] R.A. Baxter, J.J. Oliver, *Lect. Notes Artif. Int.* 1160 (1996) 83–90.
- [101] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific, 1989.
- [102] J. Rissanen, *Information and Complexity in Statistical Modeling*, Information Science and Statistics, first ed., Springer, 2007.
- [103] J. Rissanen, *Circuits, Systems, and Signal Process.* 1 (1982) 395–396.
- [104] J. Rissanen, *Ann. Stat.* 11 (1983) 416–431.
- [105] R.A. Baxter, J. Oliver, MDL and MML: similarities and differences, Technical Report TR 207, Department of Computer Science, Monash University, 1994.
- [106] J. Rissanen, *IEEE Trans. Info. Theory* 42 (1996) 40–47.
- [107] J. Rissanen, *IEEE Trans. Info. Theory* 47 (2001) 1712–1717.
- [108] Y.M. Shtarkov, *Probl. Inform. Transm.* 23 (1987) 3–17.
- [109] V. Balasubramanian, in: I.J.M.P.D. Grünwald, M.A. Pitt (Eds.), *Advances in Minimum Description Length: Theory and Applications*, MIT Press, pp. 81–99.
- [110] S. de Rooij, P. Grünwald, *J. Math. Psychol.* 50 (2006) 180–192.
- [111] J. Rissanen, *IEEE Trans. Info. Theory* 46 (2000) 2537–2543.
- [112] D.F. Schmidt, E. Makalic, *IEEE Trans. Signal Process.* 59 (2011) 479–487.
- [113] A.P. Dawid, J. Roy. Stat. Soc. Ser. A 147 (1984) 278–292.
- [114] T. Roos, J. Rissanen, in: *Proceedings of the 1st Workshop on Information Theoretic Methods in Science and Engineering (WITMSE-08)*, Tampere International Center for Signal Processing (Invited Paper).
- [115] J. Rissanen, T. Roos, in: *Proceeding of the 2007 Information Theory and Applications Workshop (ITA-07)*, IEEE Press, 2007, pp. 337–341 (Invited Paper).
- [116] D.F. Schmidt, E. Makalic, *IEEE Trans. Info. Theory* 55 (2009) 3087–3090.
- [117] D.F. Schmidt, E. Makalic, *IEEE Trans. Signal Process.* 60 (2012) 1508–1510.
- [118] M.H. Hansen, B. Yu, *J. Am. Stat. Assoc.* 96 (2001) 746–774.
- [119] F. Liang, A. Barron, *IEEE Trans. Info. Theory* 50 (2004) 2708–2726.
- [120] J. Rissanen, T. Roos, P. Myllymäki, *J. Multivariate Anal.* 101 (2010) 839–849.
- [121] D.F. Schmidt, T. Roos, in: *Proceedings of the 3rd Workshop on Information Theoretic Methods in Science and Engineering (WITMSE-10)*, Tampere International Center for Signal Processing. (Invited Paper).
- [122] C.D. Giurcăneanu, S.A. Razavi, A. Liski, *Signal Process.* (2011).

- [123] M. Hansen, B. Yu, in: *Science and Statistics: A Festschrift for Terry Speed*, of Lecture Notes—Monograph Series, vol. 40, Institute of Mathematical Statistics, pp. 145–164.
- [124] P. Kontkanen, P. Myllymäki, *Info. Process. Lett.* 103 (2007) 227–233.
- [125] T. Mononen, P. Myllymäki, in: *Proceedings of the 10th International Conference on Discovery Science*, Lecture Notes in Computer Science, Sendai, Japan, vol. 4755, pp. 151–160.
- [126] T. Silander, T. Roos, P. Myllymäki, in: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS-09)*.
- [127] J. Rissanen, T.P. Speed, B. Yu, *IEEE Trans. Info. Theory* 38 (1992) 315–323.
- [128] P. Kontkanen, P. Myllymäki, in: M. Meila, X. Shen (Eds.), in: *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, San Juan, Puerto Rico.
- [129] Y. Yang, I. Tabus, in: *IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS 2007)*, pp. 1–4.
- [130] I. Tabus, J. Rissanen, J. Astola, *Signal Process.* 83 (2003) 713–727.

# Music Mining

# 26

George Tzanetakis

*Department of Computer Science, University of Victoria, Victoria, Canada*

## 1.26.1 Introduction

During the first ten years of the 21st century we have witnessed a dramatic shift in how music is produced, distributed, and consumed. Several factors including advances in digital signal processing, faster computers, and steadily increasing digital storage capacity and network bandwidth have made digital music distribution a reality. It is now possible to purchase and listen to music as well as find all sorts of associated information about it from any computer or smart phone. Currently, portable music players and phones can store thousands of music tracks, and millions of tracks are accessible through streaming over the Internet in digital music stores and personalized radio stations.

Enabling anyone with access to a computer and the Internet to listen to essentially most of recorded music in human history is a remarkable technological achievement that would probably be considered impossible even twenty years ago. The research area of Music Information Retrieval (MIR) gradually emerged during this time period in order to address the challenge of effectively accessing and interacting with these vast digital collections of music and associated information such as meta-data, reviews, blogs, rankings, and usage/download patterns.

As a research area, data mining emerged from the interaction between the database community that needed to address the challenge of extracting useful not explicitly represented information from large collections of data, and the machine learning community which explored algorithms that can improve their performance over time as they are exposed to more data. Music mining refers to the application of ideas from the field of data mining to the extraction of useful information from large collections of music and is the topic of this chapter. It can be viewed as a subset of music information retrieval research.

Music is pervasive and plays an important role in the daily lives of most people today. It is an extremely complex human creation that somehow strongly affects our intellect and emotions. In order to create effective tools for interacting with large music collections we need to design algorithms that can extract high-level information from music signals and use that information in a variety of mining tasks. Techniques from audio signal processing have been used to extract various low-level and mid-level audio features that can subsequently be used to represent music tracks to perform various music mining tasks. In addition it is possible to apply traditional data mining techniques to other sources of music related information such as reviews, lyrics, blogs, rankings and usage patterns.

Music has several important characteristics and challenges that make it a particularly interesting research area for data mining. Similarly to image search it requires sophisticated content analysis

algorithms to extract useful information from the raw signal. At the same time it has very rich structured context information associated with it. For example a particular song especially if it is popular might be mentioned in thousands of web pages and blogs. It also has lyrics which can be analyzed and by virtue of being performed by a particular artist has many associations to other pieces of music. This rich tapestry of relevant information provides many opportunities for data mining but at the same time its heterogeneity is a challenge for many algorithms that require more homogeneous and structured data. The sheer amount of data required both for storing the audio but also for storing calculated audio features poses significant system scalability challenges and requires efficient large scale algorithms. The focus of this chapter is music mining in large music collections and does not cover tasks in MIR that deal with individual music tracks such as transcription, audio-score alignment and structural analysis among others. The choice of topics as well as how much detail they are described were to a large extent determined by the corresponding volume of published work in each music mining topic.

The remainder of this chapter is organized as follows. The first Section provides an overview of audio feature extraction for audio signals which forms the foundation of many music mining algorithms. The following sections describe various music mining tasks such as classification, clustering, tag annotation, advanced data mining, and visualization. The goal is to provide an overview of the music mining problems that researchers have explored and describe in some detail basic system configurations to solve these tasks. The particular techniques chosen are representative examples that are straightforward to explain rather than a comprehensive list of all possibilities. We also discuss open problems and future trends in music mining. The final section provides pointers to further reading about these topics.

---

## 1.26.2 Ground truth acquisition and evaluation

In the data mining literature, frequently the primary concern is the algorithm(s) used for extracting information from the data. This data is assumed to be readily available and in the format required for the particular task. However, in many specific applications areas such as music mining the data acquisition process is critical and not trivial. In mining algorithms typically there is a clear distinction between data that is somehow automatically extracted and the desired information that the mining algorithm somehow “extracts.” In order to assess how well a particular mining algorithm performs, we need to know in some way, what the “correct” answer should be. There are several general strategies that have been used to acquire this “ground truth” information. In most cases human users are involved in the process. With access to this ground truth information for a particular dataset, different music mining algorithms and systems can be evaluated, compared, and contrasted. In this section, we discuss the process of ground truth acquisition and evaluation in a generic way that is applicable to most music mining tasks. Many music mining tasks can be viewed as ways of associating music and text. Free-form text (frequently called tags) is the most general type of annotation and subsumes specific annotations such as genre, style, and emotion/motion. In the following sections issues more specific to each task are examined in more detail.

In terms of ground truth acquisition the simplest case is when the desired annotations are readily available by some external authority. For example online music stores provide genre labels for most of their music that can be directly used to train genre classification algorithms. For more specialized types of information expert annotation can be used. For example the personalized radio company Pandora utilizes music experts to annotate pieces of music with 400 attributes. Expert data is reliable and of high

quality but it is costly to acquire and therefore harder to scale to large music collections. An alternative approach is to utilize average users/listeners to perform the annotation. The time honored approach to acquiring ground truth, especially in academic work, is the survey typically of undergraduate students. Surveys can be carefully designed and therefore the data obtained tends to be reliable. However, they are time consuming and costly and therefore limited in terms of scalability. More recently there has been a surge in the use of “social” tags which are simply words entered by users to characterize their photos or music. Last.fm is a music discovery Web site that relies on such social tags. By the beginning of 2007, Last.fm’s large base of 40 million monthly users had built an unstructured vocabulary of 960,000 free-text tags and used it to annotate millions of songs. By harvesting the collective effort of millions of users social tagging can scale to large collections. However it comes with its own set of issues. As there is no restriction in the vocabulary, used there is a lot of inconsistency and noise in the data. Another important issue is that there is a sparsity (or lack of) tags for new artists/tracks which has been termed the cold-start problem. This is a specific case of the more general problem of popularity bias in which popular multimedia items tend to be recommended to most users simply because there is a lot of information about them. An interesting alternative that combines some of the control of surveys with the scalability and large number of users of social tags is the concept of annotation games. These are games in which a large group of users are presented with a song, and a list of tags. The goal of the game is to “guess” the tags that other users apply for that song. When a large group of users agree on a tag then the song has a strong association with it. Such annotation games belong to the larger category of games with a purpose in which players play the game because it is engaging and entertaining, while at the same time in the process of playing provide valuable information. The most famous such game is the ESP game that has been used for image annotation. Even though similar annotation games have been proposed for music they have not yet received large scale usage.

Assuming that ground truth for a particular task is available it is important to be able to evaluate different algorithms that attempt to solve it. Music mining is an emerging research area with a history of about ten years. Typically early work in a particular music mining task involves assembling a collection of music and associated ground truth for the task. In many cases only the original researchers have access to the data and the work is hard to replicate. Over time some of these datasets have been shared helping make more meaningful comparisons between different algorithms and systems. A big challenge in music mining is the difficulty of sharing data given the multiple copyrights associated with music information. The Music Information Retrieval Evaluation eXchange (MIREX) is an annual evaluation campaign for music information retrieval (MIR) algorithms that is coupled to the International Conference of the Society for Music Information Retrieval (ISMIR). It is organized by the graduate school of library and information sciences at the University of Illinois at Urbana-Champaign. Participating groups submit their systems that follow specific input/output conventions and they are evaluated on data that for some tasks is accessible only by the MIREX organizers and not the participants. In addition to dealing this way with copyright issues it also helps avoid overfitting which is an important problem for data mining algorithms in general. Overfitting refers to the situation where the performance of a mining algorithm in training data is misleadingly higher than its performance on data it has not encountered before. If the full datasets used for evaluation are available it is possible to over-optimize learning algorithms to fit their specific characteristics to the expense of generalization performance, i.e., the performance of the algorithm on data that it has not encountered during training. In the remainder of this chapter representative results from MIREX will be provided for the music mining tasks that are described, when they are available.

### 1.26.3 Audio feature extraction

Audio feature extraction forms the basis of many music mining tasks. Even though a full exposition is beyond the scope of this chapter, a short overview is provided as it is important in understanding what type of information is represented and how it is extracted. The goal of audio feature extraction is to calculate a succinct representation that summarizes musical information about the underlying audio signal. The representation should capture in a statistical sense the different types of musical information that humans are aware of when listening to music.

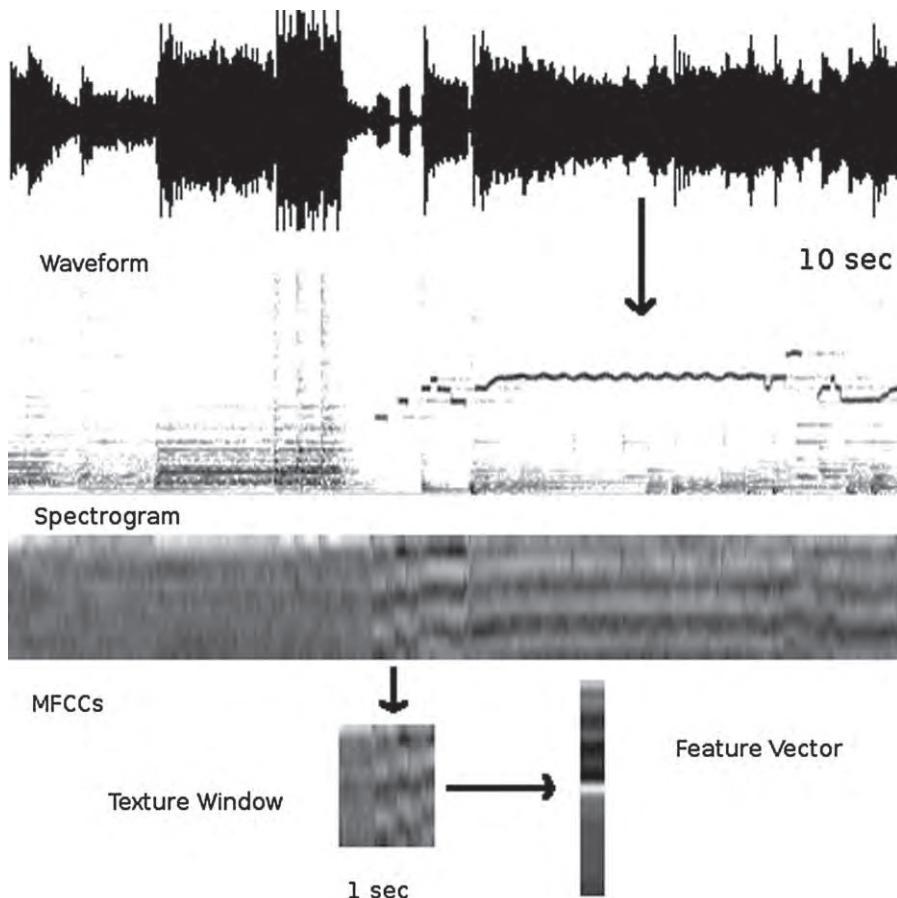
The three basic facets of music information that have mostly been explored in the existing literature are timbre, rhythm, and harmony. Rather than providing formal definitions, which are still debated among musicologists, we describe these terms informally without going into details that would require some knowledge of music theory. Timbre refers to the characteristics of the musical sound that are independent of the actual notes played and are related to the instruments playing and their sound. For example the exact same music piece played by a rock band with electric guitars and drums will sound very different than the same music piece performed by a jazz big band. Rhythm refers to the periodic repeating hierarchical structure that underlies the music independently of what instruments are playing. For example the famous beginning of the 5th Symphony by Beethoven will have the same rhythm independently of whether it is played by a symphonic orchestra or a cheap toy piano. Harmony refers to the simultaneous sounding of groups of discrete pitches/notes as well as how these groups evolve over time. For example a dance remix of a tune by the Beatles will have the same harmony or chord structure with the original while the rhythm and timbre will be completely different.

There are many variations in timbral feature extraction but most systems follow a common general template. The audio signal is broken into small slices (typically around 10–40 ms) and some form of frequency analysis, such as the Discrete Fourier Transform, is performed, followed by a summarization step in which a set of numbers (the feature vector) is calculated. This feature vector attempts to summarize/capture the content information of that short slice in time. After this stage the music track can then be represented as a sequence (trajectory) of feature vectors (points) in a high-dimensional feature space. That sequence can then be characterized using a more compact representation for subsequent classification.

Most audio features are extracted in three stages: (1) spectrum calculation, (2) frequency-domain summarization, and (3) time-domain summarization. In spectrum calculation, a short-time slice (typically around 10–40 ms) of waveform samples is transformed to a frequency domain representation. The most common such transformation is the Short Time Fourier Transform (STFT). During each short-time slice the signal is assumed to be approximately stationary and is windowed to reduce the effect of discontinuities at the start and end of the frame. This frequency domain transformation preserves all the information in the signal and therefore the resulting spectrum still has high dimensionality. For analysis purposes, it is necessary to find a more succinct description that has significantly lower dimensionality while still retaining the desired content information. Frequency domain summarization converts the high dimensional spectrum (typically 512 or 1024 coefficients) to a smaller set of number features (typically 10–30). A common approach is to use various descriptors of the spectrum shape such as the Spectral Centroid and Bandwidth. Another widely used frequency domain summarization of the Spectrum are the Mel-Frequency Cepstral Coefficients (MFCCs), a representation which originated from the speech and speaker recognition community. MFCC summarize spectral information (the energy distribution of

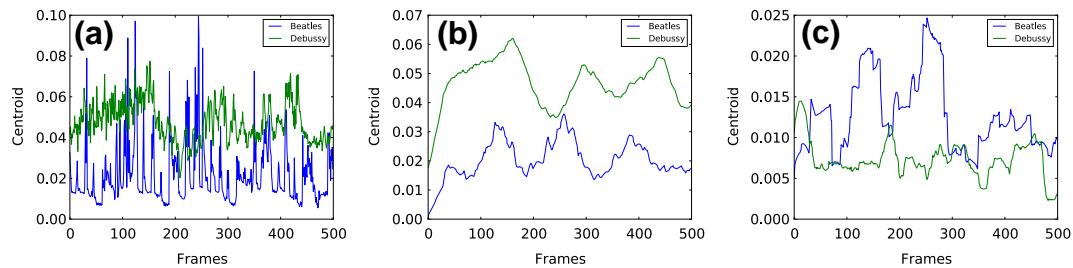
different frequencies) by taking into account, to some extent, the characteristics of the human auditory system. Such features depend on the instrumentation of a piece, how the timbral “texture” changes over time as well as how humans perceive this information. The goal of time domain summarization is to characterize the musical signal at longer time scales than the short-time analysis slices. Typically this summarization is performed across so called “texture” windows of approximately 2–3 s or it can be also performed over the entire piece of music. Figure 26.1 shows graphically feature extraction, frequency and time summarization.

Several variations on time domain summarization have been proposed. A popular approach is to fit Gaussian densities with diagonal covariance or full-covariance and then use the resulting parameters as the feature vector. Another approach has been the use of auto-regressive models that model the evolution of the feature vectors within the time window of interest.



**FIGURE 26.1**

Feature extraction and texture window.

**FIGURE 26.2**

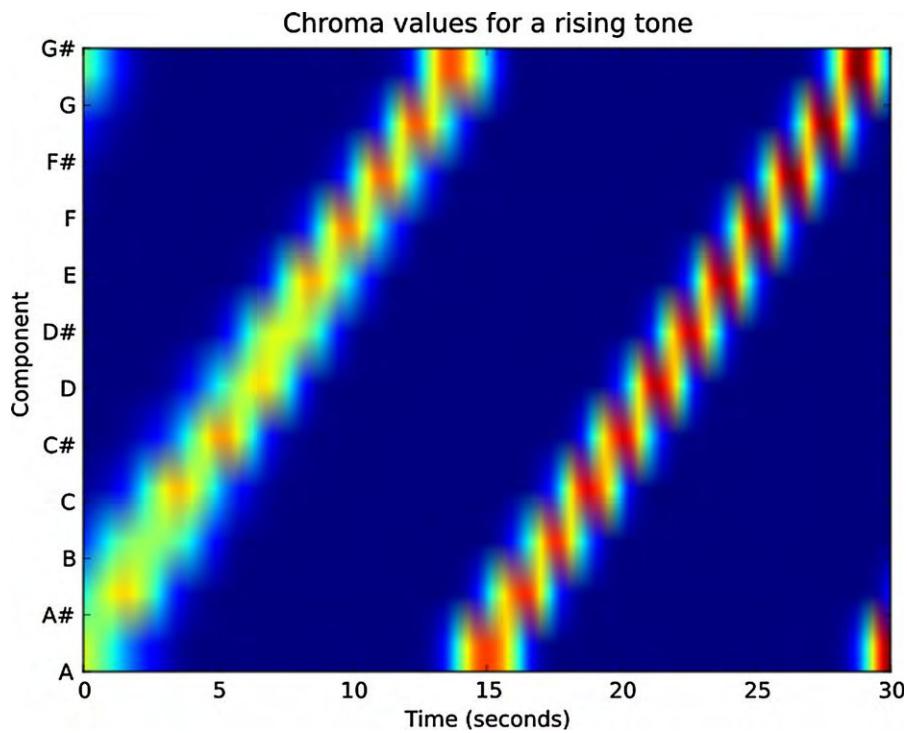
The time evolution of audio features is important in characterizing musical content. The time evolution of the spectral centroid for two different 30-second excerpts of music is shown in (a). The result of applying a moving mean and standard deviation calculation over a texture window of approximately 1 s is shown in (b) and (c).

A more detailed view of summarization (or as is sometimes called aggregation) is shown in Figure 26.2. It shows how the time evolution audio features, in this case the spectral centroid, can be summarized by applying a moving mean and standard deviation.

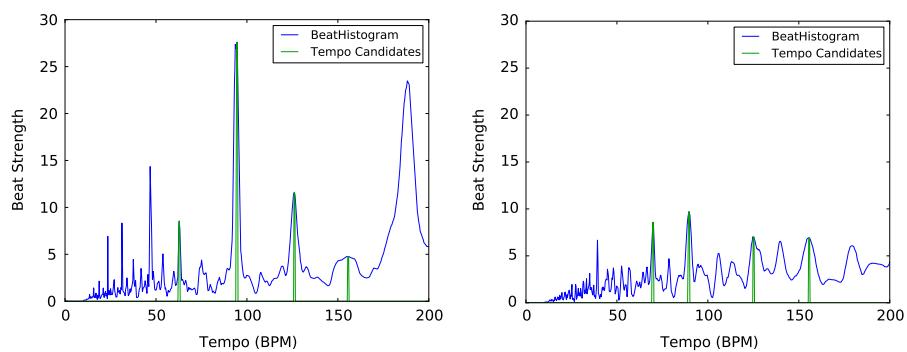
Automatic music transcription is the process of converting an audio signal to a musical score (a symbolic representation containing only information about pitch and rhythm). It is a hard problem and existing techniques can only deal with simple “toy” examples. Instead the most commonly used pitch-based representation are the pitch and pitch-class profiles (other alternative names used in literature are pitch histograms and chroma-vectors for Pitch Class Profiles). The pitch profile measures the occurrence of specific discrete musical pitches in a music segment and the pitch class profile considers all octaves equivalent essentially folding the pitch profile into 12 pitch classes. Due to space limitations we can not go into details about how pitch profiles are calculated. Broadly speaking they can either be computed by summing the energy of different frequency bins that correspond to particular pitches or alternatively multiple pitch estimation can be performed and the results can be accumulated into a profile. Figure 26.3 shows a chromagram (i.e., a sequence of chroma vectors over time) corresponding to a continuous pitch glide over two octaves. The successive activation of the chromatic scale notes as well as the warping in octaves can be observed.

Automatically extracting information related to rhythm is also important. Rhythmic information is hierarchical in nature and involves multiple related periodicities. A typical representation is a beat histogram (or sometimes called a beat spectrum) that provides a “salience” value for every possible periodicity. A typical approach applies onset detection to generate an onset strength signal that has high energy values at the time locations where there are significant changes in the audio spectrum such as the start of new notes. The periodicities of the onset strength signal at the range of human rhythm (approximately 30–180 beats/min) are calculated using autocorrelation. Another more recent approach is to identify rhythmic patterns that are characteristic of a particular genre automatically and characterize each piece as a occurrence histogram over a set of basic rhythmic patterns.

Figure 26.4 shows two example beat histograms from 30 s clips of HipHop Jazz (left) and Bossa Nova (right). As can be seen in both histograms the prominent periodicities or candidate tempos are clearly visible. Once the tempo of the piece is identified the beat locations can be calculated by locally fitting tempo hypothesis with regularly spaced peaks of the onset strength signal.

**FIGURE 26.3**

Chromagram for pitch glide over two octaves.

**FIGURE 26.4**

Beat histograms of HipHop/Jazz and Bossa Nova.

### 1.26.4 Extracting context information about music

In addition to information extracted by analyzing the audio content of music, there is a wealth of information that can be extracted by analyzing information on the web as well as patterns of downloads/listening. We use the general term musical context to describe this type of information. In some cases such as song lyrics the desired information is explicitly available somewhere on the web and the challenge is to appropriately filter out irrelevant information from the corresponding web pages. Text-based search engines such as Google and Bing can be leveraged for the initial retrieval that can then be followed by some post-processing based on heuristics that are specific to the music domain. Other types of information are not as straightforward and can require more sophisticated mechanisms such as the term weighting used in text retrieval systems, or natural language processing techniques such as entity detection. Such techniques are covered in detail in the literature as they are part of modern day search engines. As an illustrative example we will consider the problem of detecting the country of origin of a particular artist. As a first attempt one can query a search engine for various pairs of artist name and countries and simply count the number of pages returned. The country with the highest number of pages returned is returned as the country of origin. A more sophisticated approach is to analyze the retrieved web pages using term weighting. More specifically consider country  $c$  as a term. The document frequency  $DF(c, a)$  is defined as the total number of web pages retrieved for artist  $a$  in which the country term  $c$  appears at least once. The term frequency  $TF(c, a)$  is defined as the total number of occurrences of the country term  $c$  in all pages retrieved for artist  $a$ . The basic idea of term frequency-inverse document frequency weighting (TF-IDF) is to “penalize” terms that appear in many documents (in our case the documents retrieved for all artists) and increase the weight of terms that occur frequently in the set of web pages retrieved for a specific artists. There are several TF-IDF weighting schemes. For example a logarithmic formulation is:

$$TFIDF(c, a) = \ln(1 + TF(c, a)) * \ln\left(1 + \frac{n}{DF(c)}\right), \quad (26.1)$$

where  $DF(c)$  is the document frequency of a particular country  $c$  over the documents returned for all artists  $a$ . Using the above equation the weight of every country  $c$  can be calculated for a particular artist query  $a$ . The country with the highest weight is then selected as the predicted country of origin.

---

### 1.26.5 Similarity search

Similarity retrieval (or query-by-example) is one of the most fundamental MIR tasks. It is also one of the first tasks that were explored in the literature. It was originally inspired by ideas from text information retrieval and this early influence is reflected in the naming of the field as Music Information Retrieval (MIR). Today most people with computers use search engines on a daily basis and are familiar with the basic idea of text information retrieval. The user submits a query consisting of some words to the search engine and the search engine returns a ranked list of web pages sorted by how relevant they are to the query.

Similarity retrieval can be viewed as an analogous process where instead of the user querying the system by providing text the query consists of an actual piece of music. The system then responds

by returning a list of music pieces ranked by their similarity to the query. Typically the input to the system consists of the query music piece (using either a symbolic or audio representation) as well as additional metadata information such as the name of the song, artist, year of release, etc. Each returned item typically also contains the same types of meta-data. In addition to the audio content and meta-data other types of user generated information can also be considered such as ratings, purchase history, social relations and tags. Similarity retrieval can also be viewed as a basic form of playlist generation in which the returned results form a playlist that is “seeded” by the query. However more complex scenarios of playlist generation can be envisioned. For example a start and end seed might be specified or additional constraints such as approximate duration or minimum tempo variation can be specified. Another variation is based on what collection/database is used for retrieval. The term playlisting is more commonly used to describe the scenario where the returned results come from the personal collection of the user, while the term recommendation is more commonly used in the case where the returned results are from a store containing a large universe of music. The purpose of the recommendation process is to entice the user to purchase more music pieces and expand their collection. Although these three terms (similarity retrieval, music recommendation, automatic playlisting) have somewhat different connotations the underlying methodology for solving them is mostly similar so for the most part we will use them interchangeably. Another related term, that is sometimes used, is personalized radio in which the idea is to play music that is targeted to the preferences of a specific user.

One can distinguish three basic approaches to computing music similarity. Content-based similarity is performed by analyzing the actual content to extract the necessary information. Metadata approaches exploit sources of information that are external to the actual content such as relationships between artists, styles, tags or even richer sources of information such as web reviews and lyrics. Usage-based approaches track how users listen and purchase music and utilize this information for calculating similarity. Examples include collaborative filtering in which the commonalities between purchasing histories of different users are exploited, tracking peer-to-peer downloads or radio play of music pieces to evaluate their “hotness” and utilizing user generated rankings and tags.

There are trade-offs involved in all these three approaches and most likely the ideal system would be one that combines all of them intelligently. Usage-based approaches suffer from what has been termed the “cold-start” problem in which new music pieces for which there is no usage information can not be recommended. Metadata approaches suffer from the fact that metadata information is frequently noisy or inaccurate and can sometimes require significant semi-manual effort to clean up. Finally content-based methods are not yet mature enough to extract high-level information about the music.

From a data mining perspective similarity retrieval can be considered a ranking problem. Given a query music track  $q$  and a collection of music tracks  $D$  the goal of similarity retrieval is to return a ranked list of the music tracks in  $D$  sorted by similarity so that the most similar objects are at the top of the list. In most approaches, this ranking is calculated by defining some similarity (or distance) metric between pairs of music tracks. The most basic formulation is to represent each music track as a single feature vector of fixed dimensionality  $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$  and use standard distance metrics such as L1 (Manhattan) or L2 (Euclidean) or Mahalanobis on the resulting high dimensional space. This feature vector is calculated using audio feature extraction techniques as described in the previous section. Unless the distance metric is specifically designed to handle features with different dynamic ranges the feature vectors are typically normalized for example by scaling all of them so that their maximum value over the dataset is 1 and their minimum value is 0. A more complex alternative is

to treat each music track as a distribution of feature vectors. This is accomplished by assuming that the feature vectors are samples of an unknown underlying probability density function that needs to be estimated. By assuming a particular parametric form for the pdf (for example a Gaussian Mixture Model) the music track is then represented as a parameter vector  $\theta$  that is estimated from the data. This way the problem of finding the similarity between music tracks is transformed to the problem of somehow finding how similar are two probability distributions that are estimated from the samples available. Several such measures of probability distance have been proposed such as histogram intersection, symmetric Kullback-Leibler divergence and earth mover's distance. In general computation of such probabilistic distances are more computationally intensive than geometric distances on feature vectors and in many cases require numerical approximation as they can not be obtained analytically. An alternative to audio feature extraction is to consider similarity based on text such as web pages, user tags, blogs, reviews, and song lyrics. The most common model when dealing with text is the so called "bag of words" representation in which each document is represented as an unordered set of its words without taking into account any syntax or structure. Each word is assigned a weight that indicates the importance of the word for some particular task. The document can then be represented as a feature vector comprising of the weights corresponding to all the words of interest. From a data mining perspective the resulting feature vector is no different than the ones extracted from audio feature extraction and can be handled using similar techniques. In the previous section we described a particular example of text based feature extraction for the purpose of predicting the country of origin as an artist.

As an example of how a text-based approach can be used to calculate similarity consider the problem of finding how similar are two artists  $A$  and  $B$ . Each artist is characterized by a feature vector consisting of term weights for the terms they have in common. The cosine similarity between the two feature vectors is defined as the cosine of the angle between the vectors and has the property that it is not affected by the magnitude of the vector (which would correspond to the absolute number of times terms appear and could be influenced by the popularity of the artist):

$$\text{sim}(A, B) = \cos \theta = \frac{\sum_t A(t) \times B(t)}{\sqrt{\sum_t A(t)^2} \times \sqrt{\sum_t B(t)^2}}. \quad (26.2)$$

Another approach to calculating similarity is to assume that the occurrence of two music tracks or artists within the same context indicates some kind of similarity. The context can be web pages (or page counts returned by a search engine), playlists, purchase histories, and usage patterns in peer-to-peer (P2P) networks. Collaborative filtering (CF) refers to a set of techniques that make recommendations to users based on preference information from many users. The most common variant is to assume that the purchase history of a particular user (or to some extent equivalently their personal music collections) is characteristic of their taste in music. As an example of how co-occurrence can be used to calculate similarity, a search engine can be queried for documents that contain a particular artist  $A$  and  $B$ , as well as documents that contain both  $A$  and  $B$ . The artist similarity between  $A$  and  $B$  can then be found by:

$$\text{sim}(A, B) = \frac{co(A, B)}{\min(co(A), co(B))}, \quad (26.3)$$

where  $co(X)$  is the number of pages returned for query  $X$  or the co-occurrences of  $A$  and  $B$  in some context. A similar measure can be defined based on co-occurrences between tracks and artists in playlists

and compilation albums based on conditional probabilities:

$$\text{sim}(A, B) = \frac{1}{2} * \left( \frac{\text{co}(A, B)}{\text{co}(A)} + \frac{\text{co}(A, B)}{\text{co}(B)} \right). \quad (26.4)$$

Co-occurrences can also be defined in the context of peer-to-peer networks by considering the number of users that have both artists A and B in their shared collection. The popularity bias refers to the problem of popular artists appearing more similar than they should be due to them occurring in many contexts. A similarity measure can be designed to down weight the similarity between artists if one of them is very popular and the other is not (the right-hand part of the following equation):

$$\text{sim}(A, B) = \frac{C(A, B)}{C(B)} * \left( 1 - \frac{|C(A) - C(B)|}{C(\text{Max})} \right), \quad (26.5)$$

where  $C(\text{Max})$  is the number of times the most popular artist appears in a context.

### 1.26.5.1 Evaluation of similarity retrieval

One of the challenges in content-based similarity retrieval is evaluation. In evaluating mining systems ideally one can obtain ground truth information that is identical to the outcome of the mining algorithm. Unfortunately this is not the case in similarity as it would require manually sorting large collections of music in order of similarity to a large number of queries. Even for small collections and number of queries, collecting such data would be extremely time consuming and practically impossible. Instead the more common approach is to only consider the top  $K$  results for each query, where  $K$  is a small number, and have users annotate each result as relevant or not relevant. Sometimes a numerical discrete score is used instead of a binary relevance decision. Another possibility that has been used is to assume that tracks by the same artist or same genre should be similar and use such groupings to assign relevance values.

Evaluation metrics based on information retrieval can be used to evaluate the retrieved results for a particular query. They assume that each of the returned results has a binary annotation indicating whether or not it is relevant to the query. The most common one is the *F-measure* which is a combination of the simpler measures of *Precision* and *Recall*. *Precision* is the fraction of retrieved instances that are relevant. *Recall* is the fraction of relevant instances that are retrieved. As an example, consider a music similarity search in which relevance is defined by genre. If for a given query of Reggae music it returns 20 songs and 10 of them are also Reggae the precision for that query is  $10/20 = 0.5$ . If there are a total of 30 Reggae songs in the collection searched then the *Relevance* for that query is  $10/30 = 0.33$ . The *F-measure* is defined as the harmonic mean of *Precision P* and *Recall R*.

$$F = 2 \times \frac{P \times R}{P + R}. \quad (26.6)$$

These three measures are based on the list of documents returned by the system without taking into account the order they are returned. For similarity retrieval, a more accurate measure is the *Average Precision* which is calculated by computing the precision and recall at every position in the ranked sequence of documents, creating a precision-recall curve and computing the average. This is equivalent to the following finite sum:

$$AP = \frac{\sum_{k=1}^n P(k) \times \text{rel}(k)}{\#\text{relevant documents}}, \quad (26.7)$$

**Table 26.1** 20120 MIREX Music Similarity and Retrieval Results

	<b>FS</b>	<b>BS</b>	<b>P@5</b>	<b>P@10</b>	<b>P@20</b>	<b>P@50</b>
RND	17	0.2	8	9	9	9
TLN3	47	0.97	48	47	45	42
TLN2	47	0.97	48	47	45	42
TLN1	46	0.94	47	45	43	40
BWL1	50	1.08	53	51	50	47
PS1	55	1.22	59	57	55	51
PSS1	55	1.21	62	60	58	55
SSPK2	57	1.24	59	58	56	53

where  $P(k)$  is the precision at list position  $k$  and  $rel(k)$  is an indicator function that is 1 if the item at list position (or rank)  $k$  is a relevant document and 0 otherwise.

All of the measures described above are defined for a single query. They can easily be extended to multiple queries by taking their average across the queries. The most common way of evaluating similarity systems with binary relevance ground-truth is the *Mean Average Precision* (MAP) which is defined as the mean of the *Average Precision* across a set of queries.

The Music Information Retrieval Evaluation Exchange (MIREX) is an annual evaluation benchmark in which different groups submit algorithms to solve various MIR tasks and their performance is evaluated using a variety of subjective and objective metrics. Table 26.1 shows representative results of the music similarity and retrieval task from 2010. It is based on a dataset of 7000 30-second audio clips drawn from 10 genres. The objective statistics are the precision at 5, 10, 20, and 50 retrieved items without counting entries by the artist (artist filtering). The subjective statics are based on human evaluation of approximately 120 randomly selected queries and 5 results per query. Each result is graded with a fine score (between 0 and 100 with 100 being most similar) and a broad score (0 not similar, 1 somewhat similar, 2 similar) and the results are averaged. As can be seen all automatic music similarity systems perform significantly better than the random baseline (RND). The differ in terms of the type of extracted features utilized, the decision fusion strategy (such as simple concatenation of the different feature sets or empirical combinations of distances from the individual feature sets), and whether post-processing is applied to the resulting similarity matrix. There is also a strong correlation between the subjective and objective measures although it is not perfect (for example SSPK2 is better than PSS1 in terms of subjective measures but worst in terms of objective measures).

### 1.26.5.2 Cover song detection and audio fingerprinting

In addition to content-based similarity there are two related music mining problems. The goal of audio fingerprinting is to identify whether a music track is one of the recordings in a set of reference tracks. The problem is trivial if the two files are byte identical but can be considerably more challenging when various types of distortion need to be taken into account. The most common distortion is perceptual

audio compression (such as the one used for mp3 files) which can result in significant alterations to the signal spectrum. Although these alterations are not directly perceptible by humans they make the task of computer identification harder. Another common application scenario is music matching/audio fingerprinting for mobile applications. In this scenario the query signal is acquired through a low quality microphone on a mobile phone and contains significant amount of background noise and interference. At the same time the underlying signal is the same exact music recording which can help find landmark features and representations that are invariant to these distortions. Cover song detection is the more subtle problem of finding versions of the same song possibly performed by different artists, instruments, and tempo. As the underlying signals are completely different it requires the use of more high level representations such as chroma vectors that capture information about the chords and the melody of the song without being affected by timbral information. In addition it requires sophisticated sequence matching approaches such as dynamic time warping (DTW) or Hidden Markov Models (HMM) to deal with the potential variations in tempo. Although both of these problems can be viewed as content-based similarity retrieval problems with an appropriately defined notion of similarity they have some unique characteristics. Unlike the more classic similarity retrieval in which we expect the returned results to gradually become less similar, in audio fingerprinting and cover song detection there is a sharper cutoff defining what is correct or not. In the ideal case copies or cover versions of the same song should receive a very high similarity score and everything else a very low similarity score. This specificity is the reason why typically approaches that take into account the temporal evolution of features are more common.

Audio fingerprinting is a mature field with several systems being actively used in industry. As a representative example, we describe a landmark-based audio fingerprinting system based on the ideas used by Shazam, which is a music matching service for mobile phones. In this scheme, each audio track is represented by the location in time and frequency of prominent peaks of the spectrogram. Even though the actual amplitude of these peaks might vary due to noise and audio compression their actual location in the time frequency plane is preserved quite well in the presence of noise and distortion. The landmarks are combined into pairs and each pair is characterized by three numbers  $f_1$ ,  $f_2$ ,  $t$  which are the frequency of the first peak, the frequency of the second peak and the time between them. Both reference tracks and the query track are converted into this landmark representation. The triplets characterizing each pair are quantized with the basic idea being that if the query and a reference track have a common landmarks with consistent timing they are a match. The main challenge in an industrial strength implementation is deciding on the number of landmarks per second and the thresholds used for matching. The lookup of the query landmarks into the large pool of reference landmarks can be performed very efficiently using hashing techniques to effectively create an inverted index which maps landmarks to the files they originate.

To solve the audio cover song detection problem there are two issues that need to be addressed. The first issue is to compute a representation of the audio signal that is not affected significantly by the timbre of the instruments playing but still captures information about the melody and harmony (the combination of discrete pitches that are simultaneously sounding) of the song. The most common representation used in music mining for this purpose are chroma vectors (or pitch class profiles) which can be thought of as histograms showing the distribution of energy among different discrete pitches. The second issue that needs to be addressed is how to match two sequences of feature vectors (chroma vectors in this case) that have different timing and length as there is no guarantee that a cover song is played at the same tempo as the original and there might be multiple sections each with different timing.

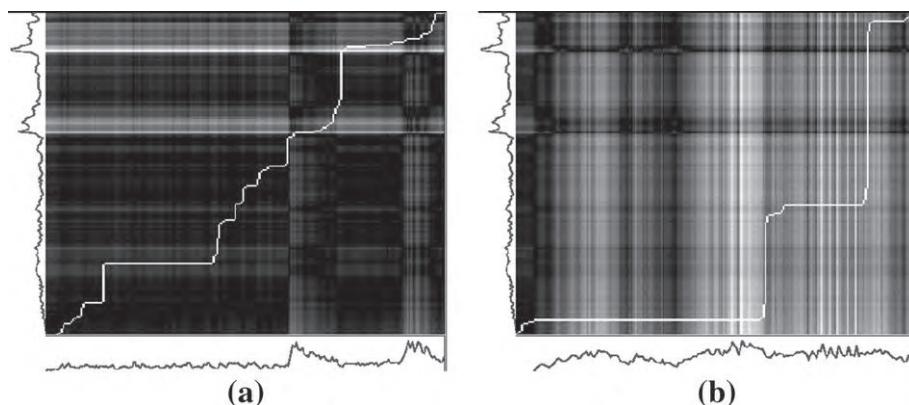
### 1.26.5.3 Sequence matching

Sequence matching algorithms are used for a variety of tasks in MIR including polyphonic audio and scores alignment, and real-time score following. More formally the problem is given two sequences of feature vectors with different lengths and timings find the optimal way of “elastically” transforming by the sequences so that they match each other. A common technique used to solve this problem, and also frequently employed in the literature for cover song detection, is dynamic time warping (DTW) a specific variant of dynamic programming. Given two time series of feature vectors  $X = (x_1, x_2, \dots, x_M)$  and  $Y = (y_1, y_2, \dots, y_N)$  with  $X, Y \in \mathbb{R}^d$  the DTW algorithm yields an optimal solution in  $O(MN)$  time where  $M$  and  $N$  are the lengths of the two sequences. It requires a local distance measure that can be used to compare individual feature vectors which should have small values when the vectors are similar and large values when they are different:

$$d : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \geq 0. \quad (26.8)$$

The algorithm starts by building the distance matrix  $C \in \mathbb{R}^{M \times N}$  representing all the pairwise distances between the feature vectors of the two sequences. The goal of the algorithm is to find the **alignment or warping path** which is a correspondence between the elements of the two sequences with the boundary constraint that the first and last elements of the two sequences are assigned to each other. Intuitively for matching sequences the alignment path will be roughly diagonal and will run through the low-cost areas of the distance matrix. More formally the alignment is a sequence of points  $(p_i, p_j) \in [1 : M] \times [1 : N]$  for which the starting and ending points must be the first and last points of the aligned sequences, the points are time-ordered and each step size is constrained to either move horizontally, vertically or diagonally. The cost function of the alignment path is the sum of all the pairwise distances associated with its points and the alignment path that has the minimal cost is called the **optimal alignment path** and is the output of the DTW.

Figure 26.5 shows two distance matrices that are calculated based on energy contours of different orchestra music movements. The left matrix is between two performances by different orchestras of the



**FIGURE 26.5**

Similarity matrix between energy contours and alignment path using dynamic time warping. (a) Good alignment (b) bad alignment.

same piece. Even though the timing and duration of each performance is different they exhibit a similar overall energy envelope shown by the energy curves under the two axes. The optimal alignment path computed by DTW is shown imposed over the distance matrix. In contrast the matrix on the right shows the distance matrix between two unrelated orchestral movements where it is clear there is no alignment and the optimal alignment path deviates significantly from the diagonal.

Hidden Markov Models (HMM) are a probabilistic sequence modeling technique. The system is modeled as going through a set of discrete (hidden) states over time following Markov transitions, i.e., each state only depends on the value of previous state. In regular Markov models the only parameters are the state transition probabilities. In HMM the states are not directly visible but their probabilistic output is visible. Each state has an associated probability density function and the goal of HMM training is to estimate the parameters of both the transition matrix and the state-dependent observation matrix. For sequence matching the estimated sequence of hidden states can be used.

#### 1.26.5.4 Cover song detection

Cover song detection is performed by applying DTW between all the query song and all the references and returning as a potential match the one with the minimum total cost for the optimal alignment path. Typically the alignment cost between covers of the same song will be significantly lower than the alignment cost between two random songs. DTW is a relatively costly operation and therefore this approach does not scale to large number of songs. A common solution for large scale matching is to apply an audio fingerprinting type of approach with efficient matching to filter out a lot of irrelevant candidates and once a sufficient small number of candidate reference tracks have been selected apply pair-wise DTW between the query and all of them.

Table 26.2 shows the results of the audio cover song detection task of MIREX 2009 in the so called “mixed” collection which consists of 1000 pieces that contain 11 “cover song” each represented by 11 different versions. As can be seen the performance is far from perfect but it is still impressive given the difficulty of the problem. An interesting observation is that the objective evaluation measures are not consistent. For example the RE algorithm performs slightly worse than the SZA in terms of mean average precision but has better mean rank for the first correctly identified cover. Table 26.3 shows the results of the MIREX 2009 audio cover song detection task for the Mazurkas collection which consists 11 different performances/versions of 49 Chopin Mazurkas. As can be seen from the results this is a easier dataset to find covers probably due to the smaller size and more uniformity in timbre. The RE algorithm is based on the calculation of different variants of chroma vectors utilizing multiple feature sets. In contrast to the more common approach of scoring the references in a ranked list and setting up a threshold for identifying covers it follows a classification approach in which a pair is either classified as reference/cover or as

**Table 26.2** 2009 MIREX Audio Cover Song Detection-Mixed Collection

	RE	SZA	TA
Mean # of covers in top 10	6.20	7.35	1.96
Mean Average Precision	0.66	0.75	0.20
Mean Rank of first correct cover	2.28	6.15	29.90

**Table 26.3** 2009 MIREX Audio Cover Song Detection—Mazurkas

	<b>RE</b>	<b>SZA</b>	<b>TA</b>
Mean # of covers in top 10	8.83	9.58	5.27
Mean Average Precision	0.91	0.96	0.56
Mean Rank of first correct cover	1.68	1.61	5.49

reference/non-cover. The SZA algorithm is based on harmonic pitch class profiles (HPCP) which are similar to chroma vectors but computed over a sparse harmonic representation of the audio signal. The sequence of feature vectors of one song is transposed to the main tonality of the other song in consideration. A state space representation of embedding  $m$  and time delay  $z$  is used to represent the time series of HPCP with a recurrence quantification measure used for calculating cover song similarity.

## 1.26.6 Classification

Classification is the task of assigning each object (in our case a music track) to one of several pre-defined categories of interest. In data mining it refers to principled ways of building classification systems using as input a set of objects with associated ground truth classification labels which is called the training set. It is a subset of the more general concept of supervised learning in which an algorithm “learns” how to improve its performance over a particular task by analyzing the results on this task provided by humans during a training phase. One of the simplest classifiers is based on techniques such as the ones described in the similarity retrieval section. A new object represented by a vector of attributes/features is classified to the category of its nearest neighbor in the training set. A common variant is to consider the classification label of the majority of  $k$  nearest neighbors where  $k$  is an odd number. In addition a variety of techniques have been proposed specifically for this task. They include rule-based classifiers, decision trees, neural networks, support vector machines and various parametric classifiers based on Bayesian decision theory such as Naive Bayes and Gaussian Mixture Models. These techniques differ in terms of the assumptions they make, the time they take to train, and the amount of data they require to work well. Their goal is to fit as well as possible the relationship between the attributes or features that characterize the music tracks and the ground truth class label. The trained model should not only predict the class label of the tracks it has encountered in training but of new music tracks it has never encountered before.

Classification techniques can be grouped into two large families: generative and discriminative. In generative approaches the classification problem is recast, through Bayesian decision theory to the problem of estimating a probability density function from a set of samples (the feature vectors of a particular class in the training set). They are called generative because the estimated model can be used to generate new random samples (features). Given a classification task of  $M$  classes/categories,  $c_1, c_2, \dots, c_M$  and an unknown object (in our case music track) represented as a feature vector  $\mathbf{x}$  the goal is to calculate the  $M$  conditional probabilities (also referred to as a posteriori probabilities)  $P(c_i | \mathbf{x})$  where  $i = 1, 2, \dots, M$ . The predicted class label for the unknown vector  $\mathbf{x}$  will then be the  $c_i$  corresponding

to the maximum conditional probability. Using the *Bayes Rule* these conditional probabilities can be written as:

$$P(c_i|\mathbf{x}) = \frac{p(\mathbf{x}|c_i)P(c_i)}{p(\mathbf{x})}. \quad (26.9)$$

The prior probabilities  $P(c_i)$  can be calculated by simply counting the number of instances belonging to each class in the training set, so the main challenge is to estimate the probability density function of the feature vectors given the class  $p(\mathbf{x}|c_i)$ . Many classification algorithms solve this problem by assuming a particular parametric form for the probability density function and then estimating the parameters from the observed samples for the particular class. For example, the simple Naive Bayes classifier assumes that the probability density function corresponding to each feature is conditionally independent given the class. Therefore it can be expressed as the product of normal distributions (one for each feature). The parameters that need to be estimated are the means and variances of each feature for the particular class. It can be shown that these correspond to the statistical means and variances of the observed samples in the training set. In music mining the most common generative modeling technique used is *Gaussian Mixture Models* (GMM) in which each class is modeled as a weighted combination of samples from  $K$  Gaussian models.

The main insight behind discriminative approaches is that what is important in a classification problem is not to model the distribution of all the observed feature vectors, but rather to focus on the areas of the feature space in which the decision of which class the vector belongs is unclear. Essentially discriminative approaches try to directly solve the classification problem by making some assumptions about a discriminative function (i.e., a function that given as input an unknown feature vector returns a predicted class) and then optimizing the parameters of this function according to some criterion. For example, the assumption can be that the discriminant function is a linear combination of the features (geometrically corresponding to a hyperplane in the feature space) and the criterion might be the number of errors in the training set. In music mining the most common discriminative model used is *Support Vector Machines* (SVM). Another way of grouping classification algorithms is into parametric approaches in which each class is characterized by a small set of parameters (both the GMM and SVM classifiers are parametric methods) and non-parametric methods in which there is no explicit parameter representation. The canonical example of a non-parametric classifier is the *Nearest Neighbor* rule which simply classifies an unknown instance with the label associated with the training instance that is closest to it in the feature space. In many music mining classification problems the predicted labels form natural hierarchies. For example Folk Metal is a subgenre of Heavy Metal and Hostility is a subordinate emotion to Anger. A straightforward way of solving hierarchical classification problems is to apply standard classification techniques for the top level categories and then train classifiers for the subordinate categories separately for each top-level category. One issue with this approach is that by making a hard decision at each level errors can be propagated. An alternative is to view the hierarchy as a probabilistic graphical model and essentially compute conditional probabilities for the decisions at each level of the hierarchy using classifiers that output probabilities over all possible classes rather than a single classification decision.

### 1.26.6.1 Genre classification

Music can be grouped into categories in many different ways and most of them have been investigated in the literature. Probably the oldest classification task that was investigated was musical genre

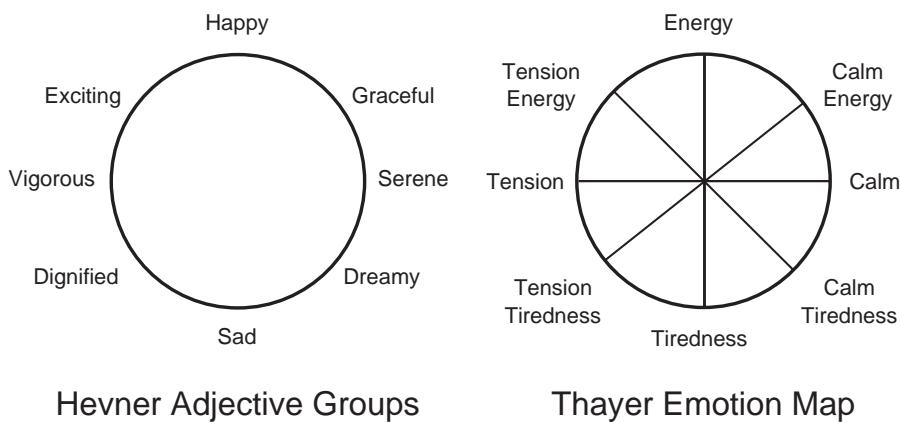
classification. Other tasks include artist/singer/performer classification, mood and emotion detection, instrument recognition and others. The boundaries between categories such as genre labels are fuzzy. Even though there is considerable agreement when listeners are asked to annotate an unknown piece of music with a genre label, there are also individual differences. Top-level genres such as classical or Reggae are more consistently identified, while more obscure genres such as folk metal or grime are meaningful only to smaller groups of listeners. Given the subjective nature of human genre annotations it is unreasonable to expect perfect computer genre classification performance (in fact the notion of perfect performance is only meaningful in relation to some ground truth which in this case will be inherently subjective). One fascinating finding is that average listeners are able to classify a music track with a label from a set of pre-defined top-level genres, with accuracy better than random with exposure to as little as 250 ms (1/4 of a second or approximately the time it takes to say the word “word”) and require only 3 s to reach their best classification performance. This indicates that low level audio-related features carry sufficient information for reliable genre classification. In a well-known user study (details can be found in the Further Reading section) a mean subject agreement of about 70% to the genres assigned by music companies was reported. Further studies have shown that on the same dataset computer classification achieves results comparable to some of the “worst” performing humans (69%) but not as good as the “best” performing humans (95%). In this case the ground-truth is defined as the genre labels that is assigned by the majority of the 27 users that participated in the study rather some external authority. Therefore the “worst” and “best” performance refer to subject agreement with the majority rather than any form of music knowledge. It also has been pointed out that listeners probably can be clustered into groups so that the perception of genres within a particular group is more consistent than across groups. In the majority of published work in automatic genre classification these issues are not considered and the ground truth is known and externally provided.

Automatic genre classification is one of the most popular topics in music mining to a large extent due to the simplicity of obtaining ground truth, the availability of datasets for which there are published results, and the direct mapping to classification techniques. It has also served as a good initial test-bed for experimenting with alternative audio and music feature sets. At the same time for a lot of music of interest the genre labels are already available (although for some cases like World music they are too generic to be useful).

### 1.26.6.2 Emotion and mood classification

Listeners can easily identify emotions and moods that are present in pieces of music. It is also known that music can induce emotions and moods to listeners. Therefore it is desirable to utilize emotion and mood related information in music retrieval systems. Unlike other types of audio classification such as genre in which the classification labels are to some extent pre-defined and readily available for particular music tracks, one of the challenges in emotion and mood classification is deciding what labels should be used and obtaining them for a particular set of music tracks.

Roughly speaking mood is a more persistent state of mind and can encompass different emotions whereas emotions are more instinctive. Most of the literature in emotion and mood classification relies on various schemes for describing emotions/moods from psychology. For example Hevner experimentally found eight adjective groups for describing music based on a user study of 450 subjects listening to 26 pieces of classical music. These groups are: dignified, sad, dreamy, serene, graceful, happy, exciting,

**FIGURE 26.6**

Different ways of organizing emotions.

vigorous and were arranged in a circle in the order they were listed such that the changes are gradual. Figure 26.6 shows this representation.

Schemes of describing emotion can be roughly divided into two general families: models based on hierarchies of words describing the different emotions and two dimensional (or sometimes even three dimensional) continuous emotion spaces where each axis corresponds to a pair of adjectives having opposite meanings (for example, happy and sad). A particular emotion is characterized as a point in the emotional space. The use of words to represent emotion (or affect) can be problematic as multiple emotions can be experienced at the same time and there is individual variability in how a particular experience is assessed. Figure 26.6 shows an example of emotion grouping and an example of an emotion space.

In terms of data mining techniques, standard classification approaches as the ones described above can be used. The most common sources of training data in the literature are either audio features, text features based on analyzing lyrics, or more generally tags which are described below. Hierarchical classification techniques can also be employed to deal with multiple levels of classification. In order to deal with the fact that multiple emotion/mood words might apply to the same music track a multi-label classification approach can be employed. In traditional classification there is a single ground truth label (from a predefined set of categories) for each training instance and a single predicted label (from the same set of categories) for each testing instance. In multi-label classification there are multiple labels (still from a predefined set of categories) associated with each instance. One can categorize multiple-label classification approaches into two broad families. The first family consists of problem transformation methods that convert a multi-label classification problem to a set of standard classification problems. For example each label can be treated separately as a binary classification problem with instances that are annotated with the label being positive examples and instances that do not contain the label as negative examples. This approach requires training  $K$  binary classifiers where  $K$  is the number of distinct labels. The second family consists of classification approaches that can be directly applied to

multi-label classification problems. A simple example is the *Nearest Neighbor* classifier in which a new instance is annotated by all the labels that are associated with the training instance that is closest to it in the feature space.

When dealing with a continuous emotional space both the ground truth and the predicted outcome are continuous rather than categorical. In data mining, regression refers to the prediction of a continuous output label from a set of continuous attributes. There is a large variety of regression algorithms which similarly to classification algorithms can be categorized as parametric and non-parametric. In many cases they are based on similar ideas to classification algorithms. For example support vector regression (SVR) is a counterpart to support vector machines (SVMs) and ensemble boosting regression (AdaBoost.RT) is the counterpart of the AdaBoost ensemble classification algorithm.

### 1.26.6.3 Evaluating classifier performance

The goal of a classifier is to be able to classify objects it has not encountered before. Therefore in order to get a better estimate of its performance on unknown data it is necessary to use some of the instances labeled with ground truth for testing purposes and not take them into account when training. The most common such evaluation scheme is called  $K$ -fold cross-validation. In this scheme the set of labeled instances is divided into  $K$  distinct subsets (folds) of approximately equal sizes. Each fold is used for testing once with the  $K - 1$  remaining folds used for training. As an example if there are 100 feature vectors then each fold will contain 10 feature vectors with each one of them being used one time for testing and  $K - 1$  times for training. The most common evaluation metric is classification accuracy which is defined as the percentage of testing feature vectors that were classified correctly based on the ground truth labels. When classifying music tracks a common post-processing technique that is applied is the so-called artist filter which ensures that the feature vectors corresponding to tracks from the same artist are not split between training and testing and are exclusively allocated only to one of them. The rationale behind artist filtering is that feature vectors from the same artist will tend to be artificially related or correlated due to similarities in the recording process and instrumentation. Such feature vectors will be classified more easily if included in both training and testing and maybe inflate the classification accuracy. Similar considerations apply to feature vectors from the same music track if each track is represented by more than one feature vector, in which case a similar track filter should be applied.

The most common evaluation metric for automatic classification is accuracy which is simply defined as the number of correctly classified instances in the testing data. It is typically expressed as a percentage. Additional insight can be provided by examining the confusion matrix which is a matrix that shows the correct classifications in the diagonal and shows how the misclassification are distributed among the other class labels.

Table 26.4 shows classification results from MIREX 2010. The classification results are shown as percentage accuracy. They are sorted based on the performance on the largest of the datasets considered (the Genre column). This dataset consists of 7000 clips each 30 s long. The following ten genres are all equally represented (700 clips for each genre): blues, jazz, country, baroque, classical, romantic, electronica, hiphop, rock, metal. The mood dataset consists of 600 30-second clips classified into 5 mood clusters. The labeling was done by human judges. Table 26.5 shows the 5 mood clusters used. The Latin dataset consists of 322 audio files representing 10 Latin music genres (axe, bachata, bolero, forro, gaucha, merengue, pagode, sertaneja, tango) sourced from Brazil and labeled by music experts. These genres are

**Table 26.4** 2010 MIREX Classification Tasks

	<b>Genres</b>	<b>Mood</b>	<b>Latin</b>
SSPK1	73.64	63.83	79.86
BRPC1	70.67	58.67	70.75
BRPC2	70.00	59.00	—
GR1	69.80	60.67	60.18
FE1	69.64	60.83	69.32
RRS1	67.89	61.67	62.53
BPME2	67.66	54.67	—
MW1	67.57	54	37.93
TN4	66.66	57.5	48.54
WLB1	66.11	55.5	68.11
GP1	64.27	63.17	66.9
RK1	64.03	54.83	50.85
TN1	63.37	55.5	36.83
MBP1	63.29	54.0	61.02
HE1	61.16	54.17	50.76
RJ1	61.07	54.83	61.98
JR2	60.94	51.17	60.74
JR4	60.93	51.17	60.27
RK2	60.54	47.67	42.14
MP2	60.43	36.17	57.30
JR1	60.01	46.33	56.49
RJ2	59.73	50.17	59.75
JR3	59.54	46.83	57.95
WLB2	48.50	57.67	34.99

**Table 26.5** 2010 MIREX Mood Clusters

Cluster 1	Passionate	Rousing	Confident	Boisterous	Rowdy	
Cluster 2	Rolling	Cheerful	Fun	Sweet	Amiable	Good natured
Cluster 3	Literate	Poignant	Wistful	Bittersweet	Autumnal	Brooding
Cluster 4	Humorous	Silly	Campy	Quirky	Whimsical	Witty
Cluster 5	Aggressive	Fiery	Tense	Intense	Volatile	Visceral

more differentiated by rhythmic characteristics than the other datasets considered. For computing these classification accuracies, for all datasets, a three-fold cross-validation with artist filter (i.e., all songs of an artists are either part of the training or testing) approach was used. The algorithms differ in terms of the exact details of the feature extraction and the type of supervised learning classifier they utilize.

#### 1.26.6.4 Clustering

Clustering is the problem of partitioning a finite set of objects (music tracks in our case) into groups called clusters such that objects belonging to the same cluster are similar to each other, and objects belonging to different clusters are dissimilar. Similarly to classification, this is a well investigated problem in the area of data mining for which several algorithms have been proposed. In music mining it can be used to automatically organize a music collection into coherent groups, or identify users with different musical interest as well as automatically construct hierarchies of music tags. Similarly to classification typically some combination of audio features and text such as lyrics is utilized in music clustering. One of the classic algorithms for clustering is *K*-means which partitions  $N$  instances into  $K$  clusters such that each instance belongs to the “nearest” cluster and clusters are characterized by the mean of the instances that are assigned to them. The most common algorithm for *K-Means* clustering uses an iterative refinement technique consisting of two steps. In the first assignment step each instance is assigned to the cluster with the closest mean. The initial means can be either random or somehow spread in the feature space. In the second update step the means characterizing each cluster are updated to reflect the instances that have been assigned to them. The two steps are repeated with this new set of cluster means until the assignments no longer change. A somewhat similar approach can be used with the so called *EM algorithm* for finding clusters based on a Gaussian Mixture Model.

---

#### 1.26.7 Tag annotation

The term “tag” refers to any keyword associated to an article, image, video, or piece of music on the web. In the past few years there has been a gradual shift from manual annotation into fixed hierarchical taxonomies, to collaborative social tagging where any user can annotate multimedia objects with tags (so called folksonomies) without conforming to a fixed hierarchy and vocabulary. For example, Last.fm is a collaborative social tagging network which collects roughly 2 million tags (such as “saxophone,” “mellow,” “jazz,” “happy”) per month and uses that information to recommend music to its users. Another source of tags are “games with a purpose” where people contribute tags as a by-product of doing a task that they are naturally motivated to perform, such as playing casual web games. For example TagATune is a game in which two players are asked to describe a given music clip to each other using tags, and then guess whether the music clips given to them are the same or different.

Tags can help organize, browse, and retrieve items within large multimedia collections. As evidenced by social sharing websites including Flickr, Picasa, Last.fm, and You Tube, tags are an important component of what has been termed as “Web 2.0.” The goal of automatic tag annotation is to automatically predict tags by analyzing the musical content without requiring any annotation by users. Such systems typically utilize signal processing and supervised machine learning techniques to “train” autotaggers based on analyzing a corpus of manually tagged multimedia objects. Music classification can be viewed as a specialized restricted form of tag annotation where there is fixed vocabulary and only one tag applies to each music track.

There has been considerable interest for automatic tag annotation in multimedia research. Automatic tags can help provide information about items that have not been tagged yet or are poorly tagged. This avoids the so called “cold-start problem” in which an item can not be retrieved until it has been tagged. Addressing this problem is particularly important for the discovery of new items such as recently

released music pieces in a social music recommendation system. Another also automatic approach is to use text mining techniques to associate tags to particular music tracks.

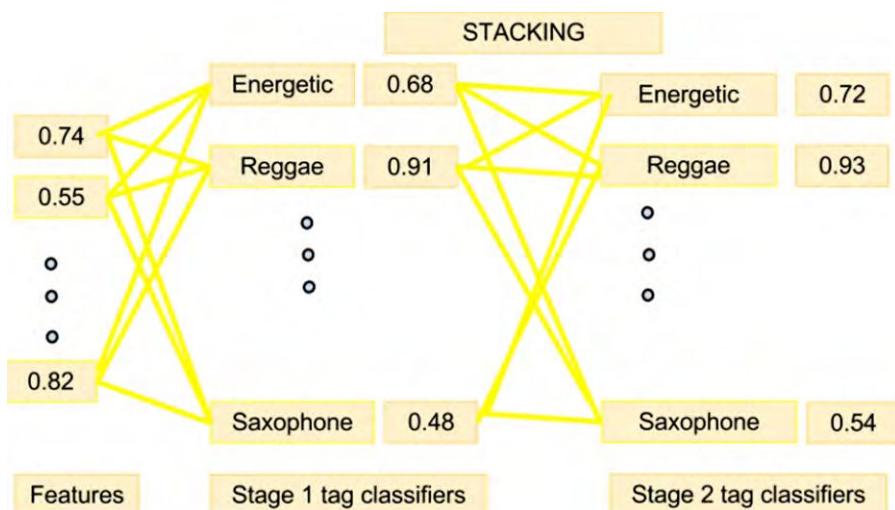
The training data for an automatic tag annotation system is typically represented as a *tag-track* matrix  $X$  where each element  $x[t, s]$  represents the strength of association between a particular tag  $t$ , and a particular song  $s$ . For example, the strength of association for a particular entry in the matrix can be the number of users that annotated that particular song with a particular tag.

From a machine learning perspective automatic tag annotation can be viewed as a variation of multi-label classification. In contrast to traditional classification in which each item is assigned one of  $k$  mutually exclusive class labels, in multi-label classification each item can be assigned to multiple labels (tags). Many different approaches to multi-label classification have been proposed in the literature. They leverage feature information computed from a training set of examples annotated with multiple labels to train models that can subsequently be used to predict labels for new examples. There are some unique characteristics and related challenges when the ground truth tag data is obtained from the web. The ground truth training data is *noisy* in the sense that the tags can contain synonyms (“calm” and “mellow”), misspellings (“chello”) and hierarchical relations (“symphony” and “classical”). In addition the data is *sparse* meaning that there can be few training examples for a given tag. Finally, the absence of a tag cannot always be taken to mean that the tag is not applicable, as it might be the case that the users have simply not yet considered that tag for the particular multimedia item. This phenomenon has been termed weak labeling in contrast to regular strong labeling.

A common straightforward approach to auto-tagging is to train  $K$  binary classifiers that classify each of the  $K$  tags independently. Instances that contain a tag are considered positive, and instances that do not contain it are considered negative. A new instance is annotated with all the tags that the binary classifiers predict as positive. Topic models such as *Latent Dirichlet Allocation* (LDA) make the assumption that tags can be grouped into an unknown number of higher level groups and build a probabilistic model around this assumption. Pre-processing and post-processing techniques that take into account semantic similarities between tags can also be used to improve the results. For example misspellings can be merged in a preprocessing step. A data driven approach that attempts to capture some of the dependencies between tags directly from the data is called *stacking*.

*Stacking* is a method of combining the outputs of multiple independent classifiers for multi-label classification. The first step of using stacking for multi-label classification is to train  $|V|$  individual tag classifiers using a training set  $(\mathbf{x}_i, \mathbf{y}_i)$ , where  $\mathbf{x}_i$  denotes the feature vector for instance  $i$  and  $\mathbf{y}_i$  is the associated set of labels. The output of these classifiers (binary or probabilistic)  $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{|V|}(\mathbf{x})$  where  $\mathbf{x}$  is the input feature vector that can then be used as a feature to form a new feature set. Let the new feature set be  $z_1, z_2, \dots, z_{|V|}$ . This feature set, together with the original ground truth labels  $(\mathbf{z}_i, \mathbf{y}_i)$ , is then used for training a second stage of stacking classifiers. The goal is to have the stacking classifiers make use of information like the correlation between tags and the accuracy of the first stage classifiers to improve the annotation performance. For example suppose that the stage 1 performance for the tag “opera” is not very good but that most of the examples with the tag “opera” receive high probabilities for the tags “classical” and “voice” at stage 1. The stacking stage 2 can take into account this information from other tags and improve annotation performance: something not possible during stage 1, in which each tag is treated independently. Figure 26.7 shows this process as a block diagram.

Evaluation of automatic tagging systems is not trivial. In general, the evaluation metrics used are generalizations of commonly used evaluation metrics for single label classification. An annotated “training”

**FIGURE 26.7**

Stacking for automatic music tag annotation.

set of instances is used to “train” the classifier, and then used to “predict” the tags for a set of instances in a “test” set. We can also distinguish between evaluation metrics that are based on a predicted set of discrete tags (sometimes referred to as the annotation task) and ones that are based on a predicted set of tag affinities/probabilities (sometimes referred to as the ranking task) for each instance in the testing set. A common approach is to treat any classification decision equally and simply count the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to derive well-known measures such as precision, recall and F-measure. In the case of probabilistic output, multiple score thresholds can be considered as possible boundaries for binarization, in which case it is common to use *Receiver operating characteristic* (ROC) curves. An ROC curve is a plot of the true positive rate as a function of the false positive rate. The ROC curve can be summarized by the area under curve (AUC-ROC) which can be found by integrating the ROC curve and is upper bound by 1.0. Random guessing in a retrieval task results in an AUC-ROC of 0.5. Different tag annotation methods can have different operating point characteristics in terms of the trade-off between true positives and false positives. A final complication is that calculating metrics over the entire set of tags can be misleading as good performance on “popular” tags that appear in many instances will dominate. However, typically a more balanced response where all tags are considered is desired. In order to address this concern, evaluation metrics averaged across tags are used. Finally it is important to note that in most cases evaluation metrics based on annotated ground truth underestimate what the true performance of the system would be if evaluated by humans. The reason is that frequently, predicted tags that humans would consider applicable are not present in the ground truth and therefore evaluated as mistakes.

Table 26.6 shows both binary (F-measure) and affinity (AUC-ROC) evaluation metrics for automatic tag annotation from MIREX 2009 using the MajorMiner data-set. This data-set consists of 2300 clips

**Table 26.6** MIREX 2009 Tag Annotation Results (MajorMiner, Mood)

Measure	BP2	CC4	GP	GT2	HBC	LWW2
MajorMiner						
F-measure	0.29	0.26	0.01	0.29	0.04	0.31
AUC-ROC	0.76	0.75	—	0.79	0.74	0.80
Mood						
F-measure	0.19	0.18	0.08	0.21	0.06	0.70
AUC-ROC	0.63	0.64	—	0.65	0.66	0.80

selected at random from 3900 tracks. Each clip is 10 s long. The clips represent a total of 1400 different tracks on 800 different albums by 500 artists. To give a sense of the diversity of the music collection, the following genre tags have been applied to these artists, albums, and tracks on Last.fm: electronica, rock, indie, alternative, pop, britpop, idm, new wave, hiphop, singer-songwriter, trip-hop, post-punk, ambient, jazz. The MajorMiner game has collected a total of about 73,000 taggings, 12,000 of which have been verified by at least two users. In these verified taggings, there are 43 tags that have been verified at least 35 times, for a total of about 9000 verified uses. These are the tags are used in this task. The table also shows the results in the Mood dataset which consists of 3469 unique songs and 135 mood tags organized into 18 mood tag groups which where used as tags. The songs are Western pop songs mostly from the USPOP collection. Each song may belong to multiple mood tag groups. The main rationale on songs selection is: if more than one tag in a group were applied to a song, or if one tag in a group was applied more than once to a song, this song is marked as belonging to this group.

In this task, the participating groups submitted variants of the same algorithm. Due to space constraints we only show the top performing variant of each group. The evaluation was performed using 3-fold cross validation using artist filtering, i.e., the training and test sets contained different artists. The results are averaged across tags. The BP2 system is based on spectral and chroma features and for classification applies both feature selection and model selection. For model selection instead of using classification accuracy, as is common, they use a custom measure designed to deal better with unbalanced data-sets and over-fitting. Each tag is classified separately. The CC4 system is based on MFCC and delta-MFCC that are then transformed into a super-vector using a universal background approach. In this approach, a Gaussian Mixture Model is trained using a large corpus of music that is separate from the files considered for tag annotation. This universal background model is then adapted to better fit the data of the tag dataset. The parameters of the adapted GMM are then used as features and an ensemble classifier consisting of a linear support vector machine and AdaBoost classifier is utilized for classification. Each tag is treated separately. The GPP system uses spectral and chroma features followed by GMM modeling both at the frame level and track level. Each tag is treated as a separate classification problem. GT2 also utilized spectral and chroma features followed by a layer of tag-specific linear support vector machines followed by a stacking layer of also linear support vector machines. The HBC system utilizes bag of codewords representation (basically an occurrence histogram of different codewords which are computed by  $k$ -means clustering-this approach is also known as vector quantization) with MFCC and

delta MFCC as the original feature vectors. For tag annotation a codeword bernoulli average approach is utilized. Unfortunately it is hard to draw any reliable conclusions from these results. The exact details of the features and tag annotation approach are not the same, so it is hard to tell if the observed differences between algorithms are due to the feature set used or the tag annotation algorithm.

### 1.26.8 Visualization

Visualization is the display of information in such a way that relationships among data items and attributes can be analyzed and understood. It takes advantage of the strong pattern recognition properties of the human visual system. Traditional visualizations of music, such as the time domain waveforms and spectrograms, popular in audio editors convey very little information about the musical content and are focused on single music tracks. In this section we focus on visualizations of audio collections and associated information about them with specific emphasis on visualizing large collections of music. As we have already discussed a common approach in automatic analysis of music is to represent each track as a feature vector of fixed dimensionality (typical numbers range from 10 to 1000 dimensions). Dimensionality reduction methods try to transform these feature vectors to either 2 or 3 dimensions so that they can be visualized in a natural way by considering the transformed feature vector as a coordinate/point in a visual space. One of the most common technique for dimensionality reduction that can be used for visualization is Principal Component Analysis (or PCA). An alternative is the use of self-organizing maps (SOM) which attempt to perform both dimensionality reduction and clustering. These two methods underlie the majority of proposed music collection visualization systems. In this section we describe these two algorithms and describe generic visualization interfaces that can be built using them. Additional details such as how the user interacts with the generated visualization, the ability to zoom in and out, display hierarchies, etc. can be found in the further reading section.

Principal component analysis (PCA) converts a set of feature vectors with possibly correlated attributes into a set of feature vectors where the attributes are linearly uncorrelated. These new transformed attributes are called the principal components and when the method is used for dimensionality reduction their number is less than the number of original attributes. Intuitively this transformation can be understood as a projection of the original feature vectors to a new set of orthogonal axes (the principal components). The projection is such that each succeeding axis explains the highest variance of the original dataset possible, with the constraint that it is orthogonal to the preceding component. In a typical application scenario, where each song is represented by a 70 dimensional feature vector, the application of PCA to these feature vectors can then be used to convert them to 3 dimensional feature vectors which can be visualized as points in a 3D space. A common way of calculating PCA is based on the covariance matrix which is defined as:

$$C = \frac{1}{N} \sum B \times B^T, \quad (26.10)$$

where  $T$  denotes the transpose operator and  $B$  is the matrix resulting from subtracting the empirical mean of each dimension of the original data matrix consisting of the  $N$  feature vectors. The eigenvectors and eigenvalues of this covariance matrix are then computed and sorted in order of decreasing eigenvalue and the first  $K$  where  $K$  is the desired number of reduced dimensions are selected as the new basis

vectors. The original data can then be projected into the new space spanned by these basis functions, i.e., the principal components. PCA is a standard operation in statistics and it is commonly available in software package dealing with matrices.

One of the potential issues with PCA for music visualization is that, because it tries to preserve as much as possible the distances between points from the original feature space to the transformed feature space, it might leave large areas of the available visualized space empty of points. This is particularly undesirable in touch based interfaces, such as tablets, where in the ideal case, everywhere a user might press should trigger some music. Self-organizing maps are an approach that attempts to perform both dimensionality reduction and clustering while mostly preserving topology but not distances. It can result in more dense transformed feature spaces that are also discrete in nature in contrast to PCA which produces a transformed continuous space that needs to be discretized.

The SOM is a type of neural network used to map a high dimensional input feature space to a lower dimensional representation. It facilitates both similarity quantization and visualization. It was first documented in 1982 and since then it has been applied to a wide variety of diverse clustering tasks. The SOM maps the original  $d$ -dimensional feature vectors  $X \in \mathbb{R}^d$  to two discrete coordinates  $I \in [1 \dots M]$  and  $J \in [1 \dots N]$  on a rectangular grid.

The traditional SOM consists of a 2D grid of neural nodes each containing a  $n$ -dimensional vector,  $\mathbf{x}(t)$  of data. The goal of learning in the SOM is to cause different neighboring parts of the network to respond similarly to certain input patterns. This is partly motivated by how visual, auditory and other sensory information is handled in separate parts of the cerebral cortex in the human brain. The network must be fed a large number of example vectors that represent, as closely as possible, the kinds of vectors expected during mapping. The data associated with each node is initialized to small random values before training. During training, a series of  $n$ -dimensional vectors of sample data are added to the map. The “winning” node of the map known as the *best matching unit* (BMU) is found by computing the distance between the added training vector and each of the nodes in the SOM. This distance is calculated according to some pre-defined distance metric which in our case is the standard Euclidean distance on the normalized feature vectors.

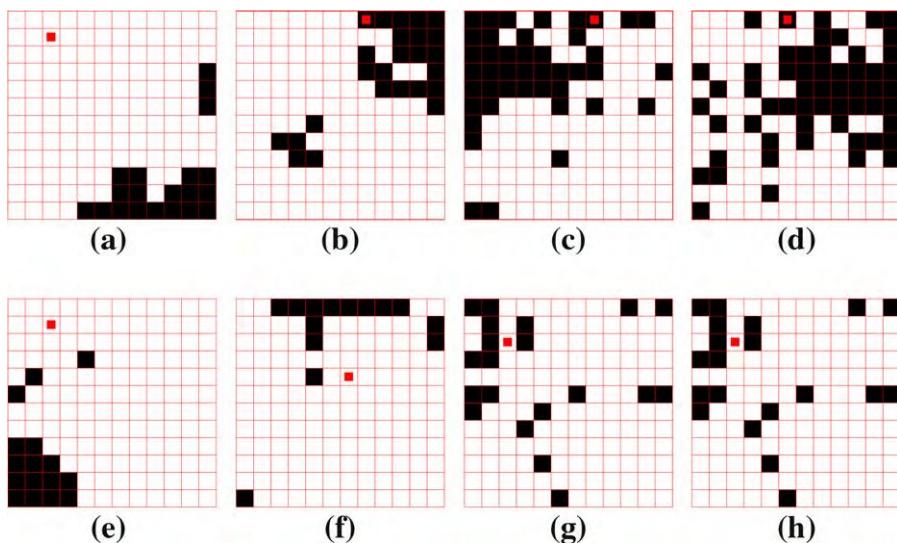
Once the winning node has been defined, it and its surrounding nodes reorganize their vector data to more closely resemble the added training sample. The training utilizes competitive learning. The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The time-varying learning rate and neighborhood function allow the SOM to gradually converge and form clusters at different granularities. Once a SOM has been trained, data may be added to the map simply by locating the node whose data is most similar to that of the presented sample, i.e., the winner. The reorganization phase is omitted when the SOM is not in the training mode.

The update formula for a neuron with representative vector  $\mathbf{N}(t)$  can be written as follows:

$$\mathbf{N}(t+1) = \mathbf{N}(t) + \Theta(v, t)\alpha(t)(\mathbf{x}(t) - \mathbf{N}(t)), \quad (26.11)$$

where  $\alpha(t)$  is a monotonically decreasing learning coefficient and  $x(t)$  is the input vector. The neighborhood function  $\Theta(v, t)$  depends on the lattice distance between the BMU and neuron  $v$ .

Figure 26.8 illustrates the ability of the automatically extracted audio features and the SOM to represent musical content. The top subfigures (a)–(d) show how different musical genres are mapped to different regions of the SOM grid (the black squares are the ones containing one or more songs from

**FIGURE 26.8**

Topological mapping of musical content by the self-organizing map. (a) Classical, (b) metal, (c) hiphop, (d) rock, (e) Bob Marley, (f) radiohead, (g) Led Zappelin (h) Dexter Gorden.

each specific genre). As can be seen Classical, Heavy Metal and HipHop are well-localized and distinct whereas Rock is more spread out reflecting its wide diversity. The SOM is trained on a collection of 1000 songs spanning 10 genres. The bottom subfigures (e)–(h) show how different artists are mapped to different regions of the SOM grid. The SOM in this case is trained on a diverse personal collection of 3000 songs spanning many artists and genres. It is important to note that in all these cases the only information used is actual audio signal that is automatically analyzed and no meta data. The locations of the genres/artists are emergent properties of the SOM and demonstrate how it organizes semantically the data.

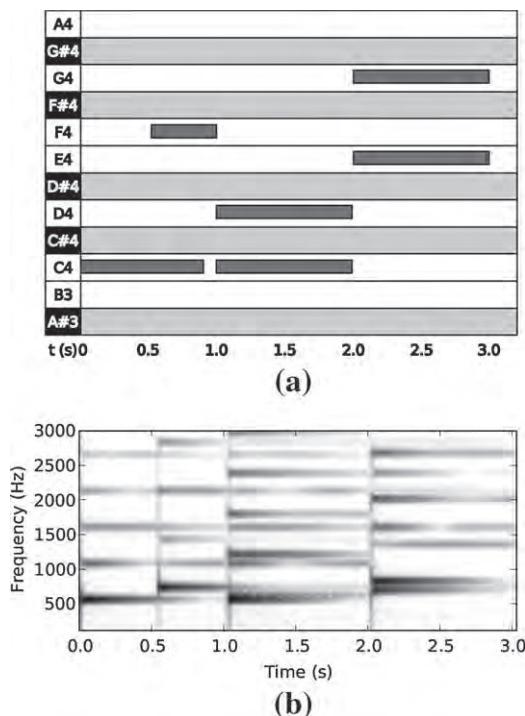
### 1.26.9 Advanced music mining

In addition to the music mining tasks that have been presented in this chapter, there are several additional topics and tasks that are more briefly described in this section. The decision to not cover them in more detail, had more to do with the limited amount of published work related to them and their usage of more recent and complex mining algorithms, rather than their importance. In many cases, they also lack commonly available data-sets and agreed upon evaluation methodologies. Multiple instance learning is a classification technique in which ground truth is provided for sets (bags) of instances rather than individual instances. A classic example in music mining is that frequently it is possible to obtain classification or retrieval ground truth for artists but the desired classification granularity is for songs. As multiple songs correspond to the same artist and the ground truth labeling does not necessarily apply to all of them, this problem is a natural fit for multiple instance learning. Semi-supervised learning is

a type of machine learning that makes use of both labeled and unlabeled data for training. It useful in scenarios where there is a limited amount of ground truth data available but large amounts of data that are unlabeled are also available.

### 1.26.9.1 Symbolic music mining

Music especially classical and to some extent popular music can be represented symbolically using representations that are similar to a music score, i.e., they encode which notes are played and at what time. A score is an abstract and structured representation. It can be interpreted in different ways by varying timing or even instrumentation depending on the performer. It can be viewed as a platonic ideal of a song that is normalized with respect to many details such as timbre, pitch and timing variations that complicate audio analysis. At the same time, because the information is discrete and structured it enables types of processing that are very difficult to perform with audio signals. For example it is possible to search efficiently for melodic patterns in large collections of symbolic data. There are several possible symbolic representations for music. The simplest form consists of simply the start and duration of a set of discrete notes or pitches. *MIDI* (Music Interface for Digital Instruments) files mostly contain this information. More expressive formats, such as *Music XML* can express additional information



**FIGURE 26.9**

Piano roll symbolic representation (a) and associated spectrogram (b).

such as graphic characteristics of the musical scores. Figure 26.9 relates a spectrogram representation with a corresponding symbolic representation using the so called piano roll notation. Algorithms from text mining, discrete mathematics and theoretical computer science are used in this research. It is also frequently driven by musicological considerations.

In the same way that the canonical example of music mining for audio signals is automatic genre classification, the canonical example of symbolic music information is searching for a particular monophonic melodic sequence in a database of melodies. This task has, for example, applications in *Query-by-humming* (QBH) systems where the user sings a melody and the best match is returned as a music file that can be played. Even though some systems only check for exact matches, in most cases, some form of approximate matching is utilized. Melodies are represented as one dimensional strings of characters, where each character represents either a discrete pitch or interval between succeeding pitches. The large literature on string matching algorithms such as editing distances, finding the longest common subsequence, or finding occurrences of one string in another can then be utilized for symbolic music mining. When melody matching needs to be performed in polyphonic audio or the query itself is polyphonic more sophisticated techniques based on computational geometry are utilized. Finally most of the music mining tasks described for audio signals can also be applied to symbolic music signals provided an appropriate feature representation is used. For example features computed from *MIDI* file such as average interval size, highest note, lowest note, etc. can be used as a feature representation for classification tasks.

### 1.26.9.2 Bridging symbolic and audio music mining

*Automatic transcription* refers to the process of essentially converting an audio recording to a symbolic representation. Figure 26.9 shows this process graphically. As can be seen the overlapping harmonic structure of notes makes it harder to separate them in a polyphonic context. Automatic transcription is a hard problem, and although significant progress has been made, it is still not very accurate for real world recordings. There are various simpler tasks related to bridging audio and symbolic representations that have been investigating with more successful results. Audio chord and key detection compute an abstract symbolic representation, i.e., a list of chord symbols for a music track. As an example application, using dynamic time warping (DTW) over chroma representations it is possible to find the correspondence between a symbolic representation and an audio recording. This problem is termed *Polyphonic audio alignment* or score following when it is performed in a real-time fashion. Combinations of symbolic and audio representations can also be used to enable new forms of querying musical data. For example in query-by-humming systems, the user sings or hums a melody and an audio recording that contains the melody is returned. It is also possible to analyze the internal structure of a music piece in terms of repeated sections such as the chorus or a bridge. This process has been termed *structure analysis*. By combining symbolic and audio representations we gain more insight in the complexity and structure of music. In turn, this can inform the more general music mining tasks we have described in this chapter. A task that has received some attention in the popular press but for which there has been no convincing demonstration is hit song science or popularity prediction. The idea is that somehow by analyzing audio and text features one might be able to predict whether a song will be a hit or how popular it will be before it is released on the market.

## 1.26.10 Software and datasets

Although frequently researchers implement their own audio feature extraction algorithms, there are several software collections that are freely available that contain many of the methods described in this chapter. They have enabled researchers more interested in the data mining and machine learning aspects of music analysis to build systems more easily. They differ in the programming language/environment they are written, the computational efficiency of the extraction process, their ability to deal with batch processing of large collections, their facilities for visualizing feature data, and their expressiveness/flexibility in describing complex algorithms.

Table 26.7 summarizes information about some of the most commonly used software resources as of the year 2012. The list is by no means exhaustive but does provide reasonable coverage of what is available in 2012. Some of these links contains collections of code and others more integrated frameworks. Most of the toolkits support some form of audio feature extraction. Some also include music mining algorithms but in many cases researchers rely on other general purpose machine learning and data mining software. Several of the figures in this chapter were created using *Marsyas* and some using custom MATLAB code. In general, although there are exceptions, software in MATLAB tends to be less efficient and more suited for prototyping than C/C++ software that is more efficient for large scale problems. An alternative to using a software toolkit is to utilize audio features provided by a web service such as the Echonest API.

Over time, several audio datasets and their associated ground truth have been collected and used to evaluate different music mining tasks. For more mature music mining tasks they have enabled, to some degree, comparison of different systems and reproducibility of results. Table 26.8 summarizes information about some of the common datasets used and their characteristics as available in 2012. Almost all of the datasets can be used for experiments in mining especially for tasks that can be performed

**Table 26.7** Software for Audio Feature Extraction (in 2012)

Name	URL	Programming Language
Auditory Toolbox	<a href="http://tinyurl.com/3yomxwl">tinyurl.com/3yomxwl</a>	MATLAB
CLAM	<a href="http://clam-project.org/">clam-project.org/</a>	C++
D.Ellis Code	<a href="http://tinyurl.com/6cvtdz">tinyurl.com/6cvtdz</a>	MATLAB
HTK	<a href="http://htk.eng.cam.ac.uk/">htk.eng.cam.ac.uk/</a>	C++
jAudio	<a href="http://tinyurl.com/3ah8ox9">tinyurl.com/3ah8ox9</a>	Java
Marsyas	<a href="http://marsyas.info">marsyas.info</a>	C++/Python
MA Toolbox	<a href="http://pampalk.at/ma/">pampalk.at/ma/</a>	MATLAB
MIR Toolbox	<a href="http://tinyurl.com/365oojm">tinyurl.com/365oojm</a>	MATLAB
Sphinx	<a href="http://cmusphinx.sourceforge.net/">cmusphinx.sourceforge.net/</a>	C++
VAMP plugins	<a href="http://www.vamp-plugins.org/">www.vamp-plugins.org/</a>	C++

**Table 26.8** Datasets for Music Mining (in 2012)

Name	URL	Clips	Ground Truth
BEATLES	<a href="http://isophonics.net/datasets">isophonics.net/datasets</a>	179	Structure, Chord Beat
BILLBOARD	<a href="http://tinyurl.com/9mocd8y">tinyurl.com/9mocd8y</a>	649	Structure, Chord, Beat
CAL500	<a href="http://tinyurl.com/9j98jj6">tinyurl.com/9j98jj6</a>	500	Tags
GTZAN	<a href="http://tinyurl.com/9rff8js">tinyurl.com/9rff8js</a>	1000	Genre, Key
LASTFM	<a href="http://tinyurl.com/9obzlfn">tinyurl.com/9obzlfn</a>	20K (artists)	Tags, Collaborative Filtering
LATIN	—	3227	Genres
MGNTUNE	<a href="http://tinyurl.com/8vkcoqx">tinyurl.com/8vkcoqx</a>	25863	Tags
MSD	<a href="http://tinyurl.com/9mr4j8x">tinyurl.com/9mr4j8x</a>	1000000	Genre, Tags, Ratings
RWC	<a href="http://tinyurl.com/cb9tpfk">tinyurl.com/cb9tpfk</a>	315	Genre, Chords, Structure
YAHOO	<a href="http://tinyurl.com/c8pbcb8">tinyurl.com/c8pbcb8</a>	717 million	Ratings

without ground truth such as similarity retrieval. In addition, several of them provide ground truth for one or more tasks. In many cases, they also contain computed audio features which is particular useful for researchers coming from the more general data mining research. The academic music information retrieval community has been fortunate to have help from industry in creating datasets. For example, the *Magnatagatune* (MGTTUNE) dataset was created with help from the *Magnatune* recording label. Another example of a dataset created with help from industry (EchoNest) is the recent and impressive in scope *Million Song Dataset* (MSD). It is a freely-available collection of audio features and metadata for a million popular music tracks. It also contains additional data such as sets of cover songs, lyrics, song-level tags and similarity and user data. Several datasets are also associated with MIREX tasks. In many cases, they are not accessible for experimentation and participants have to submit their algorithms which are executed by the MIREX organizers in order to get evaluation results.

### 1.26.11 Open problems and future trends

Music information retrieval and more specifically music mining are relatively new emerging research areas so there are many unexplored directions with significant challenges that can be investigated. Music information is multi-faceted, multi-disciplinary, multi-cultural, multi-modal and all these aspects complicate music mining. In this section some open problems and future trends are briefly described. In many of the music mining tasks that have been described in this chapter the assumptions they make are at the extremes of more nuanced differences encountered in the real world. For example music recommendation systems are either general in the sense that for a particular query they recommend the same set of similar tracks independently of who the user is, or they are highly personalized. Similarly, human genre annotations are not consistent but at the same time are not completely personalized. The reality is that probably listeners form social groups/clusters with similar taste. There is a lot of interesting work that can be done to leverage this group membership for more effective recommendations. There is a lot of context information that can also be taken into account. Typical listeners have different listening

preferences depending on the time of the day or the activity they are performing. Given the ubiquity of mobile devices that have geolocation capabilities can we leverage this information to adjust music recommendation to particular activities for a particular user? Recent advances in neuro-imaging have also made possible the monitoring of brain activity while listening to music. The resulting data is high dimensional and difficult to interpret but maybe in the future it will help lead to a better understanding of how our brain interprets music. Such understanding might radically change how we think about music mining.

A big area of future research is audio feature extraction. Existing features are rather crude, low level statistical descriptions that clearly do not capture a lot of the structure and complexity of music. A particularly interesting area that is receiving a lot of attention at the moment in the machine learning community is *Deep Belief Networks (DBN)*. They are a family of machine learning algorithms that are able to learn higher level abstract representations from lower level input signals in the context of classification problems. Another interesting challenge is that music unfolds over time at multiple levels. Many existing approaches to audio feature extraction and music mining ignore this temporal evolution or approximate it using rather crude models of dynamics. The majority of existing systems utilize one source of information and even in the cases where multiple source of information are utilized, their fusion is performed simply and directly. The integration of information from multiple facts at many different time scales is a big challenge for existing mining algorithms.

Another big limiting assumption that the majority of existing algorithm for music mining make is that the music is characterized statistically as a mixture without taking into account that it is composed of individual sound sources. At the same time, it is clear that as humans we pay a lot of attention and are able, to a large extent, to characterize and follow these individual sound sources in complex mixtures over time. This is true for both musically trained listeners and ones that are not. Separating a complex mixture of sounds to the individual components is called sound source separation and is a problem with a lot of existing literature but still far away from being solved. Even if it is solved it is unclear how all these individual sound sources, and their corresponding information can be combined for higher level music mining tasks.

Historically MIR originated from work in digital libraries and text retrieval. Therefore it has retained this focus on processing of large archives of existing music. However, new music is created, performed every day and increasingly computers are used for its production, distribution and consumption. There is a lot of potential in using MIR techniques in the context of music creation and especially live music performance. However, many of the developed techniques are not designed to work in real-time and be interactive.

The amount of data that needs to be processed constantly poses scalability challenges. For example if we expand the scope of research from recorded music by labels to all the amateur video and audio recordings uploaded on the web it will increase the number of files to be processed by one or maybe even two orders of magnitude. As audio signal processing and machine learning techniques become more sophisticated they became more computationally demanding. Important issues of scalability and parallel implementation will continue to arise making techniques that are practical in a few thousand tracks obsolete or practically infeasible in collections of a million music tracks. As collections get bigger an important consideration is music and recording quality. To most existing music mining systems a piece performed by the same combination of instruments by professional musicians and a high school cover band essentially have the same representation.

The way listeners react to music is constantly changing and varies among different age groups and cultures. Even though western popular music has been dominating music listening around the world, there is a lot of interesting work that can be done in applying music mining techniques to analyze other types of music from around the world. Such work falls under the area of computational ethnomusicology. The large diversity of music cultures can lead to specific mining problems that are variations of existing ones or completely new.

To conclude, mining music information is a complex, challenging and fascinating area of data mining. The challenges of scalability, time evolution, heterogeneous structured and unstructured representations, and personalization among others pose significant challenges to existing mining algorithms. Progress in music mining has the potential to lead to significant advances to data mining in general and the converse is also probably true. Finally music mining has already and will probably continue to affect the way music is produced, distributed, and consumed.

---

### 1.26.12 Further reading

In this section a number of pointers to relevant material to this chapter are provided. The list is not comprehensive but is a good representation of activity in the field. Music Information Retrieval is a relative new field with a history of a little bit more than 10 years [1]. Currently there is no comprehensive textbook that covers it fully. However, there are several related books and overview articles that are great starting points for learning more about the field. Orio [2] is an excellent, although somewhat outdated, tutorial and review of the field of MIR. A more recent collection of articles mostly focusing on music data management and knowledge discovery can be found in Shen et al. [3]. Many of the music mining tasks described in this chapter are covered in more detail in a recent collection of articles on music mining edited Li et al. [4]. The chapters on audio feature extraction, mood and emotional classification, web-based and community-based music information extraction, human computation for music classification, and indexing music with tags are particularly relevant.

The basic types of audio feature extraction, i.e., timbre, pitch and rhythm appear in a classic early paper on automatic genre classification [5]. Tempo induction refers to the specific problem of estimating a global tempo for a piece of music but the basic algorithms used are common to any type of rhythm analysis. Gouyon et al. [6] provide an experimental investigation of different tempo induction algorithms. Various types of pitch representations have been investigated especially in the context of chord and key detection [7]. An early seminal work showing how to combine text and audio features for extracting music information is Whitman and Smaragdis [8] and subsequent work has explored the text analysis of lyrics [9], web-based data [10] and microblogs [11].

The strategies and challenges of ground truth acquisition have been explored for music similarity [12], genre classification [13], and tag annotation [14]. An overview of the Music Information Retrieval eXchange (MIREX) for the years 2005–2007 can be found in Downie [15]. An early critical review of content-based music similarity approaches can be found in Aucouturier and Pachet [16]. It has been followed by many publications mostly focusing on how music similarity can be improved with better audio features, integration with text and context features, and design of better similarity metrics [16, 17]. High level overviews of audio fingerprinting systems have been provided by Wang [18] and Haitsma and Kalker [19], and a good review of different algorithms can be found in Cano et al. [20]. Two well

known cover song detection systems are Ellis and Poliner [21] and Serra et al. [22]. A comparative study of different component choices for cover song retrieval systems can be found in Liem and Hanjalic [23].

Classification, clustering, and regression are standard problems in data mining and pattern recognition and therefore well covered in many textbooks [24,25]. There is a large literature in automatic genre classification [5,26,27], in many cases showing that feature sets based on different musical aspects such as harmony and rhythm are all important and their combination gives better results than they do in isolation. Hierarchical classification can be performed by modeling the hierarchy as a Bayesian network [28]. A user study investigating the perception of top-level music genres by average listeners can be found in [29] and a comparison between automatic classification and annotation by users can be found in Lippens et al. [13]. An overview and critical discussion of genre classification can be found in McKay and Fujinaga [30]. Representative examples of emotion spaces that have been used in music mining are the groupings of adjectives used to describe classical music pieces by Hevner [31] as well as the 2D emotional space described by Theyer [32]. Emotion/mood recognition [33] using audio and text features has been covered in [9,34] among others. Regression can also be used for automatic emotion analysis of music [35]. Li et al. propose a clustering algorithm that combines both lyrics and audio features to perform bimodal learning [36].

Work on associating music with text using audio content analysis and machine learning started as early as 2002 [8], not using tags per se, but using keywords extracted from web-pages that ranked highly in search engine results for particular artists. Around 2007–2008, as social tag annotation became more common, some of the first papers that focused on automatic tag annotation for music started appearing in the literature, using different classification approaches and audio feature sets. For example, AdaBoost was used for tag prediction in Eck et al. [37]. A Gaussian Mixture Model over the audio feature space is trained for each word in a vocabulary in the seminal paper on semantic annotation by Turnbull et al. [38]. This work also provided the *CAL500* dataset that since then has frequently been used to evaluate tag annotation systems. Stacking for music tag annotation was originally proposed in [39]. The user evaluation of automatic tag annotation system as well as an machine learning approach that works with an open tag vocabulary has been described by Law et al. [40].

Various approaches to visualization in audio-based music information retrieval are surveyed in Cooper et al. [41]. The use of self-organizing maps in music visualization was popularized by the Islands of Music system that rendered the resulting self-organized map as a set of islands (areas of the grid where many songs were mapped) surrounded by ocean (areas of the grid where fewer songs were mapped) [42]. A number of music mining tasks have been formulated as multiple instance learning by Mandel and Ellis [43]. Symbolic music information retrieval can be done using polyphonic queries and references using a variety of methods based on string and geometric approaches [44]. Structure analysis is typically performed using self-similarity matrices [45]. The most common approach for polyphonic audio score alignment is based on calculating the distance (or similarity) matrix between two sequences typically of chroma vectors and performing dynamic time warping [46,47]. A excellent overview of signal processing methods for music transcription has been edited by Klapuri and Davy [48]. A critical view on hit song science can be found in Pachet and Roy [49]. The problem of identifying similar artists using both lyrics and acoustic data has been explored using a semi-supervised learning approach in which a small set of labeled samples are supplied for the seed labeling and then used to build classifiers that improve themselves using unlabeled data [50].

---

## Glossary

AdaBoost	a family of classification algorithms
Artist Filter	a preprocessing technique used in music mining task that ensures that tracks of the same artist are either part of the training set or the testing set but not both
Audio Fingerprinting	an algorithm for determining whether two digital audio files correspond to the same underlying recording
Beat Histogram (Spectrum)	a representation of the rhythmic content of a piece of music in terms of the amount of energy in different rhythm-related periodicities that are present
Chroma	a representation of the pitch content of a piece of music
Chromagram	a sequence of chroma vectors that can be visualized as an image
Collaborative Filtering	a technique for recommending items based on the purchase history of users
Cosine Similarity	a method of computing similarity between two vectors based on the cosine of the angle between them
Discriminatory Classifiers	a family of classification algorithms in which the goal is to discriminate between the classes directly without modeling the distribution of all the sample feature vectors but only the ones that affect the decision boundary
EM-algorithm	a algorithm for training Gaussian Mixture Models
Emotion Space	a multi-dimensional representation that maps different human emotions and their relations
F-Measure	a performance measure for retrieval algorithms
Games with a Purpose (GWAP)	computer games that in addition to being engaging and entertaining provide useful information for training computers
Gaussian Models	a parametric family of probability density functions
Gaussian Mixture Model	a method for modeling a probability density function as a combination of Gaussian components
Generative Classifiers	a family of classification algorithms in which the training data is modeled by estimating probability density functions that can generate new sample feature vectors
Genre	a categorical label that characterizes a particular type of music
Ground Truth	the information needed for training and evaluating a mining algorithm
Harmonic Pitch Class Profiles	a representation of pitch content for audio signals similar in nature to Chroma
Harmony	the structure of piece of music in terms of combinations of discrete notes or pitches

Hidden Markov Models (HMM)	a probabilistic techniques for modeling sequence of feature vectors
ISMIR	the annual International conference of the Society for Music Information Retrieval
K-Fold Cross Validation	a methodology for evaluating classification performance
K-Means	a standard clustering algorithm
Last.fm	an internet music recommendation service
Mean Average Precision:	a performance measure for retrieval algorithms across multiple queries
Mel-Frequency Cepstral Coefficients (MFCC)	a set of audio features frequently used in MIR that originated in speech processing
Musical Instrument Digital Interface (MIDI)	a symbolic format for transmitting and storing music related information
Music Information Retrieval (MIR)	the research area dealing with all aspects of retrieving information from music signals that are stored digitally
Music Information Retrieval Evaluation Exchange (MIREX)	an annual campaign for evaluating different algorithms on a variety of MIR tasks
Music XML	a symbolic format for representation all aspects of notated music
Naive Bayes	a family of parametric classification algorithms
Nearest Neighbor Classifier	a family of non-parametric classification algorithms
Pandora	an internet personalized radio station
Pitch Class Profiles	a representation of the pitch content of a piece of music
Precision	a performance measure for retrieval algorithms
Principle Component Analysis	a technique for dimensionality reduction
Query-by-Humming (QBH)	the task of matching a sung melody to a database of music tracks in either symbolic, or audio format
Recall	a performance measure for retrieval algorithms
Receiver Opeating Characteristic (ROC)	a graphical representation used to characterize retrieval systems
Rhythm	the structure of a piece of music over time
Self-Organizing Maps (SOM)	a dimensionality reduction and clustering techniques that maps high dimensional feature vectors into a discrete set of locations
Short Time Fourier Transfrom (STFT)	a transformation of a time varying signal to a time-frequency representation
Similarity Retrieval	a music mining task where a query music track is provided by the user and a set of tracks that are similar to it are returned by the system
Spectrum	a representation of a signal in terms of the amounts of different frequencies
Stacking	a methodology for combining binary classifiers for multi-label classification
Support Vector Machines	a family of discriminative classification algorithms

Tags	words from an open vocabulary supplied by users to characterize multimedia objects such as music tracks, images, and videos
Term Weighting	a technique from text mining in which different words get assigned different weights when considering the similarity of documents
Timbre	the properties that characterize a particular sound source when compared with other sound sources of the same pitch and loudness

---

## Acknowledgments

The author would like to thank Tiago Fernandes Tavares and Steven Ness for help with preparing the figures for this chapter. Moreover, the author would like to thank the many researchers from a variety of disciplines who have helped music information retrieval and, more specifically, music mining grow and evolve. Their efforts have made the work described in this chapter possible and the author is proud to have been a part of this amazing community from its inception.

*Relevant Theory:* Signal Processing Theory and Machine Learning

See this Volume, [Chapter 9](#) Discrete Transforms

See this Volume, [Chapter 16](#) Kernel Methods and SVMs

See this Volume, [Chapter 20](#) Clustering

---

## References

- [1] J. Downie, D. Byrd, T. Crawford, Ten years of ISMIR: Reflections on challenges and opportunities, in: Proceedings of the 10th International Society for Music Information Retrieval Conference, 2009, pp. 13–18.
- [2] N. Orio, Music retrieval: A tutorial and review, vol. 1. Now Pub, 2006.
- [3] J. Shen, J. Shepherd, B. Cui, L. Liu, Intelligent music information systems: Tools and methodologies, Information Science Reference, 2008.
- [4] T. Li, M. Ogihara, G. Tzanetakis (Eds.), *Music Data Mining*, CRC Press, 2012.
- [5] G. Tzanetakis, P. Cook, Musical genre classification of audio signals, *IEEE Trans. Audio Speech Lang. Process.* 10 (5) (2002) 293–302.
- [6] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, P. Cano, An experimental comparison of audio tempo induction algorithms, *IEEE Trans. Audio Speech Lang. Process.* 14 (5) (2006) 1832–1844.
- [7] E. Gómez, Tonal description of polyphonic audio for music content processing, *INFORMS J. Comput.* 18 (3) (2006) 294–304.
- [8] B. Whitman, P. Smaragdis, Combining musical and cultural features for intelligent style detection, in: Proc. Int. Symposium on Music Inform. Retriev.(ISMIR), 2002, pp. 47–52.
- [9] X. Hu, J. Downie, A. Ehmann, Lyric text mining in music mood classification, *American music* 183 (5049) (2009) 2–209.
- [10] P. Knees, E. Pampalk, G. Widmer, Artist classification with web-based data, in: Proceedings of the International Conference on Music Information Retrieval, 2004, pp. 517–24.
- [11] M. Schedl, On the use of microblogging posts for similarity estimation and artist labeling, in: Proceedings of the International Conference on Music Information Retrieval, 2010.
- [12] D. Ellis, B. Whitman, A. Berenzweig, S. Lawrence, The quest for ground truth in musical artist similarity, in: Proc. ISMIR, vol. 2, 2002, pp. 170–177.

- [13] S. Lippens, J. Martens, T. De Mulder, A comparison of human and automatic musical genre classification, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 2004 (ICASSP'04), vol. 4, IEEE, 2004, pp. iv–233.
- [14] D. Turnbull, L. Barrington, G. Lanckriet, Five approaches to collecting tags for music, in: Proceedings of the 9th International Conference on Music Information Retrieval, 2008, pp. 225–30.
- [15] J. Downie, The music information retrieval evaluation exchange (2005–2007): a window into music information retrieval research, *Acoust. Sci. Technol.* 29 (4) (2008) 247–255.
- [16] J. Aucouturier, F. Pachet, Music similarity measures: what's the use, in: Proceedings of the ISMIR, Citeseer, 2002, pp. 157–163.
- [17] T. Li, M. Ogihara, Content-based music similarity search and emotion detection, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004 (ICASSP'04), vol. 5, IEEE, 2004, pp. V–705.
- [18] A. Wang, The shazam music recognition service, *Commun. ACM* 49 (8) (2006) 44–48.
- [19] J. Haitsma, T. Kalker, A highly robust audio fingerprinting system, in: Proc. ISMIR, vol. 2002, 2002, pp. 144–148.
- [20] P. Cano, E. Batle, T. Kalker, J. Haitsma, A review of algorithms for audio fingerprinting, in: IEEE Workshop on Multimedia Signal Processing, IEEE 2002, pp. 169–173.
- [21] D. Ellis G. Poliner, Identifying cover songs with chroma features and dynamic programming beat tracking, in: IEEE International Conference on Acoustics, Speech and Signal Processing 2007, ICASSP 2007, vol. 4, IEEE, 2007, pp. IV–1429.
- [22] J. Serra, E. Gómez, P. Herrera, X. Serra, Chroma binary similarity and local alignment applied to cover song identification, *IEEE Trans. Audio Speech Lang. Process.* 16 (6) (2008) 1138–1151.
- [23] C. Liem, A. Hanjalic, Cover song retrieval: A comparative study of system component choices, in: Proc. Int. Conf. on Music Information Retrieval (ISMIR), 2009.
- [24] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Fourth (ed.) Academic Press, 2008.
- [25] P. Tan, M. Steinbach, V. Kumar, et al. *Introduction to Data Mining*, Pearson Addison Wesley Boston, 2006.
- [26] T. Li, M. Ogihara, Q. Li, A comparative study on content-based music genre classification, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2003, pp. 282–289.
- [27] A. Meng, P. Ahrendt, J. Larsen, Improving music genre classification by short time feature integration, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 2005 (ICASSP'05), vol. 5, IEEE, 2005, pp. v–497.
- [28] C. DeCoro, Z. Barutcuoglu, R. Fiebrink, Bayesian aggregation for hierarchical genre classification, in: Proceedings of the International Conference on Music Information Retrieval, 2007, pp. 77–80.
- [29] R. Gjerdingen, D. Perrott, Scanning the dial: the rapid recognition of music genres, *J. New Music Res.* 37 (2) (2008) 93–100.
- [30] C. McKay, I. Fujinaga, Musical genre classification: is it worth pursuing and how can it be improved, in: Proceedings of the Seventh International Conference on Music Information Retrieval, 2006, pp. 101–106.
- [31] K. Hevner, Experimental studies of the elements of expression in music, *Am. J. Psychol.* 48 (2) (1936) 246–268.
- [32] R. Thayer, *The Biophysiology of Mood and Arousal*. Oxford University Press, New York, 1989.
- [33] Y. Kim, E. Schmidt, R. Migneco, B. Morton, P. Richardson, J. Scott, J. Speck, D. Turnbull, Music emotion recognition: a state of the art review, in: Proc. 11th Int. Symp. Music Information Retrieval, 2010, pp. 255–266.
- [34] E. Schmidt, Y. Kim, Prediction of time-varying musical mood distributions using kalman filtering, in: Ninth International Conference on Machine Learning and Applications, 2010 (ICMLA), IEEE, 2010, pp. 655–660.

- [35] Y. Yang, Y. Lin, Y. Su, H. Chen, A regression approach to music emotion recognition, *IEEE Trans. Audio Speech Lang. Process.* 16 (2) (2008) 448–457.
- [36] T. Li, M. Ogihara, S. Zhu, Integrating features from different sources for music information retrieval, in: Sixth International Conference on Data Mining 2006, ICDM'06, IEEE, 2006, pp. 372–381.
- [37] D. Eck, P. Lamere, T. Bertin-Mahieux, S. Green, Automatic generation of social tags for music recommendation, in: Advances in Neural Information Processing Systems, vol. 20, 2007.
- [38] D. Turnbull, L. Barrington, D. Torres, G. Lanckriet, Semantic annotation and retrieval of music and sound effects, *IEEE Trans. Audio Speech Lang. Process.* 16 (2) (2008) 467–476.
- [39] S.R. Ness, A. Theocharis, G. Tzanetakis, L.G. Martins, Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs, in: Proc. ACM Multimedia, 2009.
- [40] E. Law, B. Settles, T. Mitchell, Learning to tag from open vocabulary labels, in: Principles of Data Mining and Knowledge Discovery, 2010, pp. 211–226.
- [41] M. Cooper, J. Foote, E. Pampalk, G. Tzanetakis, Visualization in audio-based music information retrieval, *Comput. Music J.* 30 (2) (2006) 42–62.
- [42] E. Pampalk, S. Dixon, G. Widmer, Exploring music collections by browsing different views, *Comput. Music J.* 28 (2) (2004) 49–62, (Summer 2004).
- [43] M. Mandel, D. Ellis, Multiple-instance learning for music information retrieval, in: Proc. ISMIR, 2008, pp. 577–582.
- [44] K. Lemström, A. Pienimäki, On comparing edit distance and geometric frameworks in content-based retrieval of symbolically encoded polyphonic music, *Musicae Scientiae* 11(Suppl. 1) (2007) 135.
- [45] R. Dannenberg, Listening to NAIMA: an automated structural analysis of music from recorded audio, in: International Computer Music Conf. (ICMC), 2002.
- [46] G.T.N. Hu, R.B. Dannenberg, Polyphonic audio matching and alignment for music retrieval, in: Workshop on Applications of Signal Processing to Audio and Acoustics, 2003.
- [47] M. Müller, *Information Retrieval for Music and Motion*, Springer-Verlag, New York, Inc., 2007.
- [48] A. Klapuri, M. Davy (Eds.), *Signal Processing Methods for Music Transcription*, Spring, 2006.
- [49] F. Pachet and P. Roy, Hit song science is not yet a science, in: Proc. Int. Conf. on Music Information Retrieval (ISMIR), 2008, pp. 355–360.
- [50] T. Li, M. Ogihara, Music artist style identification by semi-supervised learning from both lyrics and content, in: Proc. ACM Int. Conf. on Multimedia, 2004, pp. 364–367.

# Index

## A

- AdaBoost, 1472, 1477–1478, 1487
- Adaptive CoSaMP algorithm (AdCoSaMP), 1335
- Adaptive decomposition design
  - block size, transform, 459
  - coding improvements, 457
  - pre-/post-processing support, 458
- Adaptive filters, 604–606
  - acoustic echo in hands-free telephony, 619–620
  - algorithms, 619
  - blind equalization, 734
  - constraints, 746
  - convex sets projection, 745
  - cooperative estimation, 738
  - data model, 692
  - distributed region, 743
  - echo path, 624
  - echo-cancellation, 624, 630
  - finite-precision arithmetic, 718
  - IIR filters, 744
  - inputs and outputs, 620
  - orthogonality condition, 626
  - reduced-rank, 746
  - regularization, 729
  - set-membership algorithm, 745
  - solution, 627
  - statistical analysis, 689
  - structure, 626
  - subband domain, 737
  - theory, 620
  - transform-domain, 737
- Adaptive projected subgradient method (APSM)
  - formula, 959
  - kernel algorithm, 960
  - projection base, 924–925
- Adaptive Resonance Theory (ART), 1123–1124
  - vigilance parameter, 1124–1125
- Adatron algorithm, 834
- Algorithm *see specific entries*
- AMUSE algorithm, 1181
- Analog approximation problem, digital filters
  - frequency transformations, 266
  - standard lowpass, 263
  - typical requirements, 262
- Analog reconstruction filter, D/A converter, using, 187
- Analog signal acquisition systems,
  - continuous-time system, 3–4

- Analog signal processing (ASP)
  - in information processing, 169
  - practical applications, 169
  - quantization schemes, 169–170
- Analog signals, defined, 29
- Analog to Digital Converter (ADC)
  - converter performance, 215
  - frequency response, 215
  - oversampling, 227, 230
  - quantization errors, 219
  - round-off errors, 222, 225
- Approximation approach, image/video
  - processing applications
  - complex-coefficient transforms, 444
  - direct scaling, 433
  - integer DCT design, 441
  - lifting step, 437, 439
  - lossless color transform design, 439
  - structural design, 436
- Arithmetic operations, digital filters
  - addition, 321, 324
  - bit-serial addition and subtraction, 322
  - multiplication, 322
  - ripple-carry addition, 322
  - subtraction, 321
- Audio feature extraction, 1456
- Autoregressive (AR) models
  - input-output relationship, 598–599
  - parameters, 598–599
  - process, 595
- Axiomatic definition, 76

## B

- Back-propagation algorithm
  - defined, 822
  - numerical techniques, 822
  - recurrent networks, 828
  - training error minimization, 823
- Balian-Low theorem, 564
- Bandpass sampling, 195
  - random process, 165
- Batch learning, algorithmic ingredients, 953
- Bayesian information criterion (BIC)
  - in model selection, 1426
  - linear models, example, 1429
  - prior distribution effects, 1429
  - properties, 1427

- Bayesian network (BNs)
  - conditional independence properties, 1003
  - defined, 1001
  - discriminative learning, 1032
  - for expert systems, 1053
  - factorization properties, 1001
  - generative learning, 1018
  - maximum likelihood, 1019
  - parameter learning, 1022
  - speech processing, 1046
  - structure, 1028
  - time-series modeling, 1046
- Biclustering
  - algorithms, 1133
  - in bioinformatics community, 1128
  - relational data sets, 1129
- Blind source extraction, 1185
  - multi-block tensor factorizations, 1208
  - temporal structures, 1186
- Block matrices, inverses of, 639
- Boltzmann machine, 826–827
- Bounded-input bounded-output (BIBO)
  - in LTI system, 58
  - system stability, 58–59
  
- C**
- Canonical Correlation Analysis (CCA), 1226
  - fundamentals, 858
  - two set variables, 866
  - sparsification, 1175
- Cascade correlation, 831–833
- Case study, time-frequency analysis, machine learning, 1350
- Chomsky hierarchy, 829
- Classification task
  - linear mapping, 893–894
  - parameter estimation, 887
  - purpose, 888–889
  - in RKHS, 895
  - training points, 889–890
- Clustering. *See also* Clustering algorithms;
  - Clustering applications
  - Cauchy-Schwarz objective function, 1401
  - music mining, 1473
- Clustering algorithms
  - deep learning, 1135
  - hierarchical clustering (HC), 1118
  - mixture density-based, 1121
  - neural network-based, 1123
  - open issues and problems, 1140
  - spectral, 1126
- subspace, 1127
- validation, 1139
- Clustering applications
  - motion clustering, 1140
  - MRI images, 1139
  - vision-guided robot, 1140
- Cognitive radio
  - digital signal processing (DSP), 339
  - frequencies, 400–401
  - receiver devices, 402
- Commensurate-length transmission, 279
  - Richards' variable, 280
  - unit elements, 281
- Common component analysis (CCA)
  - multi-block tensor data, 1219
  - power method, 1226
- Common formulation, 646
  - channel equalization, 646–647
  - useful tasks, 646–647
- Complex variables and multi-channel
  - filtering, 661
  - optimum error, 662
  - optimum filters, 662
- Computational learning theory
  - agnostic PAC model, 800
  - halving algorithm, 802
  - mistake bound model, 801
  - noise models, 800
  - PAC model, 798
  - perceptron (online algorithm), 805
  - weighted majority, 803
  - winnow (online algorithm), 805
- Computational properties, digital filter
  - cyclic scheduling, 319
  - latency and throughput, 318
  - maximal sample rate, 318–319
- Continuous-time signals
  - defined, 30
  - Fourier transform, uses, 73–74
  - sampling, 173–174, 181
- Continuous-time system
  - analog signal acquisition systems, 3–4
  - input signal, 4
  - linear and time-invariant (LTI), 4, 31
  - output signal, 4
- Continuous Wavelet transform, 578–579
- Contourlet transform, applications, 515
- Convex optimization, 828
- Convolution
  - for causal system, 45
  - frequency-domain, 72
  - graphic interpretation, 36
  - in LTI system, 55

- C**
- Convolution (*Continued*)
    - in time-domain system, 45–48
    - input and output signal, 34
    - mathematical properties, 36, 74
    - transfer function, 55–56
    - zero-state solution, 35, 56
  - CP model
    - additional constraints, 1214
    - data tensor, 1212–1213
    - illustration, 1213
    - multilinear terms, 1213
    - power method, 1214–1215
    - restrictive trilinear, 1225
    - special form, 1219
    - Tucker model, advantages, 1213–1214
  - Cross-validation, choosing architecture, 823, 828
  - Cumulative probability distribution function (CDF)
    - defined, 120–121
    - probability density function, 122
  - Curvelet transform
    - advantages, 534
    - amplitude and angular windows, 500
    - applications, 497, 514
    - coefficient decay rate, 504
    - continuous, 499, 501
    - discrete, 506, 512
    - localized functions, 533
    - properties, 503
    - reconstruction formula, 505
    - reproducing formula, 505
    - reverse operations, 512
    - USFFT implementation, 512–513
- D**
- Deconvolution, 645, *see also* Inverse system identification
  - Deep learning architecture, 818, 824
    - semi-supervised clustering, 1135
  - Deterministic signals
    - Poisson's formula (summation), 179, 181
    - uniform sampling, 178
  - Digital down converter (DDC)
    - architecture, 403
    - band pass filter, 356, 359
    - building blocks, 356
    - frequency shifting, 346
    - resampling operations, 341–342
  - Digital filter
    - ADC output, 399–400
    - bandwidth reduction, 362–364
    - complexity, 360
    - defined, 366
    - design techniques, 343
    - domain characteristics, 360
    - multirate processing, 341
    - non-recursive filters, 351
    - resampling theory, 339, 363–364
    - sampling clock, 352
    - signal processing, 349–350
    - system memories, 350–351
    - task performance, 351–352
  - Digital signal processing (DSP)
    - of analog signals, 175–177
    - communication system, 178
    - in information processing, 169
    - practical application, 169
    - quantization schemes, 169–170
  - Digital up converter (DUP)
    - center frequency, 378
    - interpolating filters, 400
    - simulation results, 415
  - Direct scaling approach, image/video
    - processing applications
    - integer DCT design, 435
    - rotation angle, 436
  - Discrete spaces, frames in, 571
  - Discrete-time signals
    - frequency responses, 238
    - input/output relations, 234
    - mixed systems, 232
    - Poisson's summation formula, 235
    - second-order continuous-time-input, 238, 240
    - sequences, 172–173
    - stability factor, 237
  - Discrete-time systems
    - classification, 81
    - defined, 79
    - in Linear time-invariant continuous-time systems, 83
    - infinite word length, 79
    - input-stability criterion, 97
    - input/output sequence, 81
    - MATLAB application, 101
    - multi-rate interval, 79
    - output-stability criterion, 96
    - stability in, 96, 98
  - Doubly resistively terminated lossless networks, 266
    - digital filters, 266, 275
    - design, 273
    - element sensitivity, 268
    - error elements, 269
    - lossy elements, 270
    - maximal power transfer, 266
    - passband sensitivity, 268

Doubly resistively terminated lossless networks (*Continued*)  
 reflection function, 267  
 terminating errors, 269

**E**

Energy conservation method, steady-state analysis, 707  
 Energy signals, distortion measurement, 192, 194  
 Erasures, 570  
 Euclidean divergence, 1397–1398  
 Excess mean-square error (EMSE), 631  
   adaptation parameters, 713  
   step-size, 713  
   traditional statistical method, 701  
 Extreme learning machine, 824, 836

**F**

Factor graphs (FGs)  
 BN conversion, 1013  
 defined, 1011  
 error-correcting codes, 1055  
 Feed-forward network, 847  
 Filter banks  
   cascade sequence, 523  
   DF, 529  
   directional, 523, 525, 531  
   Laplacian pyramid, 520, 529  
   quincunx, 524  
 Finite vector spaces  
   analysis operators, 572  
   Gram matrix, 573  
   matrices, 572  
   synthesis operators, 572  
 Fisher discriminant analysis  
   classification function, 868  
   fundamentals, 858  
 Fourier transform  
   application, 472  
   continuous, 471  
   definition, 471–472  
   discrete, 472, 497  
   energy preservation, 471  
   input function, 472, 497  
   properties, 471, 481  
   pseudo-polar coordinates, 545  
   temporal and/or spatial data sequences, 467  
   windowed, 473  
 Frame bounds ratio, 569  
 Frame characterization problem, 566  
 Frame coefficients, 562

Frame expansion, 564  
 Frame of translates, 574–575  
 Frames  
   common approaches, 574  
   construction, 574  
   decomposition, 567  
   dilation, 574  
   inverse operator, 567  
   modulation, 574  
   operator, the, 566  
   reconstruction, 567  
   translation, 574  
 Frequency response  
   defined, 60  
   magnitude plot, 61–64  
   RLC series circuit, 61  
   symmetry property, 62  
   voltage application, 60  
 Frequency response masking (FRM) structure,  
   digital filter  
   basics, 307  
   cosine-modulated transmultiplexer, 315  
   filter bank, 314  
   half-band filter, 311  
   hilbert transformer, 311  
   multirate filter, 312  
   multistage, 310  
   recursive filter, 316  
   two dimension, 316  
 Fuzzy c-means (FCM), 1120  
 Fuzzy clustering, 1115, 1121

**G**

Gabor frames, 574, 576  
   fast analysis and synthesis operators, 583  
   Heisenberg boxes, 581  
   in discrete spaces, 583  
 Gabor system  
   elements location, 577  
   windowed Fourier frame, *see* Weyl-Heisenberg or Gabor frame  
 Gaborgram, 581  
 Gain vector, 608  
 Games with a purpose, 1454–1455, 1474  
 Gaussian mixture model, 1123, 1131  
   for audio feature, 1487  
 EM algorithm, 1474  
   music mining classification, 1468  
   tag annotation, 1477–1478  
   weighted combination, 1468–1469

Gaussian process  
 applications, 877  
 Bayesian learning approach, 872–873  
 covariance function, 873–874  
 defined, 874  
 fundamentals, 858  
 regression approach, 872–873  
 variational approximation, 877  
 Gaussian vectors, fourth-order moments, 715  
 General modular construction, image/video processing applications, lifting scheme, 451  
 Generative learning  
 of Bayesian networks, 1018  
 principles, 1016  
 Gradients, 637–638  
 Gradient descent  
 adaptive matrix, 839  
 neural network process, 833  
 pseudocode, 821–822  
 recurrent network, 827  
 stochastic decomposition, 822, 837  
 training error minimization, 823  
 Graph theory, 996

## H

HALS algorithm, 1198  
 Hard clustering, 1115, 1120  
 Hebbian learning, 817–820, 826, 837  
 Hierarchical clustering  
 arbitrary shapes, 1120  
 controlling process, 1143  
 coupled two-way clustering (CTWC) algorithm, 1134–1135  
 defined, 1115  
 example, 1118  
 image segmentation, 1139  
 open issues, 1140–1141  
 properties, 1117–1118  
 software implementation, 1123  
 squared error criterion, 1120  
 structure, 1118  
 Higher-order approach, image/video processing applications  
 blocking problems, 449–450  
 butterflies, 436–437, 441, 455–456  
 coding efficiency, 454  
 HD photo or JPEG-XR transform, 457  
 low-order building blocks, 436–437, 451–452  
 optimization, design, 449  
 pre-/post-processing operators, 452  
 scaling factors, 436–439, 457

Hilbert spaces, Fréchet's differentiation, 915  
 Hodgkin-Huxley model, 819  
 Hopfield network, 826

## I

Image/video processing applications  
 approximation approach, 433  
 attenuation at mirror frequencies, 433  
 design strategy, 431  
 direct scaling approach, 435–436  
 energy compaction, 432  
 $4 \times 4$  transformer design, 434  
 orthogonality, 432  
 stop-band attenuation, 432–433  
 zero DC leakage, 433  
 Implicit vs. physical models, deconvolution, 652  
 Importance sampling (IS), 1093  
 Impulse response  
 cascade system, 36  
 in causal system, 37, 59  
 convolution integral, 34  
 defined, 33  
 frequency response, linear system, 60  
 in Fourier transform, 76  
 in Laplace transform, 55–56  
 in real positive poles, 59, 62  
 linear time-invariant system, 32  
 stimuli estimation, 34, 39  
 voltage application, 36  
 Impulse signal, 31  
 in narrow pulse, 30  
 Input and output signals, 34  
 Independent Component Analysis (ICA), 841  
 approaches, 1180  
 higher order statistics, 1183  
 multi-block data, 1220–1222  
 MRMI algorithm, 1407  
 Inference  
 approximate, 1044  
 joint probability distribution, 1037  
 types, 1036  
 Information based learning  
 areas of application, 1379  
 nonparametric estimators, 1388  
 probability density functions (PDFs), 1380  
 reproducing kernel Hilbert space (RKHSs), 1391  
 theoretical descriptions, 1381

Information based learning (*Continued*)

- Input and output signal values,  
quantization, 216

## Interconnection networks, digital filter, 288

- direct interconnection of adaptors, 296
- parallel adaptors, 294
- series adaptors, 291
- symmetric two-port adaptor, 289
- three-port parallel adaptor, 296
- three-port series adaptor, 292
- two-port parallel adaptor, 295
- two-port series adaptor, 292

## Interference cancellation, 642

- applications of, 642

## Inverse frames, 568

## Inverse gabor frames, 582

- prototype function, 583

## Inverse system identification, 645

- in communications, 645

- simplified baseband communications  
system, 645

**J**Joint cumulative probability distribution  
function, 132

## Joint model, 116

## Joint probability density function, 132–133

**K**

## Kernel methods

- applications, 877
- Bayesian learning approach, 872
- canonical correlation analysis (CCA), 866
- computational issues, 875
- Fisher discriminant analysis, 868
- foundations, 858
- Gaussian process, 872
- multiple learning, 876
- open issues, 878
- principal component analysis, 865
- properties, 860
- support vector machines (SVMs), 869
- types, 862

## Kernel trick

- in machine learning, 858
- optimization problems, 872
- properties, 858
- ridge regression, 859

## Kernel, neural network, 835

- arbitrary, 839

## nonlinear components, 842

- structure, 836, 847–848
- trick, 836

**L**

## Ladder structures, digital filters

- design, 275
- lowpass filters, 273
- waves, 303

## Lattice structures, digital filters, 275

- analog structure, 276
- design, 278
- reactances realization, 277
- wave description of two-ports, 276

## Learning vector quantization, 825, 837, 847

## Least mean squares (LMS)

- algorithm, 629–631, 675, 678
- convex functions, 670
- convexity, 669
- counter-example, 667
- finite-precision effects, 718
- joint probability density function, 666
- linear functions, 916
- particle filters, 669
- recursive function, 959
- statistical analysis, 696
- under non-sufficiently rich input, 660
- variable step-size, 730

## Levinson-Durbin recursion, 601–603

## Linear complex least-mean squares, 663

- complex gradient, 665
- optimum error power, 665
- orthogonality condition, 665
- real or complex circular adaptive  
filters, 664

## Linear least-mean squares estimation

- linearly parametrized classes, 649, 676
- Wiener solution, 650

## Linear and time-invariant (LTI) system

- amplitude scaling, 55
- continuous-time systems, 31
- defined, 31–32
- discrete-time systems, 83
- eigensignals, 55, 58, 72
- final-value theorem, 48
- frequency-domain, 72
- impulse response, 59
- Initial-value theorem, 48
- pole location, 58
- properties, 31–32
- signal analysis function, 67

Linear and time-invariant system (*Continued*)  
 stability function, 58  
 step response, 33  
 transfer function, 55–56

**M**

Machine learning  
 adaptive CoSaMP algorithm (AdCoSaMP), 1335  
 algorithm, computational demand, 1318  
 classical regularization schemes, 765–767  
 classification (ITL framework), 1385  
 coherent dictionaries, 1347  
 compressed sensing, 1300  
 compressed sensing matching pursuit (CSMP)  
   algorithms, 1310  
 computational aspects, 769  
 cosparcity, 1348  
 dimensionality reduction technique, 1303  
 feature selection, 1385  
 filtering, 1385  
 frequency resolution, 1352–1353  
 Gabor frames, 1353  
 Gabor transform, 1350  
 geometric interpretation, 1289  
 greedy algorithms, 1306  
 iterative shrinkage algorithms (ISTA), 1311  
 $\ell_0$  norm minimizer, 1286, 1291  
 $\ell_1$  norm minimizer, 1287–1288  
 least absolute shrinkage and selection operator  
   (LASSO), 1276, 1332–1333  
 least angle regression (LARS) algorithm, 1310  
 linear parametric models, application, 765  
 MSE learning curves, 1341  
 mutual coherence, 1293, 1295  
 noisy measurements, robust signal, 1299  
 nonparametric approaches, 768  
 norms, functional analysis, 1274  
 OMP solution, 1308  
 online (time-recursive) schemes, 1331  
 optimization parameters, 1272, 1342  
 parameter estimation, 1272, 1275  
 promoting algorithms (sparsity), 1306  
 properties, signal representation, 1343–1344  
 restricted isometry property (RIP), 1296  
 signal representation, sparse, 1280  
 solution, sparsity constraint, 1284–1285  
 sparse adaptive parallel projection onto convex  
   sets method (SpAPSM), 1336  
 sparse vectors, 1357  
 sparsity-awareness, variations, 1325  
 sub-Nyquist sampling, 1304

switch in position 1, 1286, 1293  
 switch in position 2, 1289, 1293  
 switch in position 3, 1291  
 time-frequency analysis, case study, 1350  
 $\ell_2$  norm minimizer, 1285  
 types, 1271  
 weighted 1 ball, 1339

Machine learning, theoretical perspectives, 775.  
*see also* Computational learning theory  
 binary function, VC dimension, 787  
 classification and regression, 778–779  
 concentration inequalities, 783  
 data driven penalties, 792  
 empirical means, 782  
 entropy logarithm, 787  
 ERM convergence, 781  
 fixed function class, 779  
 hold-out estimation, 793  
 loss function, 777  
 PAC-Bayesian analysis, 796  
 penalty function, 791  
 probabilistic formulations, 776  
 rademacher complexity, 784  
 real valued functions, 789  
 risk function, 777  
 sample compression, 793  
 stability properties, 795  
 structural risk minimization (SRM), 790  
 Markov Chain Monte Carlo (MCMC) methods, 1077  
 acceptance probabilities, 1107  
 finite case, 1078  
 Markov network (MNs)  
   conditional independence, 1007  
   defined, 989  
   factorization properties, 1007  
 Markov random fields (MRFs)  
   error correcting codes, 1046  
   for image analysis, 1053  
 Matching pursuit. *see* Machine learning,  
   greedy algorithm  
 MATLAB application, discrete-time systems  
   space state function, 105  
   transfer function, 105  
 Matrix analysis, 639  
 Matrix inversion Lemma, 640  
 Mean squared error, 823, 827–828  
 Metropolis Hastings (MH) algorithm  
   convergence rate, 1083  
 Gibbs sampler, 1088  
 illustration, 1083  
 proposal density, 1081–1088, 1103–1104  
 symmetric proposal, 1084–1088  
 target density, 1088

- Minimum description length (MDL) principle, 1442  
 application, 1446  
 Bayesian inference, 1445  
 linear model, example, 1446  
 sequential variants, 1444
- Minimum distance estimation, 1418, 1424
- Minimum error entropy (MEE)  
 back propagation, 1396  
 ITL criterions, 1385
- Minimum message length (MML)  
 applications, 1442  
 Bayesian inference, 1440  
 invariance, 1439  
 linear models, example, 1441  
 model selection, 1434  
 other approximations, 1440  
 parameter estimation, 1439  
 selection consistency, 1439  
 Wallace-Freeman approximation, 1437
- Model selection  
 Akaike information criterion, 1418  
 Bayesian approaches, 1424  
 by compression, 1431  
 distance based criteria, 1431  
 linear model, example, 1416, 1423  
 Markov-Chain Monte-Carlo Bayesian methods, 1430  
 Minimum description length (MDL) principle, 1442  
 minimum distance estimation, 1418  
 minimum message length (MML), 1434  
 nested sets, 1416  
 nonnested sets, 1416, 1424  
 Schwarz Information Criterion, 1426  
 simulation criteria, 1447  
 Wallace-Freeman message length approximation (MML87), 1437
- Monte Carlo (MC) method  
 advanced methods, 1101  
 application, 1065  
 finite state case, 1078  
 illustration, 1065–1066  
 open issues and problems, 1111  
 principle, 1066  
 random sampling, 1065  
 sequential importance, 1092, 1096  
 stationary distribution, 1078  
 target distribution, 1080
- Motivation-acoustic echo cancellation, 620  
 echo signal estimation, 623  
 general echo canceller configuration, 621  
 measured impulse response, 621–622
- Multilayer perceptrons (MLPs)  
 supervised learning, 1394
- Multilinear BSS, Tucker decompositions, 1216
- Multimedia coding and processing  
 filter bank connection, 428  
 lapped transform connection, 430  
 notation, 425  
 optimal orthogonal transform, 427  
 transform fundamentals, 426
- Multiple kernel learning  
 recent developments, 858, 878  
 sequential minimal optimization (SMO)  
 algorithm, 877
- Multirate filter  
 application, 352–353, 363  
 defined, 341, 349  
 sampling clock, 351–352  
 single processing architecture, 341–342
- Multiresolution analysis  
 coding length criterion, 490  
 decomposition procedure, 521–522  
 denoising and modeling, 489  
 estimation problems, 488  
 LP filter bank technique, 516, 523, 529  
 MRA theory, 479  
 noise model, problem statement, 489  
 numerical experiments, 492  
 properties, 480  
 tight frame analysis, 522  
 wavelets, 479  
 worst case noise, 491
- Music mining  
 advanced techniques, 1480  
 algorithm designing, 1453  
 audio feature extraction, 1456  
 audio fingerprinting, 1464  
 automatic transcription, 1482  
 classification, 1468  
 clustering, 1473  
 context extraction, 1460  
 cover song detection, 1464, 1467  
 datasets, 1483  
 Gaussian models, 1468–1469  
 Genre classification, 1469  
 ground truth acquisition, 1454  
 mirex task, 1473  
 mood, 1470  
 open problems, 1484  
 organizing emotion, 1470–1471  
 performance evaluation, 1472  
 sequence matching, 1466  
 similarity retrieval, 1460, 1463  
 software, 1483  
 symbolic representation, 1481  
 tag annotation, 1474  
 visualization, 1478

**N**

- Neural gas, 845–846
- Nonlinear autoregressive models, 612–613
- Non-negative Matrix Factorization (NMF)
  - alpha-beta divergences, 1202–1203, 1205
  - basic models, 1190
  - convolutive, 1206
  - large-scale problems, 1200
  - non-smooth case, 1193
  - parameter estimation, 1195
  - robust algorithms, 1202
  - sparsity constraints, 1190
  - special forms, 1191
- Non-parametric signal models, 595
- Nonuniform sampling
  - DC offset, 210
  - error metrics, 208
  - generalizations, 204
  - problem formulations, 205
  - reconstruction, 207
  - relaxed problems, 206
  - time-interleaved ADCs, 204
  - time-varying FIR system, 207
- Normalized least-mean-square (NLMS)
  - algorithm, 678–679
  - affine projections algorithm (APA), 737
  - matrix inversion lemma, 680
  - proportionate, 731
  - unified statistical analysis, 701
- Normalized tight frames, 570
- Norms and spectral radius, 641
- NP-hard problem, 823

**O**

- Online learning. *see also* Reproducing kernel Hilbert spaces (RKHSs)
  - application, 970
  - convex analytic toolbox, 941, 953, 958
- Optimal mean-square error, 652
- Optimum filtering, 647
  - error power, 647
  - goals, 648
  - quadratic cost functions, 648
- Ordinary differential equation (ODE)
  - constant coefficients, example, 39
  - differentiation property, 75–76
  - for input voltage, 40
  - Laplace transform application, 54
  - pole-zero plot or pole-zero diagram, 57
- Orthogonality condition, 651
  - cross-correlation vector evaluation, 653

optimum error, 655

- Overfitting
  - machine learning task, 890
  - regularization method., 891–892, 895
  - in RLS algorithm, 929
- Oversampling
  - in A/D converters, 227, 230
  - Nyquist theorem, 210
  - reconstruction, 210, 213

**P**

- Parameter learning
  - Bayesian, 1022
  - classification rate, 1035
  - maximum likelihood, 1019
- Parametric estimation
  - deterministic and stochastic signals, 591
  - stochastic (random) signal models, 593
  - wide-sense stationary random signals, 594
- Parametric modeling, 599
  - deterministic auto-correlation
    - sequence, 601–603
  - impulse response, 601
  - input-output pairs observations, 599–601
  - joint process estimation, 604
  - linear prediction, 601–603
  - Wiener solution, 604–606
- Partial Least Squares (PLS) methods, 1177
- Perceptron
  - activation function, 820–821, 834
  - classifier, 820
  - defined, 819
  - Hopfield networks, 826–827
  - support vector machine (SVM), 833, 835–836
  - training, error minimization, 823, 835
- Piecewise linear modeling, 612–613
  - context tree example, 613–614
  - overfitting, 613
- Polar coordinates
  - Cartesian coordinates, 497, 499
  - definition, 497, 508
  - numerical conversion, 534
  - pseudo, 545
- Polyphase channelizer
  - application, 342, 383, 411
  - defined, 420
  - digital filters, 339
  - down conversions, 365
  - operating parameters, 342
  - output sampling frequency, 383, 392
  - software defined radios (SDRs), 364

- Polyphase filter bank  
 down converting, 371  
 DSP applications, 341  
 input signals, 420  
 output sample rate, 384
- Positive-definite matrices, 640
- Postfiltering. *see* Echo cancellation
- Prediction, 644  
 applications, 644  
 schemes, 644
- Principal component analysis (PCA), 842, 845  
 application, 1156  
 basic algorithms, 1166  
 Hotelling deflation procedure, 1173–1174  
 kernel methods, 1174  
 large scale problems, 1166, 1218  
 number determination, 1159  
 orthogonal directions, 865  
 power methods, 1169  
 self-association principle, 1167  
 sparse, 1170  
 standads, 1158
- Probabilistic graphical model (PGM)  
 application, 991  
 applications, 1046  
 efficient inference, 992, 1035  
 generative learning, 991–992, 1033  
 graph theory, 996  
 in approximate inference, 1044  
 in Bayesian networks (BNs), 1001  
 in loopy graphs, 1045  
 in Markov Chains (MC), 1014  
 websites, 1057
- Probability, 120  
 axiomatic approach, 115  
 Bayesian rule, 116  
 classical or a priori approach, 114  
 conditional model, 116–117  
 cumulative distribution function, 120  
 density function, 123  
 Gaussian distribution, 125, 129  
 interval estimate, 140  
 joint cumulative probability distribution function, 132  
 joint model, 116  
 joint probability density function, 132–133  
 mass function, 123  
 mean-ergodicity, 148  
 multiple random variable distributions, 131  
 Poisson distribution, 125  
 random variable sample, 127  
 relative frequency or a posteriori approach:, 114–115  
 steady-state vector, 155  
 total model, 117
- transition model, 117  
 uniform distribution, 124
- Probability density functions (PDFs)  
 information based learning, 1380  
 relevant principle, 1402
- Probability theory, 992
- Prony's method, 603
- Q**
- Quantization schemes, sparsification, 938–939, 966
- R**
- Random process  
 bandpass, 165  
 complex, 153  
 continuous- or discrete-time model, 141  
 continuous-time white noise  $w(t)$ , 159  
 defined, 128, 141  
 distributions, 141  
 ergodicity, 148  
 first order moments, 142  
 Gaussian distribution, 151  
 linear time-invariant (LTI) systems, 161, 165  
 Markov chains, 154  
 for noisy signal, 149  
 Poisson distribution, 152  
 second order moments, 142  
 sequences, 141  
 spectral description, 155, 160  
 statistical independence, 142  
 time averages, 146  
 WSS processes, 145, 159
- Random sequence  
 defined, 143  
 Discrete-time white noise, 159  
 for discrete-time case, 141  
 in Markov chains, 154  
 mean-ergodicity, 148  
 power spectrum, 167  
 spectoral density, 157–158  
 time average, 147
- Random variable  
 binomial distribution function, 125  
 characteristic function, 130, 136–137  
 complex, 138, 153  
 conditional distribution, 126  
 conditional distributions, 134  
 cumulative density function, 121–122  
 expected value of a real scalar function, 135  
 Gaussian distribution function, 124, 128, 137  
 inversion techniques, 1071

- Random variable (*Continued*)  
 low-order moments, 128  
 Markov's inequality, 129  
 mass probability function, 123  
 moment generating function, 136–137  
 multiple distributions, 131  
 orthogonal function, 135  
 parameter estimation, 139  
 Poisson distribution function, 125  
 probabilistic models, 119  
 probability density function, 122  
 processes and sequences, 142  
 statistical independence, 142  
 statistical mean, 127  
 statistical variation, 135–136  
 statistically independent, 134  
 stochastic process, 141  
 transformation function, 130  
 transformation in, 1073  
 uniform distribution function, 123–124  
 vector transformation, 137
- Real time recurrent learning, 827–828
- Recurrent neural network  
 bicausal neural networks, 830–831  
 defined, 819  
 Elman style, 827  
 simple models, 828–829  
 small weights, 830  
 temporal sequences, 827  
 time dependent signals, 818  
 training complexity, 828  
 without training, 828
- Recursive Least Squares (RLS)  
 kernel algorithm, 938–939  
 linear function, 929  
 weighting coefficient, 933
- Region of convergence (ROC)  
 finite duration signals, 43  
 in Fourier transform, 72–73  
 integral, defined, 43  
 in LTI system, 59  
 of Laplace transform, 43–44  
 on s-plane, 44–45  
 pole location, 49
- Regression task  
 in kernel APSC algorithm, 960–961  
 in RKHS, 959  
 size estimation, 953–954
- Regularization  
 cost function., 891–892  
 loss functions, 970–971
- Machine Learning task, 890  
 overfitting problem, 891–894, 929  
 user-defined parameter., 921
- Rejection sampling, 1075  
 Relevance learning, 847  
 Reproducing kernel Hilbert spaces (RKHSs), 860–861  
 complexification, 927–928  
 cost functions, 915  
 examples of kernels, 902  
 infinite dimensions, 895, 965  
 Information estimators, 1393  
 ITL costs, 1392  
 Kernel RLS, 933–934  
 nonlinear function, 917–918, 958  
 online learning schemes, 915–916  
 properties, 897–898, 904  
 regularization problems, 1245
- Resampling algorithm, sequential importance, 1094
- Reservoir computing, 818, 824, 828
- Resonance circuits, digital filter, 298  
 first-order circuits, 299  
 second-order parallel, 301  
 second-order richards' structures, 301  
 second-order series, 300
- RJ-MCMC algorithm  
 acceptance probabilities, 1107–1111  
 construction techniques, 1104  
 illustration, 1110  
 target density, 1107
- RLC circuit  
 initial condition, 40, 52  
 transfer function, 61  
 voltage series, 39, 75–76  
 voltage-current relationship, 50
- RLS algorithm  
 advantages, 681  
 comparing rate convergence, 687  
 conventional type, 685  
 dichotomous coordinate-descent (DCD)  
 function, 721, 724–725  
 error sequence, 681–682  
 finite-precision effects, 719  
 practical implementation, 686  
 unified statistical analysis, 701
- Robust filters, non-Gaussian noise, 732
- S**
- Sampling theorem  
 aliasing, examples, 182  
 anti-aliasing filter, 183  
 continuous-time signal, 181  
 distortion measurement, 192  
 reconstruction process, 184–185

- Self-organizing feature maps (SOFM), 1123  
 Self-organizing map (SOM), 843  
 Semi-supervised learning  
     challenges, 1263  
     co-regularization, 1244  
     co-training framework, 1243  
     E-step parameter, 1247  
     expectation-maximization (EM)  
         approach, 1246, 1248  
     Finite-dimensional approximation, 1254  
     graph Laplacian regularization, 1250  
     graph-based, 1249  
     iterated graph Laplacian regularization, 1252  
     large-scale computing, 1260  
     likelihood function, 1246  
     low density separation, 1241–1242  
     M-step parameter, 1247  
     manifold regularization, 1256  
     measure-based regularization, 1257  
     multiresolution analysis (MRA), 1258  
     partially observed data, 1240  
     self-training, 1249  
     structured outputs, 1259  
     theoretical analysis, 1263  
     transductive support vector machines  
         (TSVMs), 1241–1242  
     unlabeled data value, 1249  
**Sequences**  
     discrete-time signals, 79, 88  
     Fibonacci, 82  
     of signal values, 79  
**Sequential parametric modeling**, 606  
**Shannon Sampling theorem**, 575  
**Shearlet transform**  
     advantages, 534  
     anisotropic components, 534  
     applications, 534, 546  
     cascade algorithm, 543  
     continuous, 497  
     edge orientation, 547, 550  
     frequency domain, 536  
     image features, 548, 552  
     localized function, 533  
     normal directions, 539  
     numerical implementation, 540, 543  
     properties, 550  
     translation parameters, 541  
     vertical direction, 541  
**Short Time Fourier Transform**, 578–579  
**Signal analysis and synthesis**, 565, 580  
     common frame constructions,  
         examples, 566  
     perfect reconstruction filter bank, 561  
     system model, 565  
**Signal processing, frames**  
     characterization of, 569  
     dual frame, the, 564  
     frame coefficients, 562  
     notation, 562  
     overcompleteness, 561  
     signal representation, 561  
     windowed Fourier analysis schemes, 565  
**Signal processing theory**  
     application areas, 3  
     adaptive filtering, 23  
     continuous-time signal and systems, 3–4  
     digital processing system, 8  
     discrete-time signal and systems, 5, 18  
     filter banks, 16  
     filter structure (digital), 13  
     frames, 22  
     multirate processing, 14  
     parameter estimation, 23  
     random signals, 7  
     shear operation, in matrix theory, 437  
     stochastic process, 7  
     wavelets, 16  
**Signal representation**  
     basis search, 487  
     classification, 171  
     functional bases, 470  
**Signals and systems, parametric models**, 594  
     autoregressive (AR) models, 595  
     autoregressive moving average (ARMA)  
         models, 598  
     input-output transfer function, 596–597  
     moving average (MA) models, 597  
     tracking colored noise, 597  
**Singular value decomposition (SVD)**  
     application, 1156  
     basic algorithms, 1166  
     data analysis, 1162  
     large scale problems, 1166, 1218  
**SOBI algorithm**, 1182  
**Software defined radios (SDRs)**  
     base-band shifting, 402  
     concept, origin of, 397  
     definition, controversies, 395–396  
     multirate structures, 341  
     optimizing issues, 342  
     output sampling frequency, 383  
     polyphase channelizers, 339, 342, 364  
     signal parameters, 395, 397, 400  
**Sparse PCA**  
     deflation technique, 1173  
     standard extension, 1170

Sparse vectors  
 matrix completion, 1357, 1362  
 robust PCA, 1360, 1363

Sparsification  
 coherence-based strategy, 920  
 computational complexity, 965  
 nonlinear modeling., 892  
 novelty criterion, 919, 923  
 quantization strategy, 938–939, 966  
 simulation results, 968  
 surprise-based strategy, 920–921  
 various strategies, 924, 926

Speech and audio coding applications, modulated design, 460

Speed and precision, tradeoff between convergence rate and mis-adjustment, compromise, 632

eigenvalue spread of matrix, 636–637

gradient algorithm, 634

mis-adjustment, 631

quadratic cost function, 619–620

steepest descent algorithm performance, 636

State-space description  
 defined, 84  
 filter, defined, 106  
 frequency domain equations, 88  
 general realization, 84–85, 93  
 IIR system structure, 91–93  
 input–output transfer function, 94, 100  
 transfer function, 110  
 vector equations, 86

Stochastic algorithms, 674  
 linearly parametrized classes, 674  
 output, 675

Stochastic gradient (LMS) algorithm, 609

Stochastic processes, sampling  
 error power, 202  
 reconstruction, 197  
 uniform, 195

Sub-Nyquist sampling  
 machine learning, 1304

Subspace clustering  
 matrix factorization, 1130  
 popular algorithms in, 1127

Sufficiently rich signals, 657

Support vector machine (SVM)  
 application, 847  
 fundamental kernel methods, 858  
 linear classifier, 834  
 nonlinear patterns, 857–858  
 structural risk minimization, 833

Symmetric matrices, 640

Synapse, simple perceptron, 819

**T**

Tensors  
 basic models, 1210  
 decompositions models, 1210  
 multilinear operations, 1208

Theoretical descriptions, information based learning  
 divergence, 1382  
 entropy, 1381  
 mutual information, 1383

Tight frames, 569–570

Time-frequency content analysis, 585  
 coefficients obtained, 586  
 step based decomposition  
 algorithm, 586–587  
 Wigner-Ville distribution (WD), 585

Time-varying AR (1) model, 616

Topographic mapping, 843, 845–846

Training data  
 estimation problem, 953  
 fitting sets, 884–886, 888  
 hinge loss functions, 971  
 learning task, 958  
 quantization, 921

Transfer function  
 Bode Plots, 63–64  
 convolution integral, 56  
 in Laplace transform, 55  
 initial condition, 57  
 pole location, 58  
 pole-zero diagram, 57

**U**

Undermodeling, 653

Undersampling  
 distortions in, 190  
 examples, 190

Unit-step signal, 31

Universality, 612, 615  
 sequential adaptive algorithm, 615–616  
 superior steady state performance, 615  
 universal linear predictor performance, 616

Universal model order estimation, 609  
 min-max optimal performance, 611–612  
 number of parameters increased, 609–610

Unsupervised learning, 1116–1117, 1135

**V**

Vapnik-Chervonenkis dimension, 825  
 Vector quantization, loss minimization, 1121

**W**

Wave digital filters, 282  
 current, 284  
 descriptions, 284  
 design, 282  
 power, 285  
 reflectance function, 285  
 voltage, 284

Wave-flow building blocks, 285  
 circuit elements, 285  
 circulator, 288  
 matched termination, 286  
 open-ended unit element, 285, 287  
 resistive load, 286  
 short-circuited unit element, 286–287  
 voltage signal source, 287

Wavelet frame, 575, 578  
 defined, 530  
 directional filter banks, 531  
 Heisenberg boxes, 581  
 Laplacian filter banks, 529  
 Laplacian pyramid, 522  
 multiresolution analysis, 479  
 subsampling, 474

Wavelet transform  
 applications, 514  
 classical 2-D function, 533–534  
 continuous signal, 477  
 dilations and translations, 473–474  
 dimensionality reduction, 478  
 dyadic sampling, 517  
 8-tap orthogonal filter, 448  
 filter banks, 444–445  
 5-/3-tap symmetric filter, 447  
 4-tap orthogonal filter, 447  
 Laplacian pyramid, 522

lifting step, 448  
 local cosine bases, 485  
 lowpass filter, 545  
 mathematical properties, 467  
 multiscale analysis, 476  
 9-/7-tap symmetric filter, 448  
 packet bases, 486  
 properties, 481  
 reconstruction formula, 476  
 smoothness, 481  
 spectral factorization, 444, 446  
 symmetry, 482  
 trous algorithm, 516  
 vanishing moments, 481

Weight-error, covariance matrix, 693–694  
 Weyl-Heisenberg or Gabor frame, 575  
 Whiteness, 1161  
 Widely-linear complex  
 filter, 663  
 least-mean squares, 662  
 Wirtinger's Calculus, 672  
 application, 926–927  
 in complex LMS, 925–926  
 in Pure Complex KLMS (PCKLMS), 928

Wirtinger derivatives  
 Cauchy-Riemann conditions, 671–672  
 minimization problem, 670–671  
 quadratic function, 674  
 real functions of complex variables, 672

**Z**

Zero and non-zero mean variables, 655  
 extended regressor, 656–657  
 extended weight vector, 656–657  
 z-transform, polyphase decomposition, 472