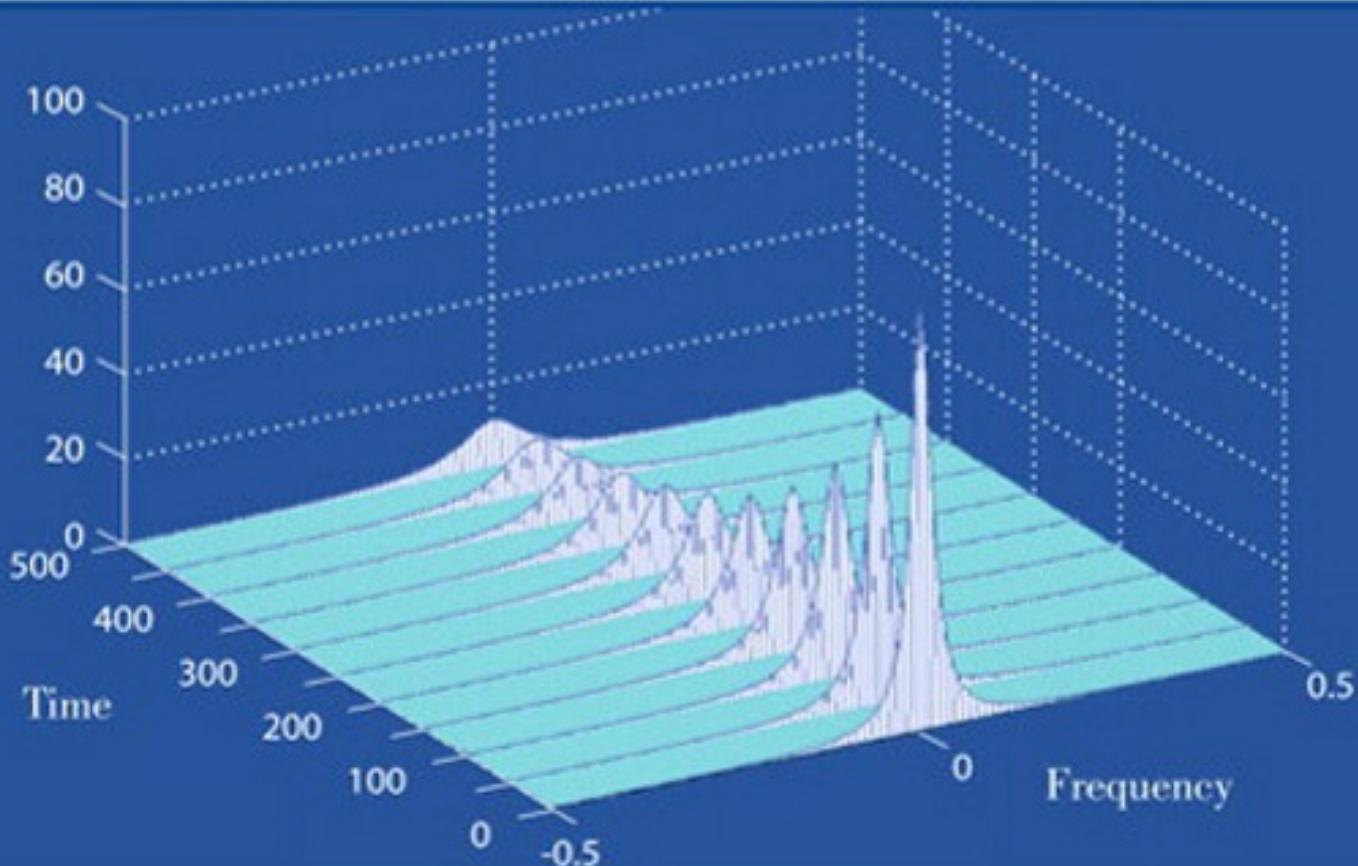


VOLUME III

# FUNDAMENTALS OF STATISTICAL SIGNAL PROCESSING

## PRACTICAL ALGORITHM DEVELOPMENT



STEVEN M. KAY



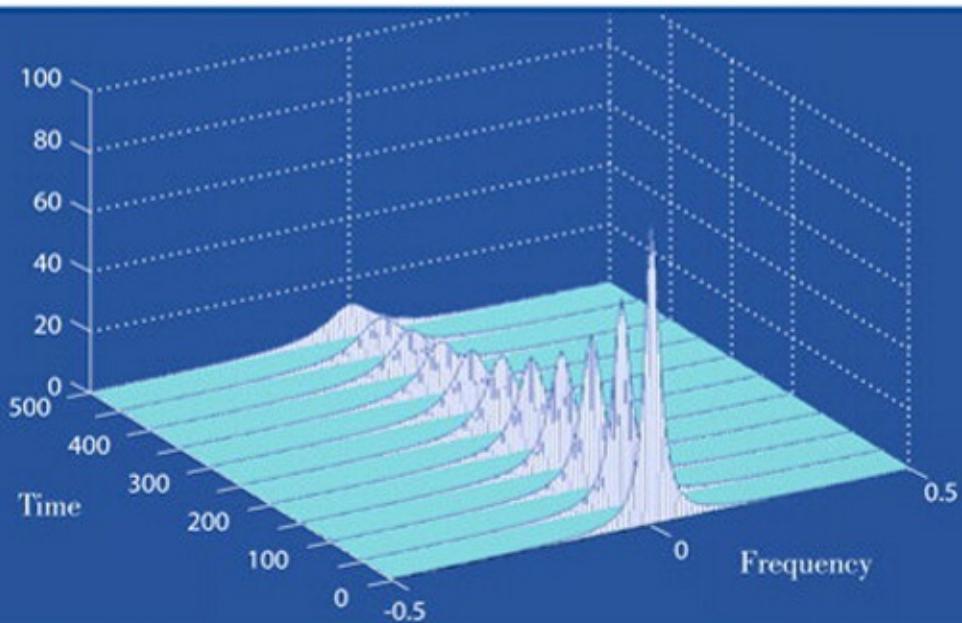
CD-ROM INCLUDED

VOLUME III

PRENTICE  
HALL

# FUNDAMENTALS OF STATISTICAL SIGNAL PROCESSING

PRACTICAL ALGORITHM DEVELOPMENT



STEVEN M. KAY



CD-ROM INCLUDED

# **Fundamentals of Statistical Signal Processing, Volume III**

## **Practical Algorithm Development**

**Steven M. Kay**



Upper Saddle River, NJ • Boston • Indianapolis • San Francisco  
New York • Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales  
(800) 382-3419  
[corpsales@pearsontechgroup.com](mailto:corpsales@pearsontechgroup.com)

For sales outside the United States please contact:

International Sales  
[international@pearson.com](mailto:international@pearson.com)

Visit us on the Web: [informit.com/ph](http://informit.com/ph)

*Library of Congress Control Number: 92029495*

Copyright © 2013 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-280803-3  
ISBN-10: 0-13-280803-X

Text printed in the United States on recycled paper at Courier Corporation in  
~~Woburn, Massachusetts~~

వ్యాచులు, వ్యాచులు

First printing, March 2013.

*To my wife Cindy, who is and will always be my anchor*

# **Contents**

[Preface](#)

[About the Author](#)

## [I Methodology and General Approaches](#)

### [1 Introduction](#)

[1.1 Motivation and Purpose](#)

[1.2 Core Algorithms](#)

[1.3 Easy, Hard, and Impossible Problems](#)

[1.4 Increasing Your Odds for Success—Enhance Your Intuition](#)

[1.5 Application Areas](#)

[1.6 Notes to the Reader](#)

[1.7 Lessons Learned](#)

[References](#)

[1A Solutions to Exercises](#)

### [2 Methodology for Algorithm Design](#)

[2.1 Introduction](#)

[2.2 General Approach](#)

[2.3 Example of Signal Processing Algorithm Design](#)

[2.4 Lessons Learned](#)

[References](#)

[2A Derivation of Doppler Effect](#)

[2B Solutions to Exercises](#)

### [3 Mathematical Modeling of Signals](#)

[3.1 Introduction](#)

[3.2 The Hierarchy of Signal Models](#)

[3.3 Linear vs. Nonlinear Deterministic Signal Models](#)

[3.4 Deterministic Signals with Known Parameters \(Type 1\)](#)

[3.5 Deterministic Signals with Unknown Parameters \(Type 2\)](#)

[3.6 Random Signals with Known PDF \(Type 3\)](#)

[3.7 Random Signals with PDF Having Unknown Parameters](#)

[3.8 Lessons Learned](#)

[References](#)

[3A Solutions to Exercises](#)

## **4 Mathematical Modeling of Noise**

[4.1 Introduction](#)

[4.2 General Noise Models](#)

[4.3 White Gaussian Noise](#)

[4.4 Colored Gaussian Noise](#)

[4.5 General Gaussian Noise](#)

[4.6 IID NonGaussian Noise](#)

[4.7 Randomly Phased Sinusoids](#)

[4.8 Lessons Learned](#)

[References](#)

[4A Random Process Concepts and Formulas](#)

[4B Gaussian Random Processes](#)

[4C Geometrical Interpretation of AR PSD](#)

[4D Solutions to Exercises](#)

## **5 Signal Model Selection**

[5.1 Introduction](#)

[5.2 Signal Modeling](#)

[5.3 An Example](#)

[5.4 Estimation of Parameters](#)

[5.5 Model Order Selection](#)

[5.6 Lessons Learned](#)

[References](#)

[5A Solutions to Exercises](#)

## **6 Noise Model Selection**

- [6.1 Introduction](#)
- [6.2 Noise Modeling](#)
- [6.3 An Example](#)
- [6.4 Estimation of Noise Characteristics](#)
- [6.5 Model Order Selection](#)
- [6.6 Lessons Learned](#)
- [References](#)
- [6A Confidence Intervals](#)
- [6B Solutions to Exercises](#)

## **7 Performance Evaluation, Testing, and Documentation**

- [7.1 Introduction](#)
- [7.2 Why Use a Computer Simulation Evaluation?](#)
- [7.3 Statistically Meaningful Performance Metrics](#)
- [7.4 Performance Bounds](#)
- [7.5 Exact versus Asymptotic Performance](#)
- [7.6 Sensitivity](#)
- [7.7 Valid Performance Comparisons](#)
- [7.8 Performance/Complexity Tradeoffs](#)
- [7.9 Algorithm Software Development](#)
- [7.10 Algorithm Documentation](#)
- [7.11 Lessons Learned](#)
- [References](#)
- [7A A Checklist of Information to Be Included in Algorithm Description Document](#)
- [7B Example of Algorithm Description Document](#)
- [7C Solutions to Exercises](#)

## **8 Optimal Approaches Using the Big Theorems**

- [8.1 Introduction](#)
- [8.2 The Big Theorems](#)

[8.3 Optimal Algorithms for the Linear Model](#)  
[8.4 Using the Theorems to Derive a New Result](#)  
[8.5 Practically Optimal Approaches](#)  
[8.6 Lessons Learned](#)  
[References](#)  
[8A Some Insights into Parameter Estimation](#)  
[8B Solutions to Exercises](#)

## **II Specific Algorithms**

### **9 Algorithms for Estimation**

[9.1 Introduction](#)  
[9.2 Extracting Signal Information](#)  
[9.3 Enhancing Signals Corrupted by Noise/Interference](#)  
[References](#)  
[9A Solutions to Exercises](#)

### **10 Algorithms for Detection**

[10.1 Introduction](#)  
[10.2 Signal with Known Form \(Known Signal\)](#)  
[10.3 Signal with Unknown Form \(Random Signals\)](#)  
[10.4 Signal with Unknown Parameters](#)  
[References](#)  
[10A Solutions to Exercises](#)

### **11 Spectral Estimation**

[11.1 Introduction](#)  
[11.2 Nonparametric \(Fourier\) Methods](#)  
[11.3 Parametric \(Model-Based\) Spectral Analysis](#)  
[11.4 Time-Varying Power Spectral Densities](#)  
[References](#)  
[11A Fourier Spectral Analysis and Filtering](#)  
[11B The Issue of Zero Padding and Resolution](#)

## [11C Solutions to Exercises](#)

### [III Real-World Extensions](#)

#### [12 Complex Data Extensions](#)

[12.1 Introduction](#)

[12.2 Complex Signals](#)

[12.3 Complex Noise](#)

[12.4 Complex Least Squares and the Linear Model](#)

[12.5 Algorithm Extensions for Complex Data](#)

[12.6 Other Extensions](#)

[12.7 Lessons Learned](#)

[References](#)

[12A Solutions to Exercises](#)

### [IV Real-World Applications](#)

#### [13 Case Studies - Estimation Problem](#)

[13.1 Introduction](#)

[13.2 Estimation Problem - Radar Doppler Center Frequency](#)

[13.3 Lessons Learned](#)

[References](#)

[13A 3 dB Bandwidth of AR PSD](#)

[13B Solutions to Exercises](#)

#### [14 Case Studies - Detection Problem](#)

[14.1 Introduction](#)

[14.2 Detection Problem - Magnetic Signal Detection](#)

[14.3 Lessons Learned](#)

[References](#)

[14A Solutions to Exercises](#)

#### [15 Case Studies - Spectral Estimation Problem](#)

[15.1 Introduction](#)

[15.2 Extracting the Muscle Noise](#)  
[15.3 Spectral Analysis of Muscle Noise](#)  
[15.4 Enhancing the ECG Waveform](#)  
[15.5 Lessons Learned](#)  
[References](#)  
[15A Solutions to Exercises](#)

## **A Glossary of Symbols and Abbreviations**

[A.1 Symbols](#)  
[A.2 Abbreviations](#)

## **B Brief Introduction to MATLAB**

[B.1 Overview of MATLAB](#)  
[B.2 Plotting in MATLAB](#)

## **C Description of CD Contents**

[C.1 CD Folders](#)  
[C.2 Utility Files Description](#)

## **Index**

## **Where Are the Companion Content Files?**

# Preface

*Fundamentals of Statistical Signal Processing: Practical Algorithm Development* is the third volume in a series of textbooks by the same name. Previous volumes described the underlying theory of estimation and detection algorithms. In contrast, the current volume addresses the *practice* of converting this theory into software algorithms that may be implemented on a digital computer. In describing the methodology and techniques, it will not be assumed that the reader has studied the first two volumes, but of course, he/she is certainly encouraged to do so. Instead, the descriptions will focus on the general concepts using a minimum of mathematics but will be amply illustrated using MATLAB implementations. It is envisioned that the current book will appeal to engineers and scientists in industry and academia who would like to solve statistical signal processing problems through design of well-performing and implementable algorithms for real systems. These systems are typically encountered in many signal processing disciplines, including but not limited to communications, radar, sonar, biomedical, speech, optical, and image processing. Additionally, due to the emphasis on actual working algorithms, the material should be of use to the myriad of researchers in statistical signal processing who wish to obtain an overview of the state of the *practical* art. Those new to the field who are concerned with sorting the wheat from the chaff in the ever-expanding arsenal of signal processing algorithms will also benefit from the exposition.

The overall goal for this book is to allow the reader to develop his/her intuition and subsequent expertise into the *practice* of statistical signal processing. To accomplish this goal we have endeavored to

1. Describe the methodology, including mathematical modeling, computer simulation, and performance evaluation, used to develop algorithms.
2. Allow the reader to assimilate the important concepts by practicing with the tools typically available. These include useful analytical results and MATLAB implementations for design, evaluation, and testing.
3. Highlight the approaches and specific algorithms that work in practice, i.e., those that have stood the test of time.
4. Illustrate application areas by describing and solving real-world problems.
5. Introduce the reader to some extensions required in practice.
6. Translate a mathematical algorithm into MATLAB code and verify the

integrity of the solution.

Pedagogically speaking, we believe that the strong reliance on MATLAB examples will aid in understanding the workings and subtleties of the various algorithms. The reader will then *learn by doing*. In the same vein, numerous analytical exercises have been embedded into the text for student practice. The full solutions are contained in the appendices of each chapter. MATLAB exercises are also given, with abbreviated solutions listed in the appendix of each chapter, and the full solutions, including executable MATLAB code, contained on the enclosed CD. At the end of many of the chapters is a section called “Lessons Learned”. These conclusions are important observations that are intended to provide insight into the inner workings of the algorithms and rules of thumb that are routinely employed. These lessons learned are often critical to the development of successful algorithms. Most of the topics chosen for inclusion have been drawn from *Fundamentals of Statistical Signal Processing: Estimation Theory*, 1993, and *Fundamentals of Statistical Signal Processing: Detection Theory*, 1998, but we have also added much material from *Modern Spectral Estimation: Theory and Application*, 1988 (all books published by Prentice Hall), since the latter book contains many of the techniques required for data simulation and analysis. Finally, it is hoped that the current book will be useful for self-study. Although this volume can be used without MATLAB as a practice tool, much of the understanding that comes from that experience would be lost.

The background assumed for the reader is a knowledge of calculus, basic linear systems, including some digital signal processing, probability and introductory random processes, and some linear and matrix algebra. As previously mentioned, we have attempted to describe the techniques without heavy reliance upon mathematics and this background material. However, in the end the algorithms are by their nature mathematical and so it must be that this goal can only partially be attained.

The author would like to acknowledge the contributions of the many people who over the years have provided stimulating discussions of teaching and research problems and opportunities to apply the results of that research. Thanks are due to my colleagues L. Jackson, R. Kumaresan, L. Pakula, and P. Swaszek of the University of Rhode Island. A debt of gratitude is owed to all my current and former graduate students. They have contributed to the final manuscript through many hours of pedagogical and research discussions as well as by their specific comments and questions. In particular, Quan Ding and Naresh Vankayalapati have contributed specific comments and helped with the exercise

solutions. Additionally, William Knight has provided valuable feedback on the manuscript. The author is indebted to the many agencies and program managers who have sponsored his research. These managers include Jon Davis, Darren Emge, James Kelly, Muralidhar Rangaswamy, Jon Sjogren, and Peter Zulch. The agencies include the Naval Undersea Warfare Center, the Naval Air Warfare Center, the Air Force Office of Scientific Research, the Office of Naval Research, the Air Force Research Labs, and the Edgewood Chemical and Biological Center. The practical experience that the author has acquired from the numerous industrial firms for which he has consulted is also greatly appreciated. As always, the author welcomes comments and corrections, which can be sent to [kay@ele.uri.edu](mailto:kay@ele.uri.edu).

—Steven M. Kay  
University of Rhode Island  
Kingston, RI

## About the Author

**Steven Kay** was born in Newark, New Jersey. He received a B.E. from Stevens Institute of Technology, Hoboken, New Jersey, in 1972, an M.S. from Columbia University, New York, New York, in 1973, and a Ph.D. from Georgia Institute of Technology, Atlanta, Georgia, in 1980, all in electrical engineering. From 1972 to 1975, he was with Bell Laboratories, Holmdel, New Jersey, where he was involved with transmission planning for speech communications and simulation and subjective testing of speech processing algorithms.

From 1975 to 1977, he attended Georgia Institute of Technology to study communication theory and digital signal processing. From 1977 to 1980, he was with the Submarine Signal Division, Portsmouth, Rhode Island, where he engaged in research on autoregressive spectral estimation and the design of sonar systems. He is presently a Professor of Electrical Engineering at the University of Rhode Island, Kingston, and a consultant to numerous industrial concerns, the Air Force, the Army, and the Navy.

As a leading expert in statistical signal processing, he has been invited to teach short courses to scientists and engineers at government laboratories, including NASA and the CIA. He has written numerous journal and conference papers and is a contributor to several edited books. He is the author of the textbooks *Modern Spectral Estimation* (Prentice Hall, 1988), *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory* (Prentice Hall, 1993), *Fundamentals of Statistical Signal Processing, Volume II: Detection Theory* (Prentice Hall, 1998), and *Intuitive Probability and Random Processes using MATLAB* (Springer, 2005). His current interests are spectrum analysis, detection and estimation theory, and statistical signal processing.

Dr. Kay is a Fellow of the IEEE, and a member of Tau Beta Pi and Sigma Xi. He has been a distinguished lecturer for the IEEE signal processing society. He has been an associate editor for the IEEE *Signal Processing Letters* and the IEEE *Transactions on Signal Processing*. He has received the IEEE signal processing society education award for outstanding contributions in education and in writing scholarly books and texts.

Dr. Kay has recently been included on a list of the 250 most-cited researchers in the world in engineering.

# **Part I. Methodology and General Approaches**

# Chapter 1. Introduction

## 1.1. Motivation and Purpose

Over the last forty years there has been a virtual explosion of ideas, approaches, techniques, and applications of digital signal processing (DSP) to commercial products and military systems. The primary journal devoted to digital signal processing, the *IEEE Transactions on Acoustics, Speech, and Signal Processing*, which was founded in 1974, was originally published bimonthly with each issue consisting of about 100 pages. Today the *IEEE Transactions on Signal Processing*, which is devoted solely to signal processing, is published monthly with a content of about 500 pages, reflecting a *tenfold increase* in papers. This does not even account for the other more specialized journals that have been spawned, such as the *IEEE Transactions on Audio, Speech, and Language Processing*, the *IEEE Transactions on Image Processing*, and others. The algorithm designer, who must choose and implement an approach, is now faced with a bewildering cornucopia of possible algorithms. Even surveying the open literature to glean a promising approach can be an overwhelming task. As a result, it is now more important than ever for the algorithm designer to have a tried and trusted arsenal at his/her disposal. These approaches may not solve the current problem in its entirety, but will at least provide a good starting point for algorithm development.

In addition to accumulating a suite of trusted algorithms, it is critical that we understand why they work as well as when they are likely to fail. DSP algorithms and more specifically *statistical signal processing* algorithms, being highly mathematical and stochastic in nature, do not yield their secrets easily. But as the designer begins to implement these algorithms and observe their behavior, his/her intuition grows along with chances for successful future algorithm choices. This intuition may only be gained through experience. We are fortunate today that it is not necessary to implement an algorithm in hardware to assess its performance. Software implementations are readily available and allow relatively painless assessments of performance. A popular and very versatile software language is MATLAB, and it is this vehicle that we will use to implement our proposed algorithms and examine their performance. Its use allows us to “play” with the proposed algorithm as well as to provide us with a “first-cut” implementation in software. In fact, a MATLAB implementation frequently leads to an implementation on a DSP chip or on some specialized

digital hardware. For these reasons we will rely heavily on MATLAB throughout this textbook.

The material contained herein are algorithms for *statistical signal processing*. On the other hand, for the processing of signals whose mathematical form is completely known and that are not subject to excessive noise, many standard techniques exist and have been found to be quite reliable. As examples, these typically include algorithms for the design of digital filters or for the computation of the discrete-time Fourier transform, i.e., the fast Fourier transform (FFT). Many excellent books describe these algorithms and their implementations [[Ingle and Proakis 2007](#), [Lyons 2009](#)]. In contrast, our purpose is to describe algorithms that can be used to analyze and extract information from *random data*. For example, the specification that a signal whose Fourier spectrum is lowpass in nature should be filtered by a digital lowpass filter prior to further processing, naturally requires the design of a digital filter with a prescribed cutoff frequency. A slightly different specification might be to filter a bandpass signal whose center frequency is *unknown*. In the first case, the specification is complete. In the second case, it remains to determine how to center the filter so as to pass the signal but hopefully remove much of the noise. The former calls for deterministic signal processing while the latter requires *estimation of the center frequency*, preferably on-line, so that if the signal center frequency changes, our algorithm will still be able to provide the appropriately centered filter. When there is uncertainty in the signal characteristics, only a *statistical* approach is appropriate.

Algorithms for the analysis of random data are highly problem specific. This is to say that each real-world signal processing problem, although generically related to many others, is unique and requires a specialized approach. Because of the seemingly endless development of new electronic systems and devices, it is not possible to use “off-the-shelf” algorithms. However, all is not lost! There exist a suite of “core” algorithms that appear at the heart of most practical signal processing systems. It is our intention to describe and implement in MATLAB these approaches in this book. A general discussion of these algorithms is given next.

## 1.2. Core Algorithms

For signal processing problems requiring the detection of a signal and estimation of its parameters, there exist some statistically sound and consequently, well accepted approaches. As examples, we mention the *matched filter* for detection, the *maximum likelihood estimator* and its frequent implementation, the *least*

*squares estimator*, for parameter estimation. It is these well accepted approaches that we intend to focus on. Hopefully, with exposure to the techniques that work in practice, the signal processing algorithm designer will at least have a good starting point from which to proceed to an actual design. Many of the core approaches, in addition to more advanced but possibly not proven-in-practice techniques, have been described in detail in the first two volumes of *Fundamentals of Statistical Signal Processing* [[Kay 1993](#), [Kay 1998](#)] and in *Modern Spectral Estimation: Theory and Application* [[Kay 1988](#)]. The latter book on spectral analysis is important for modeling of random signals and provides many useful algorithms for computer generation of these signals. The reader is encouraged to refer to these books for a fuller understanding of the theoretical underpinnings of these approaches. In this volume we

1. Describe the important algorithms used in practice,
2. Describe the assumptions required for their successful operation, and
3. Describe their performance and their limitations in practice.

This book is an attempt to accomplish these goals *without having had the exposure to the books referenced above*.

### 1.3. Easy, Hard, and Impossible Problems

Since our goal is to describe statistical signal processing algorithms that are widely used in practice, we may ask how these algorithms have attained this place of great honor. The reasons are two-fold. The first is that they “work” and the second is that they can be conveniently implemented in digital software/hardware. For an algorithm to “work”, it must meet the specifications of the system. For example, it might be that a specification calls for a parameter to be estimated. The performance of the estimator should be a relative error of no more than 2.5%, as an example. Hence, whether an algorithm “works” or not depends upon what is expected of the algorithm. If the specifications are unreasonable, then a proposed approach *or any approach* may not work. It is therefore important to assess the *feasibility* of meeting the performance requirements. For the latter example, a commonly used method for feasible parameter estimation accuracy is the *Cramer-Rao lower bound* (see [Section 8.2.1](#)). It provides a lower bound on the variance of an unbiased estimator (i.e., one that on the average yields the correct result). If the specifications cannot be met in theory, then *there is no point in proceeding with a design*. Maybe we should require more accurate sensors, if this is a possibility, and/or more data. Given a signal model and a noise model (we will discuss this further in [Chapters](#)

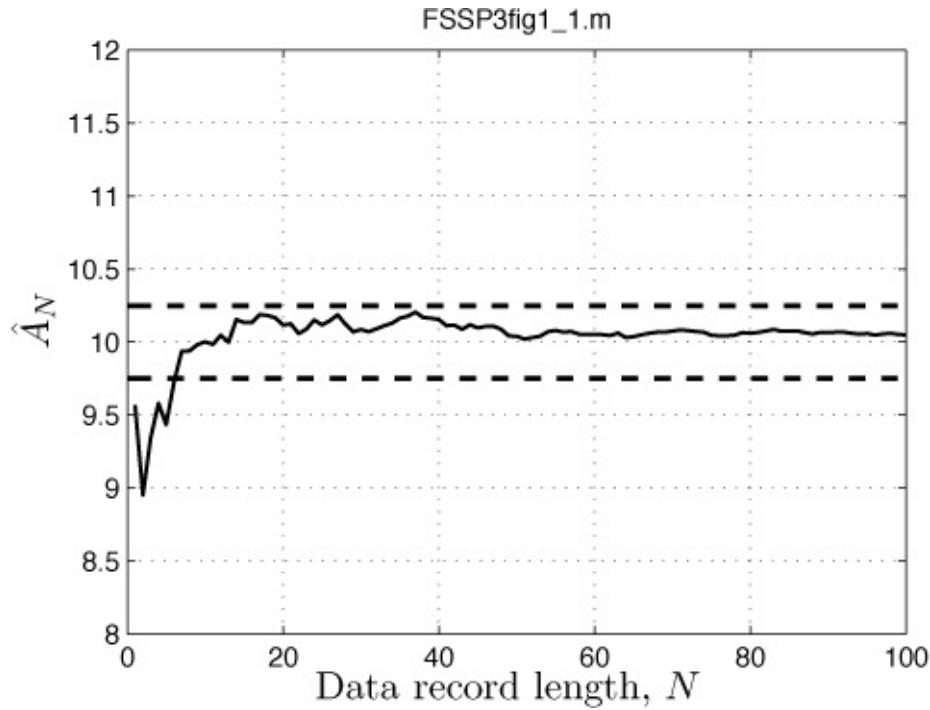
[3–6](#)), signal processing is capable of providing implementable algorithms that *extract all the available information*. This information, we hope, is sufficient to yield the desired performance. However, *it cannot do the impossible*, although we may ask it to do so! As an example, suppose we wish to estimate the value  $A$  of a constant *discrete-time* signal  $s[n]$ , also referred to as a DC level signal (presumably a continuous-time signal that has been sampled by an analog-to-digital (A/D) convertor). The signal is given as

$$s[n] = A \quad n = 0, 1, \dots, N - 1$$

and is embedded in white Gaussian noise (WGN)  $w[n]$  with power  $\sigma^2$  (see [Section 4.3](#)). The observed data is then given by  $x[n] = A + w[n]$  for  $n = 0, 1, \dots, N - 1$  and from this data we wish to determine  $A$  as accurately as possible. It is well known that the optimal way to do this is to employ the estimator given by the *sample mean*

$$\hat{A}_N = \frac{1}{N} \sum_{n=0}^{N-1} x[n].$$

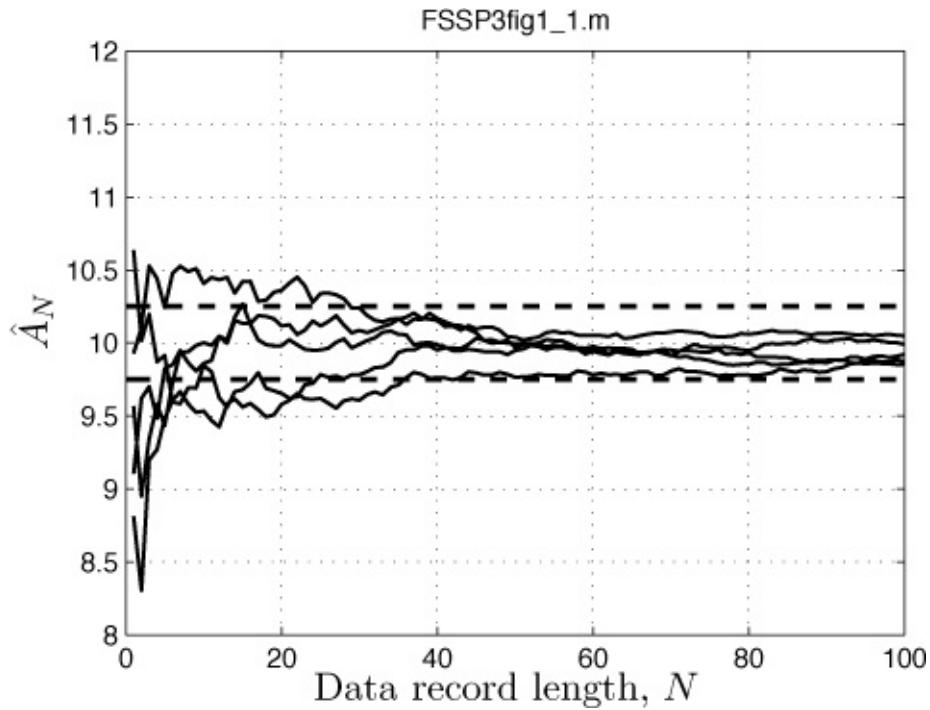
(The “hat” will always denote an estimator.) Suppose that  $A = 10$  and that our specifications require for  $N = 20$  and  $\sigma^2 = 1$  that the estimate should fall within the interval  $[A - 0.25, A + 0.25]$ , which for this choice of parameter would be  $[9.75, 10.25]$ , a maximum relative error of 2.5%. A MATLAB computer simulation is shown in [Figure 1.1](#), in which the estimate  $\hat{A}_N$  is plotted versus the data record length  $N$ .<sup>1</sup> (We have connected the points  $(\hat{A}_1, \hat{A}_2, \dots)$  in the figure by straight lines to make the viewing easier.) Since our specification was for  $N = 20$ , the results seem to indicate that it has been met, with the estimate falling between the dashed lines at 9.75 and 10.25 at  $N = 20$ .



**Figure 1.1: Estimate of DC level A versus data record length  $N$ .**

<sup>1</sup>The MATLAB program used to produce the figure is listed at the top of the figure. These programs may be obtained from the author upon request.

However, this good performance may just be fortuitous in that if we repeat the experiment, whereby a different set of WGN samples  $w[n]$  are generated, then we might obtain very different results. In [Figure 1.2](#) the results of five different experiments, corresponding to five different WGN realizations, are shown. (Each realization of  $N = 100$  noise samples will be different.)



**Figure 1.2: Five different realizations of the DC level estimator.**

Now the specification is not met for  $N = 20$  data points and only appears to be met for more than about  $N = 40$  points. In fact, if we require that the estimate fall within the dashed lines for  $N = 20$  and for 95.5% “*of the time*”, then it can be shown that the variance  $\text{var}(\hat{A}_N)$  of the estimator must satisfy

$$2\sqrt{\text{var}(\hat{A}_N)} \leq 0.25$$

or equivalently

$$\text{var}(\hat{A}_N) \leq \frac{1}{64}.$$

But the Cramer-Rao lower bound says that all (unbiased) estimators must have

$$\text{var}(\hat{A}_N) \geq \frac{\sigma^2}{N} \tag{1.1}$$

which for  $\sigma^2 = 1$  and  $N = 20$  produces a lower bound of  $1/20$ , and so this specification is impossible to meet. We will see later that the estimator  $\hat{A}_N$ , called the *sample mean* estimator, does indeed *attain* the bound and so its variance is given by equality in (1.1). Therefore, to meet the specification we would require

$$\frac{\sigma^2}{N} = \frac{1}{64}$$

which is to say that  $N$  must be at least 64 data samples for a noise power of  $\sigma^2 = 1$ . As a side note, the reader should heed the implicit lesson seen in [Figures 1.1](#) and [1.2](#). Since the results depend upon the particular noise sequence generated, *it is imperative that the experiment be repeated many times*. No conclusions can be drawn from only a few realizations, and in fact, in [Figure 1.2](#) many more realizations should be added.

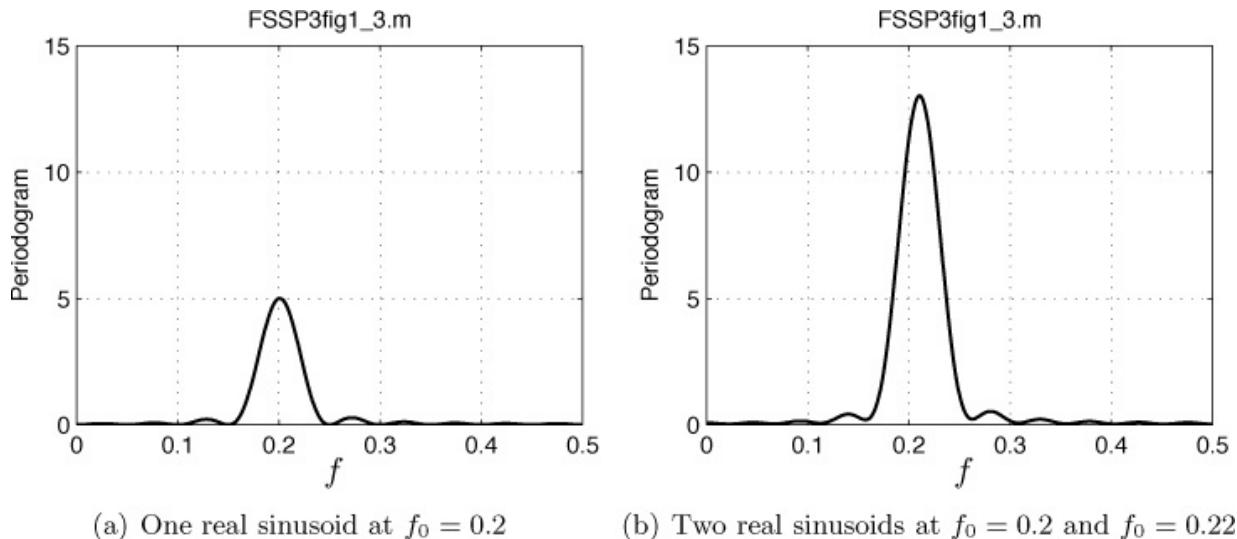
---

### Exercise 1.1 – Necessity of multiple realizations for performance analysis

- a. Run the MATLAB program `FSSP3exer1_1.m` for 100 noise realizations and for  $N = 64$ . Do about 95 of the estimates fall within the specified interval [9.75, 10.25]?
  - b. If the accuracy specification is increased to 1% maximum relative error, what should  $\text{var}(\hat{A}_N)$  be? How long does the data record length  $N$  need to be to attain this? Finally, modify the code in `FSSP3exer1_1.m` to simulate the performance for this larger value of  $N$ . How many estimates meet the specification?
- 

Some signal processing problems appear in many different fields. For example, it is of interest to be able to accurately determine the frequency of a sinusoidal signal when the signal is embedded in noise. A mathematically optimal approach is to use a *periodogram*, which is a type of spectral estimator, and pick the location of the maximum value as the estimate (see [Algorithm 9.3](#)). This works well in practice and has found widespread acceptance for numerous applications. Additionally, even if the underlying assumptions that need to be made to claim optimality are violated somewhat, the performance does not degrade radically. This is an example of a *robust algorithm*. In practice, robustness can be a critically important property of an algorithm. Few real-world data sets conform to a set of underlying theoretical assumptions. These assumptions are usually made for mathematical tractability so as to allow the derivation of an optimal approach. We might term the design of a frequency estimator of a single sinusoid as an *easy problem*, since the solution that performs well and is easily implemented (via an FFT) is the periodogram. Of course, to justify its use, the underlying assumption that we have a single sinusoid embedded in WGN, cannot be totally ignored. If other sinusoids or interfering signals are present and/or if the noise is highly correlated, then this

approach may not perform as advertised. Specifically, if a second sinusoid is close in frequency to the one of interest, then the frequency estimator can be severely biased, producing an inaccurate estimate. An example of this is given in [Figure 1.3](#). In [Figure 1.3a](#) the periodogram of a single sinusoid with an amplitude of one at a frequency of 0.2 is shown. The maximum is seen to occur at the correct location in frequency. In [Figure 1.3b](#) the periodogram is shown for the sum of two sinusoids, the desired one at a frequency of 0.2 combined with an interfering sinusoid at a frequency of 0.22, also with an amplitude of one. It is seen that the peak is now biased away from the desired frequency of 0.2. To account for this possibility we must alter our thinking and acknowledge the potential for one or more interfering sinusoidal signals. This additional complication then leads to a *hard problem* [[Kay 1988](#)]. There may not be a good solution, especially if the frequencies of the interfering sinusoids are unknown. As the reader has no doubt surmised, properly designed algorithms work well (and some can be said to be optimal in performance) when the assumptions under which they were designed are satisfied. It is therefore critical that we be able to verify that *these assumptions hold in practice*. To prepare ourselves for disappointing performance results we need to assess not only the good properties of the algorithm but also its limitations.



**Figure 1.3: Periodogram of a sinusoidal signal and also a sinusoidal signal plus an interfering signal.**

It is often the case that a seemingly difficult or *hard problem* will yield to an easier one if viewed appropriately. It is well known that the *linear signal model* (see [Sections 3.5](#) and [3.6.4](#) for a description) leads to optimally performing and easily implementable algorithms. In the real world not all signals can be said to

conform to this model. Knowing, however, of the desirable properties of the linear model, it behooves us to try and transform our nonlinear model into a linear one. Continuing with the sinusoidal signal example, we have its mathematical representation in discrete-time as

$$s[n] = A \cos(2\pi f_0 n + \phi) \quad n = 0, 1, \dots, N - 1 \quad (1.2)$$

where  $A$  is an unknown amplitude with  $A > 0$ ,  $f_0$  is a known frequency with  $0 < f_0 < 1/2$ , and  $\phi$  is an unknown phase with  $-\pi \leq \phi < \pi$ . We might wish to estimate the unknown amplitude and unknown phase. As it stands, however, the signal is nonlinear in the phase (since  $A \cos(2\pi f_0 n + \phi_1 + \phi_2) \neq A \cos(2\pi f_0 n + \phi_1) + A \cos(2\pi f_0 n + \phi_2)$ ), and this will complicate the development of any estimation algorithm. To convert this problem into a more manageable one, we could use the trigonometric identity  $\cos(C + D) = \cos(C)\cos(D) - \sin(C)\sin(D)$  to yield

$$s[n] = A \cos(\phi) \cos(2\pi f_0 n) - A \sin(\phi) \sin(2\pi f_0 n)$$

and then let  $\alpha_1 = A \cos(\phi)$  and  $\alpha_2 = -A \sin(\phi)$  (which is just a type of polar to Cartesian coordinate transformation) to produce

$$s[n] = \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n). \quad (1.3)$$

The signal is now *linear* in the unknown transformed parameters  $\alpha_1, \alpha_2$ . We have transformed the original hard problem into a relatively easy one, and furthermore, into one whose *solution is well known*. It can be shown that a good estimator of amplitude and phase based on the observed data set  $\{x[0], x[1], \dots, x[N - 1]\}$ , which consists of the sinusoidal signal plus noise, is (see [Section 3.5.4](#) and [Algorithm 9.2](#))

$$\begin{aligned} \hat{A} &= \frac{2}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n) \right| \\ \hat{\phi} &= \arctan \left( \frac{-\sum_{n=0}^{N-1} x[n] \sin 2\pi f_0 n}{\sum_{n=0}^{N-1} x[n] \cos 2\pi f_0 n} \right). \end{aligned} \quad (1.4)$$

But to do so we had to have been familiar with the easy problem (the linear signal model) and then been able to choose the appropriate transformation. In practice, knowledge of easy problems for which we have already *known* solutions is *indispensable*. This book contains many of these well known solutions.

In the real world most of the signal processing problems are *hard* (if they were not, then someone would have already solved them!). But knowing the solutions to simpler problems and building upon the intuition that familiarity with these problems brings, can lead to good solutions for more difficult problems. For example, in (1.3) we might not know the frequency  $f_0$ . How could we then estimate the frequency in addition to the amplitude and phase? It can be shown that the magnitude of the discrete-time Fourier transform of a time truncated sinusoid, i.e., of (1.2), is

$$|S(f)| = \left| \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \right| \quad (1.5)$$

$$\approx \frac{NA}{2} \left| \frac{\sin[N\pi(f - f_0)]}{N \sin[\pi(f - f_0)]} \right| \quad (1.6)$$

as long as  $f_0$  is not very close to 0 or 1/2. You are asked to verify this in the next exercise (the solution is contained in [Appendix 1A](#)).

---

### Exercise 1.2 – Derivation of discrete-time Fourier transform of truncated sinusoid

Verify the expression given in (1.6). To do so first break the real sinusoid given in (1.2) into its two complex conjugate components  $\exp(j2\pi f_0 n)$  and  $\exp(-j2\pi f_0 n)$ , insert these into (1.5) and then use the complex geometric series result

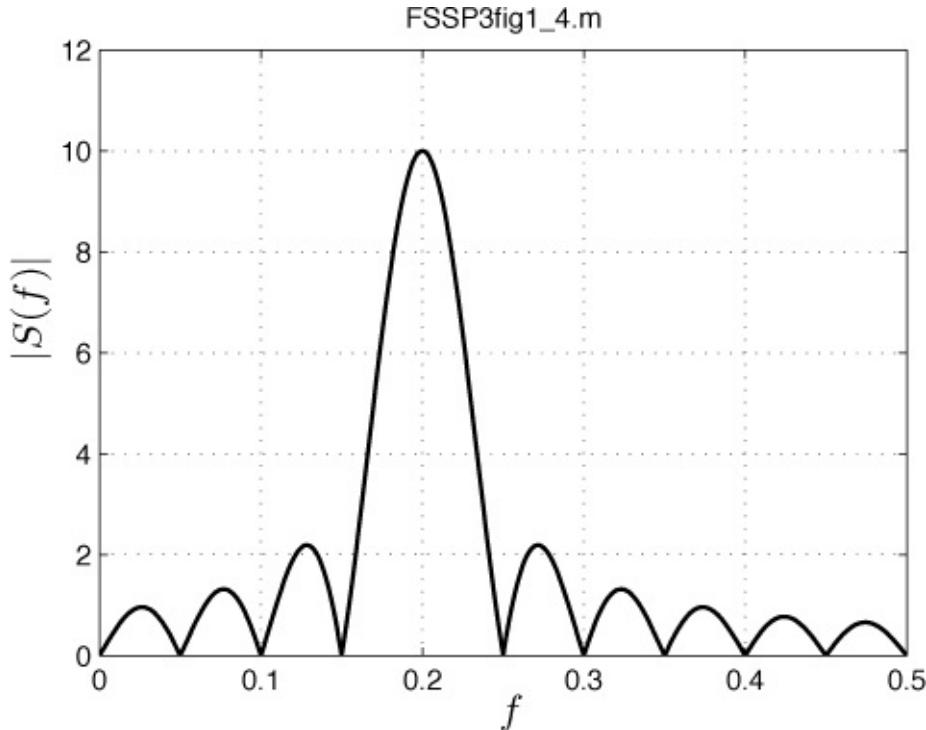
$$\left| \sum_{n=0}^{N-1} z^n \right| = \left| \frac{1 - z^N}{1 - z} \right| = \left| \frac{z^{-N/2} - z^{N/2}}{z^{-1/2} - z^{1/2}} \right|$$

where the first equality is true for all complex  $z$  and the second equality is only true for  $|z| = 1$ , where  $|\cdot|$  denotes the complex magnitude. Finally, discard the negative frequency contribution, which is negligible if we are evaluating the Fourier transform for  $f > 0$  (and  $f_0$  is not near 0 or 1/2).

---

A plot of  $|S(f)|$  is shown in [Figure 1.4](#) for  $A = 1$ ,  $N = 20$ , and  $f_0 = 0.2$ . This suggests that we might be able to estimate the frequency by using the peak location in frequency as our estimator, and in fact forms the basis for the periodogram estimator previously mentioned and illustrated in [Figure 1.3a](#). Although the effect of noise has not been included in our discussion, it does

indeed turn out that even when noise is present, the estimator performs well (see [Algorithm 9.3](#)). This result is also intuitively clear in that for large data records, i.e., as  $N \rightarrow \infty$ , the Fourier transform becomes a Dirac impulse, and so should “stand out” above any noise background. The need for a good sense of intuition cannot be overemphasized. It prevents us from pursuing algorithmic approaches that result in dead ends and it explains why a good algorithm works well, apart from the mathematical justification—if there is one. How does one obtain this intuition? Answer: *Practice, practice, practice!*



**Figure 1.4: Magnitude of discrete-time Fourier transform of time truncated sinusoid.**

## 1.4. Increasing Your Odds for Success—Enhance Your Intuition

To build up our intuition we must learn by doing. Fortunately, there are many software packages and hardware development kits that can provide us with adequate practice. Since our aim is to design the algorithms in software, our discussion will be limited to this implementation. We have found the MathWorks software package MATLAB to be particularly well-suited to the types of signal processing problems encountered in practice. Therefore, throughout the book MATLAB will be our primary vehicle for developing and testing statistical signal processing algorithms. We have even used MATLAB to

generate the textbook figures (the program used is always listed at the top of the graph and may be obtained from the author upon request) and to solve some of the exercises. The code for the exercise solutions as well as programs that implement the algorithms to be described and to perform various useful functions are provided on the enclosed CD.

Implementation of an algorithm in MATLAB is a testament to the understanding of the basic operation of the algorithm. Furthermore, testing the algorithm on controlled data sets generated within MATLAB, i.e., ones for which the signal and noise characteristics are known, is *essential* in verifying the efficacy of the algorithm. In observing the output of the MATLAB-coded algorithms, and comparing the output with the expected one, a strong sense of intuition can be attained. Finally, we can at times employ the same code used to assess the theoretical algorithm performance using a computer simulation to process actual field data, although usually not in a real-time operational sense. In all these cases, practice is essential in developing one's intuition. There will be ample opportunity to partake of this valuable tool for understanding in completing the exercises scattered throughout the book.

## 1.5. Application Areas

The statistical signal processing algorithms to be described find application in many fields. Some of these are:

1. Communications - transmission and reception of digital information  
[[Proakis and Salehi 2007](#)]
2. Radar and sonar - target detection, localization, and tracking [[Richards 2005](#), [Burdic 1984](#)]
3. Biomedicine - heart arrhythmia detection, brain computer interfaces  
[[Sörnmo and Laguna 2005](#), [Sanei and Chambers 2007](#)]
4. Image processing - medical diagnostic imaging, data compression  
[[Gonzalez and Woods 2008](#)]
5. Speech processing - speech recognition and synthesis [[Rabiner and Schafer 2007](#)]
6. Radio navigation - global positioning systems [[Pany 2010](#)].

This is but a small sampling of applications with many more being added every day. At first glance it may seem strange that these somewhat diverse fields employ nearly the same statistical signal processing algorithms. This was not always so. This confluence of approaches is mainly due to the use of the modern

digital computer for implementation of signal processing. Whether the signal is electrical, as in radar, acoustic, as in sonar and speech, or optical, as in imaging, it ultimately is reduced to a set of numbers to be input and stored within a digital computer. The transformation from a physical signal, say speech, to a set of numbers is accomplished via a transducer, say a microphone, followed by an A/D convertor to produce a set of bytes to be read and stored for further processing within a digital computer. The only difference from a signal processing perspective is the character of the signal. It can be lowpass, having most of its energy at low frequencies, as in speech, or bandpass, having its energy within a given band, as in a radar signal. It can be one-dimensional in nature, as in an acoustic signal, or two-dimensional, as in an image. The sampling rate of the A/D convertor will therefore need to be tailored to the signal under consideration and will result in different amounts of data to process. However, remarkably similar, if not identical, algorithms are used to process these widely disparate types of signals. Matched filters are used in sonar, where the signal spectrum has frequencies in the KHz range, but also in radar, where the signal spectrum has higher frequencies, usually in the GHz range. FFTs are used for one-dimensional signals such as speech, but also the two-dimensional version of the FFT is used for image analysis. As a result, it is not unusual to design signal processing algorithms based solely on a mathematical description of the signal and noise without reference to their origins. This would of course ignore any prior knowledge of the real-world constraints that the signal and noise must obey, and so any subsequent algorithm developed may have the potential for improved performance by imposing these constraints. Knowing, for example, that a signal evolved from a reflection from a moving target allows the radar designer to put a constraint on the Doppler frequency shift due to the constraint on maximum target speed.

## 1.6. Notes to the Reader

### 1.6.1. Types of Signals Considered

In describing the algorithms we will assume that the physical signal is a *real, lowpass signal* that has been sampled at an appropriate rate (at least twice the highest frequency, i.e., at least the Nyquist rate). For applications such as radar and sonar that process bandpass signals, it is customary to use complex demodulation, followed by sampling of the *in phase* and *quadrature* signals. This leads to a complex signal representation, which is slightly more complicated. The required extensions are described in [Chapter 12](#). Also, because the signals are discrete-time in nature, having been sampled by some device and

stored in a digital computer, we will always assume the received data has the form  $x[n]$ , which is a sequence of real numbers indexed by the integer  $n$ . Typically, we will use the index set  $n = 0, 1, \dots, N-1$  if the data record consists of  $N$  successive time samples. Note that we have referred to the samples as being indexed by time. However, the  $n$  index could equally well represent *spatial samples* such as would be obtained from  $N$  sensors equally spaced along a line with the spatial samples having been obtained by sampling all the sensor outputs *at a given and fixed time*.

### 1.6.2. Book Features and Notation

The MATLAB version used throughout the textbook is 7.8 (R2009A). Toolboxes are not required to run the MATLAB code, and where subprograms are called for, they are provided. A brief introduction to MATLAB is contained in [Appendix B](#). A description of the code for all programs contained on the CD is given in [Appendix C](#). In addition, a `readme.txt` file contained on the CD describes its contents. The “typewriter” font, such as used in `run_simulation.m`, indicates MATLAB program names and code.

Throughout the book there are exercises to provide the reader some practice in simple analytical manipulations and MATLAB algorithmic implementations. The solutions to the analytical exercises are contained in the corresponding chapter appendix. The MATLAB exercise solutions are only summarized, with more complete solutions found on the CD. Note the solutions were obtained using MATLAB version R2009A, and so future versions of MATLAB may produce slightly different results. The reader is *strongly encouraged* to try the exercises as they form an important pathway to understanding the material.

At the end of each chapter there is a section entitled “[Lessons Learned](#)”. These are important results, many of which have become “rules of thumb”, and thus, should be committed to memory. For some applications they may form the basis on which an algorithm is either explored for its efficacy or otherwise rejected as not being suitable for the problem at hand.

The mathematical notation for all common symbols is summarized in [Appendix A](#). The distinction between a continuous-time waveform and a discrete-time waveform or sequence is made through the symbolism  $x(t)$  for continuous-time and  $x[n]$  for discrete-time. Plots of discrete-time data such as  $x[n]$ , however, may appear continuous in time, the points having been connected by straight lines for easier viewing. An example of this is given in [Figure 1.1](#). All vectors and matrices are **boldface**, with all vectors being *column* vectors. When a random variable needs to be contrasted with its value, we will use a

capital letter, say  $X$ , to denote the random variable, and a lower case letter, say  $x$ , to denote its value. All other symbolism is defined within the context of the discussion. Also, the reader will frequently be warned of potential “pitfalls”. Common misconceptions leading to design errors will be noted and described. The pitfall or caution symbol shown below should be heeded!!



## 1.7. Lessons Learned

The lessons listed next will be a major recurring theme of our discussions. We will have much more to say about these lessons as we examine algorithms and their performance throughout the book.

- Assess the feasibility of the algorithm requirements as the first step in the design. Typically, the Cramer-Rao lower bound is used for estimation, and the probability of detection of the Neyman-Pearson likelihood ratio test is used for detection. For classification, the maximum a posteriori (MAP) classifier is used. In all cases, the probability density functions must be known.
- Reassess the goals and/or require more accurate data if the specifications are not attainable.
- Signal processing cannot do the impossible—there must be a reasonable signal-to-noise ratio for it to succeed.
- In determining the performance of an algorithm via computer simulation, repeat the experiment many times, say 1000 or more. Make sure that your performance metric is statistical in nature, i.e., variance for an estimator, probability of detection for a detector, probability of error for a classifier. Keep increasing the number of experiments until the evaluation of these metrics produces consistent results.
- Make sure that the algorithm is tested under varying operational conditions to assess robustness, also called *sensitivity*.
- Verify that the underlying algorithm assumptions hold in practice by analyzing real-world data.
- Try to transform the problem into a simpler one, for example, the linear signal model.
- Test the algorithm first in MATLAB by using controlled data sets generated within MATLAB. The results should agree with theoretical predictions, if available. The performance obtained under MATLAB

controlled conditions should be an upper bound on that for field data.

- Before proposing an algorithm, scour the open literature for similar signal processing problems in other fields and the methods employed.

## References

- Burdic, W.S., *Underwater Acoustic Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Gonzales, R.C., R.E. Woods, *Digital Image Processing*, Prentice Hall, Upper Saddle River, NJ, 2008.
- Ingle, V.K., J.G. Proakis, *Digital Signal Processing Using MATLAB*, 2nd ed., Cengage Learning, Stamford, CT, 2007.
- Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory*, Vol. I, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory*, Vol. II, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Lyons, R.G., *Understanding Digital Signal Processing*, Prentice Hall, Upper Saddle River, NJ, 2009.
- Pany, T., *Navigation Signal Processing for GNSS Software Receivers*, Artech House, Norwood, MA, 2010.
- Proakis, J., M. Salehi, *Digital Communications*, 5th ed., McGraw-Hill, NY, 2007.
- Rabiner, L.R., R.W. Schafer, *Introduction to Digital Speech Processing*, Now, Boston, 2007.
- Richards, M.A., *Fundamentals of Radar Signal Processing*, McGraw-Hill, NY, 2005.
- Sanei, S., Chambers, J.A., *EEG Signal Processing*, Wiley-Interscience, NY, 2007.
- Sörnmo, L., P. Laguna, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, Elsevier Academic Press, NY, 2005.

## Appendix 1A. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code.

These commands are for the uniform and Gaussian random number generators, respectively.

**1.1** For part a, run the code with `randn ('state', 0)` at the start of the program to initialize the random number generator. You should observe 92 estimates that meet the specifications, i.e., lie within the interval [9.75, 10.25]. For part b, we now require  $2\sqrt{\text{var}(\hat{A}_N)} \leq 0.1$  so that from the CRLB  $\text{var}(\hat{A}_N) = \sigma^2/N = 1/400$  and thus, we require that  $N = 400$ . Modifying the program and running it, produces 95 estimates that meet the specification, i.e., lie within the interval [9.9, 10.1].

## 1.2

$$\begin{aligned}
 S(f) &= \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \\
 &= \sum_{n=0}^{N-1} A \cos(2\pi f_0 n + \phi) \exp(-j2\pi f n) \\
 &= \sum_{n=0}^{N-1} \left[ \frac{A}{2} \exp(j2\pi f_0 n + \phi) + \frac{A}{2} \exp(-j2\pi f_0 n - \phi) \right] \exp(-j2\pi f n) \\
 &= \frac{A}{2} \exp(j\phi) \sum_{n=0}^{N-1} \exp[-j2\pi(f - f_0)n] \tag{1A.1}
 \end{aligned}$$

$$+ \frac{A}{2} \exp(-j\phi) \sum_{n=0}^{N-1} \exp[-j2\pi(f + f_0)n]. \tag{1A.2}$$

Now let  $z = \exp[-j2\pi(f - f_0)]$  and note that  $|z| = 1$ . Dropping the second sum we have

$$S(f) = \frac{A}{2} \exp(j\phi) \sum_{n=0}^{N-1} z^n = \frac{A}{2} \exp(j\phi) \frac{1 - z^N}{1 - z}$$

and taking its complex magnitude

$$\begin{aligned}
|S(f)| &= \frac{A}{2} \left| \frac{1-z^N}{1-z} \right| \\
&= \frac{A}{2} \left| \frac{z^{N/2}(z^{-N/2} - z^{N/2})}{z^{1/2}(z^{-1/2} - z^{1/2})} \right| \\
&= \frac{A}{2} \left| \frac{z^{-N/2} - z^{N/2}}{z^{-1/2} - z^{1/2}} \right|.
\end{aligned}$$

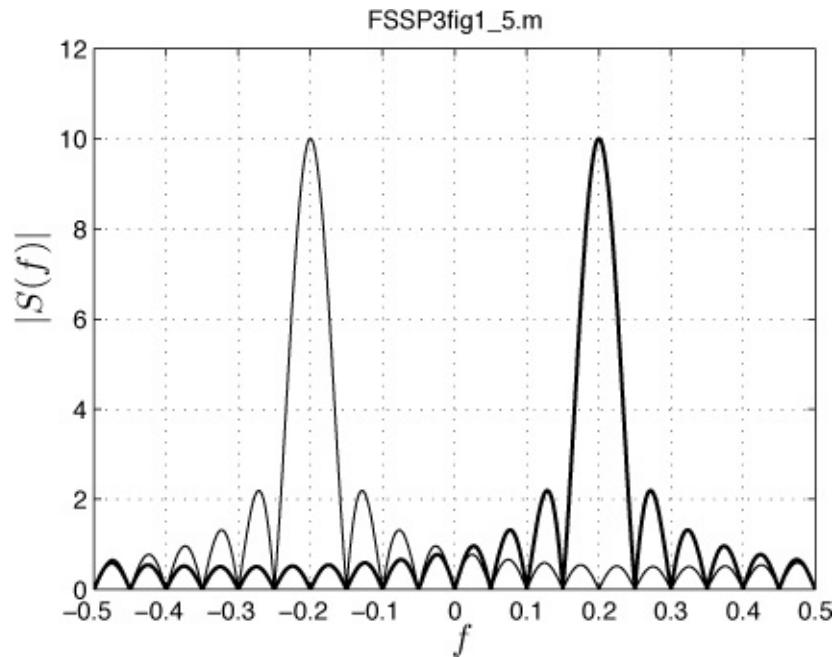
Next letting  $\alpha = -2\pi(f - f_0)$ , we have  $z = \exp(j\alpha)$  and

$$\begin{aligned}
|S(f)| &= \frac{A}{2} \left| \frac{\exp[-j\alpha(N/2)] - \exp[j\alpha(N/2)]}{\exp[-j\alpha(1/2)] - \exp[j\alpha(1/2)]} \right| \\
&= \frac{A}{2} \left| \frac{\sin(N\alpha/2)}{\sin(\alpha/2)} \right|.
\end{aligned}$$

Since  $\alpha/2 = -\pi(f - f_0)$  we have finally that

$$|S(f)| = \frac{A}{2} \left| \frac{\sin[N\pi(f - f_0)]}{\sin[\pi(f - f_0)]} \right| = \frac{NA}{2} \left| \frac{\sin[N\pi(f - f_0)]}{N \sin[\pi(f - f_0)]} \right|.$$

Note that this is approximate since we have neglected the second term in the sum, i.e., the term given by (1A.2). Since this term is the mirror image of the positive frequency term (the one we retained) and is centered about  $f = f_0$ , its contribution will be small as long as it does not “interfere” with the positive frequency term. This will be the case if  $f_0$  is not near 0 or 1/2. This is illustrated in [Figure 1A.1](#) in which we have plotted the magnitude of the terms (1A.1) (positive frequency component) and (1A.2) (negative frequency component) for the entire frequency band  $-0.5 \leq f \leq 0.5$ . The signal parameters are  $N = 20$ ,  $A = 1$ ,  $\varphi = 0$ , and  $f_0 = 0.2$ . It is seen that there is little interference in this case.



**Figure 1A.1: Magnitude of discrete-time Fourier transform of frequency components with  $f_0 = 0.2$  and  $N = 20$ . The heavier line indicates the Fourier transform magnitude of the positive frequency component given by (1A.1) while the lighter line is that for the negative frequency component given by (1A.2). Significant interactions between the two components occur when  $0 < f_0 < 1/N = 0.05$  or  $0.45 = 1/2 - 1/N < f_0 < 1/2$ .**

# **Chapter 2. Methodology for Algorithm Design**

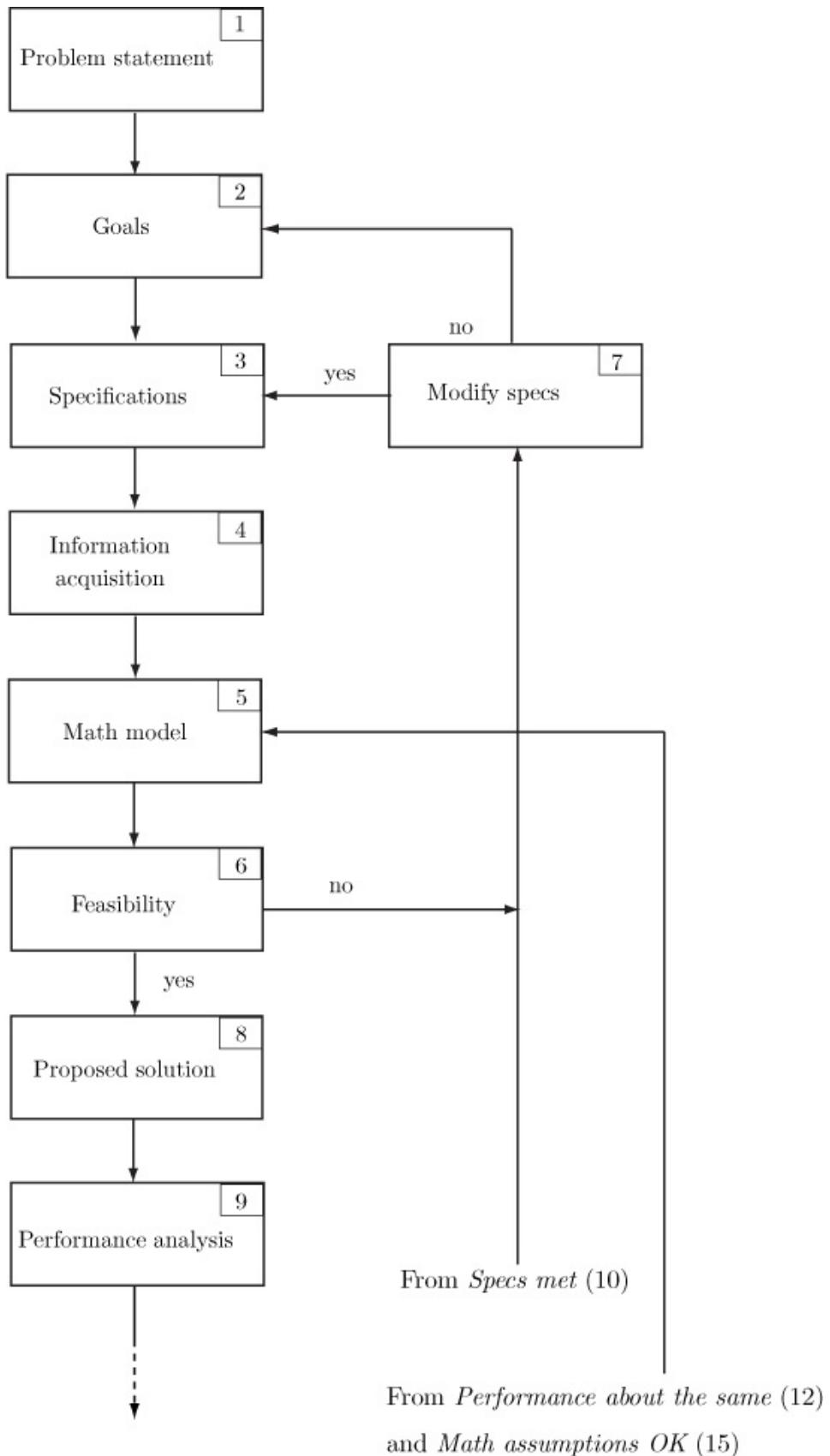
## **2.1. Introduction**

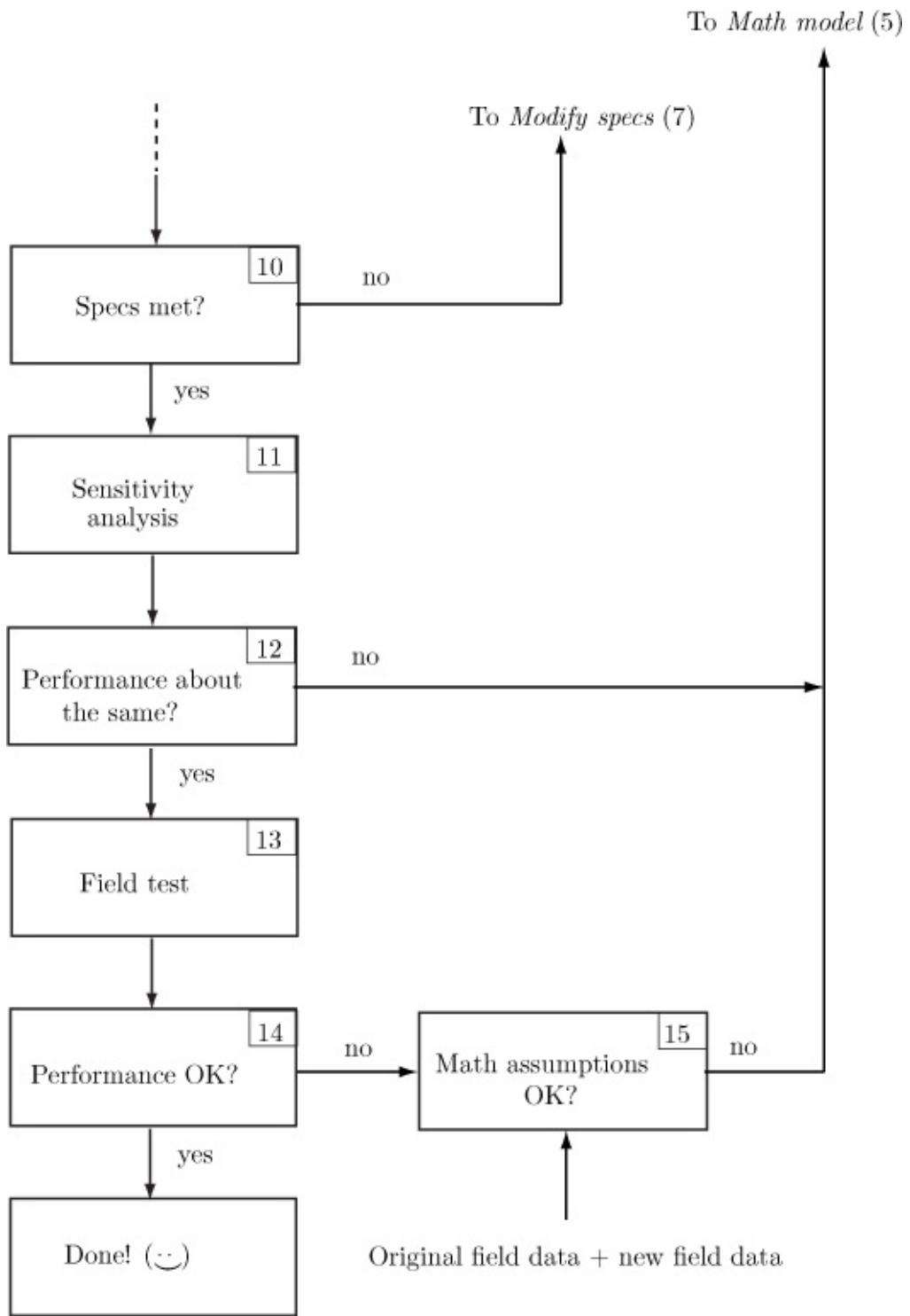
In this chapter we describe a general procedure for choosing an algorithm and determining its performance. Then, we apply the procedure to a hypothetical but realistic problem that might be encountered in practice. In particular, we consider the design of a signal processing algorithm to estimate blood flow velocity that could be implemented in a Doppler ultrasound device. This estimation problem is also of great importance in radar, sonar, and communications. The only difference between its application to these other fields is in the physical origin of the signal of interest, leading to important physical constraints on the signal characteristics. As mentioned previously in [Chapter 1](#), many signal processing problems, of which velocity estimation is one, appear quite commonly in several different fields. A good practical solution in one field is generally “leveraged” for others.

We first begin with a description of a generic approach to algorithm design, and then proceed to a specific application of this approach. Consider the following situation: you are contacted by a person, whom we will call the “sponsor”, to design a working algorithm for a signal processing system. The key component of the system is to be the algorithm, whose purpose is to process an input signal to produce some desired information at the output of the system. “Your mission, should you decide to accept it”, is to provide that algorithm and demonstrate to the sponsor that it indeed produces the desired results. How should you proceed?

## **2.2. General Approach**

A summary of the procedure is shown in [Figure 2.1](#) as a flowchart or sometimes referred to as a “roadmap”.





**Figure 2.1: Roadmap for algorithm development.**

It encompasses the problem description, performance specifications, required input information, proposed solution, and testing to verify the algorithm's performance. All of these topics will be discussed in much more detail in the

succeeding chapters. The reader should note that there is nothing unique about the general approach to be described in that many similar but somewhat different versions are used in practice. It does, however, provide a logical basis for algorithm development and can either be expanded upon or else pruned as the particular problem dictates. The following discussion is keyed to the numbered blocks shown in [Figure 2.1](#).

## 1. Problem Statement

A typical problem statement you might encounter could be to *design an algorithm to determine when an event takes place*. For example, a request might be made to design an alarm system for a home to detect an intruder on the grounds of the home. The sponsor making the request may not be at all familiar with signal processing and therefore, may not have any idea how to accomplish this.

## 2. Goals

The goals are the desired outcomes or end results of the algorithm in signal processing terms. In the previous example, it could be a detection. But probing further into the problem, the real goal could be a detection followed by a classification. The sponsor may not want an alarm to sound if a neighbor's dog has just crossed into his yard. Hence, it is important to translate the problem statement into specific signal processing goals. In a similar vein, does the system have to be fully automatic or can it have human intervention? For example, can a person check a video feed in the yard before the alarm is sounded to avert a false alarm?

## 3. Specifications

Specifications can be very vague, being qualitative, to extremely specific, being quantitative in nature. It might be required that the detection system be able to detect an event most of the time with a low false alarm rate. Or more specifically, it might require a probability of detection of  $P_D = 0.99$  with a probability of false alarm of  $P_{FA} = 10^{-6}$ , where the signal-to-noise ratio (SNR) is at least 20 dB at the receiver input. New systems might have rather loose "specs", being experimental, while existing systems would have much tighter specs. In the latter case, there is usually a *baseline system*, one that is already operational but requires improvements. The specs will then be determined by the desire to improve its performance. For example, a bar code scanner may occasionally mistake an item for a similar one. The new specs might be to have an error rate that is 10 times smaller than the current one.

## 4. Information Acquisition

To bring to bear as much knowledge as possible to the design problem it is necessary to gather all relevant information. This includes understanding the physical constraints of the signal origin, i.e., is the signal acoustic and therefore one-dimensional, or is it electromagnetic and three-dimensional, being sensed from a vector field? If, for example, a radar signal is intercepted and the data is only one-dimensional, was the antenna only capable of sensing the vertical component of the signal?

It is important to know the extent of the sensing and processing resources available. Can an array of sensors be deployed if necessary to meet the system specs? How much computer computation or throughput is available? Does the system need to process the signals and extract information in real time?

Similarly, it is critical that we ascertain whether the algorithm will need to be *adaptive*. By adaptive we mean that the algorithm may need to adapt itself to the environment that it is sensing. Adaptive algorithms are necessary for signal and noise sources that are not relatively stable. For example, a sensor that is mounted on a moving platform will encounter different physical backgrounds, as in an aircraft-mounted radar that flies over different types of terrain. Any algorithm used in this type of environment will be required to adapt to an ever-changing scenario. To answer this question it is advantageous to review the history of the current system, if there is one, to discern when the system works well and when it does not. Poor performance of a system can sometimes be explained by implicit assumptions about the operating conditions that have been violated.

Finally, it is always a good idea to do a literature survey for solutions to similar problems in the same or different fields to see how the problems are approached. Doing so will also make available information about typical generic problems encountered. An example is in surface chemical detection in which the ever-present signatures of nitrogen and oxygen must be edited out of any spectroscopic data. After this preprocessing step, the detection system could then search for “target chemicals”.

## 5. Mathematical Model Selection

*The model to be chosen is the basis for the algorithm.* It should incorporate detailed physical and statistical information about the signal and noise. This model is usually motivated by the physics of the signal and noise generation processes but pared down to make it easy enough to work with. For example, in speech recognition a time-varying linear filter (actually piece-wise time varying) is used in conjunction with glottal/throat excitation modeling to produce an

overall speech production model. Any model chosen should be accurate enough to capture the important aspects of the signal, say the spectrum, but not so complicated that algorithms based on it cannot be conveniently implemented. The latter consideration motivates the use of a linear prediction coding (LPC) model, which is an all-pole filter excited by either white noise or a train of impulses. A more accurate but much more complex model would be a pole-zero filter. The all-pole filter model leads to an easy problem since the parameters are estimated using a set of linear equations. Unfortunately, the physically more accurate pole-zero model requires a difficult nonlinear optimization to obtain accurate parameter estimates.

It is usually helpful to obtain some field data prior to choosing a model. This collected data can be analyzed to determine its statistical characteristics. Some of these are:

- a.** stationarity of data
- b.** signal and noise power spectral densities
- c.** probability density functions
- d.** sources of interference
- e.** sources of data artifacts
- f.** SNRs
- g.** sensor limitations

Properly analyzing field data is an important first step in formulating a good mathematical model.

## **6. Feasibility Study**

Once a mathematical model has been proposed, along with parameter values such as number of signals, signal amplitudes, noise power and spectral distribution, etc., it is possible to do a feasibility analysis. This is an assessment of the *best possible performance* that is achievable *by any algorithm*. If the specs call for performance in excess of this, then clearly this is not possible. In this case, one must either modify the specs or modify the goals for the algorithm. Even if the specs do prove to be attainable in theory, it is a sad but inescapable fact that algorithms perform poorer in the field than in the “lab”. No algorithm is capable of accommodating all the complexities that the real world introduces. Hence, saying that the performance is feasible and using it to predict operational performance is risky. At best the theoretical performance will be an upper bound. If the actual performance in practice is close to the theoretical, but of course poorer, then the algorithm can be said to have succeeded.

Some of the methods used to establish these upper bounds are:

- a. Estimation - Cramer-Rao lower bound (CRLB)—see [Chapters 8](#) and [9](#)
- b. Detection - Neyman-Pearson—see [Chapters 8](#) and [10](#)
- c. Classification - Maximum a posteriori classifier—see [Chapter 8](#)

An example of the use of the CRLB is given in the next section.

## 7. Specification Modifications (if necessary)

If the specs prove to be theoretically unattainable, then there is little choice but to modify them. For example, suppose a tracker is to be designed to track the positions of ten vehicles. The spec calls for a root-mean-square (RMS) error  $\sqrt{\delta_x^2 + \delta_y^2} \leq 1 \text{ m}$  for 95% of the time. Here  $\delta_x$  and  $\delta_y$  are the position errors in the  $x$  and  $y$  directions, respectively. If this is not possible, then maybe the RMS error spec can be modified to be 5 m, or maybe a 1 m error can be maintained but only for 90% of the time. Alternatively, the goals could be modified so that the desired position accuracy can be attained but only for a maximum of five vehicles.

## 8. Proposed Solution

Once the feasibility has been established it is time to do the hard work and actually configure the algorithm. It should be a direct consequence of the mathematical model proposed. *First principle* type of approaches in which some analytical work is necessary to configure the solution are

- a. Estimation - derivation of maximum likelihood estimator (MLE) or minimum mean square error (MMSE) estimator
- b. Detection - derivation of Neyman-Pearson (NP) detector, generalized likelihood ratio test (GLRT), or Bayesian likelihood ratio test
- c. Classification - derivation of maximum a posteriori probability density functions (PDFs).

A much easier approach, but not guaranteed to work as well, would be to modify a known solution in an attempt to adapt it to the problem at hand. For example, in a detection system the optimal approach, an NP detector, can be implemented if the noise power is known. Knowledge of the noise power is necessary to set the detection threshold. Since this is generally not available, the noise power is estimated and then used in the NP detector as if it were known. This may produce acceptable results. If not, then one may be forced to acknowledge this lack of essential knowledge and attempt a derivation of a detector without this knowledge. A GLRT would then suggest itself as a solution.

It is always advantageous at this stage in the development to have a “backup” approach. For example, the proposed algorithm may require a great amount of computation. A suboptimal approach might be proposed as an alternative since typically these algorithms require less computation. Of course, the performance would also be degraded. An example might be the use of the suboptimal energy detector, which requires little computation, as opposed to a GLRT, which would require a search for the maximum likelihood estimate of one or more parameters.

## 9, 10. Performance Analysis

To determine the performance of the proposed algorithm we can either attempt an analytical evaluation or use a Monte Carlo computer simulation evaluation. The former can be quite difficult, requiring many approximations and therefore, producing questionable results. In its favor, an analytical performance description provides valuable insight into the critical factors that affect performance. A computer simulation, on the other hand, is usually much easier and allows incorporation of more realistic assumptions. The disadvantage is that it can be difficult to validate the program logic, and computer coding errors are always a danger. As an example, in an estimation problem we could derive an approximate variance, assuming the data record is large. This is called the *asymptotic variance*. But if the data record is not large enough in practice, the performance prediction may not be accurate enough. Using a computer simulation, however, much the same as in [Exercise 1.1](#), we can accurately determine the variance. When the mathematical model is complicated, we usually resort to computer simulations since mathematical analysis is not tractable. Once the performance has been determined, we can ascertain whether the specs have been met. If not, then we need to modify the specs (go back to Step 7). It is also of interest at this point to determine the *performance margin*. This is typically the reduction in SNR that can be tolerated before the specs are no longer met. Even if the specs have been met, we can expect poorer performance in practice. The increased performance of the algorithm above the specs is important information. It tells us how much performance we can afford to lose in practice but still meet the specs.

## 11, 12. Sensitivity Analysis

Even if the proposed algorithm has been found to meet the specs in theory, this might not be the case in practice. One must prepare the algorithm for the practical realities of 60 Hz interference (stray voltages due to power line pickup), sensor manufacturing tolerances, nonideal analog preprocessing equipment, and so on. To do so, a *sensitivity analysis* should be conducted. For

example, if an EEG signal is being used to control a cursor on a computer screen, which requires a classification algorithm, then the effects of 60 Hz interference could be a problem for the algorithm. Simple filtering may not be possible since the EEG signal may have significant power near 60 Hz. To ascertain the sensitivity of the algorithm to the interference one would, under controlled conditions (in a computer simulation of the algorithm), introduce a 60 Hz interference of increasing level. If the performance does not degrade greatly, then it can be concluded that the algorithm is not particularly sensitive to this interference. However, if this is not the case, we may need to revise the mathematical model to account for the presence of the interference, leading to a modified algorithm. Note that it is sometimes possible to alleviate a sensitivity problem by designing a more *robust* algorithm, i.e., one that does not respond to the interference. This means that the algorithm performance would need to be *uniform* over different levels of 60 Hz interference. More often than not, however, we soon learn that the price for doing so is to decrease the performance when the interference is not present. One needs to be cognizant that the robustness property can be synonymous with the performance of the algorithm exhibiting “uniform mediocrity”.

### **13, 14, 15. Field Testing**

The “proof of the pudding” is actually trying out the algorithm under realistic operating conditions, i.e., in the “field”. The author has found that this step can be a particularly humbling experience! All too often the system does not work as well as predicted in Step 9. This is due to many factors, including testing procedures, equipment malfunctions, in addition to possible algorithm problems. If this occurs, the next step is to take the field data back to the lab for further investigative processing. There the field data can be analyzed to verify that the assumptions used to formulate the mathematical model are indeed accurate. If not, a modification of the model may then be necessary. If the assumptions prove to be accurate, then the only recourse is to obtain new field data in an attempt to determine the source of the discrepancy between the predicted and measured performance.

A last word of caution is in order. In evaluating the performance of an algorithm in the field or back at the lab using field data, it is critical that the data taken be sufficient to make a *statistically valid assessment*. As was illustrated by Example 1.1, a suitable number of experiments need to be conducted under various operational conditions to produce conclusive results. A good test plan will pay careful attention to this requirement and configure the test accordingly.

## 2.3. Example of Signal Processing Algorithm Design

We now illustrate the general design approach described in the previous section. It is centered around a hypothetical problem that could conceivably be encountered in practice. Much of the discussion has been simplified so as to not burden the reader with many of the real-world details. To enhance understanding we have chosen as an example, the design of a device that is similar to one that many people may already be familiar. In particular, the example is that of designing a specialized Doppler ultrasound device to detect a blood clot.

Ultrasound generally refers to an acoustic signal whose frequency content is above the human hearing range, typically above 25 KHz. Some of the physics behind such a device and the practical constraints involved in its design are described in [[Jensen 1996](#)]. The reader is referred there for further details.

### 1. Problem Statement

Anyone who has ever traveled long distances by air is aware that one's legs often feel stiff from a lack of movement. A potential danger of this inactivity is the formation of a blood clot in a deep vein of the leg. This condition is called a *deep vein thrombosis* (DVT), and can be life threatening if the clot breaks free and travels through the circulatory system.

The chief executive officer (CEO) of Friendly Airlines is concerned that DVT events occurring while flying will lead to liability lawsuits for his company. He approaches the University of Rhode Island Signal Processing Group (URISPG), known worldwide for their expertise in signal processing, for help with his problem. The CEO, whom we will henceforth call the sponsor, asks if a device can be designed to determine if a passenger experiencing leg pain has a DVT. Since the sponsor wishes to outfit his entire fleet of aircraft with this device, it needs to be inexpensive. Also, since a flight attendant will have to administer the test and interpret the results, it should be relatively easy to use. And of course, it must be highly accurate. The sponsor does not want to scare the passenger by telling him he has a DVT when that is not the case, but if a passenger does indeed have a DVT, it should be found.

### 2. Goals

After meeting with the sponsor, the URISPG decides that the end result of such a device should be a detection followed by a classification. First the device should be capable of determining that a clot is present (the detection stage) and then if so, of determining its relative size (the classification stage). The size of the clot is important in discussing the test results with the passenger about the urgency of seeing a doctor. This classification is to be "small, medium, or large". The goals

seeing a doctor. This classification is to be simple, meaningful, or large. The goals then are to design a detector with a given probability of false alarm and probability of detection followed by a classifier with a given probability of error of an incorrect classification.

### 3. Specifications

Since there is no device on the market to consult for performance specifications, the specs must be derived from the problem statement and goals. So as not to tell passengers that they have a DVT, when they do not, i.e., to control false alarms, we reason as follows. Friendly Airlines has 200 flights per day that are more than 4 hours in duration, a length of time for which inactivity can cause a DVT. Assuming a full passenger load of 300 persons and the probability of a leg pain complaint of  $10^{-2}$ , it is likely that there would be about 3 complaints per flight. In a week there would be about 4200 complaints. To keep the false alarm rate at no more than 1 per week, we should require the probability of false alarm  $P_{FA}$  to be less than  $1/4200 \approx 2 \times 10^{-4}$ . This establishes our  $P_{FA}$  spec. The probability of detection  $P_D$  should be quite high, and so we choose it to be 0.9. In this way the sponsor can advertise that 9 out of 10 DVTs are detected. As for classification, once a detection has been declared, we wish to classify the clot with a low probability of error  $P_e$ . We choose this to be 0.1. As the performance of all signal processing algorithms depends upon the energy-to-noise ratio (ENR), we will assume a spec of a minimum ENR of 20 dB. This means that the device should operate according to the specifications for ENRs of 20 dB or higher. The definition of the ENR will be given shortly.

### 4. Information Acquisition

By reviewing the open literature on the diagnosis of blood clots, it is found that ultrasound is a common approach. Typically, the test is done in a hospital with a fairly sophisticated and therefore, a large machine. The physical principle that the diagnosis is based upon is the Doppler effect. This effect is used to create a map of blood flow velocity, as in a carotid artery, for example. The same approach should be applicable to our problem. However, we must be aware that the device must be portable, low power, and inexpensive, as per the problem statement. Also, it needs to produce results in near real time—we cannot wait a day or so for a radiologist to read the images. Thus, the algorithm must not be overly complex. It will probably not need to be adaptive since the subject will not be moving nor will the sensor. However, the depth of the clot will be unknown, leading to a varying signal level as the signal propagates from the transmitting transducer through tissue to the clot and back again to the receiving

transducer. As an example, for a 1 MHz ultrasound transmit signal the attenuation due to absorption and scattering is about 1.5 dB/MHz-cm, meaning that the loss is  $1.5 \times 1 \times 2d = 3d$  dB if the clot is  $d$  cm in distance from the transmitting transducer. For clots in the range of 2 to 8 cm, the variation in attenuation would be in the range of 6 to 24 dB, which is quite significant. Finally, the size of the clot can vary from near 0 cm in diameter to 3 cm. This will certainly enter into our thinking about classification. Fortunately, the modeling of an ultrasonic signal as it propagates through tissue has been extensively studied. A vast number of papers and books are available in the open literature [[Jenssen 1996](#)]. We next leverage some of this knowledge in our design.

## 5. Mathematical Model Selection

### 5.A. General Model

The detection and classification of a target is a centerpiece of radar and sonar signal processing. In our problem, the “target” is the blood clot. Much of what has been learned in the radar/sonar field can be applied to modeling ultrasonic scenarios. The first step in the formulation of a model is to use the physics of the signal generation process, the transmission medium propagation characteristics, and the reflection process of the target. To discern a target from the case of no target in an artery we suppose that the clot/target is stationary, whereas the blood flow exhibits a nonzero velocity. From physical considerations the velocity can be sensed through the Doppler effect. Thus, we are naturally led to a Doppler ultrasound device. It would be helpful to have some experimental data for the cases of no clot and a clot present. For our problem, we do not assume that this is available since the device is a new one, and so has not previously been built and tested. Still, existing Doppler ultrasound machines might conceivably be able to provide some useful data.

We will assume that a continuous wave (CW) acoustic ultrasound signal is to be transmitted into the leg under examination. This signal is just a pure sinusoid of a given frequency that is present over an uninterrupted time interval. The received signal, assuming the medium is linear, will be a sinusoid with the same frequency if there is a clot (target is stationary) or a sinusoid with a different frequency if there is no clot, and the reflection of the signal is due to laminar blood flow. As previously mentioned, the received signal will be greatly attenuated due to interactions with the leg tissue. A standard model for the transmitted signal would be

$$s_T(t) = A_T \cos(2\pi F_0 t) \quad 0 \leq t \leq T$$

where  $A_T$  is the amplitude of the transmitted CW,  $F_0$  is the transmit frequency in Hz, and  $T$  is the length of the CW transmission in seconds. For the received signal a simplified, but widely used model in practice is [[Ziomek 1985](#), [Jensen 1996](#)]

$$s_R(t) = A_R \cos[2\pi(F_0 + F_d)(t - \tau) + \beta] \quad \tau \leq t \leq \tau + T. \quad (2.1)$$

The received signal is seen to incur a Doppler frequency shift of  $F_d$  Hz, a delay of  $\tau$  seconds due to the propagation time through the tissue to the clot and back to the receiving transducer, and a phase shift  $\beta$  due to the interactions with the tissue and clot. For a reflected signal from a clot  $F_d = 0$ , and for a reflected signal from blood only  $F_d \neq 0$ . Apart from the Doppler shift, which is a time-varying phenomenon, the amplitude and phase have been altered as they would be if a sinusoid were input to a linear *time invariant* system with a given frequency response. It can further be shown that the Doppler shift is given by [[Jensen 1996](#)]

$$F_d = \frac{2v}{c} F_0 \cos \theta \quad (2.2)$$

where  $v$  is the speed of the blood flow in m/s,  $c$  is the speed of sound propagation in tissue in m/s, and  $\theta$  is the angle in radians between the transducer normal and the blood flow velocity vector. A simple derivation of (2.2) for  $\theta = 0$  is given in [Appendix 2A](#). The transducer is usually a line aperture consisting of a piezoelectric material. Some typical values are  $F_0 = 1$  MHz,  $v_{\max} = 0.5$  m/s,  $c = 1540$  m/s, and  $\theta = \pi/4$  ( $45^\circ$ ), so that the maximum Doppler shift is  $F_{d_{\max}} = 459$  Hz. We will use 500 Hz as our maximum Doppler frequency. Note that the Doppler shift may also be negative, as it depends on the orientation of the transducer relative to the blood flow velocity vector. Hence, we will assume that the Doppler shift is in the interval  $[-500, 500]$  Hz. Therefore, the received signal will be a sinusoid in the frequency band  $999, 500 \leq F_0 + F_d \leq 1, 000, 500$  Hz, which is a very narrow band centered about 1 MHz. To avoid having to sample at twice the highest frequency, which would be 2,001,000 samples/sec, it is customary to demodulate the signal down to baseband and then sample the resultant lowpass signal (see [Chapter 12](#)). Note that the lowpass signal will only have a 500 Hz bandwidth and hence the sampling rate can be as low as 1000 samples/sec, greatly simplifying the hardware design. Although typically this would be accomplished using *complex demodulation* (see [Exercise 2.1](#)), for our problem we can use a real demodulator since we do not care whether the

Doppler shift is positive or negative. We will only be attempting to determine if it is zero or not. Thus, using real demodulation, which entails multiplying by  $\cos(2\pi F_0 t)$  and lowpass filtering, produces

$$s(t) = [A_R \cos[2\pi(F_0 + F_d)(t - \tau) + \beta] \cos(2\pi F_0 t)]_{LPF}$$

where LPF means *lowpass filtering*. By using a standard trigonometric identity ( $\cos(C) \cos(D) = (1/2)(\cos(C + D) + \cos(C - D))$ ), we have that

$$\begin{aligned} s(t) &= \frac{A_R}{2} [\cos[2\pi(F_0 + F_d)(t - \tau) + \beta + 2\pi F_0 t] \\ &\quad + \cos[2\pi(F_0 + F_d)(t - \tau) + \beta - 2\pi F_0 t]]_{LPF} \\ &= \frac{A_R}{2} \cos[2\pi(F_0 + F_d)(t - \tau) + \beta - 2\pi F_0 t] \\ &= \frac{A_R}{2} \cos[2\pi F_d t - 2\pi(F_0 + F_d)\tau + \beta] \\ &= \frac{A_R}{2} \cos(2\pi F_d t + \phi) \end{aligned}$$

where  $\phi = -2\pi(F_0 + F_d)\tau + \beta$  and does not depend on  $t$ . The term with frequency  $2F_0 + F_d$  is filtered out by the lowpass filer. Hence, our received signal is a real sinusoid of frequency  $F_d$ . Note that since  $\phi$  is unknown, we cannot discern a negative Doppler shift from a positive one. This is due to our use of a cosine signal for demodulation as opposed to a sine and a cosine (see also [Chapter 12](#) for details on *complex* demodulation). Since our problem is to discern a zero Doppler shift from a nonzero one, this should not be a problem and simplifies the mathematical modeling and algorithm implementation significantly.

---

### Exercise 2.1 – Complex demodulation

Repeat the previous derivation except replace  $\cos(2\pi F_0 t)$  by its *quadrature component* given by  $-\sin(2\pi F_0 t)$  to find

$$s_Q(t) = [A_R \cos[2\pi(F_0 + F_d)(t - \tau) + \beta](-\sin(2\pi F_0 t))]_{LPF}.$$

You should get  $s_Q(t) = \frac{A_R}{2} \sin(2\pi F_d t + \phi)$ . Then combine this with  $s(t) = \frac{A_R}{2} \cos(2\pi F_d t + \phi)$  to form the *complex signal* as  $\tilde{s}(t) = s(t) + js_Q(t)$ . Can  $\tilde{s}(t)$  be used to find  $F_d$  unambiguously, i.e.,

can we distinguish positive frequencies from negative frequencies?

---

Our final signal model is obtained by letting  $A = A_R/2$  and referencing the time variable to be zero at the leading edge of the returned sinusoid. Thus, we have

$$s(t) = A \cos(2\pi F_d t + \varphi) \quad 0 \leq t \leq T$$

where  $F_d = (2v/c)F_0|\cos(\theta)|$  so that  $F_d \geq 0$ . (We have also defined  $\varphi$  as  $\varphi = -2\pi F_0 \tau + \beta$ .) Next, to model errors such as due to equipment, nonlaminar blood flow, etc., we assume that the returned signal is embedded in continuous-time white Gaussian noise (WGN)  $w(t)$  so that at the receiving transducer output we observe

$$x(t) = A \cos(2\pi F_d t + \varphi) + w(t) \quad 0 \leq t \leq T.$$

Since the maximum Doppler shift is 500 Hz, we sample at the Nyquist rate of  $F_s = 1/\Delta = 1000$  samples/sec. This yields the discrete-time observations  $x(n\Delta)$  that will be input to a digital processor as

$$x(n\Delta) = A \cos(2\pi F_d n\Delta + \varphi) + w(n\Delta)$$

or by letting  $x[n] = x(n\Delta)$  and  $w[n] = w(n\Delta)$ , and defining the normalized digital frequency as  $f_d = F_d/F_s = F_d\Delta$ , we have

$$x[n] = A \cos(2\pi f_d n + \phi) + w[n] \quad n = 0, 1, \dots, N - 1 \quad (2.3)$$

where  $0 \leq f_d \leq 1/2$ ,  $T = (N - 1)\Delta$ . It can further be shown that  $w[n]$  is a discrete-time WGN process with the properties that  $E[w[n]] = 0$ ,  $\text{var}(w[n]) = \sigma^2$  for all  $n$  and all the samples are Gaussian and uncorrelated (and hence also independent) [Kay 2006, pg. 583]. The mathematical model given by (2.3) is our *selected mathematical model*.

---

### Exercise 2.2 – Discrete-time WGN obtained from continuous-time WGN

Continuous-time WGN  $w(t)$  has a power spectral density (PSD)  $P_w(F) = N_0/2$  for  $-\infty < F < \infty$ . It is a useful mathematical model that assumes the noise will subsequently be filtered. Otherwise, the noise power is infinite ( $\int_{-\infty}^{\infty} (N_0/2) dF = \infty$ ). If  $w(t)$  is lowpass filtered to  $W$  Hz, the output of

the lowpass filter  $y(t)$  has a PSD

$$P_y(F) = \begin{cases} N_0/2 & -W \leq F \leq W \\ 0 & |F| > W. \end{cases}$$

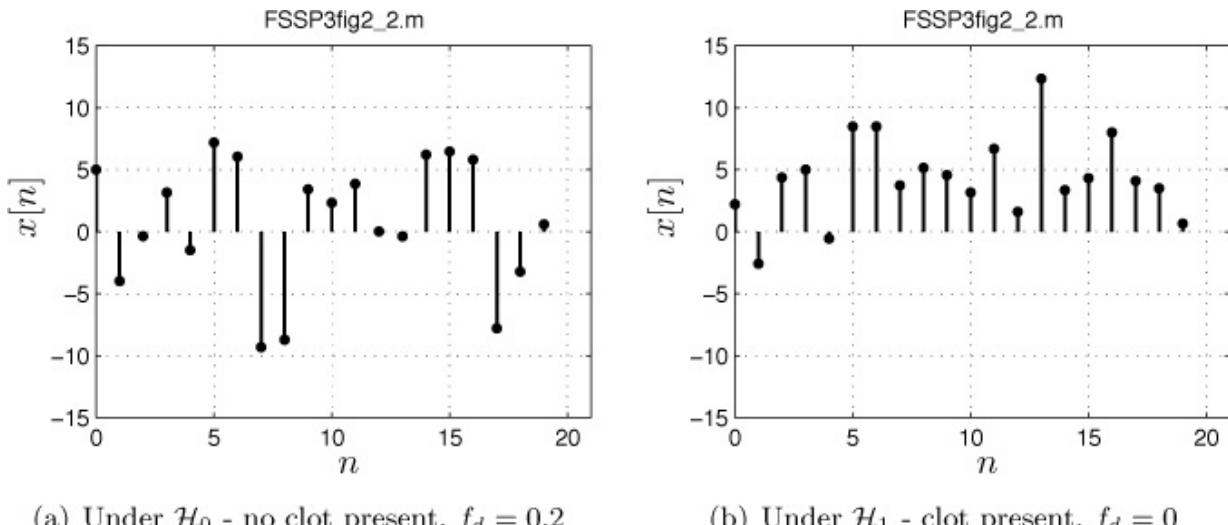
If we now sample  $y(t)$  at the Nyquist rate of  $F_s = 2W$ , the “spectrum” is replicated to produce a periodic function of frequency. Draw a picture of the replicated spectrum, which is the PSD of  $y[n] = y(n\Delta)$ , where  $\Delta = 1/F_s$ . Note that the discrete-time PSD is periodic, but its basic period extends from  $-W/F_s$  to  $W/F_s$ , which equals  $-1/2$  to  $1/2$ . Thus,  $y[n]$  has a flat PSD and so can be said to be discrete-time WGN.

### 5.B. Detection Model

Our detection problem can now be formulated based on the mathematical model of (2.3). When a clot is present, we expect  $f_d = 0$  and otherwise  $f_d > 0$ . The problem is recognized as a hypothesis test between the two competing hypotheses  $H_0$ , which is “no clot”, and  $H_1$ , which is “clot present”, and is normally written as

$$\begin{aligned} \mathcal{H}_0 : x[n] &= A \cos(2\pi f_d n + \phi) + w[n] && \text{no clot} \\ \mathcal{H}_1 : x[n] &= A \cos(\phi) + w[n] && \text{clot present} \end{aligned} \quad (2.4)$$

for  $n = 0, 1, \dots, N-1$ , and where  $w[n]$  is WGN with variance  $\sigma^2$ . An example of the data sets that might be observed under the two hypotheses is shown in Figure 2.2. Here we have used  $A = \sqrt{15}$ ,  $\sigma^2 = 15$ ,  $f_d = 0.2$ ,  $\phi = 0$ , and  $N = 20$ .



**Figure 2.2: Typical data sets observed under each hypothesis.**

---

### Exercise 2.3 – Distinguishability of sinusoids with different frequencies

Assume that  $s[n] = \cos(2\pi f_d n)$  for  $n = 0, 1, \dots, N - 1$ , where  $N = 20$ . Plot the magnitude of the Fourier transform, where the discrete-time Fourier transform is defined as

$$S(f) = \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \quad -1/2 \leq f \leq 1/2$$

for  $f_d = 0$  and compare it to the case when  $f_d = 0.2$ . Next let  $f_d = 0$  and compare it to the case when  $f_d = 0.01$ . What can you say about the distinguishability of two different frequency sinusoids if they are much closer than  $1/N$  in frequency? Hint: You will need to write a MATLAB program.

•

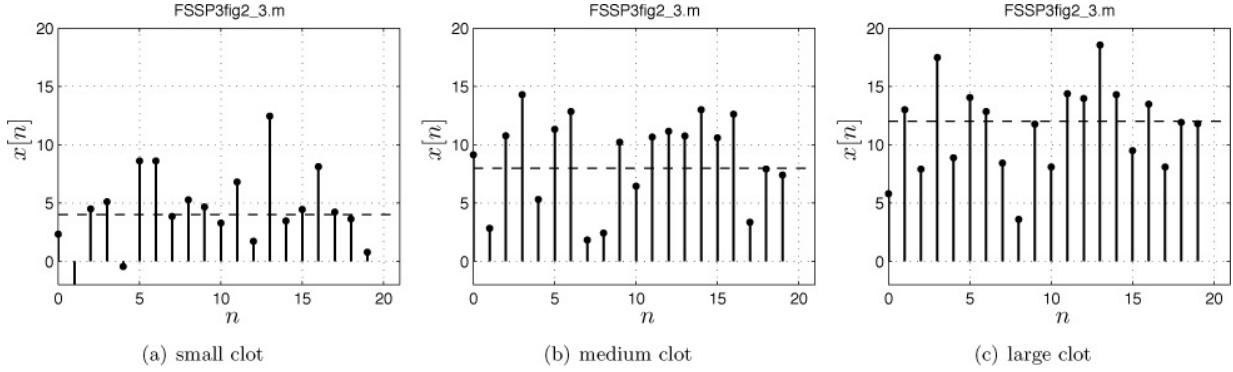
---

### 5.C. Classification Model

Next we formulate the mathematical model for the classification problem. To do so we assume that a detection has taken place and now we wish to determine the size of the blood clot. As described in the Goals, we must ascertain whether the clot is “small”, “medium” or “large”. Of course, this requires us to quantify what we mean by these designations, not an easy matter. Since a detection has occurred, the received data is of the form of  $x[n] = A + w[n]$  for  $A > 0$  (from (2.4), where we assume  $A > 0$  and  $\cos \varphi = 1$ ). We suppose that the smallest clot to be reliably detected is given by the value of  $A$  previously used for the detection illustration, i.e.,  $A = \sqrt{15} \approx 4$ . For the medium-sized clot, we will assume that  $A = 8$ , and for the large clot, we will assume that  $A = 12$ . Hence, we can now summarize the classification problem as one in which we must choose, on the basis of the received data, one of the following hypotheses

$$\begin{aligned} \mathcal{H}_0 &: x[n] = 4 + w[n] && \text{small clot} \\ \mathcal{H}_1 &: x[n] = 8 + w[n] && \text{medium clot} \\ \mathcal{H}_2 &: x[n] = 12 + w[n] && \text{large clot} \end{aligned} \tag{2.5}$$

and where it is assumed that  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is observed. As before,  $w[n]$  is assumed to be WGN with variance  $\sigma^2 = 15$ . Recall that our specification calls for  $P_e = 0.1$ . An example of the data that might be observed is shown in [Figure 2.3](#).



**Figure 2.3: Typical data sets observed for each clot size. The amplitude  $A$  of the clot is shown as the dashed line.**

## 6. Feasibility Study

### 6.A. Detection Problem

We will use the Neyman-Pearson (NP) performance bound as a means of assessing feasibility. It is an upper bound on performance because it can be shown to be optimal in terms of maximizing  $P_D$  for a constrained  $P_{FA}$  (see [Section 8.2.2](#)). Furthermore, in the hypothesis testing problem of [\(2.4\)](#) we do not know the values of  $A$ ,  $f_d$ , and  $\varphi$ . These are signal parameters whose values are critically dependent upon the specifics of the propagation, absorption, and scattering characteristics of the tissue as well as the clot and can never be known a priori. Hence, in practice they will all have to be estimated from the received data. The NP performance assumes that *these parameter values are known* and so must provide an upper bound on the performance that would be obtained in practice. To obtain this upper bound we will assume an ENR of 10 dB under  $H_0$  (no clot), which is defined as (see [\(2.4\)](#))

$$\text{ENR}_0 = 10 \log_{10} \frac{NA^2/2}{\sigma^2} = 10 \text{ dB}$$

where the value is obtained by letting  $A = \sqrt{15}$ ,  $\sigma^2 = 15$ ,  $f_d = 0.2$ ,  $\varphi = 0$ , and  $N = 20$ . Under  $H_1$  (clot present) it is slightly higher, since for these same values

$$\text{ENR}_1 = 10 \log_{10} \frac{NA^2}{\sigma^2} = 13 \text{ dB.}$$

The ENRs of 10 dB and 13 dB should be attainable in practice. We had previously assumed a minimum ENR of 20 dB, so if the specs are feasible, then there should be at least a 7 dB margin for performance degradations in practice. The detection performance will increase as the Doppler frequency shift under  $H_0$

increases. This is because the discrimination between the two signals increases as the signals are more disparate, as will be shown shortly. Here, we choose a Doppler shift under  $H_0$ , say  $f_d = 0.2$ , but recognize that it might be lower in practice. Finally, we need to specify the number of data samples used to form a detection decision. Recall that the sampling rate is 1000 samples/sec. For the model to be valid we must be assured that the transducer placement is fixed, the subject is not moving, and the blood flow velocity is fixed (it varies with heart beat cycles). Thus, we choose a small interval of time in which to observe the received signal. This ensures that the model of (2.4) remains valid, i.e., the signal is *stationary* over time. A typical value is 20 ms, yielding  $N = 20$  samples. We now have all the information necessary to find an upper bound on  $P_D$ . It can be shown that it is given by the NP performance as [Kay 1998, pg. 103] (see also [Exercise 2.4](#))

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{d^2} \right) \quad (2.6)$$

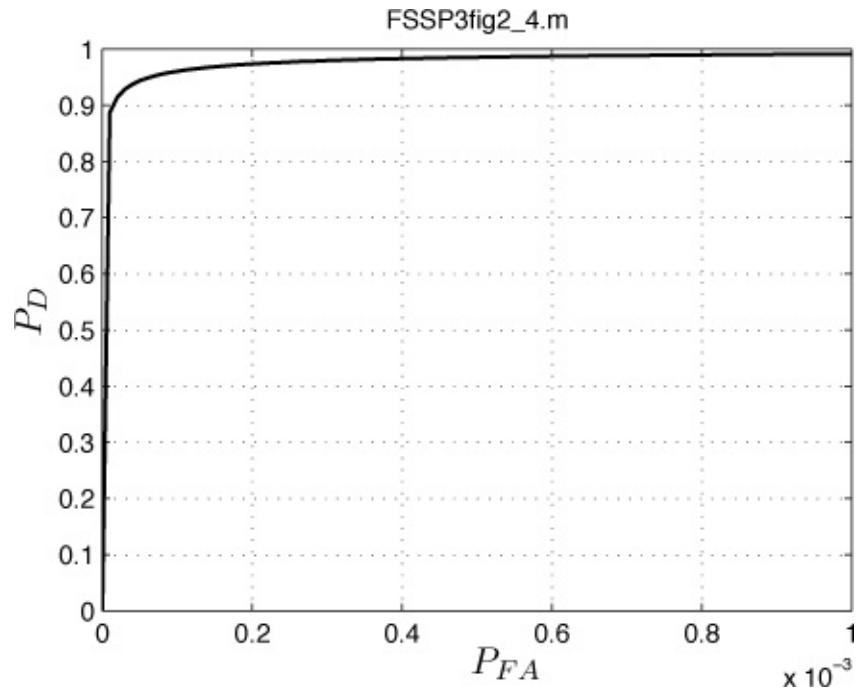
where the  $Q$  function is defined as

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{1}{2}t^2 \right) dt$$

and represents the area under the standard Gaussian PDF (with mean zero and variance one) to the right of the point  $x$ , and  $Q^{-1}(x)$  is the inverse function. (MATLAB subprograms `Q.m` and `Qinv.m` can be used to compute  $Q(x)$  and  $Q^{-1}(x)$ , respectively. They can be found on the enclosed CD.) Also,  $d^2$  is the *deflection coefficient* defined as

$$d^2 = \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (s_1[n] - s_0[n])^2 \quad (2.7)$$

where  $s_1[n] = A \cos(\varphi)$ , which is the received signal under  $H_1$ , and  $s_0[n] = A \cos(2\pi f_d n + \varphi)$ , which is the received signal under  $H_0$ . The deflection coefficient measures how far apart the two signals are relative to the noise power, and the probability of detection increases as the deflection coefficient increases. For the parameters previously chosen ( $A = \sqrt{15}$ ,  $\sigma^2 = 15$ ,  $f_d = 0.2$ ,  $\varphi = 0$ , and  $N = 20$ ), we plot  $P_D$  versus  $P_{FA}$  in [Figure 2.4](#). This graph is called the *receiver operating characteristics* (ROC).



**Figure 2.4: Neyman-Pearson upper bound on probability of detection performance.**

Note that for the spec of  $P_{FA} = 2 \times 10^{-4}$  the probability of detection is about  $P_D = 0.97$  and exceeds the spec of 0.9. Hence, the detection performance is feasible if the ENRs are  $\text{ENR}_0 = 10 \text{ dB}$  and  $\text{ENR}_1 = 13 \text{ dB}$ . Since we originally anticipated an ENR of 20 dB, this leaves about a 7 dB margin, which should be enough to accommodate the degradations encountered in practice.

---

### Exercise 2.4 – Calculation of deflection coefficient

For the parameters given,  $A = \sqrt{15}$ ,  $\varphi = 0$ ,  $\sigma^2 = 15$ ,  $N = 20$ , plot the deflection coefficient given by (2.7) versus  $f_d$  for  $0 \leq f_d \leq 1/2$ . Interpret your results. Hint: You will need to write a MATLAB program.

### Exercise 2.5 – ENR required to meet spec

Rerun the MATLAB code used to generate [Figure 2.4](#) but decrease the ENR to yield the spec of  $P_D = 0.9$  for  $P_{FA} = 2 \times 10^{-4}$ . You can do this by decreasing  $A$ . What value of  $A$  is needed to just attain the spec? The code

used for [Figure 2.4](#) is given on the CD as FSSP3exer2\_5.m.

---

### 6.B. Classification Problem

Next we examine the feasibility of achieving a probability of error of  $P_e = 0.1$  or lower in classifying the size of the blood clot. The classification problem given in [\(2.5\)](#) is a multiple hypothesis test. If the probability of each hypothesis occurring is the same, then it can be shown [[Kay 1998](#), pg. 83] that the optimal classifier has a probability of error of

$$P_e = \frac{4}{3}Q\left(\sqrt{\frac{N(\Delta A)^2}{4\sigma^2}}\right) \quad (2.8)$$

where  $\Delta A$  is the difference in the value of  $A$  between two neighboring hypotheses (see also [Section 8.2.3](#)). From [\(2.5\)](#) this is  $\Delta A = 8 - 4 = 12 - 8 = 4$ . The performance given by [\(2.8\)](#) is the best possible since the optimal classifier needs to know  $\Delta A$ , which is not usually known. For our example, we have that the lower bound on  $P_e$  is

$$P_e = \frac{4}{3}Q\left(\sqrt{\frac{20(4)^2}{4(15)}}\right) = 0.0139 \quad (2.9)$$

which meets the spec of  $P_e = 0.1$ . The margin for any practical degradations is a factor of  $0.1/0.0139 \approx 7$ .

## 7. Specification Modifications

For our example the specs are *theoretically attainable*, and therefore no modifications are necessary.

## 8. Proposed Solution

### 8.A. Detection Algorithm

Referring to [\(2.4\)](#) for the detection model, we now assume that the parameters  $A$ ,  $f_d$ , and  $\varphi$  are unknown. For the mathematical model parameters to be identifiable we need to assume that  $A > 0$  and  $-\pi \leq \varphi < \pi$ . Also, we have assumed that  $0 < f_d < 0.5$  but more realistically the minimum Doppler frequency shift under  $H_0$  is due to the minimum blood flow velocity, and therefore, we will assume that  $0.1 < f_d < 0.5$ . Thus, we need to design a detector without knowledge of the unknown parameters under each hypothesis. A detector that works well in practice in the presence of unknown parameters is the generalized likelihood

ratio test (GLRT) [Kay 1998, pg. 200] (see also [Section 8.5.2](#)). It can be shown that one should decide  $H_1$  (a clot is present) if a detection statistic  $T(\mathbf{x})$ , where  $\mathbf{x} = [x[0] \ x[1] \dots x[N - 1]]^T$ , exceeds a threshold  $\gamma$ . Specifically, we decide  $H_1$  if

$$T(\mathbf{x}) = \frac{\min_{0.1 < f_d < 0.5} \sum_{n=0}^{N-1} [x[n] - \hat{A} \cos(2\pi f_d n + \hat{\phi})]^2 - \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}{\sigma^2} > \gamma \quad (2.10)$$

where

$$\begin{aligned} \bar{x} &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] \\ \hat{A} &= \frac{2}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_d n) \right| \\ \hat{\phi} &= \arctan \left( \frac{-\sum_{n=0}^{N-1} x[n] \sin 2\pi f_d n}{\sum_{n=0}^{N-1} x[n] \cos 2\pi f_d n} \right). \end{aligned}$$

Note that the minimization must be done numerically by substituting each value of frequency in the range  $0.1 < f_d < 0.5$  into  $\hat{A}$  and  $\hat{\phi}$  and then these values into (2.10). The maximum over all the Doppler frequency shifts is taken to be the value of the *detection statistic*  $T(\mathbf{x})$  that is compared to  $\gamma$ . The threshold  $\gamma$  is chosen to satisfy the  $P_{FA}$  spec of  $2 \times 10^{-4}$ .

### 8.B. Classification Algorithm

Recall the mathematical model of (2.5). If we assume that  $\Delta A$  is known, then the  $P_e$  is given by the lower bound of (2.9). We will make this assumption but later examine the sensitivity to deviations from the *assumed known* value of  $\Delta A = 4$ . The actual classification algorithm is called a *minimum distance classifier* and decides a hypothesis as (see [Kay 1998, pg. 83] for an equivalent problem)

$$\begin{aligned} \mathcal{H}_0 &\text{ if } \bar{x} < 6 \\ \mathcal{H}_1 &\text{ if } 6 < \bar{x} < 10 \\ \mathcal{H}_2 &\text{ if } \bar{x} > 10. \end{aligned} \quad (2.11)$$

The classifier chooses the hypothesis whose estimate of  $A$ , which is the sample mean of the data  $\bar{x}$ , is closest to the amplitude value assigned to the clot.

## 9, 10. Performance Analysis

---



## Always do a computer simulation first.

Before ever testing out an algorithm in practice, *it is essential that a computer simulation of the algorithm be done first*. This has many advantages over testing on real data. Some of these are as follows.

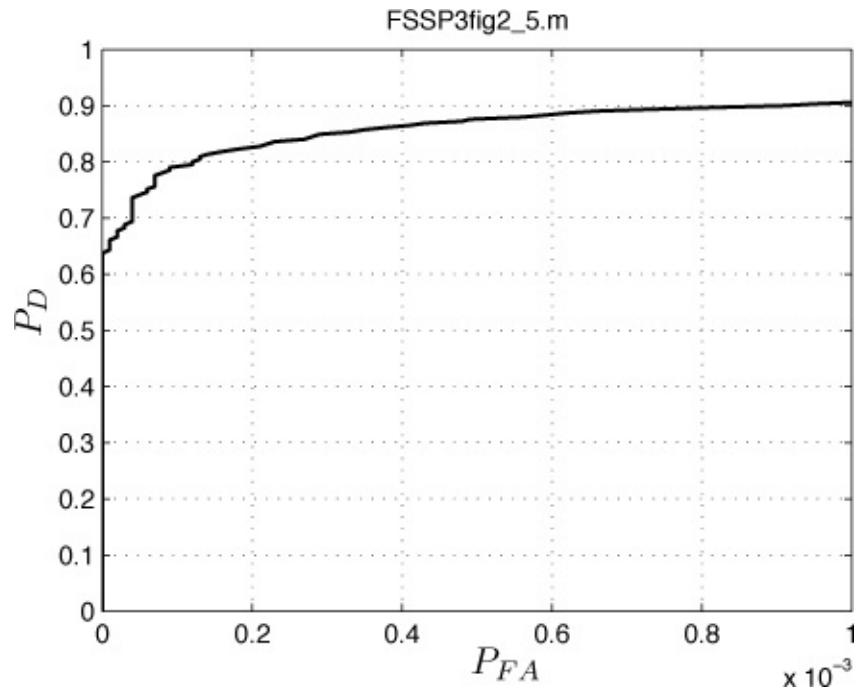
- a. We verify our understanding of the mathematics behind the algorithm by being able to program them into a working algorithm.
- b. Nuances that we may face later, such as ill-conditioning of a matrix as an example, are highlighted.
- c. Repeated trials required to ascertain performance are easily carried out, and the number of trials required for statistically stable performance results may be determined empirically.
- d. Parameter values are easily changed to determine the effect upon the algorithm performance.
- e. Different simulated data sets can be easily input to the algorithm for testing. These data sets can be guaranteed to have given statistical characteristics, unlike field data.



---

### 9.A. Detection Algorithm Performance

The exact  $P_{FA}$  and  $P_D$  for the detector of (2.10) are difficult to obtain by analytical means. Thus, we will resort to a *Monte Carlo* computer evaluation (see [Chapter 7](#) for how we do this). We will use the same parameter values of  $A = \sqrt{15}$ ,  $\sigma^2 = 15$ ,  $\varphi = 0$ , and  $N = 20$ , and also  $f_d = 0.2$ . The results are shown in [Figure 2.5](#), which should be compared with the NP upper bound given in [Figure 2.4](#).



**Figure 2.5: Computer-simulated detection performance of GLRT detector.**

It is seen that for  $P_{FA} = 2 \times 10^{-4}$ , the probability of detection is about 0.82. To meet the spec of 0.9 would require a slightly higher ENR. Note that as expected the performance is poorer than the upper bound. The methods used to carry out the computer evaluation of performance will be described more fully in [Section 7.3.2](#).



### An upper bound is hard (actually impossible) to beat!

If, on the basis of a computer simulation, the results in [Figure 2.5](#) were better than the upper bound results of [Figure 2.4](#), then we would immediately recognize that something was wrong. Presumably there is a “bug” in our computer code. Alternatively, the computer simulation may have used slightly different assumptions than those used to find the upper bound. In either case, it behooves us to find the error before proceeding further. An upper bound on performance is a good way to perform a “sanity check” on the performance of any proposed algorithm.



### 9.B. Classification Algorithm Performance

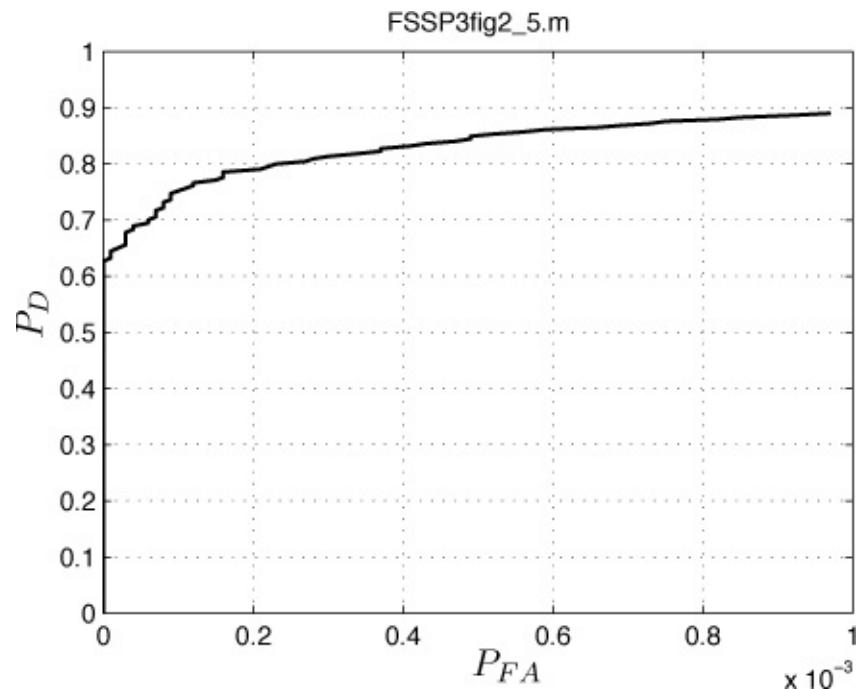
Since the classifier algorithm is an exact implementation of a minimum distance

classifier, the performance is given by (2.8). This has already been found to be  $P_e = 0.0139$  and exceeds the spec of  $P_e = 0.1$ .

## 11, 12. Sensitivity Analysis

### 11.A. Detection Algorithm Performance Sensitivity

As one example of the sensitivity of the proposed algorithm to its mathematical assumptions, consider what might happen if there were timing jitter. By this we mean that the sampling device (an A/D convertor) produces samples nonuniformly in time. To simulate this condition we could replace the time index  $n$  by  $n + u_n$ , where  $u_n$  is a uniform random variable on the interval  $[-0.5, 0.5]$ . The effect is that the sample times will be  $\{0 + u_0, 1 + u_1, \dots, N - 1 + u_{N-1}\}$  so that we observe  $x((n+u_n)\Delta)$  instead of  $x(n\Delta)$ . The sensitivity of the detector to timing jitter can be determined by reevaluating the computer simulated performance previously shown in [Figure 2.5](#). The effect of timing jitter is shown in [Figure 2.6](#).



**Figure 2.6: Computer-simulated detection performance of GLRT detector with timing jitter.**

Comparing the two figures at  $P_{FA} = 2 \times 10^{-4}$ , we see that there is only a slight loss in detection performance. Therefore, the detector is relatively insensitive to timing jitter.

### 1.1.2. Classification Algorithm Performance Sensitivity

For the classification algorithm the principle source of sensitivity would be to the assumption that  $\Delta A$  is known. This value affects the *decision regions*, which are the intervals that delineate the chosen hypothesis, as given by (2.11). As an example, say the three values for  $A$ , which were assumed to be 4, 8, and 12, are actually 4, 10, and 12. Then, using a computer simulation once more it is found that the probability of error increases from  $P_e = 0.0139$  to  $P_e = 0.1730$  so that the spec is no longer met. Clearly, the classifier performance is very sensitive to the values we assume for the amplitudes of the blood clot. It is conceivable that the proposed algorithm would need to be modified. To do so we would follow the path shown in [Figure 2.1](#) (see Step 12) and consider another mathematical model for the blood clot sizes. A more robust approach might be to measure the power in the received  $x[n]$ , and use this metric to discriminate among the blood clot sizes. Under the three hypotheses the assumed power for  $A = 4$ ,  $A = 8$ , and  $A = 12$  would be  $E[x^2[n]] = A^2 + \sigma^2 = 31$ , 79, and 159, for  $H_0$ ,  $H_1$ , and  $H_2$ , respectively. An alternative algorithm, based on the received power, chooses the hypothesis for the original assumed amplitudes as follows.

$$\begin{aligned} \mathcal{H}_0 & \text{ if } \frac{1}{N} \sum_{n=0}^{N-1} x^2[n] < 55 \\ \mathcal{H}_1 & \text{ if } 55 < \frac{1}{N} \sum_{n=0}^{N-1} x^2[n] < 119 \\ \mathcal{H}_2 & \text{ if } \frac{1}{N} \sum_{n=0}^{N-1} x^2[n] > 119 \end{aligned} \quad (2.12)$$

where the power of the received data is estimated and the hypothesis chosen whose power is closest to the estimated power. Then, for no errors in the assumed values of  $A$ , it is found via computer simulation that  $P_e = 0.0258$ , as opposed to  $P_e = 0.0139$  for the classifier based on amplitudes. The performance, as expected, is degraded since (2.12) is not the optimal classifier. However, for the case when the assumptions are violated, in that the actual values of  $A$  are 4, 10, and 12, the probability of error of the suboptimal classifier of (2.12) is found to be  $P_e = 0.1423$ , a factor increase of  $0.1423/0.0258 = 5.5$ . For the optimal classifier in the presence of amplitude assumption errors, the factor increase is  $0.1730/0.0139 = 12.44$ . Hence, the classification performance is less degraded by faulty assumptions, i.e., the algorithm is more robust. The price paid is that when the assumptions are correct, there is a performance degradation with respect to the optimal classifier, the increase in  $P_e$  being a factor of  $0.0258/0.0139 = 1.85$ . Neither algorithm meets the spec of  $P_e = 0.1$ , but since the ENR is 7 dB higher, it is possible that the spec will be attained in practice at this higher ENR.

## **13, 14, 15. Field Testing**

The next step would be to test the algorithms under real operational conditions by processing real data. Of course, field testing means applying the algorithm to persons who do not have a DVT and those who do. This requires the use of a hospital Doppler ultrasound, with the scans to be read by a radiologist. All too often, the proposed algorithm will produce inadequate performance and require us to go “back to the drawing board”! Our assumptions would then have to be examined for flaws. It is conceivable that this entire process of design and testing will require several iterations before a workable device is obtained.

## **2.4. Lessons Learned**

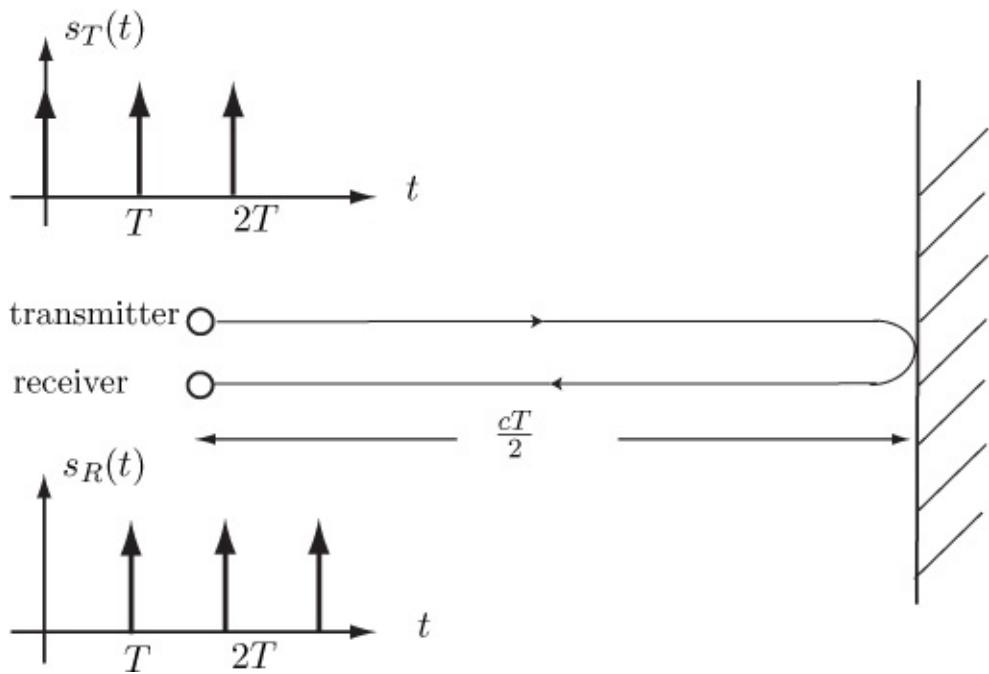
- Use complex demodulation first if a bandpass signal is to be sampled and it is important to maintain the signal characteristics.
- For simplicity of data modeling and algorithm development, always use the white Gaussian noise model for observation noise for a “first-cut” analysis of an algorithm’s performance. You can modify it later if it proves not to be accurate enough in practice.
- Sinusoids of different frequencies can be distinguished by plotting their Fourier transforms if their frequencies are spaced further than  $1/N$  cycles/sample apart.
- Always test the algorithm on computer-simulated data first, since this data can be *controlled* (unlike field data).
- Algorithm performance results are always poorer in practice than those that are obtained from computer simulation. Always build in an extra margin for practical performance degradations.
- Upper bounds on performance can be found by assuming perfect knowledge of parameters that would be unknown in practice.
- Exceedance of an upper bound on performance is a “red flag” telling you that something is wrong with the algorithm and/or its computer implementation.
- For good detection performance the energy-to-noise ratio (ENR) should be at least 15 dB.
- To maintain stationarity of the signal, a short data record is sometimes required.
- As an algorithm’s robustness increases, its performance decreases.

## References

- Jensen, J.A., *Estimation of Blood Velocities Using Ultrasound*, Cambridge University Press, NY, 1996.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- Middleton, D., “A Statistical Theory of Reverberation and Similar First-order Scattered Fields. Part I: Waveforms and the General Process”, *IEEE Trans. on Information Theory*, pp. 372–392, July 1967.
- Ziomek, L.J., *Underwater Acoustics, A Linear Systems Theory Approach*, Academic Press, NY, 1985.

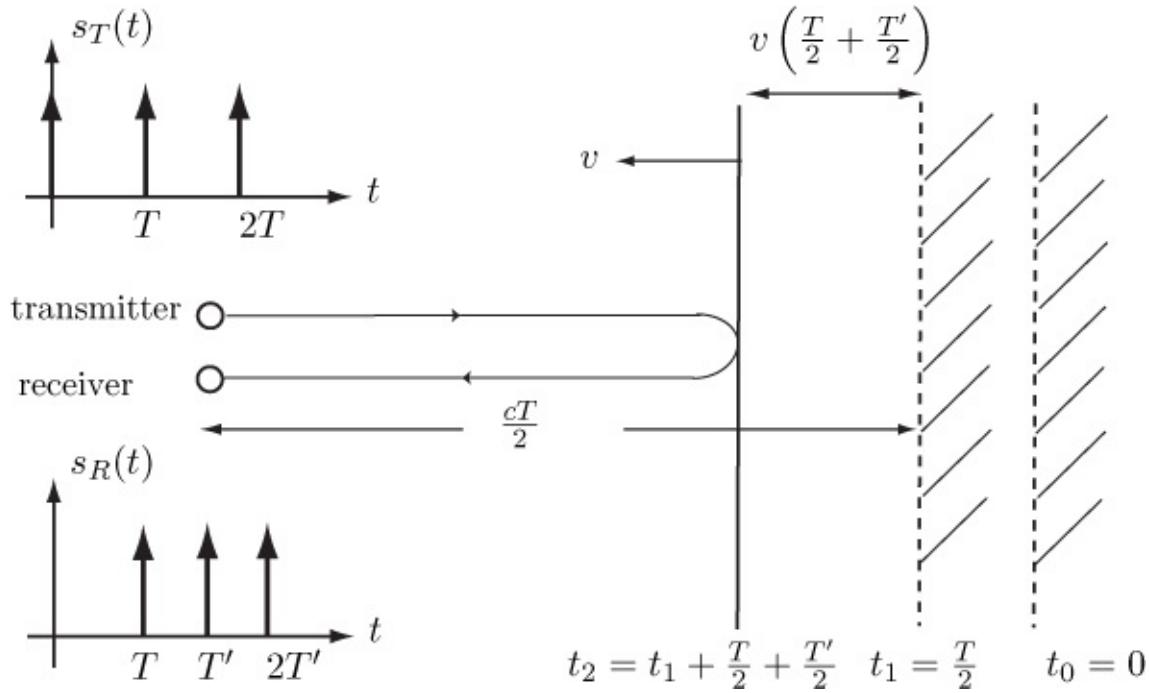
## Appendix 2A. Derivation of Doppler Effect

Consider the following experiment shown in [Figure 2A.1](#). A sequence of impulses  $s_T(t)$  spaced  $T$  seconds apart is transmitted from the position  $x = 0$  and propagates to the right as shown. At a distance of  $cT/2$  meters, where  $c$  is the speed of propagation, the transmitted signal encounters a “wall”, i.e., a reflector, and the signal bounces back toward the receiver. The first pulse that is transmitted at  $t = 0$  travels to the wall for a time of  $d/c = (cT/2)/c = T/2$  and then returns at time  $t = T$ . The first pulse returns just as the second one at  $t = T$  is being transmitted. Hence, a stream of pulses  $s_R(t)$  are received that are spaced  $T$  seconds apart. The received signal is identical to the transmitted signal, except delayed by the round trip propagation time of  $T$  seconds.



**Figure 2A.1: Reflection from a stationary wall.**

Next consider that the wall is moving to the left with speed  $v$  as shown in [Figure 2A.2](#). As before, we transmit the first pulse at  $t = t_0 = 0$ . We position the wall such that it arrives at the previous location  $x = cT/2$  at  $t = t_1 = T/2$ . The first pulse arrives back at the receiver at  $t = T$ , the same as before. When the second pulse is transmitted at  $t = T$ , the wall has moved to the left by an additional  $vT/2$  meters relative to the time the first pulse encountered the wall. As a result, the second pulse arrives at the wall at time  $t = t_2 = t_1 + T/2 + T'/2$ , where  $T' < T$ . To find  $T'$  we use the fact that the position of the wall when the second pulse hits is  $x = (cT/2) - v(T/2 + T'/2)$ . This is the position of the wall when the first pulse hits less the movement to the left while the first pulse returns to the transmitter ( $T/2$  seconds) and the second pulse arrives at the wall ( $T'/2$  seconds). The distance from



**Figure 2A.2: Reflection from a moving wall with speed  $v$ . At time  $t_0$  the first pulse is transmitted. The first pulse hits the wall at time  $t_1$ , while the second pulse hits the wall at time  $t_2$ .**

the transmitter to the wall upon impact for the second pulse is  $x = cT'/2$ . Hence, to find  $T'$

$$\frac{cT}{2} - v \left( \frac{T}{2} + \frac{T'}{2} \right) = c \frac{T'}{2}$$

which results in

$$\frac{1}{T'} = \frac{c+v}{c-v} \frac{1}{T}$$

or the frequency in pulses per second that is seen at the transmitter of the received pulse train is

$$F = \frac{1}{T'} = \frac{c+v}{c-v} F_0$$

where  $F_0 = 1/T$ . A convenient approximation that is used when  $v \ll c$ , as is usually the case, is

$$\begin{aligned} F &= \frac{1+v/c}{1-v/c}F_0 \\ &\approx \left(1+\frac{2v}{c}\right)F_0 \end{aligned}$$

where the approximation

$$\frac{1+x}{1-x} \approx 1+2x$$

has been used. The Doppler shift is seen to be given by  $(2v/c)F_0$ , where  $F_0 = 1/T$  and is the fundamental frequency of the periodic pulse train. The same reasoning applies to a sinusoidal signal. For further details on the Doppler effect that accounts for moving transmitters, receivers, and arbitrary velocity vectors, see [Middleton 1967, Ziomek 1985].

## Appendix 2B. Solutions to Exercises

To obtain the results described below, initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**2.1** Since

$$s_Q(t) = [A_R \cos[2\pi(F_0 + F_d)(t - \tau) + \beta](-\sin(2\pi F_0 t))]_{\text{LPF}}$$

we use the trigonometric identity  $\sin(A)\cos(B) = (\sin(A+B) + \sin(A-B))/2$  to yield

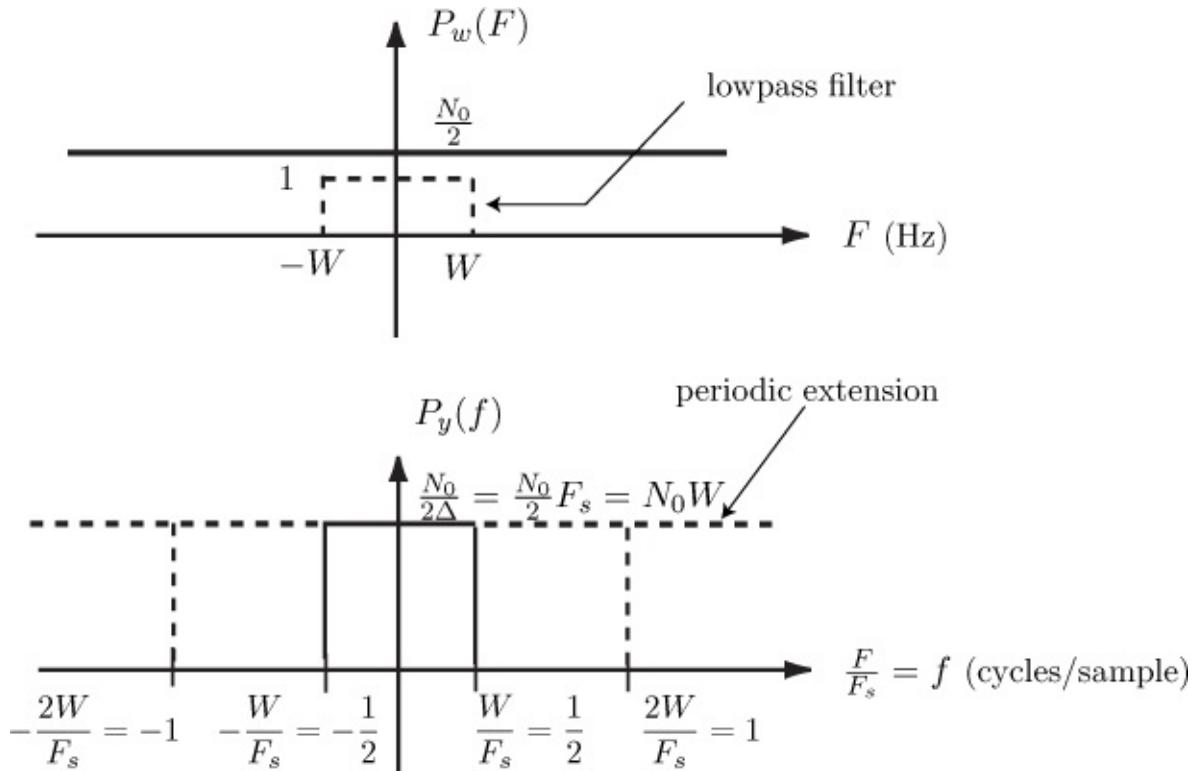
$$\begin{aligned} s_Q(t) &= -\frac{A_R}{2} [\sin[2\pi(F_0 + F_d)(t - \tau) + \beta + 2\pi F_0 t]] \\ &\quad + \sin[2\pi F_0 t - 2\pi(F_0 + F_d)(t - \tau) - \beta]]_{\text{LPF}} \\ &= \frac{A_R}{2} \sin(2\pi F_d t + \phi). \end{aligned}$$

This produces

$$\begin{aligned} \tilde{s}(t) &= s(t) + j s_Q(t) \\ &= \frac{A_R}{2} \cos(2\pi F_d t + \phi) + j \frac{A_R}{2} \sin(2\pi F_d t + \phi) \\ &= \frac{A_R}{2} \exp[j(2\pi F_d t + \phi)] \end{aligned}$$

which is a complex sinusoid at frequency  $F_d$ . The Doppler shift frequency can now be found without knowledge of  $\phi$ . For example, taking the Fourier transform of  $\tilde{s}(t)$  will produce a “sin x/x” type function with a *single* peak at  $F = F_d$ . If the Fourier transform of  $\frac{A_R}{2} \cos(2\pi F_0 t + \phi)$  were taken, there would always be two peaks, one at  $F = F_d$  and one at  $F = -F_d$ . Hence, we could not distinguish a positive Doppler frequency shift from a negative one.

**2.2** The continuous-time PSD is shown in the top half on [Figure 2B.1](#). After lowpass filtering the PSD would be  $P_y(F) = N_0/2$  over the band  $-W \leq F \leq W$  and zero otherwise. When we sample at the Nyquist rate of  $F_s = 2W$  the PSD is replicated as shown in the bottom half of the figure. Also, it is multiplied by  $F_s$  (from the sampling theorem). Finally, if we normalize the analog frequencies by dividing them by  $F_s$ , we have the periodic PSD of  $y[n]$  as shown. The basic period is from  $-W/F_s = -1/2$  to  $W/F_s = 1/2$ .



**Figure 2B.1: Power spectral densities of continuous-time WGN (top plot) and discrete-time WGN (bottom plot) after sampling at the Nyquist rate of  $F_s = 2W$ .**

**2.3** For the case where  $f_d = 0$ , you should see a single peak at  $f = 0$ . When  $f_d = 0.2$ , you will see peaks at  $-0.2$  and  $0.2$  that are clearly distinguishable from the  $f_d = 0$  case. However, when  $f_d = 0.01$ , you will see only a single peak near  $f = 0$ , and it will very difficult to distinguish this from the case of  $f_d = 0$ . The MATLAB program `FSSP3exer2_3.m`, which is contained on the CD, can be run to illustrate this solution.

**2.4** For  $f_d = 0$  the two signals are the same and therefore the deflection coefficient is zero. For  $f_d$  not near zero the deflection coefficient is about 30 or 14.8 dB. The latter value of  $d^2$  is a typical one required for good detection. The MATLAB program `FSSP3exer2_4.m`, which is contained on the CD, can be run to illustrate this solution.

**2.5** If you run `FSSP3exer2_5.m`, which is contained on the CD, but modify  $A$  to be  $\bar{A} = 0.875\sqrt{15}$ , you will attain the spec.

# Chapter 3. Mathematical Modeling of Signals

## 3.1. Introduction

As previously mentioned, we will only consider real continuous-time lowpass signals that have been sampled at the Nyquist rate. Bandpass and complex signals are described in [Chapter 12](#). In this chapter typical signal models are discussed while in [Chapter 4](#) we complete the mathematical modeling by describing noise models. It should be noted as illustrated in [Figure 3.1](#), that “One man’s signal is another man’s noise”. Thus, what we designate as a signal could equally well be considered noise. For example, the Doppler-shifted sinusoid introduced in [Chapter 2](#) was used to model a signal. But a sinusoid could also be used to represent noise, in particular, a 60 Hz interference. Hence, our delineation of signal and noise models is somewhat arbitrary. We will categorize the model as a signal model or a noise model according to its predominant usage in practice.



**Figure 3.1: Difficulty of defining a signal versus noise.**

The models to be described are simplified representations of what is observed in nature. An example is the use of damped sinusoids, also called the *all-pole model*, for speech signals [[Schafer and Rabiner 1978](#)]. Other models are the ideal representations of man-made signals. Examples include the transmitted signals that occur in a communication system such as *phase-shift keying* [[Proakis and Salehi 2007](#)] and the sinusoid as in a Doppler ultrasound. In formulating a signal model, incorporation of physical knowledge into the model will enhance its

realism, and therefore, generally lead to improved performance of any algorithm based upon the model. An example is a multipath signal, typically encountered in wireless communications, that is a sum of delayed and attenuated replicas of a transmitted signal. It can be modeled as the output of a tapped delay line, whose input is the transmitted signal.

The mathematical models for signals will be delineated according to whether they are deterministic or stochastic. The deterministic signal class contains those signals that are completely known, i.e., they have a known mathematical form, or those whose mathematical form is known except for some parameters. The random signal class, on the other hand, assumes a random process model for the signal with either a completely known probability density function (PDF) or a PDF known except for some parameters. In either case, when there are unknown parameters, we are “hedging our bets” in that we wish to acknowledge the reality that the true signal will vary somewhat in practice. For example, in the Doppler ultrasound example, we did not assume knowledge of the Doppler frequency shift. Even if the signal model cannot be completely specified, the algorithm must still perform well for all possible values of the unknown parameters. We must always keep in mind that there are two cases to be considered, one in which all parameters are known and the other in which some parameters are unknown. Each case will in general require a radically different algorithm. For the unknown parameter case the algorithm will need to be *adaptive*. Operational data then performs a dual function: It must provide enough information to not only make a signal processing decision but also to estimate the unknown parameters *on-line*. Adaptive algorithms are by definition more robust to environmental changes but can also be very tricky to determine and implement. A common example is the need to estimate and remove an underlying trend. In an electrocardiogram (ECG) there is always a *baseline* waveform that is usually of little interest [[Sörnmo and Laguna 2005](#)]. It is a slowly varying signal upon which the QRS signal, the heart beat pulse, resides. After removing the baseline waveform, the QRS signal can more easily be observed by a physician to ascertain any anomalies in the heartbeat signal. It is best to do this on-line since the baseline will change from patient to patient and also in time. Thus, it is not possible to configure an algorithm that can extract a potential heart arrhythmia with a nonadaptive algorithm.

## 3.2. The Hierarchy of Signal Models

We now begin a detailed description of the important signal model types and illustrate them with examples. Then, we will discuss some considerations in the

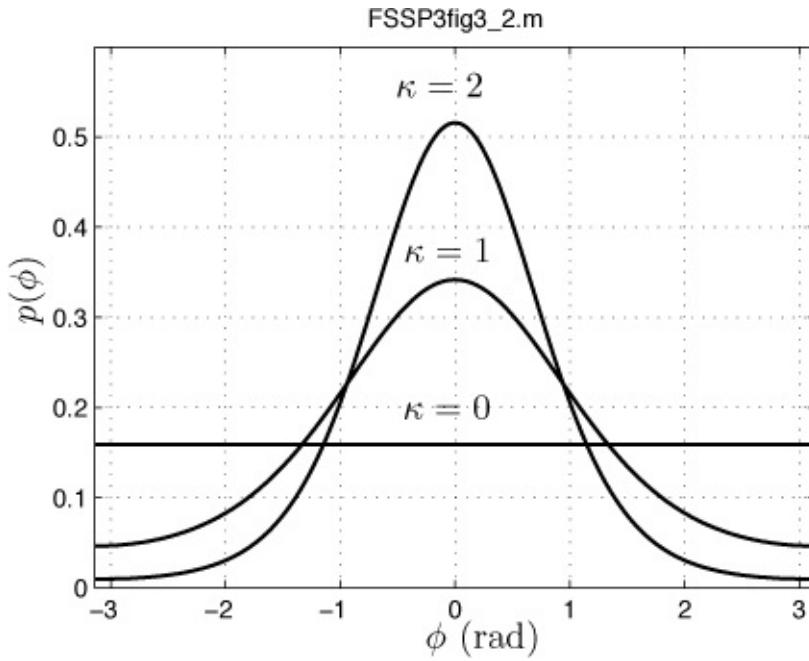
selection of a model. Referring to [Table 3.1](#), it is seen that there are four model types. For Types 1 and 3 the models are completely specified, although for Type 3 the complete specification is in terms of a *known* PDF. Type 3 modeling assumes that we do not know the phase and so have modeled it as the *outcome of a random variable* with a *known* PDF. It should be noted that a known PDF provides a complete *statistical description* of the parameter. Thus, standard techniques for designing optimal algorithms can be employed, such as Neyman-Pearson for detection and maximum a posteriori probability (MAP) classifiers for pattern recognition. For Types 2 and 4 there is an unknown parameter, which for Type 2 is the phase itself, and for Type 4 consists of one or more parameters of the PDF of the phase. An example of the latter is the use of the von Mises PDF

$$p(\phi) = \frac{1}{2\pi I_0(\kappa)} \exp[\kappa \cos(\phi)] \quad -\pi \leq \phi < \pi \quad (3.1)$$

where  $\kappa$  is an unknown concentration parameter with  $\kappa > 0$ , and  $I_0(\kappa)$  is the **modified Bessel function of order zero**. Some examples of the PDF for different values of  $\kappa$  are shown in [Figure 3.2](#). Note that if we had assumed that we knew the value of  $\kappa$ , say  $\kappa = 0$ , then the PDF of the phase would be uniform on  $(-\pi, \pi)$ , reducing a Type 4 model to a Type 3 model. If we did not wish to make this assumption, then we could use (3.1) but would have to estimate  $\kappa$  on-line. We will continue our discussion of the motivation for Type 3 signal models in [Section 3.6](#).

**Table 3.1: Different assumptions on the phase for a sinusoid signal model.**

Model type	Parameters	Example	Description of signal
1. Deterministic	known	$10 \cos[2\pi(0.1)n + \pi/4]$	completely known
2. Deterministic	unknown	$10 \cos[2\pi(0.1)n + \phi]$	$-\pi \leq \phi < \pi$
3. Random	known	$10 \cos[2\pi(0.1)n + \Phi]$	PDF of $\Phi$ known, $\Phi \sim \mathcal{U}(-\pi, \pi)$
4. Random	unknown	$10 \cos[2\pi(0.1)n + \Phi]$	PDF of $\Phi$ unknown



**Figure 3.2: PDFs for the phase of a sinusoidal model using the von Mises PDF.**

For either Types 2 or 4 we will need to design an adaptive algorithm. For example, a *phase-locked loop* is usually employed to estimate the phase on-line in a communication system [[Proakis and Salehi 2007](#)]. In general, the best model to use from a performance standpoint is Type 1. This imparts the maximum amount of knowledge to the algorithm. On the other hand, if we assume Type 1 but are in error, then very poor performance may result.

To underscore this point consider the detectability of a sinusoid using algorithms based upon Type 1 and Type 2 signal models. If the phase is modeled as deterministic and known, say  $\phi = \pi/4$ , then an optimal detector for a sinusoid in white Gaussian noise (WGN) is a *matched filter* or *replica correlator*. It decides a signal is present if [[Kay 1998](#), pg. 95] (see also [Algorithm 10.1](#))

$$T_{MF}(\mathbf{x}) = \sum_{n=0}^{N-1} x[n] \cos[2\pi(0.1)n + \pi/4] \quad (3.2)$$

$$\begin{aligned} &= \cos(\pi/4) \sum_{n=0}^{N-1} x[n] \cos[2\pi(0.1)n] \\ &\quad - \sin(\pi/4) \sum_{n=0}^{N-1} x[n] \sin[2\pi(0.1)n] > \gamma_{MF}. \end{aligned} \quad (3.3)$$

Note that the phase knowledge is used to combine the “cosine” and “sine” components of the detection statistic  $T_{MF}(\mathbf{x})$ . If this knowledge is incorrect, very poor performance may occur, as we will illustrate shortly. To avoid this degradation in performance, we can assume the phase is unknown and deterministic (Type 2 signal model). Then, we will need to implement a generalized likelihood ratio test (GLRT) that decides a signal is present if (see [Chapter 8](#))

$$T_{GLRT} = \left( \sum_{n=0}^{N-1} x[n] \cos[2\pi(0.1)n] \right)^2 + \left( \sum_{n=0}^{N-1} x[n] \sin[2\pi(0.1)n] \right)^2 > \gamma_{GLRT}. \quad (3.4)$$

The latter detector is relatively insensitive to phase in that it matches the received data to a sine and a cosine, and then adds the two values *incoherently* (as a “power sum”), as opposed to [\(3.3\)](#), in which the values are added *coherently*. It is called a *quadrature or incoherent* matched filter. It can be shown that the loss in detection performance is only about 1 dB, as measured by the energy-to-noise ratio (ENR). Thus, the lack of phase knowledge leads to a loss in performance, although in this case the loss is very small. One can then say that the GLRT is *robust* to changes in phase.

As a side note, it can also be shown that the Type 3 signal model leads to a detector with almost identical performance to the GLRT [[Brennan et al. 1968](#)]. This result appears to be very general. Algorithms that assume a parameter is deterministic and unknown and those that assume that the parameter is an outcome of a random variable with a *uniform PDF*, *will usually produce very similar performance*. The uniform PDF is required so that the parameter PDF does not favor one value of the parameter over another. This is easily done in the unknown phase case but not so easily in the case when the unknown parameter can take on values in the infinite interval  $(-\infty, \infty)$ . Then, a uniform PDF is not possible since a PDF must integrate to one. The latter can occur if the unknown parameter is the amplitude of a sinusoid.

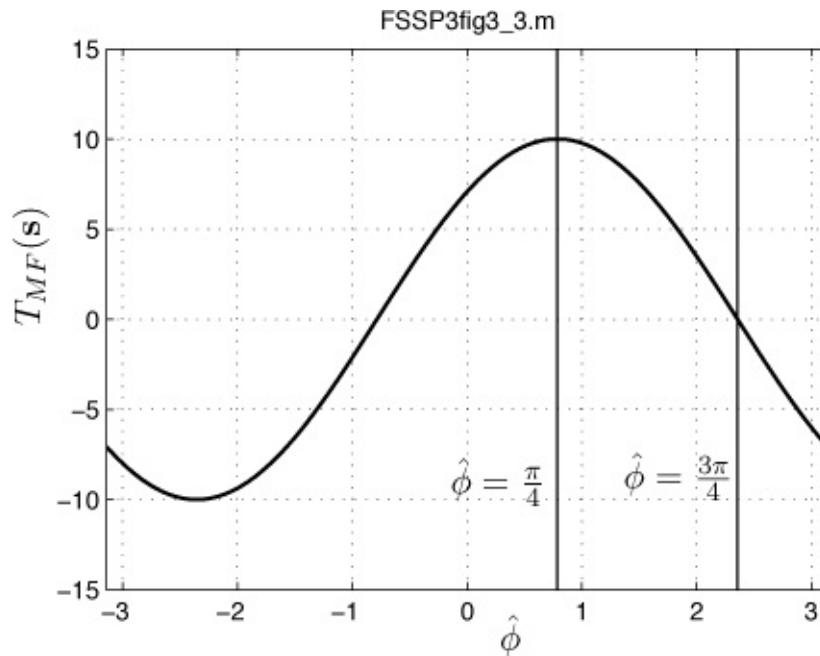
Now consider what might happen if we had assumed perfect knowledge of the phase and used the Type 1 model and consequently a matched filter, *but were wrong in our assumption*. Specifically, the true signal is  $s[n] = \cos[2\pi(0.1)n + \phi/4]$  for  $n = 0, 1, \dots, N - 1$ , where  $N = 20$ , but we assume it is

$\hat{s}[n] = \cos[2\pi(0.1)n + \hat{\phi}]$  for  $\hat{\phi} \neq \pi/4$ . To assess the performance degradation we examine the signal output of the matched filter. The signal-only output is the *correlation* between the true signal and the estimated one. From [\(3.2\)](#) this is

$$\begin{aligned}
T_{MF}(s) &= \sum_{n=0}^{N-1} s[n]\hat{s}[n] \\
&= \sum_{n=0}^{N-1} \cos[2\pi(0.1)n + \pi/4] \cos[2\pi(0.1)n + \hat{\phi}].
\end{aligned}$$

Note that if  $\hat{s}[n] = s[n]$ , i.e., there is perfect matching, then

$T_{MF}(s) = \sum_{n=0}^{N-1} s^2[n]$  and the signal output is maximized. To illustrate the degradation incurred we plot  $T_{MF}(s)$  in [Figure 3.3](#) versus the assumed value of  $\phi$ , i.e.,  $\hat{\phi}$ . Note that at  $\hat{\phi} = (3/4)\pi$ , the output is zero! A zero signal output would also occur if there were no signal present in the data and thus a noise-only decision would almost always be made. Clearly, we need to guard against this occurrence.



**Figure 3.3: Signal output of matched filter with mismatch in the phase. The true signal phase is  $\phi = \pi/4$ , while the assumed signal phase used in the matched filter is given by  $\hat{\phi}$ .**

### Exercise 3.1 – Correlation between phase-mismatched sinusoids

Prove that the correlation is zero if two sinusoids have the same frequency, which is a multiple of  $1/(2N)$ , but are mismatched in phase by  $\pi/4 = 90^\circ$ . To do so, explain the steps given below.

$$\begin{aligned}
\sum_{n=0}^{N-1} \cos(2\pi f_0 n) \cos(2\pi f_0 n + \pi/2) &= - \sum_{n=0}^{N-1} \cos(2\pi f_0 n) \sin(2\pi f_0 n) \\
&= -\frac{1}{2} \sum_{n=0}^{N-1} \sin(4\pi f_0 n) \\
&= -\frac{1}{2} \text{Imag} \left( \sum_{n=0}^{N-1} \exp(j4\pi f_0 n) \right).
\end{aligned}$$

Then, let  $N = 4$  and  $f_0 = 1/8$  to yield

$$-\frac{1}{2} \text{Imag} \left( \sum_{n=0}^3 \exp(jn\pi/2) \right)$$

and show that the complex sum is zero by drawing “phasors” at the appropriate locations.



The same type of poor performance can occur if we assume that the signal phase is the outcome of a random variable and assign to it a PDF that is inaccurate. For example, we might design a detector using the von Mises PDF for the phase, assuming a particular value of  $\kappa$ . If in practice our assumed value of  $\kappa$  is much different than the actual value, poor performance can be expected.

In summary, there are several points that an algorithm designer should keep in mind:

1. Signal models should reflect as much physical realism as possible but not be so complicated that development of algorithms becomes difficult.
2. Mathematical signal models can be either deterministic and completely known or deterministic with unknown parameters. They can also be random processes with completely known PDFs or with PDFs having unknown parameters.
3. Signal models that must estimate parameters on-line tend to be called adaptive algorithms.
4. Assuming that the signal is completely known can be risky if in practice the environment changes in time and/or the conditions in the field do not match those in the lab. In such a case, it is advisable to assume some parameters are unknown and estimate them on-line. The performance may be degraded somewhat, but the overall algorithm will perform much more uniformly, i.e., it will be more *robust*.

### 3.3. Linear vs. Nonlinear Deterministic Signal Models

Before describing specific models for deterministic signals, it is important to differentiate between **linear** and **nonlinear** models. A Type 1 signal model, that of a known **deterministic signal**, is an algorithm designer's dream.

Unfortunately, in practice we almost never know the signal completely and thus must move on to Type 2. Hence, assuming a deterministic signal with unknown parameters, there are three **subcases** that are encountered. These are the *linear model*, the *partially linear model*, and the *nonlinear model*. The first one is the easiest to work with and therefore forms *the basis for nearly all practical algorithms*. To contrast the signal parameter dependencies, we again employ a **sinusoid model** and its **equivalent form** given as (much the same as was done to obtain (1.3))

$$\begin{aligned}s[n] &= A \cos(2\pi f_0 n + \phi) \\&= \underline{A \cos(\phi)} \cos(2\pi f_0 n) - \underline{A \sin(\phi)} \sin(2\pi f_0 n) \\&= \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n).\end{aligned}\tag{3.5}$$

If the frequency is known, say  $f_0 = 0.1$ , but the amplitudes  $\alpha_1, \alpha_2$  are unknown, then

$$s[n] = \alpha_1 \cos(2\pi(0.1)n) + \alpha_2 \sin(2\pi(0.1)n)$$

and the signal is known except for the amplitudes  $\alpha_1, \alpha_2$ . This is an example of the *linear model*. In fact, it can be written in the slightly more general form as

$$s[n] = \alpha_1 h_1[n] + \alpha_2 h_2[n]$$

where  $h_1[n], h_2[n]$  are **known basis signals**. We term this a linear model since  $s[n]$  is a linear function of the *unknown parameters*  $\alpha_1$  and  $\alpha_2$ . It does not matter what form the basis signals take, only that they are known. Another linear model would be  $s[n] = \alpha_1 \exp(n) + \alpha_2 \exp(2n)$ . The importance of the linear model is that ultimately we will need to estimate  $\alpha_1, \alpha_2$ , which we term the amplitudes of the basis signals, and *the linear dependence assures us that we can find an accurate and easily implementable estimator*.

Another possibility in considering our state of knowledge about the signal in (3.5) is that the amplitudes might be **known** but the frequency is unknown. Then, the model is a *nonlinear* one as illustrated by the next exercise.

---

## Exercise 3.2 – Nonlinear dependence of sinusoid on frequency

Consider  $s[n] = g_n(f_0) = \cos(2\pi f_0 n)$ . If  $s_1[n] = g_n(0.1) = \cos(2\pi(0.1)n)$  and  $s_2[n] = g_n(-0.1) = \cos(2\pi(-0.1)n)$ , is  $g_n(a+b) = g_n(a) + g_n(b)$  for  $a = 0.1, b = -0.1$ ?

---

The last possibility is that all the parameters  $\alpha_1, \alpha_2, f_0$  are unknown. Then, the signal is **linear in the amplitudes** but **nonlinear in the frequency**. This is called a *partially linear model*. Another partially linear model is  $s[n] = Ar^n$ , where both  $A$  and  $r$  are unknown. If  $r$  were known and  $A$  unknown, then it would be a linear model. If  $A$  were known but  $r$  were unknown, it would be a nonlinear model. We next describe deterministic signal models of practical interest.

unknow做系数，并且系数跟的是一个复杂的函数，那么对于信号来说，估计这个系数是线性的；但是如果known是系数，而系数后面跟的函数是unknown，那么对于这个信号来说，估计这个函数是非线性的。

## 3.4. Deterministic Signals with Known Parameters (Type 1)

In this section we assume that all parameters are known, with a discussion of the modifications required when they are unknown (being Type 2 signals) to be given in the following section.

### 3.4.1. Sinusoids

As previously described a sinusoid signal is modeled as

$$s[n] = A \cos(2\pi f_0 n + \varphi) \quad n = 0, 1, \dots, N-1$$

where the amplitude  $A$  takes on the values  $A > 0$ , the frequency  $f_0$  takes on values  $0 < f_0 < 1/2$ , and the phase has values  $-\pi \leq \varphi < \pi$ . Note that the amplitude is constrained to be positive to avoid the problem of *nonidentifiability*, as illustrated in [Exercise 3.3](#). This problem arises when the signal can take on exactly the same sample values for two different sets of amplitudes and phases.

---

## Exercise 3.3 – Identifiability of sinusoid

Show that if  $A_1 = 1, \varphi_1 = \pi/4$  and  $A_2 = -1, \varphi_2 = (5/4)\pi$ , then  $s_1[n] = s_2[n]$ .

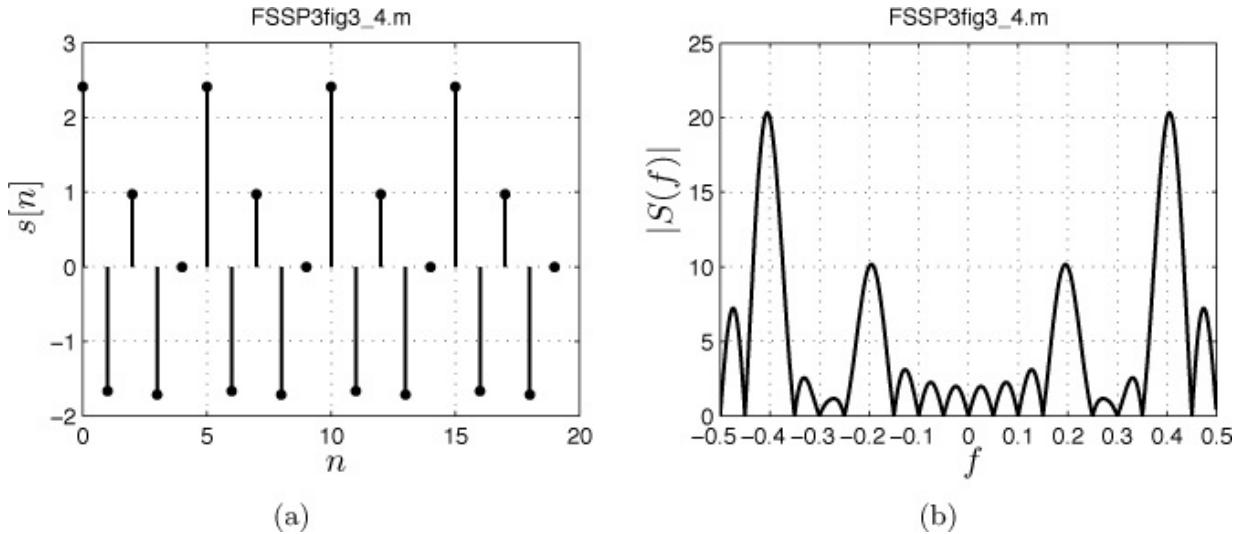
---

Also, we exclude the value of zero for frequency since then  $s[n] = A \cos(\varphi)$ , a constant sequence or a DC level. This signal is included under polynomials.

For multiple sinusoids the model is

$$s[n] = \sum_{i=1}^p A_i \cos(2\pi f_i n + \phi_i) \quad n = 0, 1, \dots, N - 1 \quad (3.6)$$

with the same restrictions on the parameters. An application of the model is in the modeling of radar interferers [[Miller et al. 1997](#)]. As an example, the signal and the magnitude of its Fourier transform, sometimes called the *spectrum*, are shown in [Figure 3.4](#) for the parameters given in the figure.



**Figure 3.4: Signal samples and the spectrum for the sum of two sinusoids with  $(A_1, f_1, \phi_1) = (1, 0.2, 0)$  and  $(A_2, f_2, \phi_2) = (2, 0.4, \pi/4)$  and  $N = 20$ .**

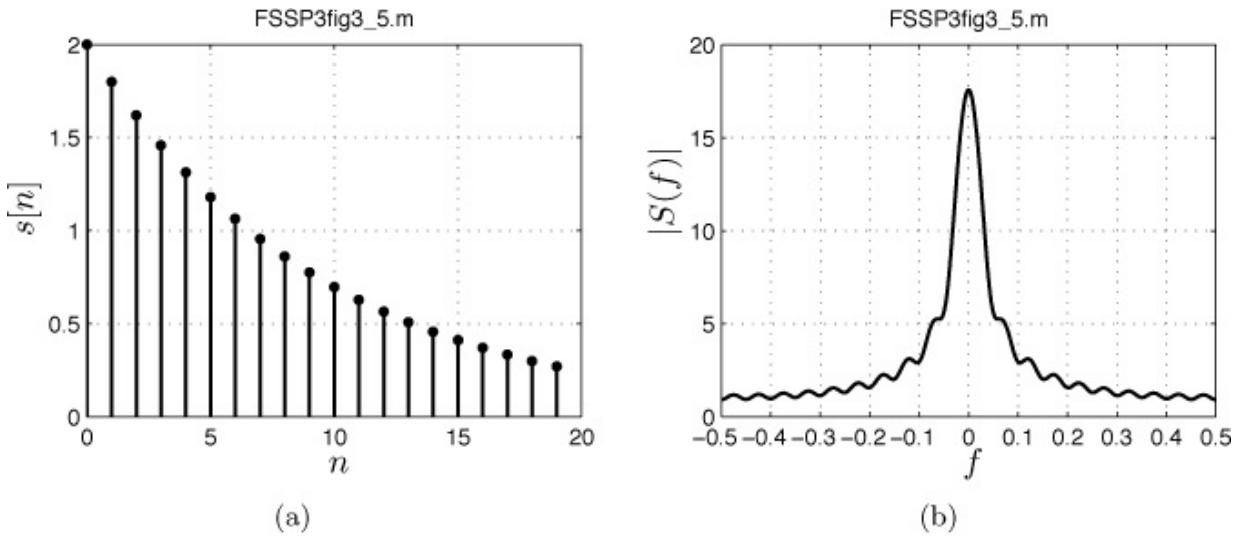
The MATLAB program `sinusoid_gen.m`, which is included on the CD, can be used to compute the signal samples and the spectrum.

### 3.4.2. Damped Exponentials

A damped exponential signal model is given as

$$s[n] = Ar^n \quad n = 0, 1, \dots, N - 1 \quad (3.7)$$

where  $-\infty < A < \infty$  and  $-1 < r < 1$ . As an example, the signal and the spectrum are shown in [Figure 3.5](#) for the parameters given in the figure.



**Figure 3.5: Signal samples and the spectrum for a damped exponential with  $A = 2$ ,  $r = 0.9$ , and  $N = 20$ .**

For multiple damped exponentials the signal is given by

$$s[n] = \sum_{i=1}^p A_i r_i^n \quad n = 0, 1, \dots, N - 1 \quad (3.8)$$

with the same restrictions on the parameters. This signal model is used more frequently when it is generalized to the damped sinusoid model as described next. The MATLAB program `sinusoid_gen.m` can be used to compute the signal samples and the spectrum.

### 3.4.3. Damped Sinusoids

Combining the two previous models, results in the damped sinusoid model given in general as

$$s[n] = \sum_{i=1}^p A_i r_i^n \cos(2\pi f_i n + \phi_i) \quad n = 0, 1, \dots, N - 1 \quad (3.9)$$

This model is used heavily in vibration analysis [[McConnell 1995](#)], in spectroscopy [[Vanhamme et al. 2001](#)], and in many other fields. The MATLAB program entitled `sinusoid_gen.m`, which is included on the CD, can be used to compute the signal samples and the spectrum.

### 3.4.4. Phase Modulated Signals

A phase-modulated signal is given as

$$s[n] = A \cos(\beta[n] + \varphi) \quad n = 0, 1, \dots, N - 1$$

where the time-varying phase  $\beta[n]$  can take on a variety of forms. As before, the amplitude must be positive for the signal to be identifiable. For example, the sinusoid is a special case in which  $\beta[n] = 2\pi f_0 n$ . More generally  $\beta[n]$  can be a polynomial in  $n$ , leading to *polynomial phase signals*. The most important one in practice is the *linear FM*, for which  $\beta[n] = 2\pi(f_0 n + \frac{1}{2}m n^2)$ . The signal becomes

$$s[n] = A \cos[2\pi(f_0 n + \frac{1}{2}m n^2) + \phi] \quad n = 0, 1, \dots, N - 1. \quad (3.10)$$

The phase is quadratic in  $n$  so that the first difference, which yields the frequency, is linear with  $n$ . Hence, it is called a linear FM (LFM) or sometimes a *chirp*. Usually, the LFM of (3.10) is obtained by sampling the continuous-time signal

$$s(t) = A \cos[2\pi(F_0 t + \frac{1}{2}m_c t^2) + \phi] \quad 0 \leq t \leq T$$

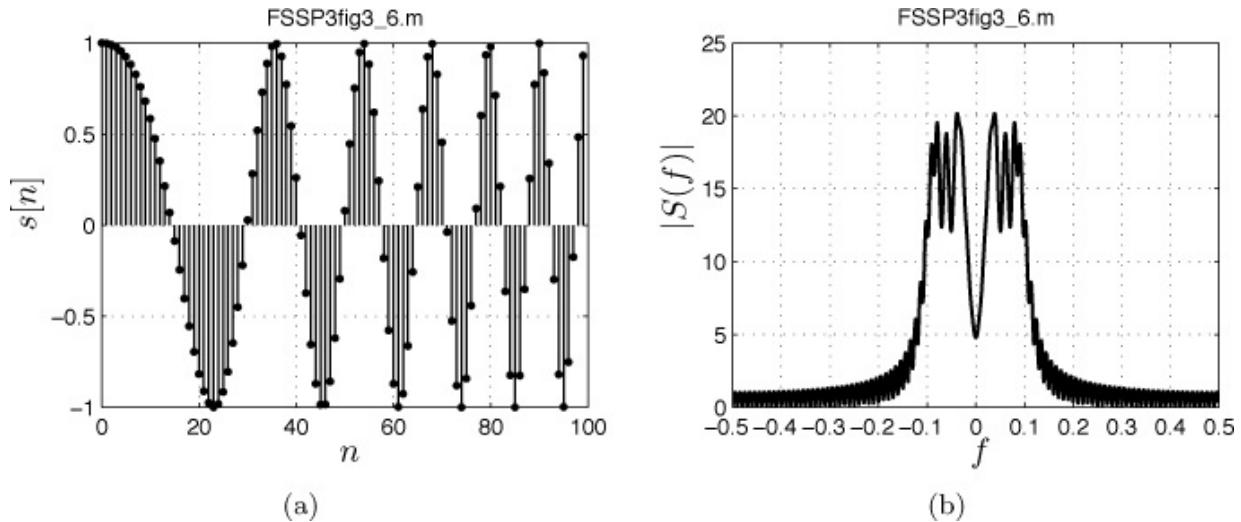
whose instantaneous frequency, obtained by differentiating the phase with respect to  $t$ , is given by

$$F_i(t) = F_0 + m_c t.$$

It is seen that the *sweep rate*, which is the rate of change in Hz per sec, is given by  $m_c$ . For the signal to reside in the digital band of  $[-1/2, 1/2]$ , we must have

$$0 \leq F_0 + m_c t \leq F_s/2 \quad 0 \leq t \leq T$$

where  $F_s$  is the sampling rate in samples/sec. As an example, the signal and the spectrum are shown in [Figure 3.6](#) for the parameters given in the figure.



**Figure 3.6: Signal samples and spectrum for a linear FM with  $A = 1$ ,  $f_0 = 0.01$ ,  $\phi = 0$ ,  $m = 0.1/(N - 1)$ , and  $N = 100$ .**

Note that the spectrum indicates energy over the frequency band  $[f_0, f_0 + m(N - 1)] = [0.01, 0.11]$ . This type of signal is widely used in sonar [[Burdick 1984](#)] and radar [[Cook and Bernfeld 1993](#)].

### 3.4.5. Polynomial Signal

The simplest types of polynomial signals are the *DC level* given by

$$s[n] = A \quad n = 0, 1, \dots, N - 1$$

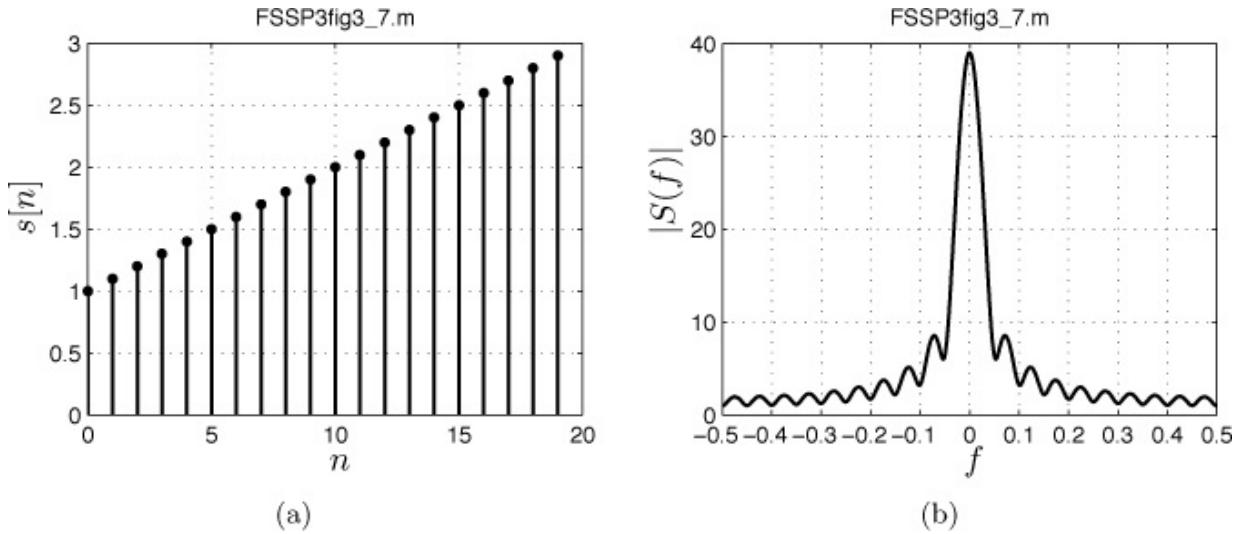
where  $-\infty < A < \infty$  and the *line* signal given by

$$s[n] = A + Bn \quad n = 0, 1, \dots, N - 1$$

where  $-\infty < A < \infty$  and  $-\infty < B < \infty$ . In addition to modeling signals, these models are frequently used to represent trends in the data [[Granger and Newbold 1986](#)]. More generally, a polynomial signal is given as

$$s[n] = \sum_{i=0}^{p-1} c_i n^i \quad n = 0, 1, \dots, N - 1 \quad (3.11)$$

where the parameters can take on any real values. The spectrum of the signal varies greatly with the values of the parameters. For a line signal, in which  $s[n] = A + Bn$ , most of its energy is at low frequencies. As an example, the signal and its spectrum are shown in [Figure 3.7](#) for the parameters given in the figure.



**Figure 3.7: Signal samples and spectrum for a line signal with  $A = 1$ ,  $B = 1$ .**

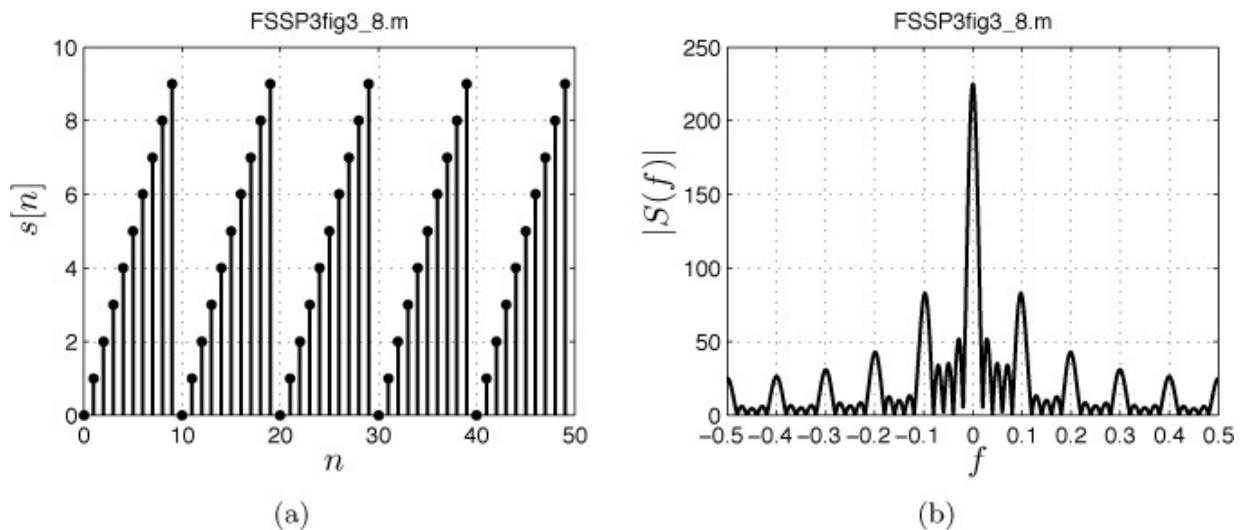
**0.1, and  $N = 20$ .**

### 3.4.6. Periodic Signal

A periodic signal is one for which the signal samples repeat after every  $M$  samples. An example is shown in [Figure 3.8](#) along with the spectrum. The periodic signal with a period of  $M$  samples is given as

$$\begin{aligned} s[n] &= g[n] & n = 0, 1, \dots, M-1 \\ &= g[n-M] & n = M, M+1, \dots, 2M-1 \\ &= g[n-2M] & n = 2M, 2M+1, \dots, 3M-1 \\ &\vdots \\ &= g[n-(K-1)M] & n = (K-1)M, (K-1)M+1, \dots, N-1 \end{aligned}$$

if there are  $K-1$  full periods and part of one period. If, however,  $N = KM$ , then there will be exactly  $K$  full periods. Here  $g[n]$  for  $n = 0, 1, \dots, M-1$  is the *basic period*. In [Figure 3.8](#) an example is shown of a periodic signal with an *integral* number of periods since  $K = N/M = 5$ .



**Figure 3.8: Signal samples and spectrum for a periodic signal with period  $M = 10$ .**

Periodic signals are routinely encountered in economics [[Granger and Newbold 1986](#)], vibration analysis [[McConnell 1995](#)], and biomedicine [[Sörnmo and Laguna 2005](#)]. A real-world example is given in [Chapter 14](#) for the modeling of a magnetic signal. It is of interest to observe that a periodic signal can also be represented using a Fourier series. Using this representation the frequencies would be given as  $f_k = 0, 1/M, 2/M, \dots, 1/2$  for  $M$  even. It is seen from

[Figure 3.8](#) that the spectrum does indeed have peaks at the harmonic frequencies of  $f_k = 0, 1/10, 2/10, \dots, 5/10$ . The fact that the spectrum does not exhibit Dirac impulses is due to the finite signal length of  $N = 50$  samples, creating a “smearing” effect in the spectrum. Instead of the theoretical impulses we observe “sin x/x” type energy lobes, whose 3 dB bandwidth is approximately  $1/N = 0.02$  cycles/sample.

## 3.5. Deterministic Signals with Unknown Parameters (Type 2)

### 3.5.1. General Considerations

In modeling signals containing unknown parameters, it is necessary to be able to estimate those parameters based on field data. The estimated parameters are then inserted back into the theoretical expressions to obtain an *estimated signal*. Then, a comparison between the field data signal and the modeled, i.e., estimated signal, can be made to determine how well the model matches the actual signal. For the signal models of the previous section there are linear and nonlinear parameters. The [polynomial signal](#) and the [periodic signal models](#) contain only linear parameters, which lead to an *easy problem*. We will begin our discussion of the estimation of linear parameters in this chapter. Further details pertaining to signal model selection are discussed in [Chapter 5](#). Note also that *the final algorithm must contain a means to estimate these parameters online, being unknown a priori*. Therefore, the [estimation methods](#) to be described to [ascertain modeling accuracy](#), based on field test data, will also be integral to our proposed algorithm, based on operational data.

### 3.5.2. Polynomial Signal Model

Consider first a special case of the polynomial signal, the line signal. It is given as  $s[n] = A + Bn$  for  $n = 0, 1, \dots, N - 1$ , where the unknown parameters are  $A$  and  $B$ . There are no restrictions on the values of these parameters. Assuming actual data  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is available, then the parameters are easily found using a *least squares* approach. It determines  $A$  and  $B$  by finding the best match between the data and the signal model in the least squares sense. Specifically, we choose the parameter values as those that minimize the least squares error criterion

$$J(A, B) = \sum_{n=0}^{N-1} (x[n] - (A + Bn))^2. \quad (3.12)$$

Note that  $x[n]$  represents the *known values* of the data and so  $J$  is only a function of  $A$  and  $B$ . Since  $J$  is quadratic in  $A$  and  $B$ , the values that minimize  $J$  are easily

found. Just differentiate  $J$  by taking the partial derivatives of  $A$  and  $B$  and set them equal to zero.

---

### Exercise 3.4 – Finding the minimum for $A$

As an even easier case, assume that  $B = 0$  in (3.12) so that

$J(A) = \sum_{n=0}^{N-1} (x[n] - A)^2$ . Find the minimum of  $J$  by taking the derivative with respect to  $A$  and set it equal to zero. Call the value that minimizes  $J$ ,  $\hat{A}$ . What is  $\hat{A}$ , and does  $\hat{s}[n] = \hat{A}$  seem reasonable as an estimate of the signal?

---

After differentiation of (3.12) and setting the partial derivatives equal to zero, we obtain the two equations

$$\begin{aligned}\sum_{n=0}^{N-1} (x[n] - A - Bn) &= 0 \\ \sum_{n=0}^{N-1} (x[n] - A - Bn)n &= 0\end{aligned}$$

or equivalently

$$\begin{aligned}\left(\sum_{n=0}^{N-1} 1\right) A + \left(\sum_{n=0}^{N-1} n\right) B &= \sum_{n=0}^{N-1} x[n] \\ \left(\sum_{n=0}^{N-1} n\right) A + \left(\sum_{n=0}^{N-1} n^2\right) B &= \sum_{n=0}^{N-1} nx[n]\end{aligned}$$

or finally in matrix/vector form

$$\begin{bmatrix} \sum_{n=0}^{N-1} 1 & \sum_{n=0}^{N-1} n \\ \sum_{n=0}^{N-1} n & \sum_{n=0}^{N-1} n^2 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum_{n=0}^{N-1} x[n] \\ \sum_{n=0}^{N-1} nx[n] \end{bmatrix}. \quad (3.13)$$

When solved, these *linear equations* produce the *least squares estimates* of  $A$  and  $B$ . The reason that the equations are linear and hence easily solved by any number of standard numerical techniques, follows from the property that  $J$  is quadratic in  $A$  and  $B$ . As mentioned in [Chapter 2](#) this is an *easy problem*. It is also an example of the *linear signal model*.

The equations given by (3.13) can also be written as

$$\mathbf{H}^T \mathbf{H} \boldsymbol{\theta} = \mathbf{H}^T \mathbf{x} \quad (3.14)$$

where

$$\mathbf{x} = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & N-1 \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} A \\ B \end{bmatrix} \quad (3.15)$$

and the matrix  $\mathbf{H}$  has dimension  $N \times 2$  and  $T$  denotes a transpose. The least squares estimator for the unknown parameters becomes

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} \hat{A} \\ \hat{B} \end{bmatrix} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \quad (3.16)$$

where  $-1$  denotes the matrix inverse of the  $2 \times 2$  matrix  $\mathbf{H}^T \mathbf{H}$ . The equations (3.14) and (3.16) generalize to any number of unknown parameters, with an example to be given shortly.

---

### Exercise 3.5 – Numerical example of matrix and vectors encountered in least squares estimator

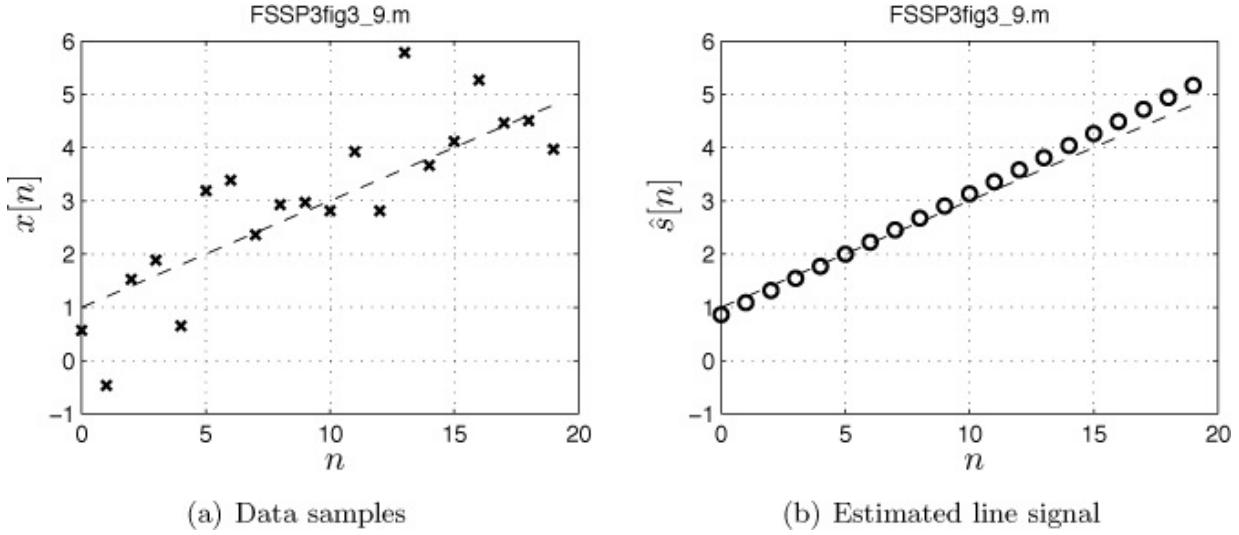
For the  $\mathbf{H}$  given in (3.15) show that for  $N = 3$ ,  $\mathbf{H}^T \mathbf{H}$  and  $\mathbf{H}^T \mathbf{x}$  are given by

$$\mathbf{H}^T \mathbf{H} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix} \quad \mathbf{H}^T \mathbf{x} = \begin{bmatrix} x[0] + x[1] + x[2] \\ x[1] + 2x[2] \end{bmatrix}.$$

Do these results agree with (3.13)?

---

An example of the use of (3.16) to estimate  $A$  and  $B$  is shown in [Figure 3.9](#). The data, indicated by x's along with the true signal  $s[n] = 1 + 0.2n$ , shown dashed with the signal samples connected by the dashed line, are shown in [Figure 3.9a](#). The estimated signal  $\hat{s}[n] = \hat{A} + \hat{B}n$ , shown by the o's, along with the true signal are shown in [Figure 3.9b](#).



**Figure 3.9: Least squares estimate of a noisy line signal. The true line signal ( $s[n] = 1 + 0.2n$ ) is shown as the dashed line, its points having been connected for easier viewing. The estimated line signal is  $\hat{s}[n] = 0.8648 + 0.2265n$  and is shown by the o's.**

To appreciate the generality of this approach for the case of *linear parameters*, we note that if  $\mathbf{s} = [s[0] \ s[1] \ \dots \ s[N - 1]]^T$  denotes the vector of signal samples, where  $s[n] = A + Bn$ , it can be written in the succinct form as

$$\mathbf{s} = \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & N - 1 \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} A \\ B \end{bmatrix}}_{\boldsymbol{\theta}}$$

where  $\mathbf{H}$  is a *known* matrix and  $\boldsymbol{\theta}$  is the vector of parameters to be estimated. In general, if we can write the signal samples as the vector  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ , where  $\mathbf{s}$  is an  $N \times 1$  vector,  $\mathbf{H}$  is a *known*  $N \times p$  matrix with  $p < N$ , and  $\boldsymbol{\theta}$  is  $p \times 1$  parameter vector, then the signal conforms to what is termed the *linear model*. Not only is the vector of unknown parameters  $\boldsymbol{\theta}$  easy to estimate (it is given explicitly as  $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$ , as was shown by (3.13) and (3.14)), and under certain conditions the estimator is optimal, but there is also a wealth of knowledge concerning the statistical performance of the estimator [Graybill 1976].

As a more general example of the linear model consider a polynomial signal. The signal model as given by (3.11) can be written in the linear model form as  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$  where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0^2 & \dots & 0^{p-1} \\ 1 & 1 & 1^2 & \dots & 1^{p-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & N-1 & (N-1)^2 & \dots & (N-1)^{p-1} \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{p-1} \end{bmatrix}.$$

Finally, to account for noise perturbations of the signal, we usually complete the data description of the linear model by adding to  $\mathbf{s}$  a noise vector  $\mathbf{w} = [w[0] \ w[1] \ \dots \ w[N-1]]^T$  so that the observed data model becomes

$$\mathbf{x} = \mathbf{H}\boldsymbol{\theta} + \mathbf{w}. \quad (3.17)$$

Typically, the noise samples are assumed to be samples of WGN, meaning that the samples are all independent Gaussian random variables with zero mean and the same variance  $\sigma^2$ . In such a case, the least squares estimator of (3.16) becomes the *maximum likelihood estimator*, which is known to be an optimal estimator in terms of estimation accuracy for this model. More details about this important property are provided in [Section 8.3](#).

### 3.5.3. Periodic Signal Model

The other linear signal model is the periodic signal *when* the signal samples within the basic period are unknown. (See also [Algorithm 9.11](#) for a random outcome modeling of a basic period.) These unknown signal samples, which we will denote by  $\{g[0], g[1], \dots, g[M-1]\}$ , can be viewed as unknown parameters. As an example, if the period is  $M = 3$  and we have a total of  $N = 6$  signal samples, then

$$\mathbf{s} = \begin{bmatrix} g[0] \\ g[1] \\ g[2] \\ g[0] \\ g[1] \\ g[2] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} g[0] \\ g[1] \\ g[2] \end{bmatrix}}_{\boldsymbol{\theta}}$$

and therefore we have immediately the linear signal model and the least squares estimator  $\hat{\boldsymbol{\theta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$ . Once we realize that  $\mathbf{s}$  can be written as  $\mathbf{H}\boldsymbol{\theta}$ , then the parameter estimation problem is solved! Also, the estimated signal, obtained by replacing the unknown parameter vector  $\boldsymbol{\theta}$  by  $\hat{\boldsymbol{\theta}}$ , is given by

$$\hat{\mathbf{s}} = \mathbf{H}\hat{\boldsymbol{\theta}} = \underbrace{\mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T}_{\mathbf{P}} \mathbf{x}. \quad (3.18)$$

The matrix  $\mathbf{P}$ , which is  $N \times N$ , and which converts the data vector  $\mathbf{x}$  into the

estimated signal vector  $\hat{\mathbf{s}}$  is called the *projection matrix* [Kay 1993, pg. 231]. To interpret its effect on the data it is easiest to use a computer evaluation of  $\mathbf{P}$  for a specific example. For the current example of  $N = 6$  we obtain the  $6 \times 6$  matrix

$$\mathbf{P} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

so that from (3.18)

$$\hat{\mathbf{s}} = \begin{bmatrix} \hat{s}[0] \\ \hat{s}[1] \\ \hat{s}[2] \\ \hat{s}[3] \\ \hat{s}[4] \\ \hat{s}[5] \end{bmatrix} = \mathbf{P}\mathbf{x} = \begin{bmatrix} \frac{1}{2}(x[0] + x[3]) \\ \frac{1}{2}(x[1] + x[4]) \\ \frac{1}{2}(x[2] + x[5]) \\ \frac{1}{2}(x[0] + x[3]) \\ \frac{1}{2}(x[1] + x[4]) \\ \frac{1}{2}(x[2] + x[5]) \end{bmatrix}. \quad (3.19)$$

The signal for the basic period  $s[0], s[1], s[2]$  is estimated by taking the first three data samples and averaging them with the last three data samples. Because the signal is assumed to be periodic this averaging does not affect the signal but does tend to average out the noise. For example,

$$\begin{aligned} \hat{s}[0] &= \frac{1}{2}(x[0] + x[3]) = \frac{1}{2}(s[0] + w[0] + s[3] + w[3]) \\ &= \frac{1}{2}(g[0] + w[0] + g[0] + w[3]) = g[0] + \frac{1}{2}(w[0] + w[3]) \end{aligned}$$

so that the noise effects are reduced.

---

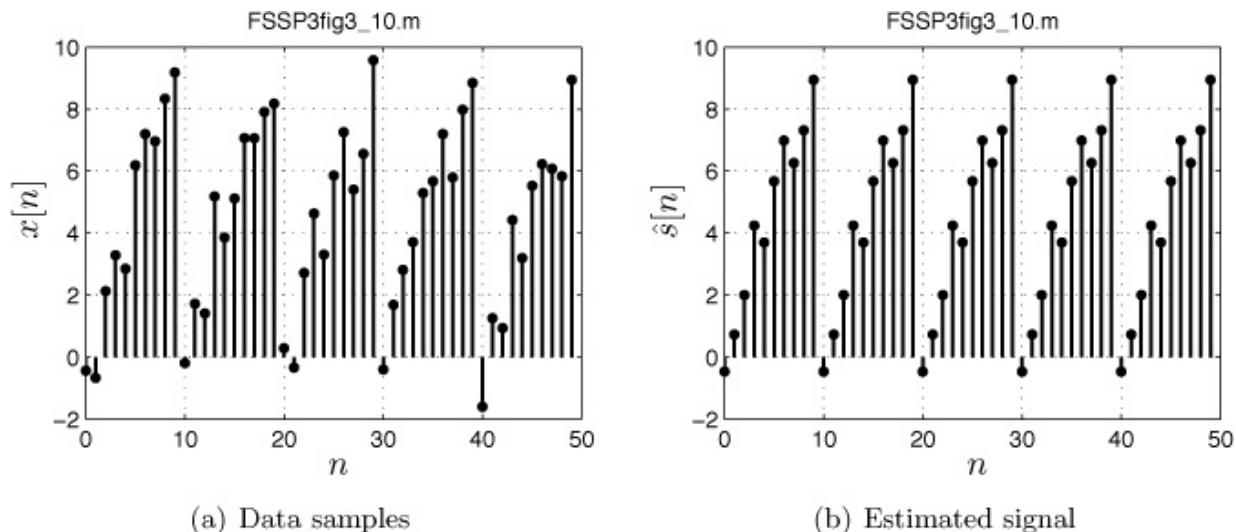
### Exercise 3.6 – Reduction of noise due to averaging

Assume that  $w[0]$  and  $w[3]$  are both zero mean and uncorrelated random variables. If each one has a variance of one, show that the variance of the average, i.e., the variance of  $\bar{w} = (w[0] + w[3])/2$ , is only 1/2. Do so by computing the expected value of  $[\frac{1}{2}(w[0] + w[3])]^2$ , which is the variance of  $\bar{w}$  since  $\bar{w}$  has a zero mean. Hint: Recall that if two random variables  $X$

and  $Y$  are uncorrelated, then  $E[XY] = E[X]E[Y]$ .

---

An example of a noise corrupted periodic signal is given in [Figure 3.10a](#) for the periodic signal previously shown in [Figure 3.8](#). White Gaussian noise of variance  $\sigma^2 = 1$  has been added to the signal to form the data. The estimated signal obtained using (3.18) is shown in [Figure 3.10b](#). Note that the estimated signal is periodic in accordance with its known form (see also (3.19) for another example).



**Figure 3.10: A noisy periodic signal and its estimated signal using the linear model least squares parameter estimator.**

It is seen that the noise effects have been mitigated somewhat. For better noise reduction we require more averaging so that the data record length  $N$  needs to be increased.

---

### Exercise 3.7 – Increasing data record for more noise reduction.

Run the program `FSSP3exer3_7.m`, which is contained on the CD, for  $N = 50$  to yield the signal sample estimates shown in [Figure 3.10b](#). Then, increase  $N$  to  $N = 1000$  and  $K$  to  $K = 100$ , and compare the results to the true signal, which is  $s[n] = n$  for  $n = 0, 1, \dots, 9$ .

---

The operation of averaging data over blocks of length equal to one period to recover the basic signal is called *comb filtering* [[Wise et al. 1976](#)]. This

nomenclature comes from the observation that in doing this type of averaging we are essentially filtering the data with a bank of narrowband filters with center frequencies located at the harmonic frequencies. Referring to [Figure 3.8b](#) we see that the spectrum has energy at  $f_k = 0, 1/10, 2/10, 3/10, 4/10, 5/10$ . The comb filtering operation attempts to pass the signal but block the noise that resides between the harmonic frequencies.

### 3.5.4. Nonlinear and Partially Linear Signals

The remaining signal types, the sinusoid, damped exponential, damped sinusoid, and phase modulated signal are also linear if the only unknown parameters are the amplitudes and phases. As shown previously, each amplitude  $A$  and phase  $\phi$  can be converted into two linear signal amplitudes  $\alpha_1$  and  $\alpha_2$ . Otherwise, the signal models are only partially linear. This is summarized in [Table 3.2](#). To illustrate the difficulty in estimating the parameters of a partially linear model consider the sinusoid model in [Table 3.2](#). Then, to estimate  $\{A_i, f_i, \phi_i\}$  or equivalently  $\{\alpha_{1_i}, \alpha_{2_i}, f_i\}$  for  $i = 1, 2, \dots, p$  we must minimize

**Table 3.2: Different assumptions on the parameters for a deterministic signal.**

Signal model	Mathematical form	Parameters	Linear parameters*
Sinusoids	$\sum_{i=1}^p A_i \cos(2\pi f_i n + \phi_i)$	$A_i, f_i, \phi_i$	$A_i, \phi_i$
Damped exponentials	$\sum_{i=1}^p A_i r_i^n$	$A_i, r_i$	$A_i$
Damped sinusoids	$\sum_{i=1}^p A_i r_i^n \cos(2\pi f_i n + \phi_i)$	$A_i, r_i, f_i, \phi_i$	$A_i, \phi_i$
Phase modulated	$\sum_{i=1}^p A_i \cos[2\pi(f_i n + \frac{1}{2}m_i n^2) + \phi_i]$	$A_i, f_i, m_i, \phi_i$	$A_i, \phi_i$
Polynomial	$\sum_{i=0}^{p-1} c_i n^i$	$c_i$	$c_i$
Periodic	$g[0], \dots, g[M-1], g[0], \dots, g[M-1], \dots$	$g[i]$	$g[i]$

\*It is assumed that  $(A_i, \phi_i)$  is transformed to  $(\alpha_{1_i}, \alpha_{2_i})$ , which are linear parameters.

$$J(\alpha_{1_i}, \alpha_{2_i}, f_i; i = 1, 2, \dots, p) = \sum_{n=0}^{N-1} \left( x[n] - \sum_{i=1}^p [\alpha_{1_i} \cos(2\pi f_i n) + \alpha_{2_i} \sin(2\pi f_i n)] \right)^2. \quad (3.20)$$

This is a *hard problem* and becomes nearly impossible for  $p$  much greater than  $p = 3$ . (See [Algorithm 9.10](#) for a good suboptimal estimator.) However, for  $p$  small a computational solution is possible. Consider  $p = 1$  so that the signal is

$$s[n] = \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n)$$

and can be written in the usual matrix/vector form as

$$\mathbf{s} = \underbrace{\begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \sin[2\pi f_0(N-1)] \end{bmatrix}}_{\mathbf{H}(f_0)} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}}_{\boldsymbol{\theta}}. \quad (3.21)$$

This appears to be in the linear model form except that  $\mathbf{H}$  is no longer a known matrix since it depends upon the unknown parameter  $f_0$ . This complicates the minimization. However, it can be shown that the minimization can be carried out by performing a numerical maximization. Specifically, we first numerically maximize the function

$$J(f_0) = \mathbf{x}^T \mathbf{H}(f_0) (\mathbf{H}^T(f_0) \mathbf{H}(f_0))^{-1} \mathbf{H}^T(f_0) \mathbf{x} \quad (3.22)$$

over  $0 < f_0 < 1/2$  to yield the maximizing value  $\hat{f}_0$ . Then, treating  $\mathbf{H}(\hat{f}_0)$  as a known matrix, the remaining parameters are given by the least squares estimator

$$\begin{bmatrix} \hat{\alpha}_1 \\ \hat{\alpha}_2 \end{bmatrix} = \left( \mathbf{H}^T(\hat{f}_0) \mathbf{H}(\hat{f}_0) \right)^{-1} \mathbf{H}^T(\hat{f}_0) \mathbf{x}.$$

The amplitude and phase estimates can now be found from

$$\begin{aligned} \hat{A} &= \sqrt{\hat{\alpha}_1^2 + \hat{\alpha}_2^2} \\ \hat{\phi} &= \arctan \left( \frac{-\hat{\alpha}_2}{\hat{\alpha}_1} \right). \end{aligned}$$

More generally for  $p$  sinusoids the numerical maximization must be carried out over the  $p$ -dimensional space for  $\{f_1, f_2, \dots, f_p\}$ , which is computationally demanding. (See [Algorithm 9.9](#) for an example when  $p = 2$ .) Similar considerations apply to the other partially linear models given in [Table 3.2](#).

---

## Exercise 3.8 – Numerical maximization

Assume that the data is noiseless so that  $x[n] = \cos(2\pi(0.12)n + \pi/4)$  for  $n = 0, 1, \dots, N-1$ , where  $N = 20$ . Substitute  $x[n]$  into (3.22) and then maximize  $J(f_0)$  over  $0 < f_0 < 1/2$  by computing the values of  $J(f_0)$  explicitly for each value of  $f_0$ . Choose the values of  $f_0$  as  $f_0 = 0.1, 0.2, 0.3, 0.4$ . Then choose the values as  $f_0 = 0.01, 0.02, \dots, 0.49$ . Do you obtain the correct value of  $f_0$ ? Hint: you will need to write a computer program to do this.

---



### The siren call of linearization

The linear signal model  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$  is such an appealing one (with the optimal estimator given explicitly as  $\hat{\boldsymbol{\theta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$ ), that many algorithms are based on an *approximate* linear model. When the true model is nonlinear, it can be reduced to a linear one by using a first-order Taylor expansion about the true value of the parameter. Of course, *the true value is unknown* since that is what we are trying to estimate. Assume, however, that we can guess at the true value and then use a Taylor expansion. Even if we don't know the true value, the argument goes that maybe we can iterate the Taylor expansion until we arrive at the true value, similar to a standard Newton-Raphson approach. As an example, let's look at estimation of the frequency of a simple sinusoid  $s[n] = \cos(2\pi f_0 n)$  by minimizing

$$J(f_0) = \sum_{n=0}^{N-1} (x[n] - \cos(2\pi f_0 n))^2. \quad (3.23)$$

Clearly, the signal is nonlinear with  $f_0$ , but we can linearize the signal about some guess, say  $f_G = 1/4$ , by using the first-order Taylor expansion,  $g(x) = g(x_G) + dg/dx|_{x_G}(x-x_G)$ , to obtain

$$\begin{aligned} \cos(2\pi f_0 n) &= \cos[2\pi f_G n] + (-2\pi n \sin[2\pi f_G n]) (f_0 - f_G) \\ &= \cos[2\pi(1/4)n] + (-2\pi n \sin[2\pi(1/4)n]) (f_0 - 1/4). \end{aligned}$$

Then, inserting this into (3.23) we have

$$J(f_0) \approx \sum_{n=0}^{N-1} \left[ x[n] - \cos\left(\frac{n\pi}{2}\right) + \left(2\pi n \sin\left(\frac{n\pi}{2}\right)\right) (f_0 - 1/4) \right]^2$$

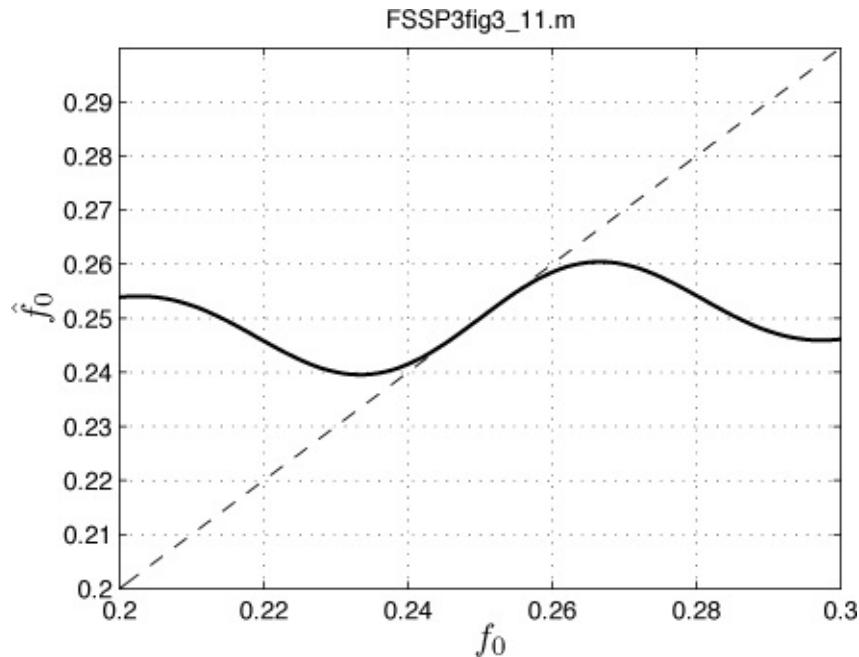
which is now a simple quadratic function of  $f_0$ . By differentiating  $J(f_0)$  with respect to  $f_0$ , setting the derivative equal to zero, and solving, we have that

$$\hat{f}_0 = \frac{1}{4} - \frac{\sum_{n=0}^{N-1} [(x[n] - \cos(n\pi/2))(2\pi n \sin(n\pi/2))]}{\sum_{n=0}^{N-1} (2\pi n \sin(n\pi/2))^2}. \quad (3.24)$$

If we assume that there is no noise present so that  $x[n] = \cos(2\pi f_0 n)$ , then

$$\hat{f}_0 = \frac{1}{4} - \frac{\sum_{n=0}^{N-1} [(\cos(2\pi f_0 n) - \cos(n\pi/2))(2\pi n \sin(n\pi/2))]}{\sum_{n=0}^{N-1} (2\pi n \sin(n\pi/2))^2}.$$

Our estimate  $\hat{f}_0$  as a function of the *true frequency*  $f_0$  is shown in [Figure 3.11](#) for values of frequency close to our initial guess of  $f_G = 0.25$ .



**Figure 3.11: Least squares estimate of frequency  $\hat{f}_0$  of linearized signal versus true frequency  $f_0$ , assuming no noise. The frequency estimation error is the difference between the solid curve and dashed line. The linearization point is  $f_G = 0.25$ .**

It is seen that as long as the true value of  $f_0$  is near our initial guess of  $f_G = 0.25$ , then the estimation error will be small. However, a large error can occur if

we linearize about a value that is not close to the true value.



## 3.6. Random Signals with Known PDF (Type 3)

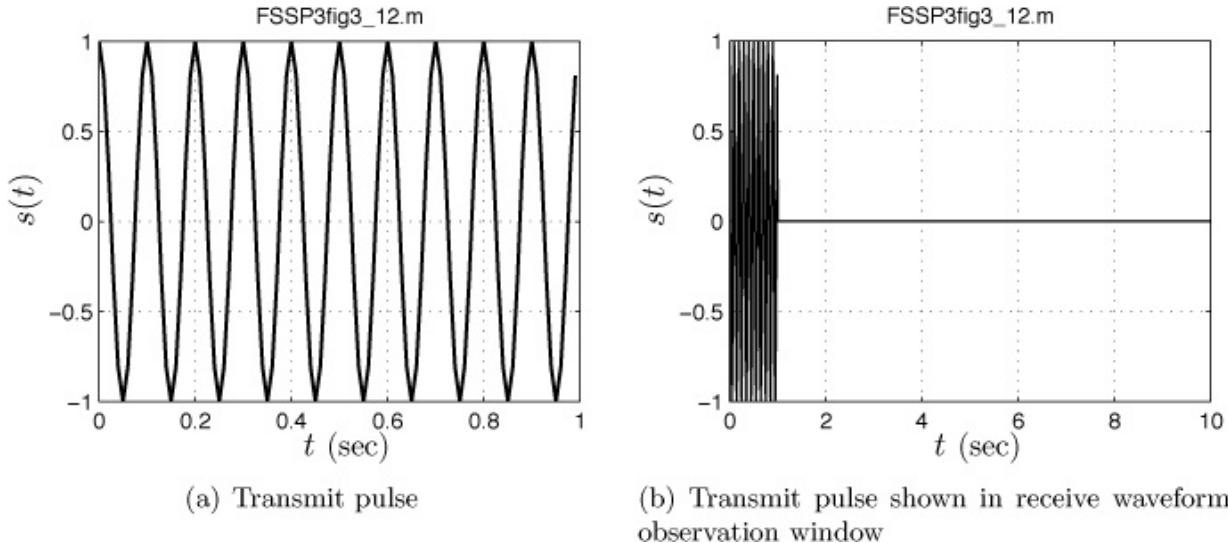
### 3.6.1. General Considerations

Some signals have parameters that are better modeled as *outcomes* of random variables. The simplest example, but one of the most useful, is a sinusoid that has been reflected by a geometrically complex target. Such a target exhibits multiple points of reflection with each point producing an echo with an attenuated amplitude and a slightly different time delay. Specifically, if the transmitted continuous-time signal is  $s(t) = A \cos(2\pi F_0 t + \varphi)$ , then the reflected signal due to  $p$  point reflectors will be

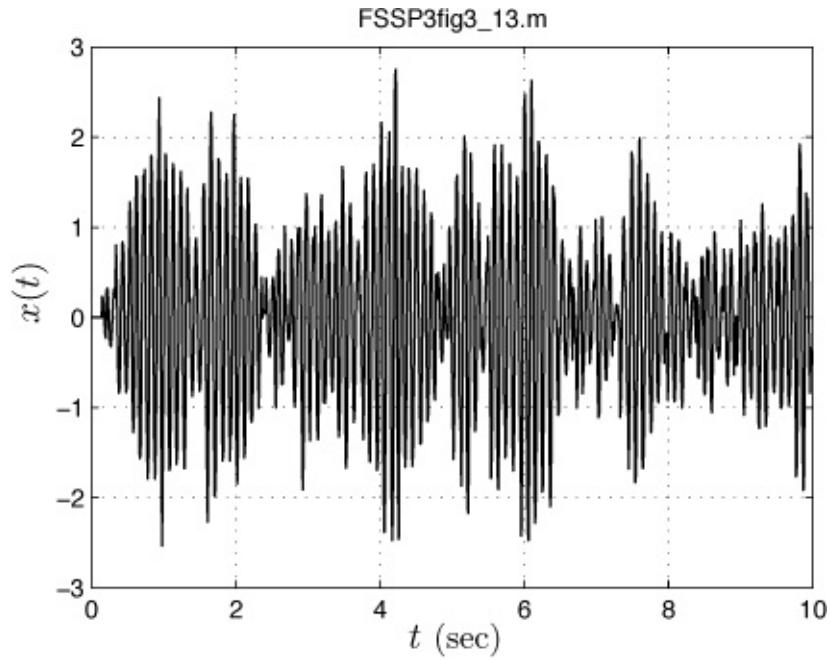
$$\begin{aligned} \sum_{i=1}^p A_i \cos[2\pi F_0(t - \tau_i) + \phi_i] &= \underbrace{\sum_{i=1}^p A_i \cos(-2\pi F_0 \tau_i + \phi_i) \cos(2\pi F_0 t)}_{\alpha_1} \\ &\quad - \underbrace{\sum_{i=1}^p A_i \sin(-2\pi F_0 \tau_i + \phi_i) \sin(2\pi F_0 t)}_{\alpha_2} \\ &= \alpha_1 \cos(2\pi F_0 t) + \alpha_2 \sin(2\pi F_0 t). \end{aligned}$$

Because  $\{A_i, \tau_i, \phi_i; i = 1, 2, \dots, p\}$  are unknown to us (they depend upon the target shape, its orientation, etc.), the best we can do in characterizing their values is to assume that we observe specific outcomes of random variables. The motivation for this type of modeling may be understood by the example shown in [Figures 3.12](#) and [3.13](#). In [Figure 3.12a](#) a sinusoid of frequency  $F_0 = 10$  Hz is transmitted for a time duration of 1 second. The received signal “window” is shown in [Figure 3.12b](#). Note that the receive time window is longer than the transmit signal duration. This ensures that all the reflections, which arrive at the receiver at different times due to the varying distances from the transmitter to the different point reflectors, may be processed. The received waveform is shown in [Figure 3.13](#) and exhibits the usual time delay *spreading* usually seen in practice. The variation in amplitude is referred to as *fading* and is due to constructive and destructive interference. Since the received waveform is a summation of all these echos, with each one being of a similar character, the central limit theorem

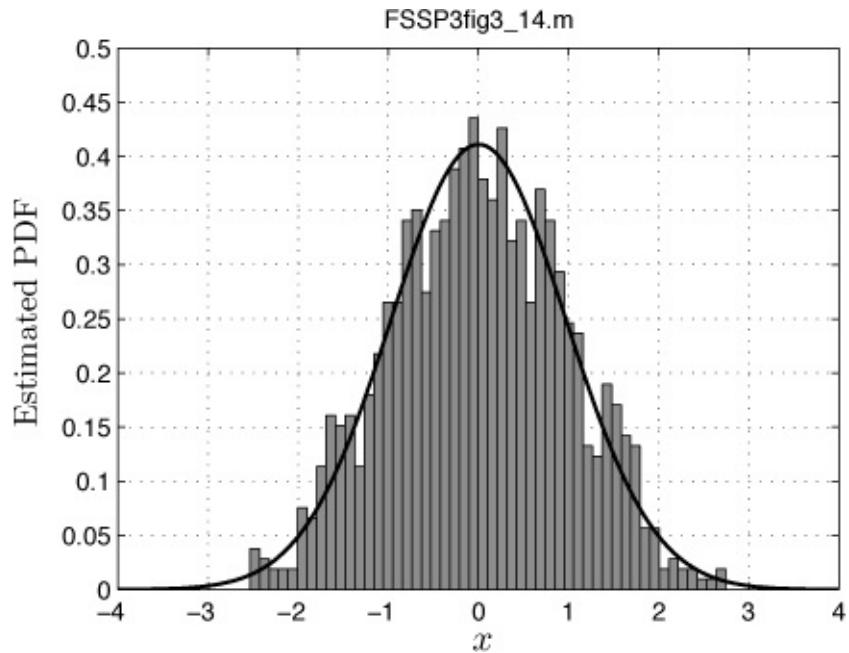
(CLT) [Kay 2006, pg. 492] applies, and the PDF of the amplitudes  $\alpha_1$  and  $\alpha_2$  are Gaussian. This is verified by estimating the PDF, with the result shown in Figure 3.14. Usually for a small target the echos are closer together than those shown in Figure 3.13, which have been accentuated for illustration purposes. Therefore, the received signal in a one second interval will appear as a constant amplitude and constant phase sinusoid but whose values are random outcomes. This model is called the *random amplitude/phase sinusoid* model and is called a Swerling I model in radar [Nathanson 1999]. It is also used as a model for the output waveform of a communication channel subject to fading [Rappaport 2002]. We now make the Gaussian assumption for the amplitudes  $\alpha_1$  and  $\alpha_2$ , which then implies that the receive signal samples will also be Gaussian distributed. A shorthand notation for a Gaussian random variable PDF that we will use is  $X \sim N(\mu, \sigma^2)$ , which is to say that random variable  $X$  “is distributed according to” a Gaussian (or sometimes called a *normal*) PDF with mean  $\mu$  and variance  $\sigma^2$ . Using this notation we assume that  $\alpha_1 \sim \mathcal{N}(0, \sigma_\alpha^2)$ ,  $\alpha_2 \sim \mathcal{N}(0, \sigma_\alpha^2)$  and that  $\alpha_1$  and  $\alpha_2$  are independent. Note that with these assumptions the amplitude of the sinusoid is  $A = \sqrt{\alpha_1^2 + \alpha_2^2}$  and is Rayleigh distributed, and the phase is  $\varphi = \arctan(-\alpha_2/\alpha_1)$  and is uniformly distributed on  $(-\pi, \pi)$ . Also, the amplitude and phase are independent random variables [Kay 2006, pg. 403]. Hence, the terminology in communications is that of a *Rayleigh fading channel*.



**Figure 3.12: Transmitted sinusoidal pulse.**



**Figure 3.13:** Received waveform consisting of many randomly overlapped and random amplitude echos.



**Figure 3.14:** Estimated PDF of the samples of received waveform shown in [Figure 3.13](#) and a Gaussian PDF fit (solid curve).

As a result of this modeling, the received discrete-time signal is given as

$$s[n] = \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n) \quad (3.25)$$

and is called the *random sinusoid model*. In matrix/vector form the signal is

$$\mathbf{s} = \underbrace{\begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \sin[2\pi f_0(N-1)] \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}}_{\boldsymbol{\theta}} \quad (3.26)$$

which is of the linear model form and *apparently* identical to the signal portion of (3.17), *except*  $\boldsymbol{\theta}$  is a vector of random variables. The PDF of  $\boldsymbol{\theta}$  is multivariate Gaussian with mean vector zero and covariance matrix  $\mathbf{C} = \sigma_\alpha^2 \mathbf{I}$ , with  $\mathbf{I}$  being the identity matrix (since  $\alpha_1, \alpha_2$  are independent random variables, hence uncorrelated random variables, and each one has variance  $\sigma_\alpha^2$ ). Denoting a random vector as being distributed according to a multivariate Gaussian PDF with mean vector  $\boldsymbol{\mu}$  and covariance matrix  $\mathbf{C}$  as  $\mathbf{X} \sim N(\boldsymbol{\mu}, \mathbf{C})$ , we therefore have that  $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_\alpha^2 \mathbf{I})$ . We next summarize the two random sinusoid models commonly used and then generalize the discussion to the *Bayesian linear model*.

### 3.6.2. Random Sinusoid - Zero Mean

The zero mean random sinusoid model is

$$s[n] = \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n) \quad n = 0, 1, \dots, N-1 \quad (3.27)$$

where  $\boldsymbol{\alpha} = [\alpha_1 \alpha_2]^T \sim \mathcal{N}(\mathbf{0}, \sigma_\alpha^2 \mathbf{I})$ . This is the one illustrated in [Figure 3.13](#).

### 3.6.3. Random Sinusoid - Nonzero Mean

In some situations the amplitudes are more accurately modeled using nonzero means as

$$\boldsymbol{\alpha} \sim \mathcal{N}(\boldsymbol{\mu}_\alpha, \sigma_\alpha^2 \mathbf{I}).$$

This is encountered for targets and channels for which the received echos are not all about the same magnitude but one echo may dominate. This is generally referred to as a *Rician fading channel* in communication [[Proakis and Salehi 2007](#)] and is also typical in radar scenarios for targets that exhibit *specular* returns [[Nathanson 1999](#)].

### 3.6.4. Bayesian Linear Model

The most general linear model for random signals allows for arbitrary basis signals  $\{h_1[n], h_2[n], \dots, h_p[n]\}$  with random amplitudes  $\{\theta_1, \theta_2, \dots, \theta_p\}$  and is expressed as

$$s[n] = \sum_{i=1}^p \theta_i h_i[n] \quad n = 0, 1, \dots, N-1.$$

As usual, it can be put into the form  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ , where

$$\mathbf{H} = \begin{bmatrix} h_1[0] & h_2[0] & \dots & h_p[0] \\ h_1[1] & h_2[1] & \dots & h_p[1] \\ \vdots & \vdots & \vdots & \vdots \\ h_1[N-1] & h_2[N-1] & \dots & h_p[N-1] \end{bmatrix}$$

and is of dimension  $N \times p$ , and  $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \sigma_\theta^2 \mathbf{I})$ . Finally, adding WGN  $\mathbf{w}$  to the model we have

$$\mathbf{x} = \mathbf{s} + \mathbf{w} = \mathbf{H}\boldsymbol{\theta} + \mathbf{w} \quad (3.28)$$

and is called the *Bayesian linear model*. It is to be contrasted with the deterministic linear model of (3.17) in that the vector of unknown parameters  $\boldsymbol{\theta}$  is now a *Gaussian random vector*. As a special case, if  $\sigma_\theta^2 = 0$  so that the amplitudes are no longer random but are given by the known deterministic value of  $\boldsymbol{\theta} = \boldsymbol{\mu}_\theta$ , then (3.28) reduces to

$$\mathbf{x} = \mathbf{H}\boldsymbol{\mu}_\theta + \mathbf{w}. \quad (3.29)$$

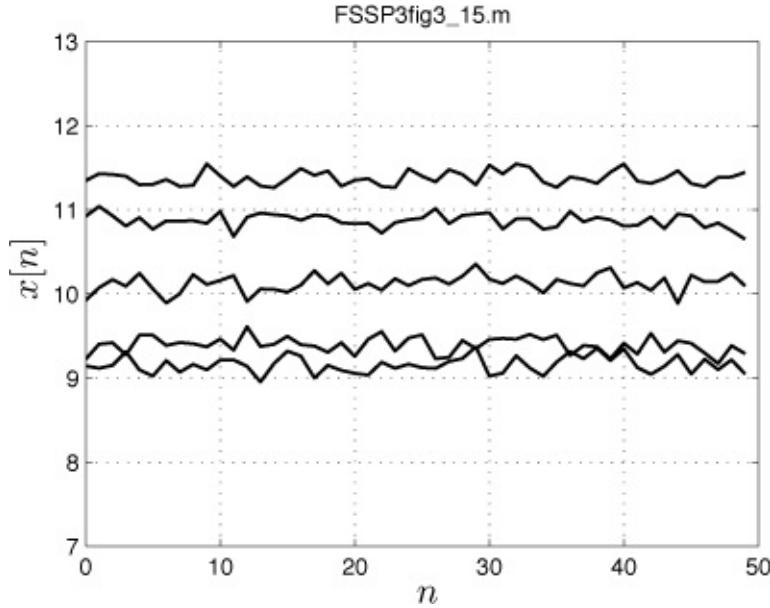
This is just the deterministic linear model, where  $\boldsymbol{\theta}$  has been replaced by  $\boldsymbol{\mu}_\theta$ , a deterministic vector. It can also be shown that for the Bayesian linear model the PDF of the data vector is [Kay 1993, pg. 326]

$$\mathbf{x} \sim \mathcal{N}(\mathbf{H}\boldsymbol{\mu}_\theta, \mathbf{H}\mathbf{C}_\theta \mathbf{H}^T + \sigma^2 \mathbf{I})$$

if  $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \mathbf{C}_\theta)$ .

As an example of the applicability of the Bayesian linear model, consider the modeling of a DC level signal  $s[n] = A$ , where the value of  $A$  is unknown but “on the average” it appears to take on the value  $A \approx 10$ . The latter may be ascertained by repeating the experiment of collecting the data and observing the values of  $A$ . We model this by letting  $A \sim \mathcal{N}(10, \sigma_A^2)$ , where  $\sigma_A^2 = 2$  so the signal is referred to as the *random DC level*. The choice of  $\sigma_A^2$  reflects our confidence in the range of values of  $A$  that we will observe in practice. For the value of  $\sigma_A^2 = 2$  we expect to observe with high probability  $\mu_A \pm 3\sigma_A = 10 \pm 3\sqrt{2}$  or values in the range  $[5.8, 14.2]$  (the mean plus or minus three standard deviations). Also, for the example we choose  $\sigma^2 = 0.01$  so that the DC level signal is not observed directly but is corrupted by noise as per (3.28). This means that we observe  $x[n]$

$= A + w[n]$ , where  $w[n]$  is WGN with variance of  $\sigma^2 = 0.01$ . Some typical observed data sets are shown in [Figure 3.15](#).



**Figure 3.15: Typical outcomes of the random DC level signal with  $A \sim N(10, 2)$  embedded in WGN with variance  $\sigma^2 = 0.01$ . The data samples have been connected with straight lines for easier viewing.**

Note that since some of the outcomes of  $A$  are less than 10, there may be some performance degradation of any algorithm based on this random modeling of a DC level signal as compared to the deterministic modeling with  $A = 10$ . It all depends upon whether the performance degradation when  $A$  is less than 10 is more than offset by the performance improvement when  $A$  is greater than 10. For example, in a Rayleigh fading communications channel there is a degradation in performance relative to a channel with no fading [[Proakis and Salehi 2007](#)].

### 3.6.5. Other Random Signal Models with Known PDF

The previous random signal models employ *known form* signals with parameters that are modeled as outcomes of random variables. There are also models that do not assume the form of the signal is known but instead use a *random process* model. These are described in [Chapter 4](#) since random process models are more routinely used for noise modeling.

## 3.7. Random Signals with PDF Having Unknown Parameters (Type 4)

This modeling of signals, although used occasionally in practice, leads to *hard*

algorithm design problems. Some details can be found in [[Kay 1998](#), pg. 302].

### 3.8. Lessons Learned

- A mathematical model can sometimes be used for both a signal model and a noise model.
- It is critical that the assumed knowledge used in developing an algorithm be valid in practice, i.e., in the operational environment of the algorithm.
- Adaptive algorithms, i.e., ones that estimate unknown parameters on-line, are more robust but will suffer in performance relative to algorithms that have a priori knowledge of the parameters.
- The linear signal model, which is one whereby the signal depends upon the unknown parameters linearly, is the most useful model in practice. It leads to optimal algorithms that are easily implementable.
- The partially linear model becomes practical when the number of nonlinear parameters is small, typically up to three.
- Attempting to convert a nonlinear signal into a linear signal using linearization techniques is fraught with danger!
- Most signal processing algorithms achieve their goals by reducing noise effects. To do so there is always some explicit or implicit type of noise averaging that takes place.
- The Gaussian PDF assumption for unknown parameters and for noise modeling is critical for developing a practical algorithm.

## References

- Brennan, L.E., I.S. Reed, W. Sollfrey, “A Comparison of Average-Likelihood and Maximum-Likelihood Ratio Tests for Detecting Radar Targets of Unknown Doppler Frequency”, *IEEE Trans. on Information Theory*, pp. 104–110, Jan. 1968.
- Burdic, W.S., *Underwater Acoustic Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Cook, C.E., M. Bernfeld, *Radar Signals*, Artech House, Boston, 1993.
- Granger, C.W.J., P. Newbold, *Forecasting Economic Time Series, Sec. Ed.*, Academic Press, NY, 1986.
- Graybill, F.A., *Theory and Application of the Linear Model*, Duxbury Press, Belmont, CA, 1976.

- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory, Vol. I*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- McConnell, *Vibration Testing, Theory and Practice*, J. Wiley, NY, 1995.
- Miller, T., L. Potter, J. McCorkle, "RFI Suppression for Ultra Wideband Radar", *IEEE Trans. on Aerospace and Electronic Systems*, pp. 1142–1156, Oct. 1997.
- Nathanson, F.E., *Radar Design Principles*, SciTech Pubs., NJ, 1999.
- Proakis, J., M. Salehi, *Digital Communications*, 5th ed., McGraw-Hill, NY, 2007.
- Rappaport, T.S., *Wireless Communications*, Prentice Hall PTR, Upper Saddle River, NJ, 2002.
- Schafer, R.W., L.R. Rabiner, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
- Sörnmo, L., P. Laguna, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, Elsevier Academic Press, NY, 2005.
- Vanhamme, L., T. Sundin, P. Van Hecke, S. Van Huffel, "MR Spectroscopy Quantification: A Review of Time-Domain Methods", *NMR in Biomedicine*, pp. 233–246, 2001.
- Wise, J.D., J.R. Caprio, T.W. Parks, "Maximum Likelihood Pitch Estimation", *IEEE Trans. on Acoustics, Speech and Signal Processing*, pp. 418–423, 1976.

## **Appendix 3A. Solutions to Exercises**

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**3.1** The first equation follows from  $\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$ . The second equation follows from  $\sin(2A) = 2\sin(A)\cos(A)$ . The third equation uses Euler's identity. Finally,  $\sum_{n=0}^3 \exp(jn\pi/2)$  can be evaluated by drawing in the complex plane arrows from the origin to each complex number. The arrows or phasors will all terminate on the

unit circle and have angles of  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ ,  $270^\circ$  leading to cancellation of all the phasors so that the sum equals zero and thus, the imaginary part is zero.

### 3.2 They are not equal since

$$\begin{aligned} g_n(a+b) &= \cos(2\pi(0.1 - 0.1)n) = \cos(0) = 1 \\ g_n(a) + g_n(b) &= \cos(2\pi(0.1)n) + \cos(2\pi(-0.1)n) \\ &= \cos(2\pi(0.1)n) + \cos(2\pi(0.1)n) = 2\cos(2\pi(0.1)n). \end{aligned}$$

### 3.3 The signals are equal for all $n$ since

$$\begin{aligned} s_1[n] &= 1 \cdot \cos(2\pi f_0 n + \pi/4) = \cos(2\pi f_0 n + (5/4)\pi - \pi) \\ &= \cos(2\pi f_0 n + (5/4)\pi) \cos(-\pi) - \sin(2\pi f_0 n + (5/4)\pi) \\ &\quad \sin(-\pi) \\ &= -\cos(2\pi f_0 n + (5/4)\pi) = s_2[n]. \end{aligned}$$

### 3.4 To find the minimum, differentiate $J(A)$ and set the derivative equal to zero to yield

$$\frac{dJ(A)}{dA} = -2 \sum_{n=0}^{N-1} (x[n] - A) = 0$$

yielding  $\hat{A} = (1/N) \sum_{n=0}^{N-1} x[n]$ , the sample mean. This is a reasonable estimator for a DC level signal embedded in *zero mean* noise. By averaging the samples, the signal will remain the same since it is a *constant* and therefore unaffected by averaging, but the noise should average out to a small quantity. It is easily shown that if  $x[n] = A + w[n]$ , where  $w[n]$  is WGN with variance  $\sigma^2$ , then  $\text{var}((1/N) \sum_{n=0}^{N-1} w[n]) = \sigma^2/N$ . Therefore, the noise power is reduced by a factor of  $N$  [Kay 1993 pg. 31] due to averaging. See also [Exercise 3.6](#) for a detailed calculation for the variance of  $\hat{A}$  for  $N = 2$ .

### 3.5 Since

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}$$

we have that

$$\mathbf{H}^T \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & 5 \end{bmatrix}$$

and

$$\mathbf{H}^T \mathbf{x} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \end{bmatrix} = \begin{bmatrix} x[0] + x[1] + x[2] \\ x[1] + 2x[2] \end{bmatrix}.$$

Note that in (3.13) we have for the elements in the matrix

$$\sum_{n=0}^2 1 = 3 \quad \sum_{n=0}^2 n = 3 \quad \sum_{n=0}^2 n^2 = 5$$

and for the vector

$$\begin{aligned} \sum_{n=0}^2 x[n] &= x[0] + x[1] + x[2] \\ \sum_{n=0}^2 nx[n] &= 0 \cdot x[0] + x[1] + 2x[2] = x[1] + 2x[2]. \end{aligned}$$

**3.6** To find the variance we note that  $\text{var}(X) = E[X^2] - E^2[X]$  and if  $X$  has zero mean or  $E[X] = 0$ , this becomes  $\text{var}(X) = E[X^2]$ . Therefore, since  $E[\bar{w}] = E[(w[0] + w[3])/2] = E[w[0]]/2 + E[w[3]]/2 = 0$ , we have that

$$\begin{aligned}
\text{var}(\bar{w}) &= \\
&= E[\bar{w}^2] && (\bar{w} \text{ has zero mean}) \\
&= E[(w[0] + w[3])/2]^2 && (\text{definition of } \bar{w}) \\
&= \frac{1}{4}E[w^2[0] + 2w[0]w[3] + w^2[3]] \\
&= \frac{1}{4}[E[w^2[0]] + 2E[w[0]w[3]] \\
&\quad + E[w^2[3]]] && (\text{linearity of expected value}) \\
&= \frac{1}{4}[\text{var}(w[0]) + 2E[w[0]]E[w[3]] \\
&\quad + \text{var}(w[3])] && (w[0], w[3] \text{ have zero mean} \\
&&& \text{and are uncorrelated}) \\
&= \frac{1}{4}[1 + 2 \cdot 0 \cdot 0 + 1] = \frac{1}{2}.
\end{aligned}$$

3.7 For  $N = 50$  you should obtain the estimated signal shown in [Figure 3.10b](#) and whose values are listed in [Table 3A.1](#). For  $N = 1000$  the estimated signal values are also shown in the table and are more accurate. The MATLAB program `FSSP3exer3_7.m`, contained on the CD, can be run to obtain the solution.

**Table 3A.1: Estimated signal obtained via averaging over periods of length ten.**

$n$	$s[n]$	$\hat{s}[n]$ for $N = 50$	$\hat{s}[n]$ for $N = 1000$
0	0	-0.4658	-0.0034
1	1	0.7343	0.9349
2	2	2.0021	2.1623
3	3	4.2443	2.9155
4	4	3.7021	3.9727
5	5	5.6720	5.0056
6	6	6.9840	6.0535
7	7	6.2607	6.8331
8	8	7.3200	7.8451
9	9	8.9395	8.8501

**3.8** For the frequency values  $f_0 = 0.1, 0.2, 0.3, 0.4$  the maximum of  $J(f_0)$  is 4.68 and occurs at  $f_0 = 0.1$ , while for  $f_0 = 0.01, 0.02, \dots, 0.49$  the maximum is 9.57 and occurs for  $f_0 = 0.12$ , which is the correct value. Clearly, the function to be maximized must be computed on a fine enough grid of points so that the value of  $f_0$  producing the maximum can be found accurately. The MATLAB program `FSSP3exer3_8.m`, contained on the CD, can be run to obtain the solution.

# Chapter 4. Mathematical Modeling of Noise

## 4.1. Introduction

The mathematical modeling of observation noise relies heavily on random process theory. The reader is referred to [[Kay 2006](#)] for much of the required background. Some of the important concepts and formulas are summarized in [Appendix 4A](#) for general random processes and in [Appendix 4B](#) for Gaussian random processes. The latter is the principal model employed in practice for noise modeling, and is therefore highlighted in this chapter.

As mentioned in [Chapter 3](#) the models to be described can also be used for some signals. This is the case when the mathematical form of the signal is unknown and so a statistical model is the only one available. An example is for speech in which the autoregressive random process model is used for unvoiced speech sounds [[Schafer and Rabiner 1978](#)]. Each sound is modeled by a Gaussian random process with a unique power spectral density (PSD). The reason for this characterization of the sound waveform is that it is quickly realized that the waveform is not reproducible. Even for the same speaker the waveform for a given sound will vary with the speed at which a person talks, the other vowels/consonants adjacent to the given one, as well as many other factors. It is this variability, this *noise-like* characteristic that the random process model is intended to capture. Generally, it is only the PSD that remains constant and which can be relied upon for modeling. And of course, for actual noise processes we have even less knowledge of the waveform, so that a random process model is the natural choice.

In this chapter we describe Gaussian as well as some nonGaussian noise models, stationary as well as some nonstationary noise models. Due to their mathematical complexity the nonGaussian and nonstationary models are given only a cursory treatment. In practice, it is important to be able to at least recognize when the stationary Gaussian noise model assumption is not valid. In such a case, it is customary to attempt to convert the nonstationary and nonGaussian data into stationary Gaussian data, if possible (see [Chapter 6](#)). When this is not possible, any subsequent signal processing algorithm may become hopelessly complicated. It is tempting to just ignore the nonGaussian/nonstationarity of the data. However, doing so can produce very poor performance of the algorithm. The interested reader is referred to [[Middleton and Spaulding 1993](#)].

All the noise models discussed are assumed to have *known* probability density functions (PDFs). When the model has a PDF with unknown parameters, the estimation of those parameters from field data for subsequent algorithm development can become exceedingly difficult. Similarly, parameter estimation as required for an “*in situ*” adaptive signal processing algorithm is a demanding task. Some methods to estimate noise parameters when noise-only data is available are described in [Chapter 6](#). These methods are useful for choosing the appropriate model.

## 4.2. General Noise Models

The noise models to be discussed are summarized in [Table 4.1](#). They are categorized by their Gaussian or nonGaussian nature and then furthermore by their PSD. These are the salient characteristics for algorithm design that must be kept in mind when choosing a model. The noise models are roughly ordered from top to bottom in terms of the easiest models to work with to the more difficult. The first three models are all Gaussian random processes (see [Appendix 4B](#)). The white Gaussian noise (WGN) model and the colored Gaussian noise models are both stationary and so are further delineated by their PSDs. The general Gaussian random process is nonstationary, precluding any definition of the PSD. Examples of the PSD of the first two models are shown in [Figure 4.1](#). The “flat” PSD shown in [Figure 4.1a](#) is also referred to as a “white” PSD, in analogy with white light, which is decomposable into a *uniform* distribution or rainbow of colors. A nonflat PSD, as shown in [Figure 4.1b](#), is therefore referred to as a “colored” PSD. The third type of noise model is a nonstationary Gaussian random process, meaning that an autocorrelation sequence (ACS) (see [Appendix 4A](#)) cannot be defined, and therefore, a definition of the PSD is not possible. An example of a realization of a nonstationary Gaussian random process is shown in [Figure 4.2](#), along with a stationary one for comparison. The processes are both Gaussian but differ in their power versus time. Specifically, they are  $w_1[n] = u[n]$  and  $w_2[n] = A[n]u[n]$ , where  $u[n]$  is WGN with variance  $\sigma_u^2 = 1$  and  $A[n] = \sqrt{0.95^n}$ . The first process has a constant power versus time, being given by

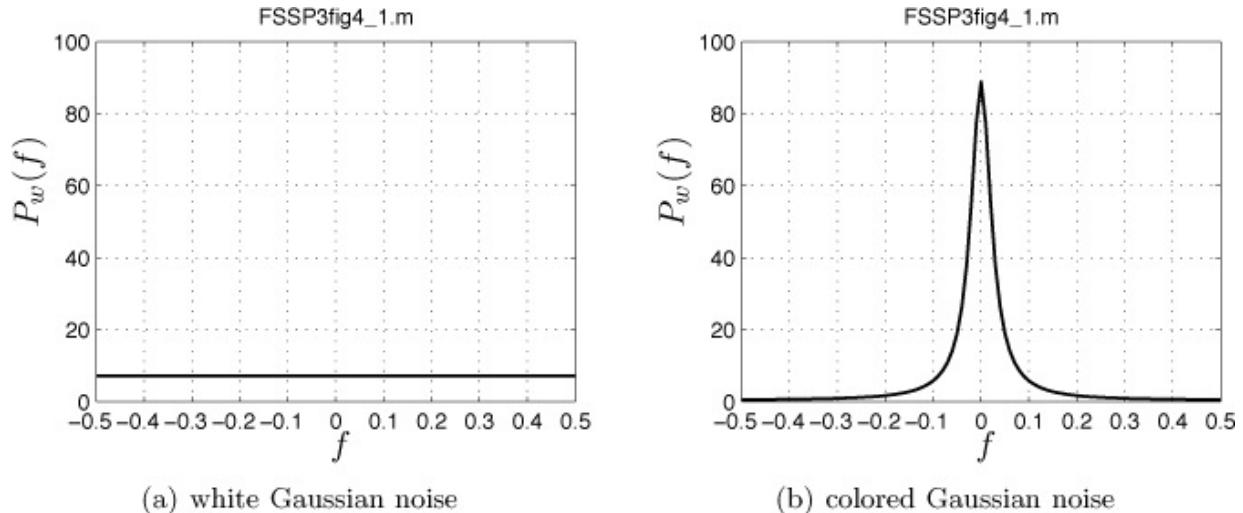
$$\begin{aligned} E[w_1^2[n]] &= E[u^2[n]] \\ &= \text{var}(u[n]) && (\text{mean of } u[n] \text{ is zero}) \\ &= \sigma_u^2 = 1 \end{aligned}$$

**Table 4.1: Noise models.**

Noise model	PDF	PSD	Stationarity
1. White Gaussian noise	Gaussian	flat	stationary
2. Colored Gaussian noise	Gaussian	nonflat	stationary
3. General Gaussian noise	Gaussian	not defined	nonstationary
4. IID <sup>†</sup> nonGaussian noise	nonGaussian	flat	stationary
5. Randomly phased sinusoids	nonGaussian <sup>††</sup>	impulsive	stationary

† IID means *independent and identically distributed*.

†† Can be modeled as Gaussian if amplitudes are modeled as Gaussian random variable outcomes.



**Figure 4.1: Power spectral densities for white Gaussian noise and colored Gaussian noise. Both PSDs have the same total power.**

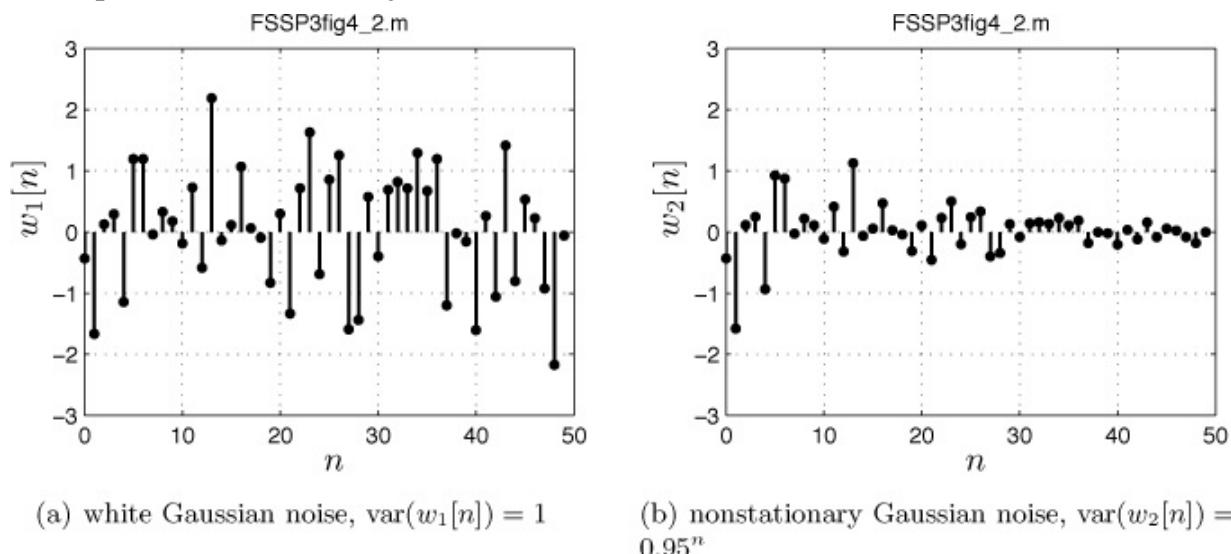
while the second process has power given by

$$\begin{aligned}
 E[w_2^2[n]] &= E[(A[n]u[n])^2] \\
 &= A^2[n]E[u^2[n]] \quad (A[n] \text{ is a constant}) \\
 &= A^2[n]\text{var}(u[n]) \quad (\text{mean of } u[n] \text{ is zero}) \\
 &= A^2[n]\sigma_u^2 \\
 &= A^2[n] = 0.95^n
 \end{aligned}$$

and is seen to vary with time. Thus,  $w_2[n]$  is a nonstationary process and should be modeled as such. For example, the fact that the noise power decreases with time indicates that any algorithm should weight the data samples more heavily at the end of the record, where the signal-to-noise ratio (SNR) is higher, than at the beginning. In many cases in practice, however, the precise noise power variation is not known (it may in fact be increasing), and so this variation in SNR cannot be leveraged. In reality, most noise processes are nonstationary if observed for a

long enough time interval.

Noise models 4 and 5 have nonGaussian PDFs. For model 4 it is assumed that each noise sample has a nonGaussian PDF. Typically, the PDF is one that exhibits “heavy tails” (an example will be given later). This produces amplitude events that would otherwise not be present for a Gaussian PDF, i.e., noise values for which  $|w[n]| > 3\sigma$ , where  $\sigma^2$  is the variance. Note that in [Figure 4.2a](#) the sample values are all less than three in magnitude. Also, the noise samples are assumed to all have the same PDF (identically distributed) and are all independent. Hence, the name *independent and identically distributed* (IID) nonGaussian noise is used. Due to the IID assumption it can be shown that the noise process is stationary and has a flat PSD.



**Figure 4.2: Realizations of stationary and nonstationary Gaussian random processes.**

Noise model 5, randomly phased sinusoids, is identical to the sinusoidal signal model given in [Chapter 3](#) (see [Table 3.1](#), model type 3). It is a sum of randomly phased sinusoids with fixed amplitudes and having different frequencies. Its PSD is a sum of impulses at the given frequencies. Its PDF is difficult to determine. However, as will be discussed shortly, it can be modeled as a Gaussian random process if the amplitudes are assumed to be outcomes of random variables with a certain PDF. We next examine each of these models in more detail.

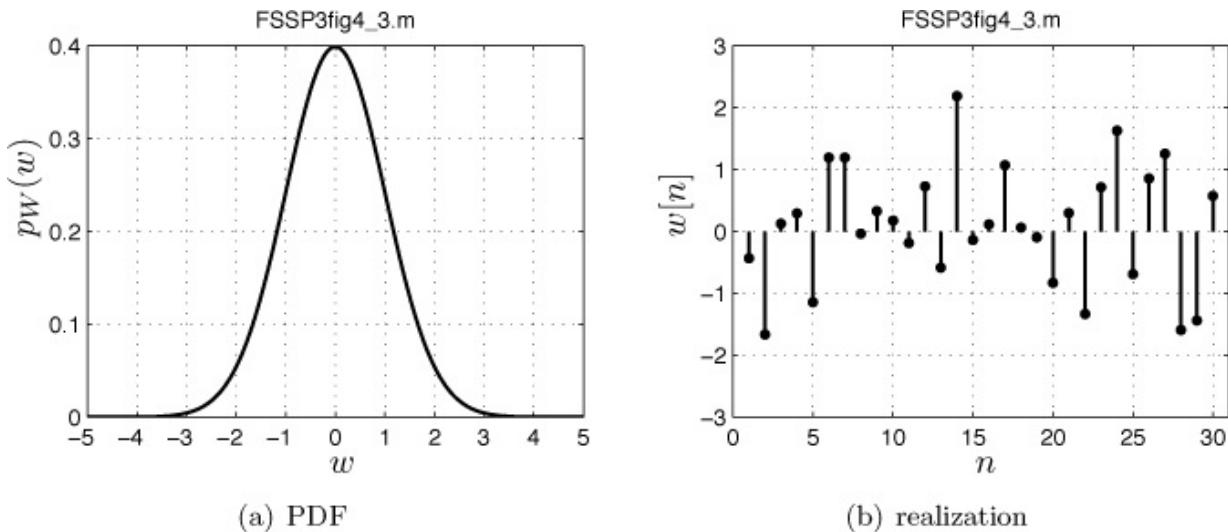
### 4.3. White Gaussian Noise

WGN has the following important properties:

1. Each noise sample in the data set  $\{w[0], w[1], \dots, w[N - 1]\}$  has the same PDF (identically distributed) and the PDF is Gaussian, as given by

$$p_W(w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}w^2\right) \quad -\infty < w < \infty \quad (4.1)$$

An example is shown in [Figure 4.3](#) for  $\sigma^2 = 1$ . Note that the probability of a sample exceeding  $3\sigma = 3$  or being less than  $-3\sigma = -3$ , is very small because the PDF is near zero for  $|w| > 3$ . In fact, as seen in [Figure 4.3b](#) there are no samples that exceed 3 in magnitude for this typical realization. Also, it can be shown that the mean  $E[w[n]] = 0$  for all  $n$  and the variance  $\text{var}(w[n]) = \sigma^2$  for all  $n$ .



**Figure 4.3: PDF and realization of WGN with  $\sigma^2 = 1$ .**

2. Each noise sample is independent of every other noise sample. This means that we cannot predict any sample based upon *observing* the others. We might say that this random process is very “noise-like”.
3. As a result of the properties that  $w[n]$  is composed of IID samples, the PSD is flat with

$$P_w(f) = \sigma^2 \quad -1/2 \leq f \leq 1/2.$$

and is shown in [Figure 4.1a](#) for  $\sigma^2 = 7.2$ .

The WGN random process is typically used in communication systems to model observation noise (called *additive* WGN in that field). Also, it is used in radar/sonar to model the ambient noise and for virtually any signal processing system subject to electronic noise. To generate realizations of WGN the

MATLAB program `WGNgendata.m`, which is contained on the CD, can be used.

---

### Exercise 4.1 – Generating white Gaussian noise

Using the program `WGNgendata.m`, generate a data record (also referred to as an *outcome* or as a *realization*) of WGN with variance  $\sigma^2 = 4$ . If  $N = 1000$  samples are generated, how many exceed 3 in magnitude? The expected number is given by the total number of samples multiplied by the probability that a sample exceeds 3. Specifically, it is

$$\begin{aligned} NP[w[n] > 3] &= NQ\left(\frac{3}{\sqrt{\sigma^2}}\right) \\ &= NQ(3/2) = 0.0668N \end{aligned}$$

where

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp(-t^2/2) dt$$

is the *right-tail probability* for a  $N(0, 1)$  random variable. (The right-tail probability for a point  $x$  is the probability that a random variable exceeds the value  $x$ ). It can be computed using the MATLAB program `Q.m`, which is contained on the CD. For  $N = 1000$  we expect about 67 exceedances.

•

---

---

### Exercise 4.2 – Estimating the power

Using the  $N = 1000$  samples of the previous exercise, estimate the noise power by using

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=0}^{N-1} w^2[n].$$

Is it close to the true value of  $\sigma^2 = 4$ ? If not, try increasing  $N$ . We will have much more to say about estimating the properties of a noise process in [Chapter 6](#).

•

---

## 4.4. Colored Gaussian Noise

Clutter in radar [[Nathanson 1999](#)] and reverberation in sonar [[Burdic 1984](#)] both have PSDs that are decidedly nonflat in the band of interest. An example of such a nonflat PSD has been shown in [Figure 4.1b](#). That PSD can be categorized as a *lowpass PSD*, whereby most of its power is below about  $f = 0.1$ . A particularly useful model for a nonflat PSD is the autoregressive (AR) model. It takes the mathematical form

$$P_w(f) = \frac{\sigma_u^2}{|1 + a[1] \exp(-j2\pi f) + \cdots + a[p] \exp(-j2\pi fp)|^2} \quad (4.2)$$

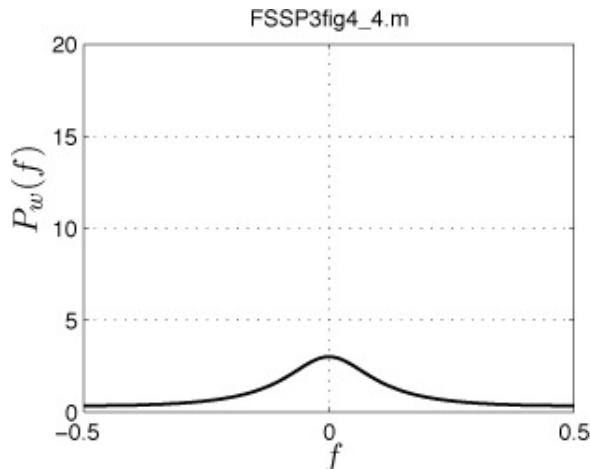
where  $\sigma_u^2 > 0$  is an *excitation noise variance parameter* and  $\{a[1], a[2], \dots, a[p]\}$  are the AR *filter parameters*. These designations will be justified shortly. The *order* of the model is given by the number of filter parameters  $p$ . We usually include the model order in the description by referring to it as an AR( $p$ ) PSD model. Its usefulness is based on the following properties:

1. It can be used to model any PSD as long as the model order  $p$  is chosen large enough.
2. Its parameters can be easily and accurately estimated from field data as described in [Chapter 11](#) and implemented using [Algorithms 11.3](#) and [11.4](#).
3. Computer simulation of realizations is easily accomplished.

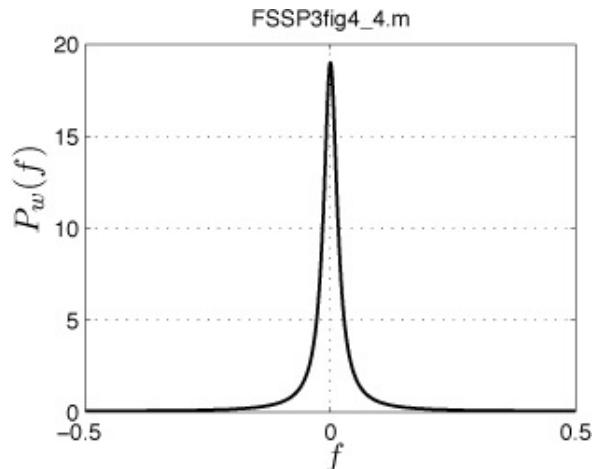
As an example, consider the AR(1) model given by

$$P_w(f) = \frac{\sigma_u^2}{|1 + a[1] \exp(-j2\pi f)|^2}. \quad (4.3)$$

For  $a[1] = -0.5$  and  $\sigma_u^2 = 0.75$ , the PSD is shown in [Figure 4.4a](#), while for  $a[1] = -0.9$  and  $\sigma_u^2 = 0.19$ , the PSD is shown in [Figure 4.4b](#). Both PSDs have the same total average power (area under the curves). However, the one with the larger value of  $|a[1]|$  is more narrowband. By proper choice of the AR filter parameter any bandwidth can be modeled. Once the value of  $a[1]$  has been chosen to specify the bandwidth, the value of  $\sigma_u^2$  can be used to specify the total average power, which is given by  $\sigma_u^2 / (1 - a^2[1])$ .



(a)  $a[1] = -0.5, \sigma_u^2 = 0.75$



(b)  $a[1] = -0.9, \sigma_u^2 = 0.19$

**Figure 4.4: AR(1) power spectral density models.**

---

### Exercise 4.3 – AR(1) PSD - another example

If  $a[1] = 0$ , what type of PSD is being modeled?

---



---

### Exercise 4.4 – AR(1) PSD - lowpass or highpass?

Choose for the AR parameters  $a[1] = 0.9$  and  $\sigma_u^2 = 0.19$ . Using ARpsd.m, which is contained on the CD, obtain values of the AR(1) PSD and the corresponding frequencies and plot the results. Compare the PSD against the one shown in [Figure 4.4b](#). What can you say about the effect of the sign of  $a[1]$  upon the PSD?

---

To more clearly understand the implicit model underlying (4.3) we note the relationship between the PSDs of the random processes at the input to a linear shift invariant filter and at the output. If  $u[n]$  is the input and  $w[n]$  is the output, and the filter frequency response is given by  $H(f)$ , then the PSDs are related as

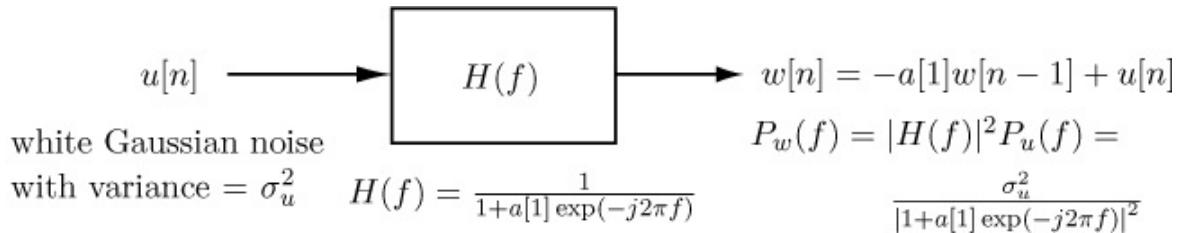
$$P_w(f) = |H(f)|^2 P_u(f). \quad (4.4)$$

This relationship is somewhat analogous to that for Fourier transforms for the input and output of a filter, which for deterministic signals would be  $W(f) = H(f)U(f)$ . The difference is that for random processes we are interested in powers

(actually *average* powers) as opposed to amplitudes. Now if we assume that  $P_u(f) = \sigma_u^2$ , a flat PSD, and also a filter frequency response of

$$H(f) = \frac{1}{1 + a[1] \exp(-j2\pi f)} \quad (4.5)$$

then the use of (4.4) results in the PSD at the output of the filter being given by the AR(1) model of (4.3). This is summarized in [Figure 4.5](#).



**Figure 4.5: AR(1) model for colored noise  $w[n]$ .  $H(f)$  is the frequency response of the coloration filter.**

Note that the input random process  $u[n]$ , which is WGN, is being filtered to alter the power at different frequencies. This produces the colored noise process  $w[n]$  at the output. Also, we can realize the output random process, which is the result of passing  $u[n]$  through a one-pole filter, by using the time domain description

$$w[n] = -a[1]w[n-1] + u[n] \quad (4.6)$$

where  $u[n]$  is white Gaussian noise with variance  $\sigma_u^2$ . Because  $u[n]$  excites the filter it is commonly referred to as *excitation noise*. For filter stability the filter coefficient, i.e., the AR(1) filter parameter, should satisfy  $|a[1]| < 1$ .

---

### Exercise 4.5 – Filter frequency response

Plot the magnitude of the filter frequency response given by (4.5) for  $-1/2 \leq f \leq 1/2$  using  $a[1] = -0.9$ . Hint: You will need a computer to do this.

---



---

### Exercise 4.6 – Power spectral density expressed as a real function

Show that the AR(1) PSD given by

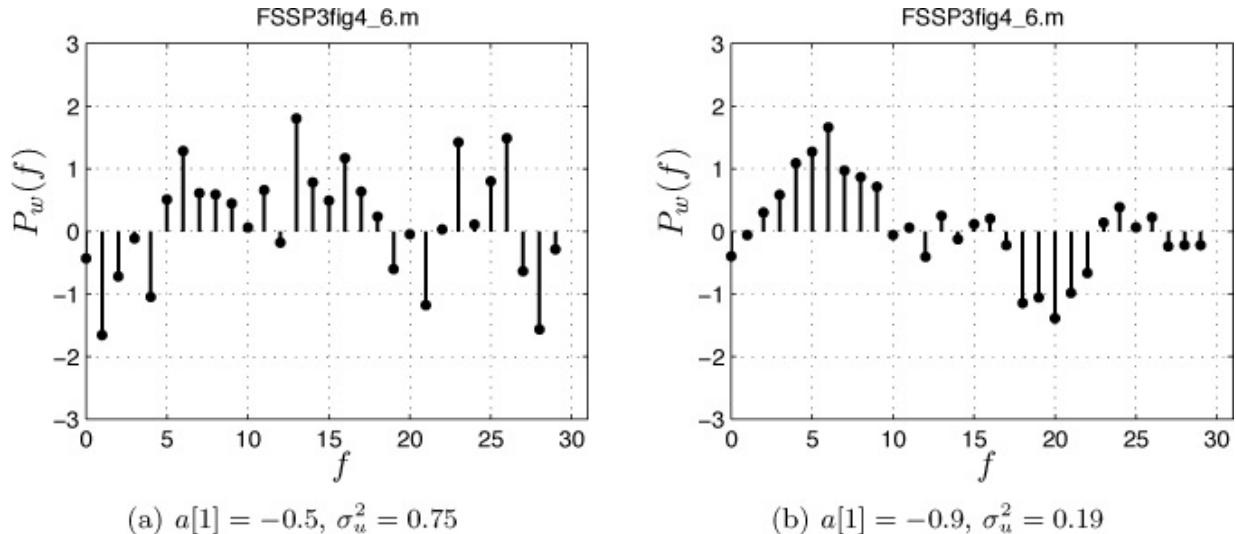
$$P_w(f) = \frac{\sigma_u^2}{|1 + a[1] \exp(-j2\pi f)|^2}$$

can also be expressed as

$$P_w(f) = \frac{\sigma_u^2}{1 + a[1] + 2a[1] \cos(2\pi f)}.$$

If  $a[1] < 0$ , show that the peak occurs at  $f = 0$  and if  $a[1] > 0$ , it is at  $f = 1/2$ .

Using (4.6) we can easily generate computer simulated realizations of an AR(1) random process. For the two AR(1) PSDs shown in [Figure 4.4](#) realizations are shown in [Figure 4.6](#). Note that the realization for which  $|a[1]|$  is small ([Figure 4.6a](#)) is more “noise-like” than the one that has a larger value of  $|a[1]|$  ([see Figure 4.6b](#)), which is more heavily *correlated*. This is because the PSD for the realization shown in [Figure 4.6a](#) contains *higher frequencies*, causing the waveform to *change faster*.



**Figure 4.6: Typical realizations for the AR(1) random process model whose power spectral densities are given in [Figure 4.4](#).**

To model bandpass PSDs, i.e., PSDs with peaks not at  $f = 0$  or  $f = 1/2$ , we require an AR model with  $p > 1$ . Since from (4.2) the filter frequency response is given by

$$H(f) = \frac{1}{1 + a[1] \exp(-j2\pi f) + \cdots + a[p] \exp(-j2\pi fp)} \quad (4.7)$$

and completely determines the PSD dependence on frequency, we need to design an *all-pole* filter with the desired frequency response. This is a digital filter design problem, for which many approaches exist. As a simple example, to

design a bandpass PSD with a peak at  $f = f_0$ , we need only place a set of complex conjugate poles in the z-plane at  $z_1 = r \exp(j2\pi f_0)$  and  $z_2 = r \exp(-j2\pi f_0)$ . This leads to a *system function* (the z-transform of the impulse response [Jackson 1988]) of

$$\begin{aligned}\mathcal{H}(z) &= \frac{1}{(1 - z_1 z^{-1})(1 - z_2 z^{-1})} \\ &= \frac{1}{(1 - r \exp(j2\pi f_0) z^{-1})(1 - r \exp(-j2\pi f_0) z^{-1})} \\ &= \frac{1}{1 - 2r \cos(2\pi f_0) z^{-1} + r^2 z^{-2}}.\end{aligned}$$

The frequency response is given by

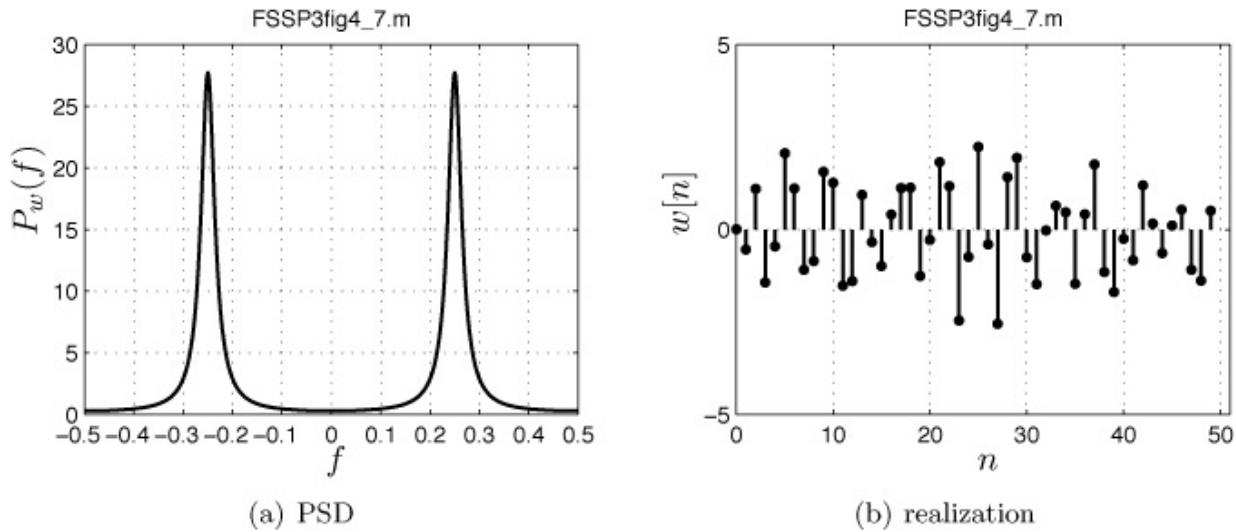
$$\begin{aligned}H(f) &= \mathcal{H}(z)|_{z=\exp(j2\pi f)} \\ &= \frac{1}{1 - 2r \cos(2\pi f_0) \exp(-j2\pi f) + r^2 \exp(-j4\pi f)}.\end{aligned}$$

Thus, we have from (4.7) with  $p = 2$  that

$$\begin{aligned}a[1] &= -2r \cos(2\pi f_0) \\ a[2] &= r^2.\end{aligned}\tag{4.8}$$

As an example, if we choose  $r = 0.9$ , placing the poles close to the unit circle, and  $f_0 = 0.25$  so that the pole angle is  $2\pi f_0 = 2\pi/4 = \pi/2$ , the PSD of the AR(2) model is shown in [Figure 4.7a](#). The AR(2) excitation noise variance, which serves to scale the total power in the PSD, was chosen to be  $\sigma_u^2 = 1$ . To generate a realization of  $w[n]$  we use the second-order recursive difference equation

$$w[n] = -a[1]w[n - 1] - a[2]w[n - 2] + u[n]\tag{4.9}$$



**Figure 4.7: Bandpass PSD modeled using an AR(2) model with complex conjugate poles at  $0.9 \exp(\pm j2\pi(0.25))$  and a typical realization.**

with a typical realization shown in [Figure 4.7b](#). To generate realizations of an AR( $p$ ) random process the MATLAB program `ARgendata.m`, which is contained on the CD, can be used.

To gain an appreciation of the effect of pole placement upon the PSD, we can use the usual geometrical argument as summarized in [Appendix 4C](#). A MATLAB program `pole_plot_PSD.m`, which is contained on the CD, allows the poles to be placed in the unit circle of the z-plane by the positioning of a cursor, and then computes and displays the corresponding AR PSD, assuming that  $\sigma_u^2 = 1$ .

---

**Exercise 4.7 – Effect of pole placement on AR(2) PSD**

Using the program `pole_plot_PSD.m`, place a pair of complex conjugate poles at approximately  $0.9 \exp(\pm j2\pi(0.25))$ . The instructions for running the program are contained in the program code. Does the PSD appear as shown in [Figure 4.7a](#)? Repeat the experiment but increase the radii of the poles to approximately  $r = 0.98$ . Explain your results. Hint: You may not be able to position the cursor exactly at the correct radius so just approximate it!

•

---

The program `pole_plot_PSD.m` can also be used to produce the AR filter parameters  $\{a[1], a[2], \dots, a[p]\}$  obtained from the chosen poles. These parameters, along with a value for  $\sigma_u^2$ , can be input to `ARgendata.m` to produce a corresponding realization of data, as in [Figure 4.7b](#).

---

**Exercise 4.8 – Generating a realization of AR(2) noise**

Using `ARgendata.m`, which requires the AR parameters to be input, generate and plot a realization of a  $N = 100$  sample data record of the AR(2) random process for  $r = 0.9$ ,  $f_0 = 0.25$ , and  $\sigma_u^2 = 1$ . Then, increase the pole radii to  $r = 0.98$  and repeat the experiment. Explain your results. Hint: See [\(4.8\)](#) to set the AR filter parameters.

•

---

The previous discussion and computer experiments were intended to lend insight into the properties of the AR PSD model, and in particular the types of PSDs it is capable of modeling. We have only explored AR models of low order. For some PSDs the order may need to be significantly higher. Such is the case if the PSD exhibits a zero at a frequency. Even though a high enough order will always suffice, the difficulty encountered in practice is that attempts to *estimate* a large number of AR parameters from field data may fail. Hence, model building and subsequent use of a large order model may lead to poorly performing algorithms. We will revisit this limitation in [Chapter 11](#). Suffice it to say that for good modeling and algorithm design, the order  $p$  should be small *relative to the data record length  $N$* .

Dispensing with the problem of estimating the AR parameters from data, we now present a general procedure for AR modeling based on *exact knowledge of the PSD to be approximated*. It proceeds as follows:

1. Choose the desired PSD  $P_w(f)$  to be modeled. The PSD can be specified using a mathematical form, say  $P_w(f) = \exp(-f^2/2)$ , if known. Otherwise, specify the PSD by using finely spaced samples in frequency, say  $P_w(f_l)$  for  $f_l = 0, 1/(2L), 1/L, \dots, 1/2$  for a large value of  $L$ .
2. Choose a value for  $p > 2M$ , where  $M$  is the number of peaks (local maxima) in the PSD.
3. If the mathematical form of the PSD is known and the inverse Fourier transform can be found analytically, proceed to Step 4. Otherwise, numerically compute the inverse Fourier transform of  $P_w(f)$  to yield the discrete-time ACS  $\hat{r}_w[k]$  for  $k = 0, 1, \dots, p$  as

$$\begin{aligned}\hat{r}_w[k] &= \int_{-\frac{1}{2}}^{\frac{1}{2}} P_w(f) \exp(j2\pi fk) df \\ &\approx \Delta f \sum_{l=-L}^L P_w(f_l) \exp(j2\pi f_l k) \\ &= \Delta f \sum_{l=-L}^L P_w(f_l) \cos(2\pi f_l k)\end{aligned}$$

where  $f_l = l/(2L)$ ,  $\Delta f = 1/(2L)$ , and  $L$  is large, say  $L = 2048$ . Also, use  $P_w(f_l) + P_w(f - l)$  for  $l < 0$ . We use a “^” for the ACS to remind us that the subsequent AR PSD  $\hat{P}_w(f)$  (and therefore its inverse Fourier transform

$\hat{r}_w[k]$ ) is only an approximation to the true PSD  $P_w(f)$ .

4. Using the ACS values, solve the *Yule-Walker equations* for the AR filter parameters which are

$$\begin{bmatrix} \hat{r}_w[0] & \hat{r}_w[1] & \dots & \hat{r}_w[p-1] \\ \hat{r}_w[1] & \hat{r}_w[0] & \dots & \hat{r}_w[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_w[p-1] & \hat{r}_w[p-2] & \dots & \hat{r}_w[0] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} \hat{r}_w[1] \\ \hat{r}_w[2] \\ \vdots \\ \hat{r}_w[p] \end{bmatrix}. \quad (4.10)$$

The Levinson recursion algorithm can be used to solve these equations in a computationally efficient manner [Kay 1988]. The MATLAB subprogram `YWsolve.m`, which is contained on the CD, can also be used to solve these equations. Once the equations have been solved for the AR filter parameters, the excitation noise variance is found as

$$\hat{\sigma}_u^2 = \hat{r}_w[0] + \hat{a}[1]\hat{r}_w[1] + \dots + \hat{a}[p]\hat{r}_w[p].$$

5. Compute the AR( $p$ ) modeled PSD by substituting the obtained AR parameters into (4.2) to obtain

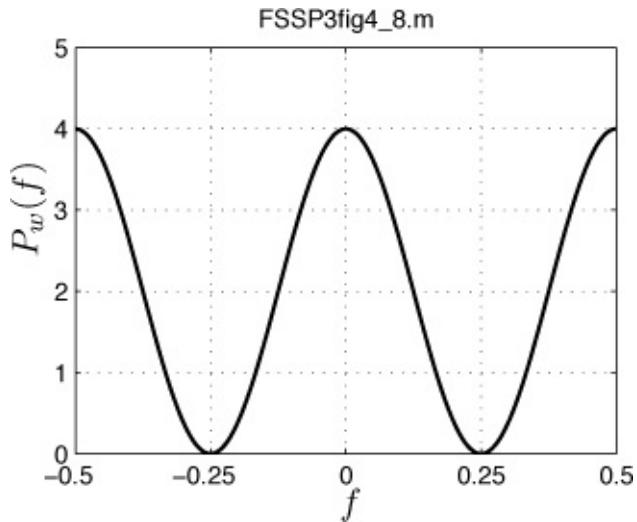
$$\hat{P}_w(f) = \frac{\hat{\sigma}_u^2}{|1 + \hat{a}[1]\exp(-j2\pi f) + \dots + \hat{a}[p]\exp(-j2\pi fp)|^2}$$

6. Compare the obtained AR PSD model  $\hat{P}_w(f)$  to the desired PSD  $P_w(f)$ , and if not satisfactory, increase  $p$  and go to Step 3.

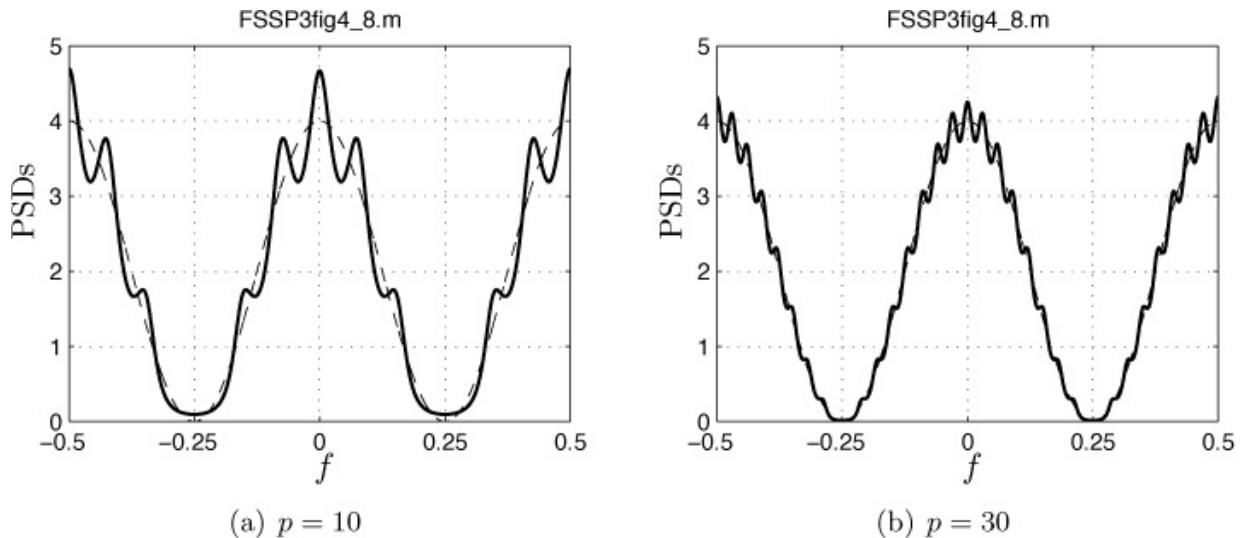
As an example, consider the modeling of a PSD exhibiting a zero and given mathematically by

$$P_w(f) = 4[1 - \cos(2\pi(f - 1/4))][1 - \cos(2\pi(f + 1/4))] \quad -1/2 \leq f \leq 1/2$$

as shown in [Figure 4.8](#). This type of PSD is a worst case since the AR model has only poles, and can only approximate zeros. Hence, it is not unexpected that for a small model order  $p = 10$ , the fit is poor, as shown in [Figure 4.9a](#). However, if we increase the model order to  $p = 30$ , a much better fit is obtained, as seen in [Figure 4.9b](#). It is typical of an AR model that the approximation is one that exhibits an “equi-ripple” type of fit. The MATLAB program `ARpsd_model.m`, which is contained on the CD, can be used to obtain the fitted AR parameters if the samples of the desired PSD are input.



**Figure 4.8: Desired PSD, exhibiting a zero at  $f = 0.25$ .**



**Figure 4.9: Desired PSD, exhibiting a zero, and the AR PSD model. The true PSD  $P_w(f)$  is shown dashed, and the AR PSD model  $\hat{P}_w(f)$  is shown as the solid curve.**

### Exercise 4.9 – A practice example

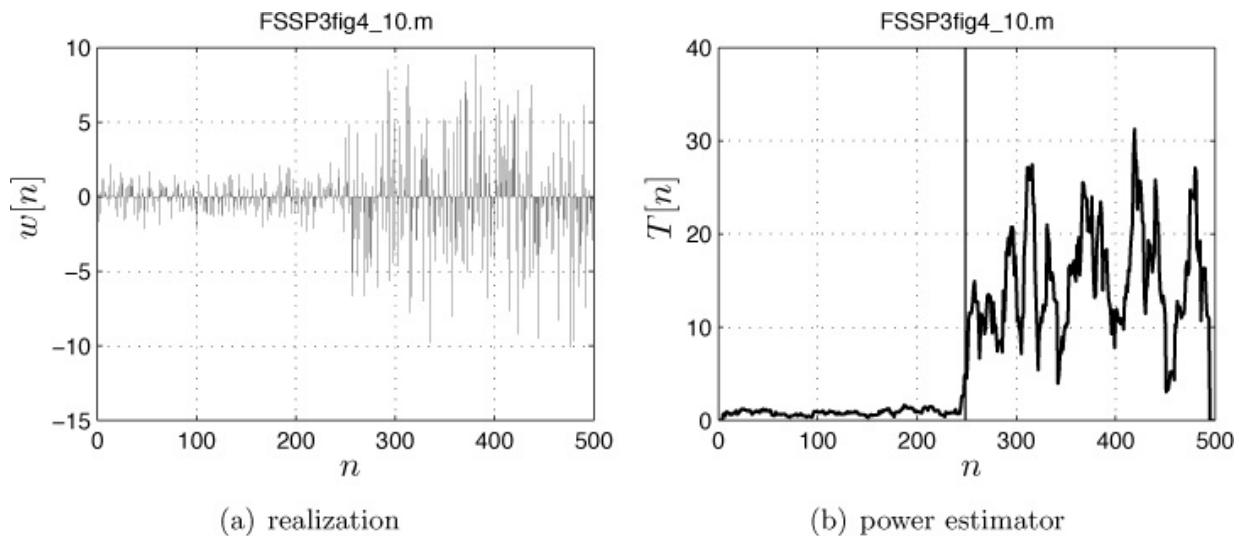
Assume we wish to model the PSD  $P_w(f) = \exp(-10|f|)$  for  $-1/2 \leq f \leq 1/2$ .

Compute the samples for  $P_w(f)$  and input them, along with an appropriate value for  $p$  to the program `ARpsd_model.m`. The output of the program will produce the AR PSD model. Plot it and compare to the true PSD. What order model is needed for a good approximation?

## 4.5. General Gaussian Noise

In the previous two sections we presented Gaussian noise models for stationary noise data. Stationarity allowed us to define a PSD. Nonstationary noise processes such as was illustrated in [Figure 4.2b](#) do not. Nonstationary Gaussian random processes can occur in practice due to:

1. *Abrupt* changes in the characteristics of the random process as illustrated in [Figure 4.10a](#).



**Figure 4.10: Random process with an abrupt change in power at  $n = M = 250$  and the sliding window power estimator. In [Figure 4.10b](#) the points have been connected by straight lines for easier viewing.**

2. *Gradual* changes as was illustrated in [Figure 4.2b](#).

In the case of abrupt changes it may still be possible to use a stationary random process model if we use *two different* models, one prior to the change and the other after the change. For example, in [Figure 4.10a](#) the model actually used to generate this plot is

$$w[n] = \begin{cases} w_1[n] & n = 0, 1, \dots, M - 1 \\ 2w_1[n] & n = M, M + 1, \dots, N - 1 \end{cases}$$

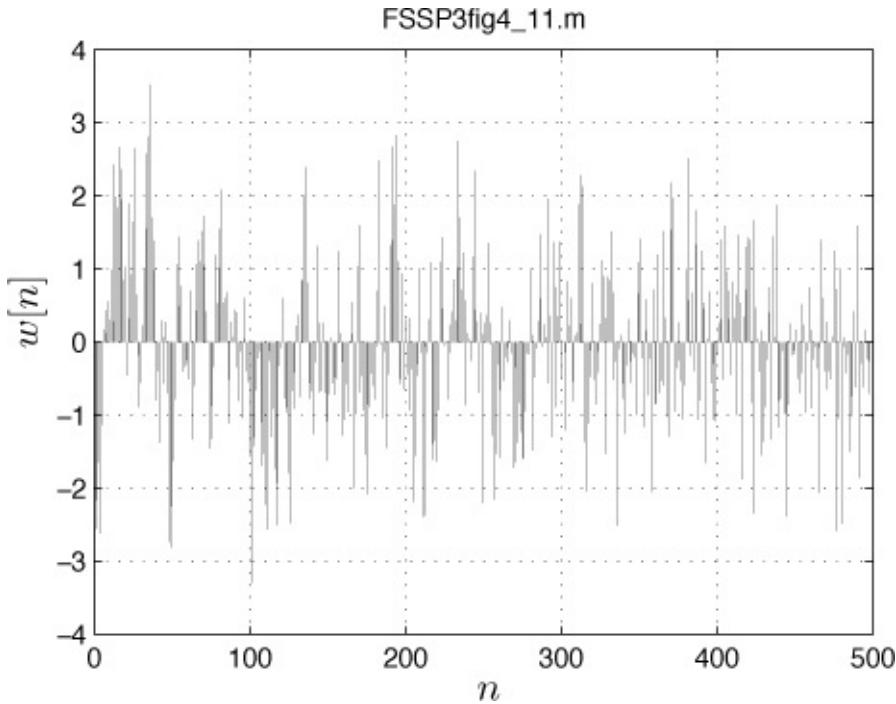
where  $w_1[n]$  is WGN with variance  $\sigma^2 = 1$  and  $M = 250$ . To employ this model we need knowledge of the *transition time*  $n = M$ . If, however, it is unknown, then conceivably it might be estimated on-line. An estimator as simple as a *sliding window* power estimator might suffice as shown in [Figure 4.10b](#). The

time at which the change takes place is indicated by a vertical line, and it is seen that the power estimator easily finds the transition point. The power estimator is given by

$$T[n] = \frac{1}{L+1} \sum_{k=n-L/2}^{n+L/2} w^2[k]$$

where  $L + 1$ , the window length, has been chosen to be  $L + 1 = 11$ . We do not pursue the modeling of this type of nonstationarity further but refer the reader to [[Basseville and Nikiforov 1993](#)] and [[Kay 1998, Chapter 12](#)].

The second type of nonstationarity is exhibited by data with a more gradual change in characteristics. Such is the case illustrated in [Figure 4.11](#) for which the power distribution with frequency changes slowly over time. This type of nonstationary random process is referred to as *locally stationary*, in that over a small enough time interval we can assume the realization *could have been generated by a stationary random process*. A good example is in the modeling of speech sounds, for which a consonant waveform appears to be locally stationary if viewed over a time window of 20 ms. In radar and sonar the clutter and reverberation, respectively, are locally stationary if viewed over a short enough time window of duration  $T$  sec. The length of the time window must be small enough so that the transmitted waveform ensonifies a region in space for which the reflections are all about the same size, and also for which the propagation loss is fairly constant. This requirement equates to  $cT \ll d$ , where  $d$  is the spatial distance for which the ensonified region is homogeneous and  $c$  is the speed of propagation. When this assumption is satisfied, the received noise process due to an ensemble of reflectors is said to be *stationary in range*.



**Figure 4.11: Realization of a random process with a slowly varying distribution of power with frequency.**

To model a locally stationary random process we can modify the AR(1) model of the previous section to allow for *temporal variation* in the filter and/or excitation noise parameters. Modifying the notation of (4.6) slightly to emphasize the time variation we have

$$w[n] = -a[1, n]w[n - 1] + u[n] \quad (4.11)$$

where  $a[1, n]$  is a *time-varying filter parameter* and  $u[n]$  is zero mean Gaussian noise but with a *variance at time n* of  $\sigma_u^2[n]$ . This model was used to generate the realization shown in [Figure 4.11](#). In particular, for [Figure 4.11](#) we have chosen  $\sigma_u^2[n] = \sigma_u^2 = 1$  so that the  $u[n]$  is WGN (a stationary random process), but a time-varying filter parameter given by

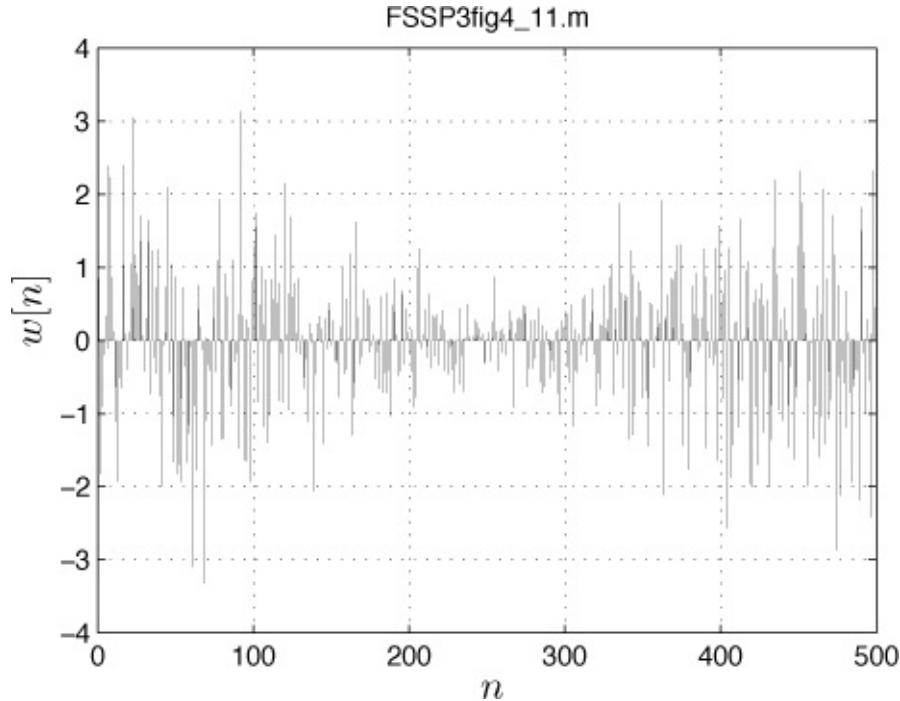
$$a[1, n] = -0.9 + \frac{n}{N} \quad n = 0, 1, \dots, N - 1 \quad (4.12)$$

to yield a nonstationary AR process. Over the duration of the realization consisting of  $N = 500$  samples, the AR(1) filter parameter evolves from  $a[1, 0] = -0.9$  to  $a[1, N - 1] \approx 0.1$ . This explains the high correlation of  $w[n]$  near the beginning of the realization and the more noise-like behavior towards the end. Also, note that because of the slow variation of  $a[1, n]$  with time, we could assume that the random process is essentially stationary over a small time window. For example, because the PSD does not change greatly for a *stationary*

AR(1) random process with the filter parameter  $a[1] = -0.9$  modified to  $a[1] = -0.8$ , we could say that a nonstationary AR(1) random process is *locally stationary* if the time window length  $\Delta n$  satisfies  $(\Delta n)/N = 0.1$  (see (4.12)). Thus, a time window of  $\Delta n = 50$  samples might allow us to use stationary processing for each 50 sample time window. Referring to [Figure 4.11](#) this is a reasonable choice since the character of the waveform does not appear to change radically over 50 samples.

The model given by (4.11) is termed a *dynamical model*. It has found widespread application as a signal model that is capable of modeling vehicle dynamics [[Bar-Shalom et al. 2001](#)], hence the name. The interested reader should consult [[Kay 1993, Chapter 13](#)] for more details and the extension of the dynamical signal model to vector random processes and its use in Kalman filters.

The model of (4.11) can also be used to model Gaussian random processes with time-varying power, an example of which is given in [Figure 4.12](#).



**Figure 4.12: Realization of a nonstationary AR(1) random process with time varying power.**

In this case we have chosen  $a[1, n] = 0$  for all  $n$  so that  $w[n] = u[n]$  and therefore the noise samples are independent. However, the power of  $u[n]$  is chosen to be time varying and is given by

$$\sigma_u^2[n] = 1 + 0.9 \cos\left(\frac{2\pi}{N}n\right).$$

This explains the large power seen at  $n = 0$  and  $n = N = 500$ , for which  $\sigma_u^2[n] = 1.9$  and the smaller power at  $n = N/2 = 250$ , for which  $\sigma_u^2[n] = 0.1$ .

Finally, a *random walk*, sometimes called a *Wiener process*, can also be easily modeled. The random walk is defined as

$$w[n] = \sum_{k=0}^n u[k] \quad n \geq 0 \quad (4.13)$$

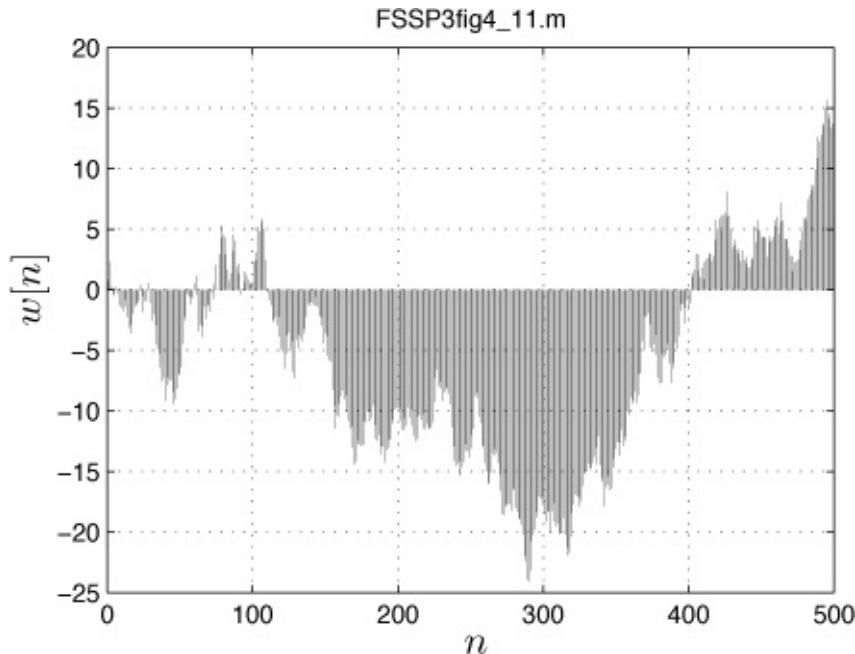
where  $u[k]$  is WGN with variance  $\sigma_u^2$  (as per the definition of WGN, the variance is constant). This type of model is generally used to model “drift”, which is the accumulation of small perturbations due to equipment errors. To see that this is a special case of (4.11), we let  $a[1, n] = -1$  and  $\sigma_u^2[n] = \sigma_u^2$ . Then, with an assumed starting time of  $n = 0$ , (4.11) becomes

$$w[n] = \begin{cases} w[n-1] + u[n] & n \geq 0 \\ 0 & n < -1 \end{cases} \quad (4.14)$$

and by “recursing” (4.14) forward with  $w[-1] = 0$ , we have

$$\begin{aligned} w[0] &= u[0] \\ w[1] &= w[0] + u[1] = u[0] + u[1] \\ w[2] &= w[1] + u[2] = u[0] + u[1] + u[2] \\ &\text{etc.} \end{aligned}$$

which is (4.13). A realization of the random walk is shown in [Figure 4.13](#).



**Figure 4.13: Realization of random walk (Wiener process).**

Its power is given by

$$\text{var}(w[n]) = (n + 1)\sigma_u^2 \quad (4.15)$$

exhibiting a nonstationarity due to an increasing power with time.

---

### Exercise 4.10 – Power versus time of random walk

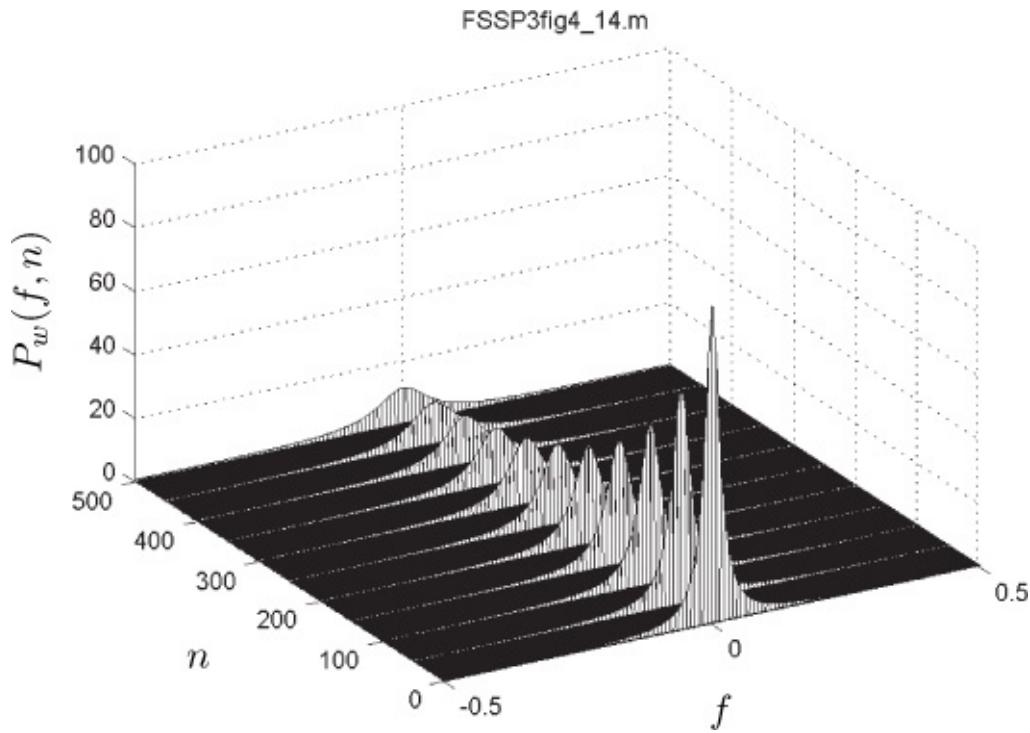
Verify the expression of (4.15). Recall that the variance of the sum of two *independent* random variables is the sum of the variances.

---

Although a nonstationary random process does not have a PSD, it is common practice to use a time-varying PSD for locally stationary random processes. Assuming the nonstationarity is a gradual one, i.e., locally stationary, we can define a time-varying PSD for the process of (4.11) as [Thostheim 1976]

$$P_w(f, n) = \frac{\sigma_u^2[n]}{|1 + a[1, n] \exp(-j2\pi f)|^2}. \quad (4.16)$$

As an example, if we let  $a[1, n] = -0.9 + 0.2(n/N)$  for  $n = 0, 1, \dots, N-1$  and  $\sigma_u^2[n] = 1$  in (4.16) and plot  $P_w(f, n)$  for  $n = 0, 50, 100, \dots, 500$ , we have the plot shown in [Figure 4.14](#). As expected, the PSD bandwidth widens as time increases and decreases in amplitude in the passband (at  $f = 0$ ) due to the decrease in the magnitude of the AR(1) filter parameter. The latter ranges from  $-0.9$  at  $n = 0$  to  $-0.7$  at  $n = 500$ .



**Figure 4.14:** Time-varying power spectral density for the AR(1) dynamical noise process. The AR(1) filter has a time-varying filter parameter and constant excitation noise variance.

The PSD shown in [Figure 4.14](#) is useful for modeling spectra typically observed for reverberation in active sonars [[Knight et al. 1981](#)].

### Exercise 4.11 – Nonstationary models with higher orders

Generalize the time-varying PSD given in [\(4.16\)](#) to one with  $p$  AR filter parameters. Could you choose the filter parameters to display a peak in the time-varying PSD at  $f = 0.25$  and a decreasing value of the peak with time? Also, give the dynamical model that would correspond to this time-varying PSD. Hint: Let  $p = 2$  and refer back to [Figure 4.7](#) for a stationary AR random process.



### Exercise 4.12 – Verification of the PSD shown in [Figure 4.14](#)

Using `ARpsd.m`, which is contained on the CD, plot the PSD for  $\sigma_u^2 = 1$  and  $a[1] = -0.9$ ,  $a[2] = -0.8$  and  $a[3] = -0.7$ . Compare your results to

those shown in [Figure 4.14](#). Note at which times the AR(1) filter parameter takes on these values from the expression  $a[1, n] = -0.9 + 0.2(n/N)$ .

---

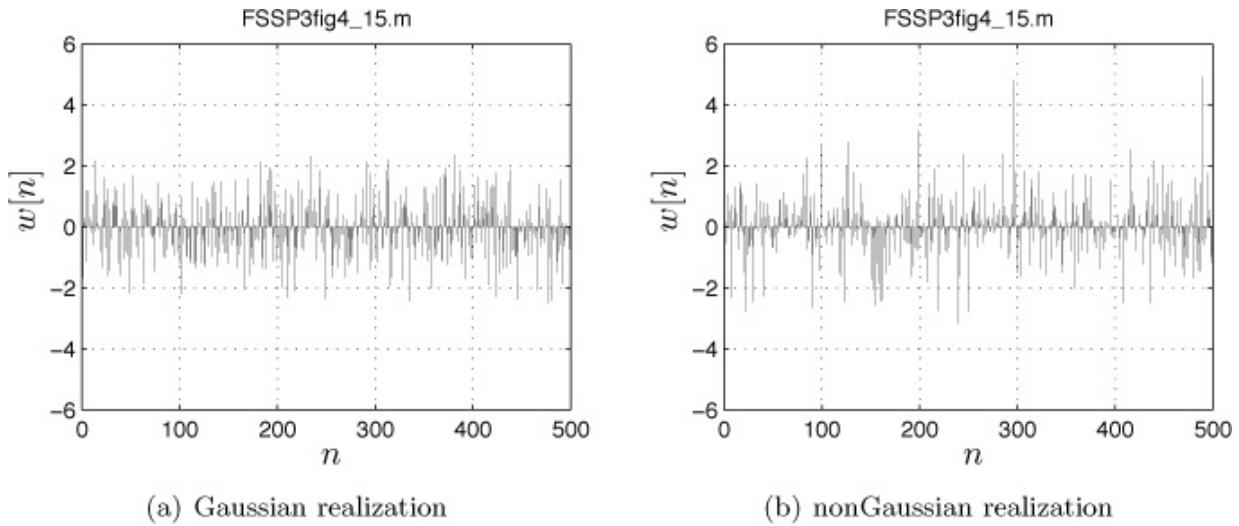
So far in our discussions we have assumed a zero mean for the noise. It frequently occurs that there is an underlying trend, which can be nonstationary. To complete the nonstationary noise model we can incorporate this trend by using one of the signal models described in [Chapter 3](#). For example, a common one might be the use of a line signal model. A linearly increasing noise mean could be due to an offset voltage that might be passed through an integrator, for example. Being undesirable we treat this component as part of the noise.

Nonstationary random processes present an especially difficult modeling problem. This is because without some physical a priori knowledge of their origin to motivate a model, one has to resort to an analysis of field data. The field data is then used to estimate and verify the proposed noise model. However, because of the nonstationarity of the data *we must be able to estimate the parameters because they change!* For a rapid rate of change this is nearly impossible. For a slow rate of change we can assume that the data is locally stationary and hence gather enough stationary data to obtain a good parameter estimate.

## 4.6. IID NonGaussian Noise

The first three models listed in [Table 4.1](#) have Gaussian PDFs. In this section we consider noise samples that can have a nonGaussian PDF. This means that the *normal* (yes, a play on words!) rule of thumb of the amplitude being “less than or equal to three standard deviations” or  $-3\sigma < w[n] < 3\sigma$  no longer applies. As a result, for a fixed average noise power  $\sigma^2$  the noise samples may exhibit large “spikes”. An example of this is shown in [Figure 4.15](#), in which 500 samples are shown for Gaussian and nonGaussian noise, both of which have  $\sigma^2 = 1$ . The particular nonGaussian noise PDF used to generate the realization of noise samples shown in [Figure 4.15b](#) is the *Laplacian* PDF given by

$$p_W(w) = \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\sqrt{\frac{2}{\sigma^2}}|w|\right) \quad -\infty < w < \infty. \quad (4.17)$$

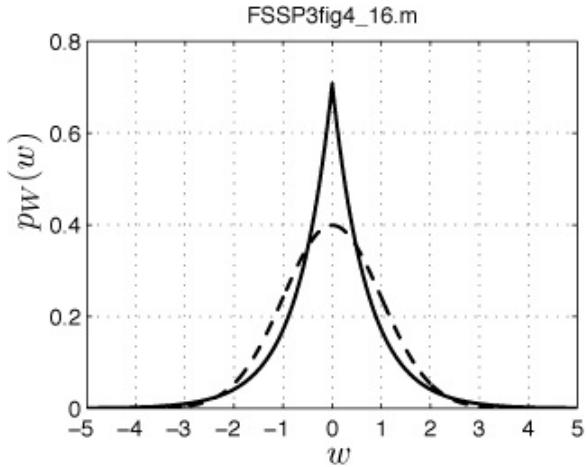


**Figure 4.15: Realizations of Gaussian and IID nonGaussian noise. Each process has a variance  $\sigma^2 = 1$ .**

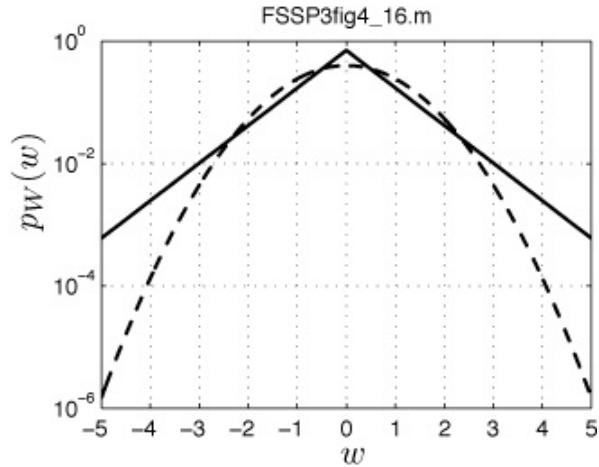
It is seen that there are no Gaussian noise samples that exceed  $\pm 3\sigma = \pm 3$  but many nonGaussian noise samples that do so. This is because the probability of exceeding  $3\sigma$ , which can be computed from

$$P[w[n] > 3\sigma] = \int_{3\sigma}^{\infty} p_W(w)dw$$

is larger for the Laplacian PDF than for the Gaussian PDF as given by (4.1). It is said that the area in the “tails” of the PDF is larger. In fact, for a Gaussian PDF the tails fall off as  $\exp(-aw^2)$ , while for the Laplacian PDF, they fall off only as  $\exp(-a|w|)$ , a much slower rate. To appreciate the differences we plot the two PDFs on linear and logarithmic scales in [Figure 4.16](#). It is seen that the values of the Laplacian PDF are much larger for  $w > 3\sigma = 3$ , and at  $w = 5\sigma = 5$ , it is more than two orders of magnitude larger (a factor of more than 100).



(a) linear vertical scale



(b) logarithmic vertical scale

**Figure 4.16: Probability density functions for a Gaussian PDF (dashed curve) and Laplacian PDF (solid curve), both for  $\sigma^2 = 1$ .**

Large noise spikes are indicative of a nonGaussian PDF. Thus, it has been used to model such physical noise processes as man-made noise [[Middleton 1973](#)], acoustic transients [[Dwyer 1983](#)], and geomagnetic noise [[Schweiger 1982](#)].

### Exercise 4.13 – Probability of a spike

Use the MATLAB program `Laplacian_gendata.m`, which is contained on the CD, to generate samples of Laplacian noise. Next, estimate the probability that the Laplacian noise will exceed 3 if  $\sigma^2 = 1$ . Compare this to the probability that Gaussian noise will exceed 3, which is 0.0013. Do so by running the program with  $N = 100,000$  and count the number of exceedances and divide by  $N$ . Then, compare your value to the theoretical value obtained by integrating the PDF from 3 to  $\infty$  as

$$P[w > 3] = \int_3^\infty p_W(w) dw$$

where  $p_W(w)$  is given by (4.17).



Since the nonGaussian noise model is assumed to consist of IID samples, the random process can be shown to be stationary. Its PSD, because of the independence assumption, is flat and is given by

$$P_w(f) = \sigma^2 \quad -1/2 \leq f \leq 1/2.$$

To model nonGaussian noise processes with an arbitrary PSD is difficult. This is because, unlike a Gaussian random process whose PSD can be shaped by linear filtering and still remain Gaussian, the filtering of a nonGaussian random process will radically change its PDF. The interested reader may wish to consult [[Kay 2010](#)] for one possible approach to this problem. Hence, for our purposes all nonGaussian noise processes will be assumed to be IID and hence, have a flat PSD.

---



### **Linear filtering alters the PDF of an IID nonGaussian random process**

The Gaussian random variable enjoys a reproducing property that few other random variables possess. It can be scaled and added to other independent Gaussian random variables and still remain Gaussian. Hence, if  $x_1$  and  $x_2$  are Gaussian random variables and are independent, then  $y = a_1x_1 + a_2x_2$  is still a Gaussian random variable. This result relies on the fact that convolving two Gaussian functions produces another Gaussian function, although wider in width and with a diminished amplitude. Since all linear filtering operations, even time-varying ones, produce linear combinations of the inputs, this Gaussian *reproducing property* will hold. Thus, a Gaussian random process at the input to a linear filter will produce a Gaussian random process at the output. The brave reader might try to convolve two Laplacian PDFs together to see if the result is still a Laplacian PDF.




---

Generation of IID nonGaussian noise samples is generally accomplished by using a nonlinear transformation of a uniformly distributed random variable  $u$  on  $(0, 1)$ . Outcomes of the latter random variable are easily generated in MATLAB using the `rand(N, 1)` command to produce  $N$  outcomes. The required nonlinear transformation is the *inverse cumulative distribution function*. Specifically, defining the cumulative distribution function (CDF)  $F_X(x)$  as

$$u = F_X(x) = \int_{-\infty}^x p_X(t)dt$$

the required nonlinear transformation  $g(u)$  is given by  $x = g(u) = F_X^{-1}(u)$ , where  $F_X^{-1}(\cdot)$  denotes the inverse function of  $F_X(x)$ . If a uniformly distributed random variable on  $(0, 1)$  is transformed as such, then the output random variable  $x$  will have the desired PDF  $p_X(x)$ . As an example, for a Laplacian random variable with PDF given by (4.17) with  $\sigma^2 = 1$  the result is

$$g(u) = \begin{cases} \frac{1}{\sqrt{2}} \ln(2u) & 0 < u \leq 1/2 \\ \frac{1}{\sqrt{2}} \ln \frac{1}{2(1-u)} & 1/2 < u < 1. \end{cases} \quad (4.18)$$


---

### Exercise 4.14 – Modifying the variance of a random variable

The transformation given in (4.18) produces an outcome of a Laplacian random variable with variance  $\sigma^2 = 1$ . To obtain a Laplacian random variable with an arbitrary variance we need only form  $\sigma g(u)$ , where  $\sigma = \sqrt{\sigma^2}$ . Prove that this is true by determining the variance of  $y = \sigma x$ , where  $x$  is a zero mean random variable. Hint: Use the fact that  $E[g(x)] = \int_{-\infty}^{\infty} g(x)p_X(x)dx$  and the definition of the variance for a zero mean random variable.




---

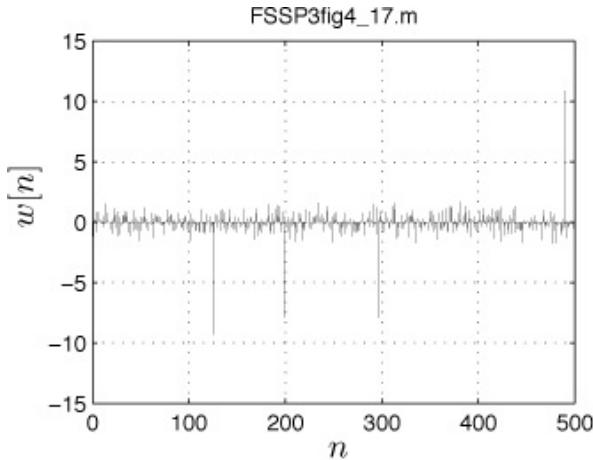
A second type of nonGaussian PDF that is useful in practice is the *Gaussian mixture* PDF. It is easily generated if uniform and Gaussian random number generators are available. The latter is available in MATLAB as the function `randn(N, 1)`, which will generate  $N$  outcomes of a Gaussian random variable, each with a zero mean and unity variance. Also, it can be extended to the generation of multivariate Gaussian mixture PDFs, useful in classification problems [Duda et al. 2001]. A simple example is the two component Gaussian mixture PDF given by

$$p_W(w) = (1 - \epsilon) \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2\sigma_1^2}w^2\right) + \epsilon \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_2^2}w^2\right). \quad (4.19)$$

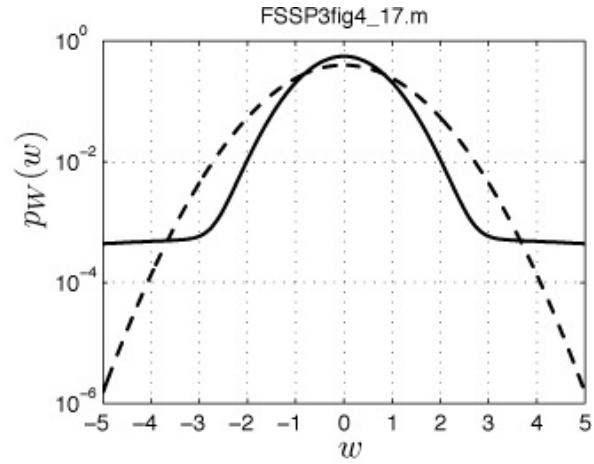
It mixes two Gaussian PDFs with variances  $\sigma_1^2$  and  $\sigma_2^2$ , and with probabilities  $1 - \epsilon$  and  $\epsilon$ , respectively. Typically,  $\sigma_2^2$  is large relative to  $\sigma_1^2$ . Thus, with probability  $\epsilon$  (chosen to be small), the outcome will potentially be large in magnitude due to the large value of  $\sigma_2^2$ . A large level amplitude  $|w| > \sigma_2$  will occur with probability of about  $0.32\epsilon$ . An example of a realization is given in Figure 4.17a along with the PDF shown in Figure 4.17b as the solid curve. The PDF parameters are

$\sigma_1^2 = 1/2$ ,  $\sigma_2^2 = 50$ , and  $\epsilon = 0.01$ . Note that the average power or variance can be shown to be

$$\sigma^2 = (1 - \epsilon)\sigma_1^2 + \epsilon\sigma_2^2$$



(a) realization



(b) PDF

**Figure 4.17: Realization and PDF for Gaussian mixture noise. The Gaussian mixture PDF is shown as the solid curve and a Gaussian PDF of the same variance ( $\sigma^2 = 1$ ) by the dashed curve for comparison of the PDF “tails”.**

which for this example is  $\sigma^2 \approx 1$ . Many spikes with amplitude  $|w| > \sigma_2 \approx 7$  are seen. The probability of observing a spike is the probability of a high variance outcome, which is  $\epsilon = 0.01$ , times the probability that it exceeds  $\sigma_2$  in magnitude, which is 0.32. Thus, the spike probability is 0.0032 and for  $N = 500$ , we expect about 1.6 spikes. We actually observe 4. This modeling is especially appropriate when we are trying to model noise that is Gaussian most of the time (with probability  $1 - \epsilon$ ) but occasionally is contaminated by spikes (with probability  $\epsilon$ ).

### Exercise 4.15 – Relative frequency of occurrence of spikes

Use the MATLAB program `Gaussmix_gendata.m`, which is contained on the CD, with the previously given values of  $\sigma_1^2$ ,  $\sigma_2^2$ , and  $\epsilon$  to generate 100,000 samples of noise. How many spikes of values  $|w| > 7$  do you observe? Is this what you would expect?



The presence of spikes, although infrequent, can produce poorly performing signal processing systems. In an energy detector each spike may cause a threshold crossing if the detector was designed assuming only Gaussian noise and therefore, a small probability that the noise samples would exceed  $3\sigma$ . For the Gaussian mixture example with a probability of a spike of 0.01, the probability of false alarm could increase from a design value, say  $P_{FA} = 10^{-7}$  to  $P_{FA} = 2 \times 10^{-3}$ .

## 4.7. Randomly Phased Sinusoids

For noise that is due to interference such as 60 Hz power line pickup, machine noise, or a narrowband jammer the usual model is a sum of sinusoids. This is nearly the same model as used for a signal (see [Table 3.2](#)) and is given by

$$w[n] = \sum_{i=1}^p A_i \cos(2\pi f_i n + \phi_i) \quad (4.20)$$

except that the sinusoidal parameters  $\{A_i, f_i, \phi_i\}$  are almost always unknown. This is because  $w[n]$ , being an interference, is not under our control. Also, to model interferences that are periodic we can use [\(4.20\)](#) if we restrict the frequencies to be harmonically related. Then, a Fourier series representation is appropriate if the frequencies are restricted to be  $f_i = i/M$ ,  $i = 1, 2, \dots, M/2$ , assuming a zero DC component. Periodic interferences are encountered frequently, and are referred to as electromagnetic interference (EMI), due to powerlines and arcing motors.

The assumptions on the sinusoidal parameters will determine the PDF of  $w[n]$ . Two models are typically used. They are both known generically as the *randomly phased sinusoid* model. The first assumes that each phase  $\phi_i$  is uniformly distributed on  $(-\pi, \pi)$  and all the phases are independent of each other. In this case the PSD is

$$P_w(f) = \sum_{i=1}^p \frac{A_i^2}{4} \delta(|f| - f_i) \quad -1/2 \leq f \leq 1/2.$$

The PDF of  $w[n]$  is difficult to determine since it will be nonGaussian. On the other hand, if we further assume that the amplitudes are Rayleigh distributed random variables and independent of all other amplitudes and phases, then the PDF will be Gaussian. In particular, letting  $P_i = E[A_i^2]/2$ , the PSD becomes

$$P_w(f) = \sum_{i=1}^p \frac{P_i}{2} \delta(|f| - f_i) \quad -1/2 \leq f \leq 1/2$$

and  $w[n]$  is a Gaussian random process [Kay 2006, pg. 689]. In either case, the PSD is a set of Dirac impulses with each pair located at  $\pm f_i$ .

As already described in [Section 3.6.1](#), for the case of Rayleigh distributed amplitudes we can write  $w[n] = \sum_{i=1}^p w_i[n]$  as a sum of cosines and sines. Each  $w_i[n]$  takes the form

$$\begin{aligned} w_i[n] &= A_i \cos(2\pi f_i n + \phi_i) \\ &= \alpha_{1i} \cos(2\pi f_i n) + \alpha_{2i} \sin(2\pi f_i n) \end{aligned} \quad (4.21)$$

where  $\alpha_i \sim \mathcal{N}(\mathbf{0}, \sigma_{\alpha_i}^2 \mathbf{I})$ , and  $\sigma_{\alpha_i}^2 = P_i$ . Thus, it is easy to generate a Gaussian randomly phased sinusoid and thus the sum of sinusoids given by [\(4.20\)](#).

---

### **Exercise 4.16 – Mean and power of a randomly phased sinusoid with Gaussian sine and cosine amplitude components**

Show that the mean of  $w_i[n]$  is zero and the power, i.e., variance is  $P_i$ .

Hint: For zero mean random variables  $x$  and  $y$ , it follows that  $\text{cov}(x, y) = E[xy] - E[x]E[y] = E[xy]$  and if they are also uncorrelated, i.e,  $\text{cov}(x, y) = 0$ , then  $E[xy] = 0$ .




---

## **4.8. Lessons Learned**

- A white Gaussian noise random process is the starting point for modeling noise. We use it unless the noise samples are heavily correlated and/or exhibit nonGaussian behavior, which is typically the presence of amplitude “spikes”.
- Conversion of a flat, i.e., white, PSD model to a nonflat, i.e., colored, PSD model is accomplished via linear filtering.
- Noise waveforms appear to be slowly varying and/or sinusoidal for narrowband PSDs and rapidly varying or more “noise-like” for broadband PSDs.
- The AR model is a convenient one for colored noise. It can model any PSD for a large enough number of filter parameters.
- Most noise processes are nonstationary if observed for a long enough time

interval. Analyze short segments to be able to apply algorithms designed for stationary noise.

- Use a dynamical AR model for nonstationary, but locally stationary, noise modeling.
- The Gaussian noise assumption must be discarded if noise spikes are observed. A rule of thumb is that for a data record of  $N$  samples, a spike occurs if it exceeds  $3\sigma$  in magnitude. Thus, we would expect to observe more than  $0.0026N$  spikes in the record if the noise is nonGaussian.
- It is very hard to jointly model both the PSD and PDF for nonGaussian noise. Use an IID nonGaussian model since most algorithms are particularly sensitive to outliers and less so to noise coloration.

## References

- Bar-Shalom, Y., X. Rong Li, T. Kirubarajan, *Estimation with Application to Tracking and Navigation*, J. Wiley, NY, 2001.
- Basseville, M., I.V. Nifkorov, *Detection of Abrupt Changes: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Burdic, W.S., *Underwater Acoustic Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Duda, R.O., P.E. Hart, D.G. Stork, *Pattern Classification, Second Ed.*, J. Wiley, NY, 2001.
- Dwyer, R., “A Technique for Improving Detection and Estimation of Signals Contaminated by Under Ice Noise”, *Journal Acoustical Society of America*, Volume 74, Issue 1, pp. 124–130, July 1983.
- Jackson, L., *Signals, Systems, and Transforms*, Addison-Wesley, NY, 1988.
- Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory, Vol. I*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., *Intuitive Probability and Random Processes using MATLAB*, Springer, NY, 2006.
- Kay, S., “Representation and Generation of NonGaussian Wide Sense Stationary Random Processes with Arbitrary PSDs and a Class of PDFs”,

- IEEE Trans. on Signal Processing*, pp. 448–458, July 2010.
- Knight, W., R.G. Pridham, S. Kay, “Digital Signal Processing for Sonar”, *Proc. of the IEEE*, pp. 1451–1506, Nov. 1981.
- Middleton, D., “Man-Made Noise in Urban Environments and Transportation Systems: Models and Measurements”, *IEEE Trans. on Communications*, Vol. 21., pp. 1232–1241, Nov. 1973.
- Middleton, D., A.D. Spaulding, “Elements of Weak Signal Detection in NonGaussian Noise Environments”, in *Advances in Statistical Signal Processing*, Vol. 2, pp. 142–215, JAI Press, Greenwich, CT, 1993.
- Nathanson, F.E., *Radar Design Principles*, SciTech Pubs., NJ, 1999.
- Schafer, R.W., L.R. Rabiner, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
- Schweiger, J., “Evaluation of Geomagnetic Activity in the MAD Frequency Band (0.04 to 0.6 Hz)”, Masters Thesis, Naval Postgraduate School, Oct. 1982.
- Tjostheim, D., “Spectral Generating Operators for NonStationary Processes”, *Advances Applied Probability*, Vol. 8, pp. 831–846, 1976.

## Appendix 4A. Random Process Concepts and Formulas

A *discrete-time* random process  $x[n]$  is a sequence of random variables defined for every integer  $-\infty < n < \infty$ . A random process is wide sense stationary (WSS) if it has a *mean*

$$E(x[n]) = \mu$$

that does not depend on  $n$  (it is a constant for all  $n$ ), and an autocorrelation sequence (ACS)

$$r_x[k] = E[x[n]x[n + k]]$$

that depends only on the *lag*  $k$  between the two samples and not their absolute positions, i.e., not on  $n$ . When the random process is used to model noise, we usually assume that  $\mu = 0$ . Some useful properties of the ACS are

$$\begin{aligned} r_x[0] &\geq |r_x[k]| \quad \text{all } k \\ r_x[-k] &= r_x[k] \quad \text{(sequence is even)} \end{aligned}$$

and  $r_x[0]$  is positive. The value of  $r_x[0]$  gives the average power  $r_x[0] = E[x^2[n]]$

of the random process.

The Fourier transform of the ACS is the power spectral density (PSD) defined as

$$P_x(f) = \sum_{k=-\infty}^{\infty} r_x[k] \exp(-j2\pi fk) \quad -1/2 \leq f \leq 1/2 \quad (4A.1)$$

and this relationship is known as the Wiener-Khintchine theorem. Note that the PSD is periodic with period one (we always use as our basic period the interval  $[-1/2, 1/2]$ ). It must be a real and nonnegative function, since it yields the average power in a band by integrating over that band as

$$\begin{aligned} \text{Average power in band } [f_1, f_2] &= \int_{-f_2}^{-f_1} P_x(f) df + \int_{f_1}^{f_2} P_x(f) df \\ &= 2 \int_{f_1}^{f_2} P_x(f) df \end{aligned}$$

where the latter step used the even or symmetry property, which is  $P_x(-f) = P_x(f)$ . We can also write the PSD as

$$P_x(f) = \sum_{k=-\infty}^{\infty} r_x[k] \cos(2\pi fk) \quad -1/2 \leq f \leq 1/2$$

since  $r_x[k]$  is an even sequence. Since the PSD is the Fourier transform of the ACS, the inverse Fourier transform of the PSD yields the ACS as

$$r_x[k] = \int_{-\frac{1}{2}}^{\frac{1}{2}} P_x(f) \exp(j2\pi fk) df.$$

As a special case, if the random process is white noise, i.e., all the samples are uncorrelated, then  $r_x[k] = \sigma^2 \delta[k]$ , where  $\delta[k]$  is the discrete-time delta sequence. Then, the PSD becomes from (4A.1)  $P_x(f) = \sigma^2$ . Finally, a more intuitive but equivalent definition of the PSD is given by

$$P_x(f) = \lim_{M \rightarrow \infty} \frac{1}{2M+1} E \left[ \left| \sum_{n=-M}^M x[n] \exp(-j2\pi fn) \right|^2 \right].$$

If a WSS random process  $u[n]$  is input to a linear shift invariant (LSI) system, then the output random process  $x[n]$  is also WSS. The PSDs are related as

$$P_x(f) = |H(f)|^2 P_u(f)$$

where  $H(f)$  is the frequency response of the LSI system.

The ACSs can also be related. If the impulse response of the LSI system is given by  $h[n]$ , which is the inverse Fourier transform of  $H(f)$ , then

$$\begin{aligned} r_x[k] &= h[k] * h[-k] * r_u[k] \\ &= \sum_{m=-\infty}^{\infty} h[k-m] \sum_{l=-\infty}^{\infty} h[-l] r_u[m-l] \end{aligned} \quad (4A.2)$$

where  $*$  denotes discrete-time convolution.

Further details are found in [Kay 2006].

## Appendix 4B. Gaussian Random Processes

The Gaussian random process is without a doubt the most important theoretical and practical noise model used in signal processing. Specifically, the important characteristics of a Gaussian random process are

1. It is physically motivated by the central limit theorem (see also [Figures 3.13](#) and [3.14](#)).
2. It is a mathematically tractable model.
3. The joint PDF of any set of samples is a multivariate Gaussian PDF, which enjoys many useful properties.
4. Only the first two moments, the mean sequence and the covariance sequence, are required to completely describe it. As a result,
  - a. in practice the joint PDF can be estimated by estimating only the first two moments.
  - b. if the Gaussian random process is wide sense stationary, then it is also stationary.
5. The processing of a Gaussian random process by a linear filter does not alter its Gaussian nature, but only modifies the first two moments. The modified moments are easily found.

More specifically, a Gaussian random process is one for which any set of samples  $x[n_0], x[n_1], \dots, x[n_{N-1}]$  are jointly distributed according to a multivariate Gaussian PDF. If the samples are taken at successive times to generate the random vector  $\mathbf{x} = [x[0] \ x[1] \ \dots \ x[N-1]]^T$ , then assuming a zero mean WSS random process, the covariance matrix takes the form

$$\mathbf{C} = \begin{bmatrix} r_x[0] & r_x[1] & \dots & r_x[N-1] \\ r_x[1] & r_x[0] & \dots & r_x[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[N-1] & r_x[N-2] & \dots & r_x[0] \end{bmatrix} \quad (4B.1)$$

and has the special form of a *symmetric Toeplitz matrix*, where each northwest-southeast diagonal of the covariance matrix has the same element. The multivariate PDF can be written as

$$p_X(\mathbf{x}) = \frac{1}{(2\pi)^{N/2} \det^{1/2}(\mathbf{C})} \exp \left[ -\frac{1}{2} \mathbf{x}^T \mathbf{C}^{-1} \mathbf{x} \right]. \quad (4B.2)$$

Note that only the covariance matrix of (4B.1) or equivalently the ACS is needed to specify the entire PDF. This can be estimated in practice (see [Chapter 6](#)). Additionally, for large data records, i.e., when  $N$  is large, the PDF of (4B.2) can be approximated as [Kay 1993, pg. 80]

$$\ln p_X(\mathbf{x}) \approx -\frac{N}{2} \ln 2\pi - \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left[ \ln P_x(f) + \frac{I(f)}{P_x(f)} \right] df$$

where  $I(f)$  is the periodogram given by

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f n) \right|^2.$$

The approximation is valid whenever  $N$  is large enough so that  $r_x[k] \approx 0$  for  $k > N$ . Using this expression we can find simple forms for the maximum likelihood estimator and the Cramer-Rao lower bound (see for example [Algorithms 9.7, 9.8, and 10.4](#)). The practical utility is that the processing may be done in the frequency domain, leading to more intuitive approaches as well as computational advantages afforded by the use of the fast Fourier transform (FFT).

If the input to an LSI system is a Gaussian random process, then so is the output. This allows the PDF to be easily found at the output of a filter by using the formula (4A.2) to compute the output ACS sequence for use in (4B.1) and then in (4B.2).

For nonlinear processing of data the Gaussian random process yields a simple expression for the higher order moments. Assuming a zero mean WSS Gaussian random process, it can be shown that the fourth-order moment is [Kay 2006, pg. 684]

$$E[x[m]x[n]x[k]x[l]] = r_x[m-n]r_x[k-l] + r_x[m-k]r_x[n-l] + r_x[m-l]r_x[n-k].$$

## Appendix 4C. Geometrical Interpretation of AR PSD

Consider an AR(2) PSD given by

$$P_w(f) = \frac{\sigma_u^2}{|1 + a[1] \exp(-j2\pi f) + a[2] \exp(-j4\pi f)|^2}. \quad (4C.1)$$

The poles of the denominator polynomial  $A(z) = 1 + a[1]z^{-1} + a[2]z^{-2}$  can be found by rooting the polynomial  $z^2A(z)$ . Assume that the poles are complex conjugates and are given by  $r \exp(\pm j2\pi f_0)$ , where  $r < 1$ . Then, we can write the denominator of  $P_w(f)$  by letting  $z = \exp(j2\pi f)$  as

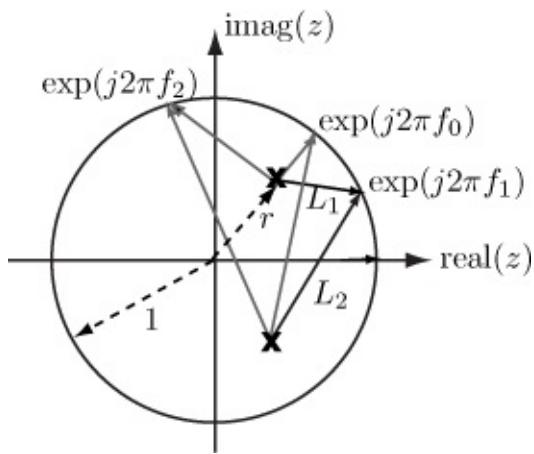
$$\begin{aligned} 1 + a[1] \exp(-j2\pi f) + a[2] \exp(-j4\pi f) &= 1 + a[1]z^{-1} + a[2]z^{-2} \\ &= z^{-2}(z^2 + a[1]z + a[2]) \\ &= z^{-2}(z - r \exp(j2\pi f_0)) \\ &\quad \cdot (z - r \exp(-j2\pi f_0)) \end{aligned}$$

and therefore we have that

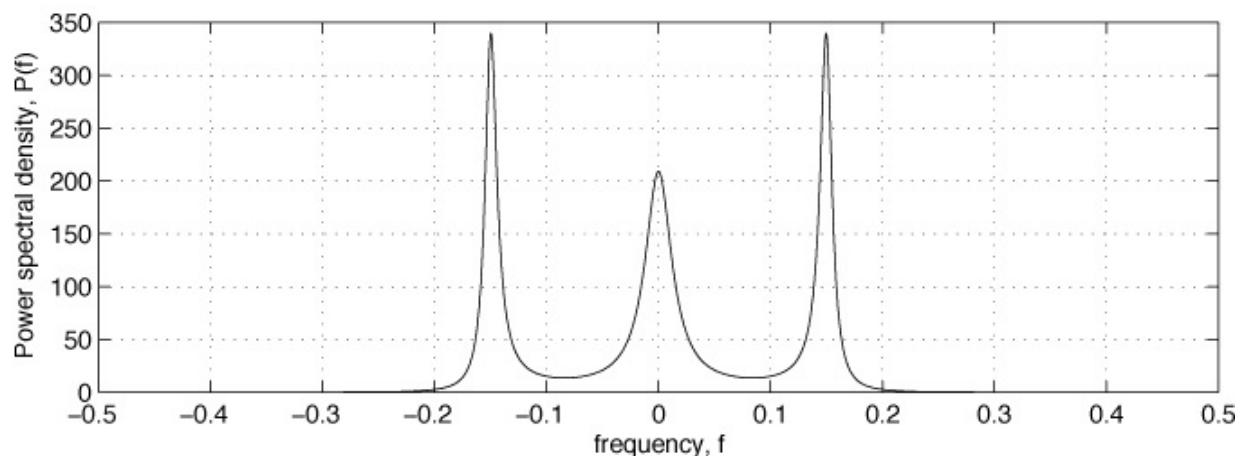
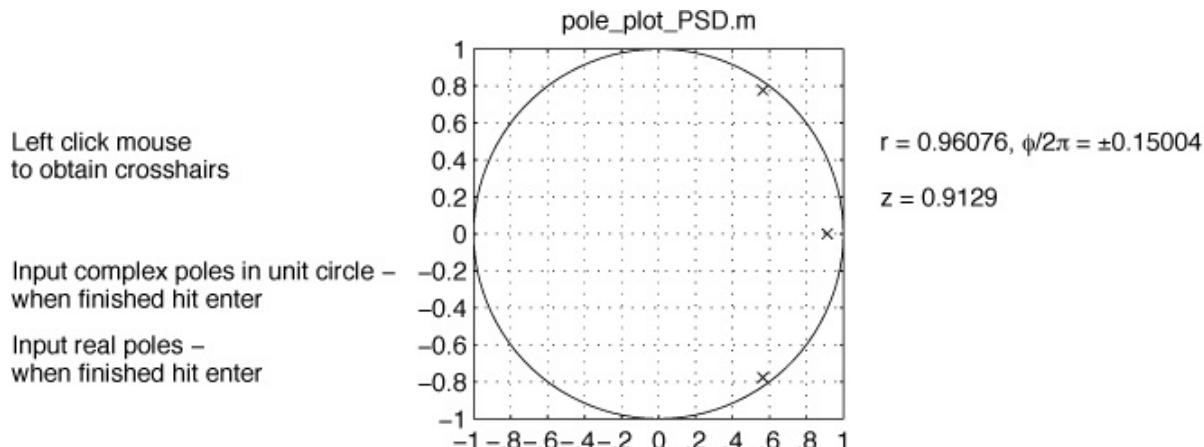
$$\begin{aligned} |1 + a[1] \exp(-j2\pi f) + a[2] \exp(-j4\pi f)| &= |z^{-2}(z - r \exp(j2\pi f_0)) \\ &\quad \cdot (z - r \exp(-j2\pi f_0))| \\ &= |(z - r \exp(j2\pi f_0))| \\ &\quad \cdot |(z - r \exp(-j2\pi f_0))| \end{aligned}$$

for  $-1/2 \leq f \leq 1/2$  or equivalently for  $z = \exp(j2\pi f)$  on the unit circle in the  $z$ -plane. Consider the first factor  $|(z - r \exp(j2\pi f_0))|$  and note that it is the *length of the vector* drawn from the pole  $z_{p1} = r \exp(j2\pi f_0)$  to a point on the unit circle, say  $z = \exp(j2\pi f)$ . This is shown in [Figure 4C.1](#) for several frequencies.

Note that the vector length  $L_1$  is smallest at  $f = f_0$ , and since this factor is in the denominator of [\(4C.1\)](#), the PSD is largest at this frequency. The poles for negative frequencies (those in the bottom half of the unit circle) have less effect on the PSD since their lengths change less for positive frequencies. As the frequency moves away from  $f = f_0$ , the length of the vector increases and hence the PSD decreases. In fact, the PSD value at frequency  $f_1$  is just  $1/(L_1 L_2)^2$ , assuming  $\sigma_u^2 = 1$  in [\(4C.1\)](#). This interpretation allows one to visualize the effect of pole placement on the AR PSD. Another example using the program `pole_plot_PSD.m` is shown in [Figure 4C.2](#).



**Figure 4C.1: Illustration of vector length in complex z-plane. A vector is drawn from each pole (denoted by the “x”, where  $z_{p1} = r \exp(j2\pi f_0)$  and  $z_{p2} = r \exp(-j2\pi f_0)$ ) to a point on the unit circle.**



**Figure 4C.2: Pole plot and corresponding PSD using the program**

`pole_plot_PSD.m`.

## Appendix 4D. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**4.1** By running `WGNgendata.m` you should obtain 50 outcomes exceeding 3. The MATLAB program `FSSP3exer4_1.m`, which is contained on the CD, can be run to obtain the solution.

**4.2** For  $N = 1000$  the estimate is  $\hat{\sigma}^2 = 3.5646$  while for  $N = 10000$  it is  $\hat{\sigma}^2 = 4.0087$ . The MATLAB program `FSSP3exer4_2.m`, which is contained on the CD, can be run to obtain the solution.

**4.3** If  $a[1] = 0$ , then from (4.3),  $P_w(f) = \sigma_u^2$ , which is just a flat PSD.

**4.4** The PSD obtained is the same as shown in [Figure 4.4b](#) except it is shifted to the right so that it is “centered” about  $f = 1/2$ . Note that because we plot the periodic PSD only over the frequency interval  $[-1/2, 1/2]$ , the PSD appears to be split. Plotting it over a wider interval, say  $[-1, 1]$ , will reveal the shifting to the right by  $1/2$ . The reader can liken this shifting to a *circular shift* on a circle. The PSD can now be said to be a highpass PSD. The effect of negating the sign of  $a[1]$  is to convert the lowpass PSD to a highpass PSD. The MATLAB program `FSSP3exer4_4.m`, which is contained on the CD, can be run to obtain the solution.

**4.5** The plot should indicate a lowpass type of filter with a peak at  $f = 0$  of  $|H(0)| = 10$ . The MATLAB program `FSSP3exer4_5.m`, contained on the CD, can be run to obtain the solution.

**4.6** The denominator can be expressed in terms of real quantities by the following algebraic equivalences.

$$\begin{aligned}|1 + a[1] \exp(-j2\pi f)|^2 &= (1 + a[1] \exp(-j2\pi f))^*(1 + a[1] \exp(-j2\pi f)) \\&= (1 + a[1] \exp(j2\pi f))(1 + a[1] \exp(-j2\pi f)) \\&= 1 + a[1] \exp(-j2\pi f) + a[1] \exp(j2\pi f) + a^2[1] \\&= 1 + a^2[1] + a[1](\exp(-j2\pi f) + \exp(j2\pi f)) \\&= 1 + a^2[1] + 2a[1] \cos(2\pi f).\end{aligned}$$

The peak or maximum of the PSD occurs when the denominator is minimized. The only term that depends on the sign of  $a[1]$  is  $2a[1]$ .

$\cos(2\pi f)$ . Thus, the minimum of the denominator is attained when the latter term is most negative. For  $a[1] < 0$ , this is at  $f = 0$  and for  $a[1] > 0$ , this is at  $f = 1/2$ . The denominator value becomes  $1 + a^2[1] - 2|a[1]| = (1 - |a[1]|)^2$  in each case.

- 4.7** For  $r = 0.9$  the PSD should appear similar to that in [Figure 4.7a](#). When the radius of the pole is increased to  $r = 0.98$  the peak value of the PSD at  $f_0 = 0.25$  should be greatly increased. Note that the exact pole locations as input by the user cursor are printed out on the screen. In general, as the radius of the pole increases toward one, the PSD peak value will keep increasing. Also, the bandwidth will become very small. This is easily explained since the length of the vector from the pole to the unit circle will decrease (see [Appendix 4C](#)).
- 4.8** For complex conjugate poles with  $r = 0.9$  the data record should appear similar to that seen in [Figure 4.7b](#). For  $r = 0.98$  the PSD is very narrow, as observed from [Exercise 4.7](#), and resembles the PSD of a sinusoid at  $f_0 = 0.25$  (the number of samples per cycle should be 4). The data record is therefore observed to resemble a sinusoid, at least over a short time interval. The MATLAB program `FSSP3exer4_8.m`, which is contained on the CD, can be run to obtain the solution.
- 4.9** For  $p = 4$  the fit is fair but for  $p = 10$  the fit is nearly perfect except for a slight error at  $f = 0$ . The latter is due to the approximation of a “cusp” (nondifferentiable at that point) by the inverse of a polynomial. The MATLAB program `FSSP3exer4_9.m`, which is contained on the CD, can be run to obtain the solution.
- 4.10** We use the property that the variance of a sum of random variables  $x$  and  $y$  is  $\text{var}(x + y) = \text{var}(x) + \text{var}(y) + 2\text{cov}(x, y)$ . Therefore, for independent (and hence uncorrelated) random variables for which  $\text{cov}(x, y) = 0$ , we have that  $\text{var}(x + y) = \text{var}(x) + \text{var}(y)$ , and this extends to any number of independent random variables. Thus, for independent random variables  $\{u[0], u[1], \dots, u[n]\}$  (all having the same variance  $\sigma_u^2$ ) we have

$$\begin{aligned}
\text{var}(w[n]) &= \text{var} \left( \sum_{k=0}^n u[k] \right) \\
&= \sum_{k=0}^n \text{var}(u[k]) \\
&= \sum_{k=0}^n \sigma_u^2 = (n+1)\sigma_u^2.
\end{aligned}$$

**4.11** The generalization to a time-varying AR( $p$ ) PSD is

$$P_w(f, n) = \frac{\sigma_u^2[n]}{|1 + a[1, n] \exp(-j2\pi f) + \cdots + a[p, n] \exp(-j2\pi fp)|^2}.$$

Choosing  $p = 2$  to obtain two time-varying “poles” at  $r \exp(\pm j2\pi f_0)$ , we would have

$$\begin{aligned}
a[1, n] &= -2r[n] \cos(2\pi f_0 n) \\
a[2, n] &= r^2[n]
\end{aligned}$$

where  $f_0 = 0.25$  and  $r[n]$  is any decreasing sequence, for example  $r[n] = 0.9 - n/(2N)$  for  $n = 0, 1, \dots, N-1$ . The excitation noise variance could be chosen as constant, for example,  $\sigma^2[n] = 1$ . The dynamical model would then be

$$\begin{aligned}
w[n] &= -a[1, n]w[n-1] - a[2, n]w[n-2] + u[n] \\
&= 2r[n] \cos(2\pi f_0 n)w[n-1] - r^2[n]w[n-2] + u[n]
\end{aligned}$$

where  $u[n]$  is WGN with variance  $\sigma_u^2 = 1$ .

**4.12** By running `ARpsd.m` for the different AR(1) filter parameter values and plotting the PSDs obtained, you should see the same PSD as in [Figure 4.14](#). The PSDs at times  $n = 0$ ,  $n = N/2 = 250$  and  $n = N = 500$  should correspond to the PSDs observed for  $a[1] = -0.9$ ,  $a[1] = -0.8$ , and  $a[1] = -0.7$ , respectively. The MATLAB program `FSSP3exer4_12.m`, which is contained on the CD, can be run to obtain the solution.

**4.13** The estimated probability of a spike is the number of spike occurrences out of 100,000 divided by 100,000. This is found to be  $676/100000 = 0.00676$ . The theoretical probability for  $\sigma^2 = 1$  is

$$\begin{aligned}
P[w > 3] &= \int_3^\infty \frac{1}{\sqrt{2}} \exp(-\sqrt{2}w) dw \\
&= -\frac{1}{2} \exp(-\sqrt{2}w) \Big|_3^\infty \\
&= \frac{1}{2} \exp(-3\sqrt{2}) = 0.00718.
\end{aligned}$$

Note that this is considerably larger than that obtained for a Gaussian random variable, which is 0.0013. The MATLAB program FSSP3exer4\_13.m, which is contained on the CD, can be run to obtain the solution.

**4.14** The definition of the variance of a random variable  $x$  is

$$\text{var}(x) = E[(x - E[x])^2] = \int_{-\infty}^{\infty} (x - E[x])^2 p_X(x) dx$$

which for a zero mean random variable, i.e.,  $E[x] = 0$ , becomes

$$\text{var}(x) = E[x^2] = \int_{-\infty}^{\infty} x^2 p_X(x) dx.$$

Now let  $y = \sigma x$  so that the variance is  $\text{var}(y) = \text{var}(\sigma x) = E[(\sigma x)^2]$ . The latter step is valid since the mean of  $\sigma x$  is zero if the mean of  $x$  is zero. Thus,

$$\begin{aligned}
\text{var}(\sigma x) &= E[(\sigma x)^2] \\
&= \int_{-\infty}^{\infty} (\sigma x)^2 p_X(x) dx \quad (\text{use the hint with } g(x) = \sigma x) \\
&= \int_{-\infty}^{\infty} \sigma^2 x^2 p_X(x) dx \\
&= \sigma^2 \int_{-\infty}^{\infty} x^2 p_X(x) dx = \sigma^2 \text{var}(x).
\end{aligned}$$

**4.15** The probability of a sample exceeding  $\sigma_2 = 7$  in magnitude is the probability of the high variance outcome occurring, which is  $\epsilon = 0.01$  times the probability that the outcome exceeds  $\sigma_2$  in magnitude, which is 0.32, yielding  $0.0032 \times 100,000 = 320$  spikes. By running the program we actually observe 320 spikes, just a lucky set of outcomes! The MATLAB program FSSP3exer4\_15.m, which is contained on the CD, can be run to obtain the solution.

**4.16** Since  $w_i[n]$  can be written as  $w_i[n] = \alpha_{1i} \cos(2\pi f_i n) + \alpha_{2i} \sin(2\pi f_i n)$ , we

have

$$\begin{aligned}
E[w_i[n]] &= E[\alpha_{1i} \cos(2\pi f_i n) + \alpha_{2i} \sin(2\pi f_i n)] \\
&= E[\alpha_{1i}] \cos(2\pi f_i n) + E[\alpha_{2i}] \sin(2\pi f_i n) \quad (\text{expected value is linear operation}) \\
&= 0 \quad (\text{mean of each } \alpha_i \text{ is zero}).
\end{aligned}$$

The power is given by  $E[w_i^2[n]]$  so that

$$\begin{aligned}
E[w_i^2[n]] &= E[(\alpha_{1i} \cos(2\pi f_i n) + \alpha_{2i} \sin(2\pi f_i n))^2] \\
&= E[\alpha_{1i}^2 \cos^2(2\pi f_i n) + 2\alpha_{1i}\alpha_{2i} \cos(2\pi f_i n) \sin(2\pi f_i n) \\
&\quad + \alpha_{2i}^2 \sin^2(2\pi f_i n)] \\
&= E[\alpha_{1i}^2] \cos^2(2\pi f_i n) + 2E[\alpha_{1i}\alpha_{2i}] \cos(2\pi f_i n) \sin(2\pi f_i n) \\
&\quad + E[\alpha_{2i}^2] \sin^2(2\pi f_i n) \quad (\text{expected value is linear operation}).
\end{aligned}$$

But the mean of each  $\alpha_i$  is zero and so  $E[\alpha_i^2] = \text{var}(\alpha_i) = \sigma_{\alpha_i}^2$ . Also, because the covariance matrix of  $\alpha_i$  is diagonal, the two components of  $\alpha_i$  are uncorrelated. Thus,

$$\begin{aligned}
E[\alpha_{1i}\alpha_{2i}] &= \text{cov}(\alpha_{1i}, \alpha_{2i}) \quad (\text{since they are zero mean}) \\
&= 0 \quad (\text{uncorrelated}).
\end{aligned}$$

Finally,

$$\begin{aligned}
E[w_i^2[n]] &= E[\alpha_{1i}^2] \cos^2(2\pi f_i n) + E[\alpha_{2i}^2] \sin^2(2\pi f_i n) \\
&= \sigma_{\alpha_i}^2 (\cos^2(2\pi f_i n) + \sin^2(2\pi f_i n)) \\
&= \sigma_{\alpha_i}^2 = P_i.
\end{aligned}$$

# Chapter 5. Signal Model Selection

## 5.1. Introduction

In [Chapters 3](#) and [4](#) we described some useful models for a signal and noise, respectively. We now examine in detail how one decides upon a particular model to represent data that will be encountered in practice. In this chapter we discuss signal model selection, with noise model selection deferred to the next chapter. Since according to Webster's dictionary a model is defined to be a mathematical description used for "guidance or imitation", we should not expect to be able to choose the *correct* one. Indeed, an exact description does not exist, so that our goal is to select a model that is sufficiently accurate to satisfy our needs. Hence, we attempt to find a suitable *approximation* to reality based on some mathematical form. At this point the reader may wish to review the algorithm flowchart shown in [Figure 2.1](#). The particular step we now consider is Step 5, *Mathematical model*.

Many signal models were described in [Chapter 3](#) for which the signal parameters were assumed known, and therefore, the model was completely specified. For example, we might assume that the signal is a sinusoid with a frequency of 1000 Hz, and with a known amplitude and known phase. Based on this model we could design a good detection algorithm, maybe something as simple as computing the power out of a narrowband filter centered at 1000 Hz. If, however, the frequency can change *significantly* from 1000 Hz in practice, then the detection algorithm may not perform as expected. In such a case, it would be better to design an algorithm that can *adapt* the filter's center frequency. In effect, the filter would have to adapt its center frequency to the data that is observed. This type of algorithm is termed a *data-adaptive algorithm*. Its use is mandated by our lack of knowledge of the signal model's parameter, in this case its frequency. A data-adaptive algorithm accomplishes its mission by *estimating the parameters of the model on-line*. In contrast to this we term an algorithm that is based on a completely known signal model, including its parameters, as a *fixed algorithm*.

In summary, signal model selection proceeds as follows.

1. Choose a general signal model *type*, such as the sinusoid model given by

$$s[n] = \sum_{i=1}^p A_i \cos(2\pi f_i n + \phi_i) \quad n = 0, 1, \dots, N - 1.$$

2. Choose the model order  $p$ , i.e., the number of sinusoids.
3. Choose the values of the parameters  $\{A_1, \dots, A_p, f_1, \dots, f_p, \varphi_1, \dots, \varphi_p\}$   
either as fixed constants to be used in a fixed algorithm or to be determined on-line in a data-adaptive algorithm.

In general, the model accuracy will depend upon its complexity. Using more parameters leads to a more detailed and potentially more accurate model. However, if those model parameters need to be estimated from data and thus are subject to statistical error, then a less accurate model may result if too many parameters are included. Hence, model order selection is an especially important task and presents itself as somewhat of a “balancing act”. An analogy might be the goal of modeling the exterior of an automobile for visual identification. As “data” one is shown a *particular* automobile. An *adequate* model would be one that incorporates four wheels, some windows, and a box-like shape, much as what a child would draw. A more *precise* model would also include the chrome wheels, the spoiler and other adornments. But the latter could hardly be counted upon to identify a general class of automobiles. The latter more detailed model has too many “parameters” to allow it to be a good *general* model.

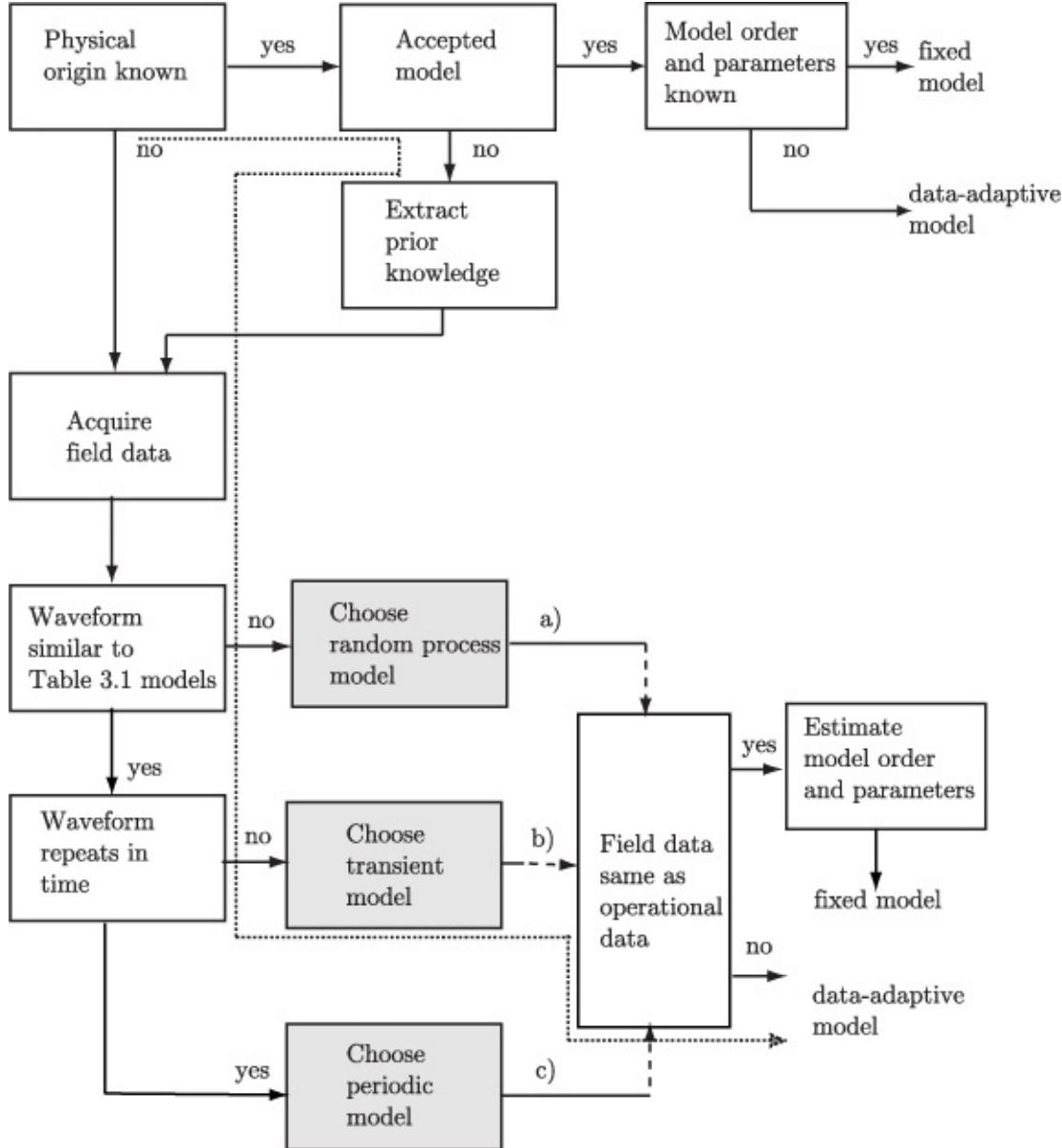
The selection of a signal model is aided by first utilizing knowledge of any physical constraints, and second, by analysis of field data. The first consideration is to guide the model selection by the physics of the signal generation process. For example, consider the signal modeling of the acoustic emissions of a helicopter. The manufacturer may have physical knowledge of the acoustic harmonic frequencies of a helicopter blade operating at a given speed, which has been established from previous testing and development studies. As already mentioned, for best algorithm performance any prior knowledge that can be *reliably* assumed should be incorporated into the algorithm as a *constraint*. The second consideration, analysis of field data, is a critical one in that it lends insight into the task at hand and can provide test data for preliminary performance analysis of the algorithm. Note that once again the same type of analysis of the field data that is used to formulate a model, may in fact be latter implemented in the actual algorithm if it is to be data-adaptive.

## 5.2. Signal Modeling

We next provide an overall approach to signal modeling, which is guided by a “roadmap”.

### 5.2.1. A Roadmap

[Table 3.2](#) given in [Chapter 3](#) listed some useful deterministic signal models. We now describe an explicit procedure for choosing a signal model and then illustrate it with an example. The overall approach is summarized by the “roadmap” given in [Figure 5.1](#). The selection process proceeds as follows.



- a) See Table 4.1
- b) damped exponentials, damped sinusoids, phase modulated signal, or polynomial
- c) harmonic sinusoids or periodic time signal

**Figure 5.1: Roadmap for signal model selection.**

We first determine if the physical origin of the signal is known and if there is an accepted model. If this is the case, then we use the model as is, with known values for the model order and parameters. This produces a *fixed* signal model. If, however, the model order and/or the parameters are unknown and may vary from data set to data set, then we will need to estimate them on-line, leading to a *data-adaptive* model.

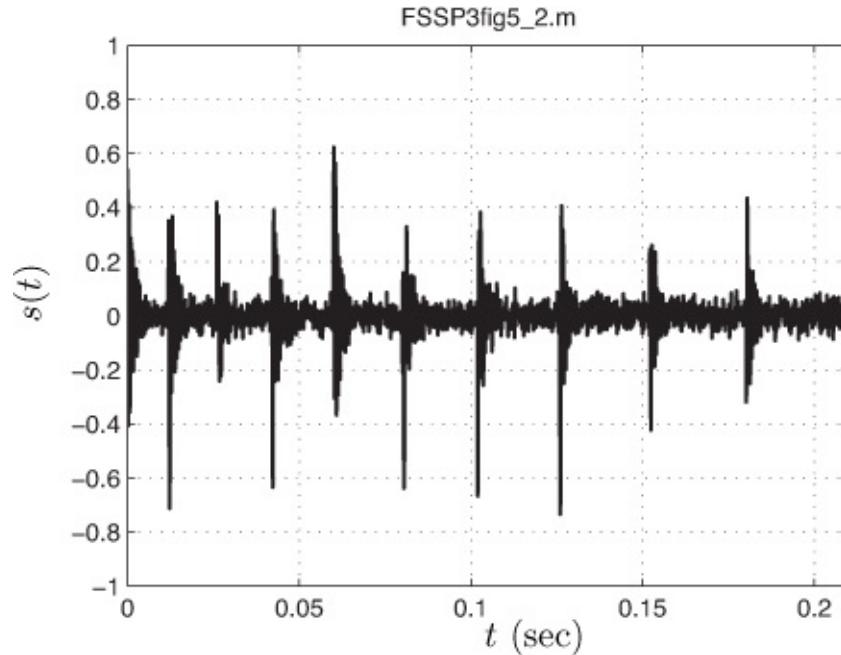
If the physical origin of the signal is unknown, or if it is known but the model is not sufficiently accurate, we next obtain some field data for study and analysis. Note that even if an acceptable model is not available, we can usually still extract information from the problem physics and use it later on to help in model selection. Having acquired field data, we ask if the waveform appears to be of a deterministic nature (a readily recognizable time series pattern) or more random in nature. If the latter, we choose a random process model from [Table 4.1](#). If the random process model parameters are known or can be estimated and assumed not to vary from data set to data set, then we are done. This produces a fixed model. If not, then we will have to estimate them on-line to produce a data-adaptive signal model.

Much the same procedure is used when the signal waveform has a definite pattern, i.e., similar to the mathematical models given in [Table 3.1](#). The two cases of interest are waveforms that repeat in time and those that do not. If the latter, we choose either damped exponentials, damped sinusoids, a phase modulated signal, or a polynomial. If it repeats with a stable pattern, i.e., is periodic in time, we choose either a set of harmonically related sinusoids (a Fourier series representation model) or a temporal signal with known or unknown signal samples within the basic period. As before, we use a fixed model if the model order and parameters are known, or are estimated and do not change from the field data used to estimate them to the operational data. Otherwise, we use a data-adaptive model.

### 5.3. An Example

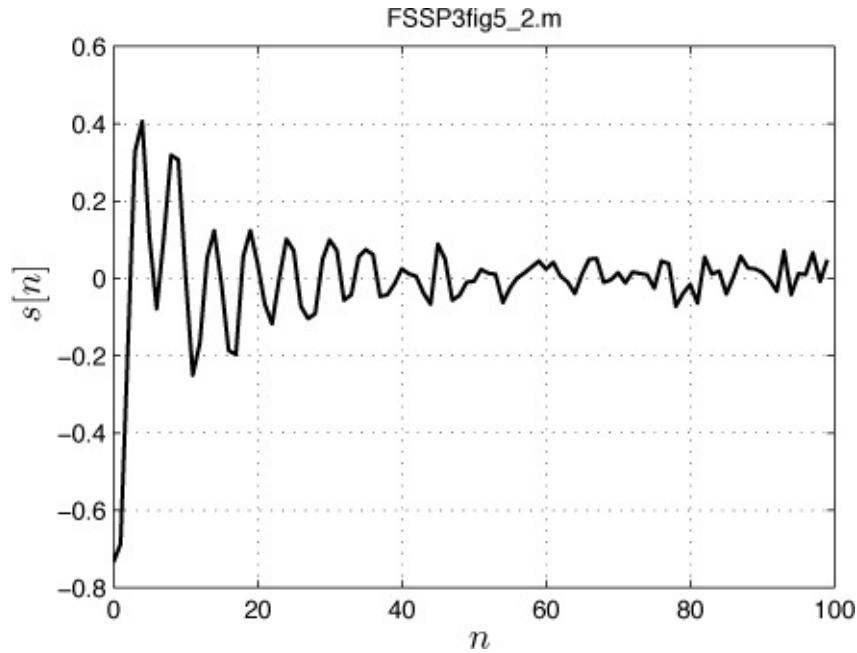
Consider the problem of detecting a fault in the bearing of a precision machine [[Liu 2008](#)]. We describe the decision-making path, which is shown in [Figure 5.1](#) as the dashed line. The physical origin of the fault is generally known, i.e., it can be due to inadequate lubrication, metal fatigue, etc., but translating that knowledge into a suitable model for the signal at the output of an accelerometer is not possible. This is mainly due to the many possible operating conditions of the machine. For example, the waveform will be dependent upon the speed of the motor which itself depends upon a load that is changing with time.

Nonetheless, from past experience it has been observed that the fault waveform, when it appears, has a spiky nature that is relatively constant in shape but whose amplitude varies with each occurrence. The spikes are nearly periodic but this assumption is violated when the machine changes speed, which is again load dependent, and cannot be predicted. An example might be as shown in [Figure 5.2](#).



**Figure 5.2: Example of accelerometer output waveform indicating a bearing fault.**

It is seen that the spike amplitudes are different and the spacing appears to increase with time, indicating a decrease in bearing speed. Hence, the waveform does not repeat, prompting the choice of a *transient* model. If we examine a single spike occurring at time  $t = 0.125$  sec, we observe the sampled waveform shown in [Figure 5.3](#).



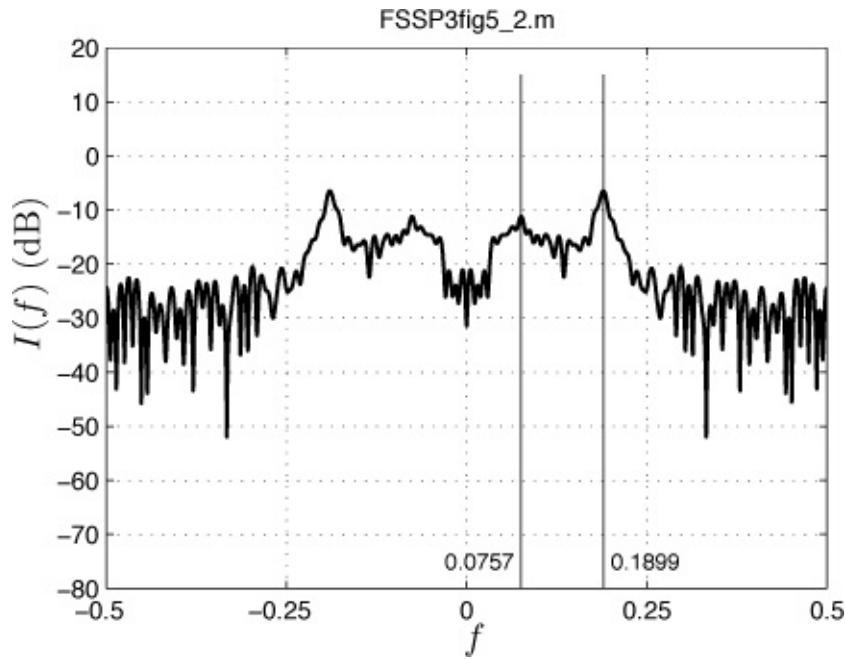
**Figure 5.3: Expanded version of sampled accelerometer output waveform showing a single spike starting at  $t = 0.125$  sec. The sample points have been connected for easier viewing.**

It has been sampled at 10,000 samples/sec so that a time interval of 0.01 seconds yields  $N = 100$  samples as shown. The waveform now appears to be a damped sinusoid or sum of damped sinusoids with some additive noise. In fact, by plotting its periodogram, which is defined as

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \right|^2 \quad (5.1)$$

and is the normalized and magnitude-squared discrete-time Fourier transform, we observe in [Figure 5.4](#) two resonances and therefore, choose a damped sinusoid model with  $p = 2$  as

$$s[n] = \sum_{i=1}^2 A_i r_i^n \cos(2\pi f_i n + \phi_i) \quad n_0 \leq n \leq n_0 + N - 1 \quad (5.2)$$



**Figure 5.4: Periodogram of time-sampled spike waveform shown in [Figure 5.3](#). The two local maxima, termed resonances, are indicated by the vertical lines. Their locations are also indicated.**

where  $n_0$  is the index of the first signal sample of the chosen spike, and  $N$  is the length of the signal.

---

### Exercise 5.1 – Length of signal

For a single damped sinusoid  $s[n] = Ar^n \cos(2\pi f_0 n + \varphi)$  with  $A = 2$ ,  $r = 0.95$ ,  $f_0 = 0.05$ , and  $\varphi = 0$ , plot the signal  $s[n]$  versus  $n$  for  $n = 0, 1, \dots, 199$ . From your plot determine  $N$ , which is the essential duration of the signal. Assuming the phase is zero, a rule of thumb is that the signal is essentially zero if

$$\left| \frac{s[N]}{s[0]} \right| < 0.001.$$

Thus, the signal length  $N$  can be found. Show that for a single damped sinusoid  $N$  is given explicitly as

$$N = \frac{\ln 0.001}{\ln r}.$$

How does this compare with your visual determination of the signal length? How could you extend this rule of thumb to the case when  $\varphi \neq 0$ ?

---

---

## Exercise 5.2 – Starting time of signal

A simple method for determining the starting time of a signal, based on noise-corrupted data  $x[n]$ , is to implement a *sliding window power detector* (see also [Section 4.5](#)). One chooses the signal start time as the time when the average power

$$T[n] = \frac{1}{L+1} \sum_{k=n-L/2}^{n+L/2} x^2[k]$$

exceeds a threshold  $y$ . Load the sampled data of the waveform plotted in [Figure 5.2](#) by using the MATLAB command `load FSSP3exer5_2`. The sampling rate is 10,000 samples/sec. Implement  $T[n]$  using a window length of  $L + 1 = 11$  and plot  $T[n]$  versus  $n$  for  $5 \leq n \leq 2094$ . Can you determine the starting times of the spikes?

---

Since the amplitudes  $A_i$ 's and phases  $\phi_i$ 's and time interval between spikes cannot be determined in advance (note how they change from spike to spike in [Figure 5.2](#)), we will use (5.2) as a signal model for a data-adaptive algorithm. The unknown parameters will need to be estimated within the algorithm. From prior experience the damping factors  $r_1, r_2$  and resonance frequencies  $f_1, f_2$  are known not to vary so that we can estimate them from the data given in [Figure 5.3](#) and then fix their values in the model. The amplitudes and phases, however, will need to be estimated in a data-adaptive algorithm. Using this data the parameter estimates obtained (the exact method to be described shortly) are

$$r_1 = 0.86 \quad f_1 = 0.08 \quad r_2 = 0.91 \quad f_2 = 0.18. \quad (5.3)$$

Note that the resonance frequencies estimated appear to be somewhat off from the ones shown in [Figure 5.4](#). This is due to the effect of noise as well as quantization of the grid search used to find the estimates, as will be explained shortly. As a result our final signal model for a given spike is

$$s[n; \theta] = A_1(0.86)^n \cos(2\pi(0.08)n + \phi_1) + A_2(0.91)^n \cos(2\pi(0.18)n + \phi_2) \quad (5.4)$$

for  $0 \leq n \leq N - 1$ . The length of the signal is  $N$ , which is about  $N = 100$  as obtained by “eyeballing” [Figure 5.3](#), and can be assumed to be known. A more accurate estimate of the signal length relies on [Exercise 5.1](#) and is  $N =$

$\ln(0.001)/\ln(0.91) = 73$ . The starting time  $n_0$  of the signal will need to be estimated on-line using the sliding window power estimator. Finally, the amplitudes and phases given as  $\boldsymbol{\theta} = [A_1 \ A_2 \ \varphi_1 \ \varphi_2]^T$  can be estimated on-line as the parameters of the linear model. We next describe the estimation procedure used to find the damping factors and frequencies given in (5.3).

## 5.4. Estimation of Parameters

As mentioned in [Section 3.5](#) it is difficult to estimate the nonlinear parameters for the models that employ them. All the signal models listed in [Table 3.2](#) are either linear or partially linear (see [Section 3.5.4](#)). The polynomial and periodic signals are examples of the linear model so that their parameters are easily estimated as

$$\hat{\boldsymbol{\theta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \quad (5.5)$$

where  $\mathbf{H}$  is the known observation matrix and  $\mathbf{x}$  is the data vector. The remaining signal models are partially linear and so first require a maximization of the function

$$\mathbf{x}^T \mathbf{H} (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \quad (5.6)$$

over the nonlinear parameters, which are embedded in  $\mathbf{H}$ . Once the nonlinear parameters have been found, they are inserted into  $\mathbf{H}$  and then (5.5) is used to produce the estimates of the linear ones. We next illustrate the procedure in more detail for the previously discussed damped sinusoid signal model.

For the waveform shown in [Figure 5.2](#) we chose the model for a single spike as

$$s[n] = A_1 r_1^n \cos(2\pi f_1 n + \phi_1) + A_2 r_2^n \cos(2\pi f_2 n + \phi_2) \quad n = 0, 1, \dots, N-1 \quad (5.7)$$

where we have referenced the time samples to begin at the start of a given spike. Also, the parameters are constrained to take on the values  $A_1 > 0$ ,  $A_2 > 0$ , and  $-\pi \leq \varphi_1 < \pi$ ,  $-\pi \leq \varphi_2 < \pi$  so that the amplitudes and phases are identifiable (see [Exercise 3.3](#)). As usual we assume that  $0 < f_1 < 1/2$  and  $0 < f_2 < 1/2$ , and finally, we need to assume that  $0 < r_1 < 1$  and  $0 < r_2 < 1$ . The assumption that the damping factor is less than one, avoids the possibility of a growing exponential, and the assumption that it is positive is needed for identifiability.

---

### Exercise 5.3 – $r > 0$ for identifiability

Show that if  $r = 1/2$ ,  $f_0 = 1/10$  and  $r = -1/2$ ,  $f_0 = 4/10$ , then each set of

parameters yields the same signal  $s[n] = r^n \cos(2\pi f_0 n)$  for  $n \geq 0$ .

---

Next performing the usual polar to cartesian coordinate transformation  $\alpha_{1_i} = A_i \cos \varphi_i$ ,  $\alpha_{2_i} = -A_i \sin \varphi_i$  in (5.7), produces

$$s[n] = (\alpha_{1_1} r_1^n \cos(2\pi f_1 n) + \alpha_{1_2} r_1^n \sin(2\pi f_1 n)) \\ + (\alpha_{2_1} r_2^n \cos(2\pi f_2 n) + \alpha_{2_2} r_2^n \sin(2\pi f_2 n))$$

which is in the linear model form with  $\boldsymbol{\theta} = [\alpha_{1_1} \alpha_{1_2} \alpha_{2_1} \alpha_{2_2}]^T$  and

$$\mathbf{H}(r_1, f_1, r_2, f_2) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ r_1 \cos(2\pi f_1) & r_1 \sin(2\pi f_1) & r_2 \cos(2\pi f_2) & r_2 \sin(2\pi f_2) \\ \vdots & \vdots & \vdots & \vdots \\ r_1^{N-1} \cos[2\pi f_1(N-1)] & r_1^{N-1} \sin[2\pi f_1(N-1)] & r_2^{N-1} \cos[2\pi f_2(N-1)] & r_2^{N-1} \sin[2\pi f_2(N-1)] \end{bmatrix}. \quad (5.8)$$

Therefore from (5.6) we must maximize the function

$$J(r_1, f_1, r_2, f_2) = \mathbf{x}^T \mathbf{H}(r_1, f_1, r_2, f_2) [\mathbf{H}^T(r_1, f_1, r_2, f_2) \mathbf{H}(r_1, f_1, r_2, f_2)]^{-1} \mathbf{H}^T(r_1, f_1, r_2, f_2) \mathbf{x}. \quad (5.9)$$

To avoid ambiguities in distinguishing the two damped sinusoids we assume that  $r_1 < r_2$ . If not, we will obtain two identical maxima and redundant parameter estimates, as for example  $(r_1, f_1, r_2, f_2) = (0.7, 0.2, 0.8, 0.4)$  and  $(r_1, f_1, r_2, f_2) = (0.8, 0.4, 0.7, 0.2)$ .

To maximize (5.9) we perform a grid search (see also [Exercise 3.8](#)) for the maximum. Since the function  $J$  is 4-dimensional, evaluating it on a fine grid of points is time consuming. Hence, the function was evaluated for the parameters at an interval of 0.05, yielding the grid  $\{0.01, 0.06, \dots, 0.96\}$  for the damping parameters and at an interval of 0.025, yielding the grid  $\{0.005, 0.03, \dots, 0.48\}$  for the frequency parameters. The maximum is found to be given by (5.3). The obtained maximum of  $J$  can therefore be in error from the true maximum by the quantization error of  $0.05/2 = 0.025$  for the damping parameter and of  $0.025/2 = 0.0125$  for the frequency. This is a standard drawback of a grid search and prompts one to try other methods such as iterative searches. *However, only a grid search is guaranteed to produce the global maximum to within the specified grid size error.* (See also [Algorithm 9.9](#) for another example of function maximization using a grid search.) To complete the discussion of the remaining partially linear models, we observe that sinusoids, damped exponentials, and

phase modulated signals also require a maximization over the nonlinear parameters.

---

### Exercise 5.4 – Another example - phase modulated signal

The phase modulated signal is given by

$s[n] = A \cos[2\pi(f_0 n + \frac{1}{2}mn^2) + \phi]$  for  $n = 0, 1, \dots, N - 1$ . Show that this can be written in the form  $s = \mathbf{H}(f_0, m)\boldsymbol{\theta}$ , where  $\boldsymbol{\theta} = [\alpha_1 \alpha_2]^T = [A \cos(\phi) - A \sin(\phi)]^T$  are the linear parameters and  $\mathbf{H}(f_0, m)$  is an  $N \times 2$  matrix containing the nonlinear parameters  $f_0$  and  $m$ . How should all the parameters  $A, \phi, f_0, m$  be estimated?

•

---

---

### Exercise 5.5 – Estimation of linear frequency modulated signal parameters

Use the results from [Exercise 5.4](#) to implement a computer simulation example. To do so

1. Generate  $N = 100$  samples of  $s[n] = A \cos[2\pi(f_0 n + \frac{1}{2}mn^2) + \phi]$  for  $n = 0, 1, \dots, N - 1$ . Let  $A = 1$ ,  $\phi = 0$ ,  $f_0 = 0.1$ , and  $m = 0.0005$ .
2. Next add WGN  $w[n]$  of variance  $\sigma^2 = 0.001$  to  $s[n]$  to form the data  $x[n] = s[n] + w[n]$ .
3. Maximize  $J(f_0, m) = \mathbf{x}^T \mathbf{H}(f_0, m) [\mathbf{H}^T(f_0, m) \mathbf{H}(f_0, m)]^{-1} \mathbf{H}^T(f_0, m) \mathbf{x}$  over  $f_0, m$  by evaluating  $J$  over the grid  $f_0 = 0.01, 0.02, \dots, 0.45$  and  $m = 0.0001, 0.0002, \dots, 0.001$ .

Are your estimates of frequency and sweep rate accurate? Next find estimates of the amplitude and phase. Repeat the experiment if  $\sigma^2 = 1$ .

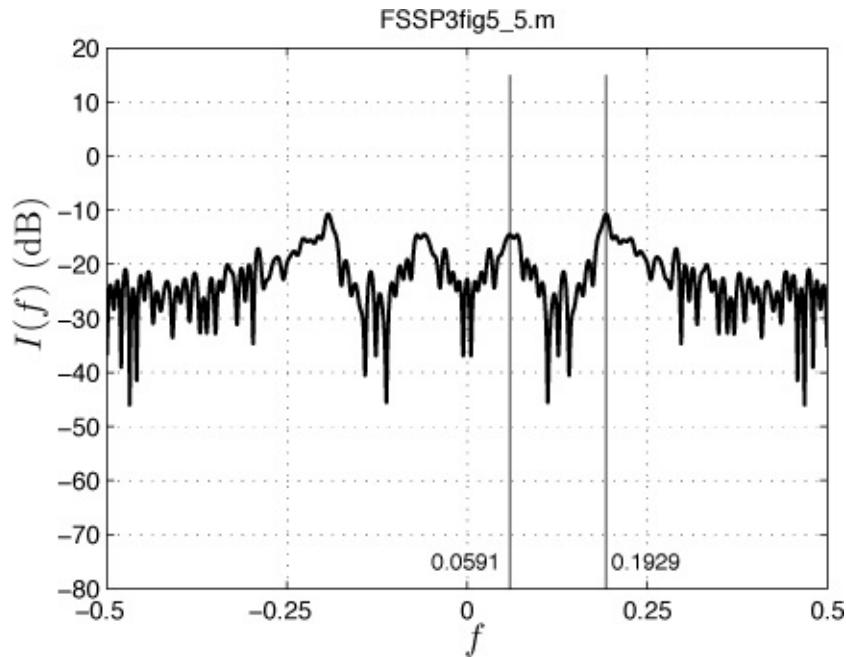
•

---

## 5.5. Model Order Selection

We have so far assumed that the model order  $p$ , which is the number of damped sinusoids as in [\(5.2\)](#), where  $p = 2$  for example, is known. Based on knowledge of  $p$  we then estimated the model parameters, with the value of  $p$  necessary to

specify  $\mathbf{H}$  as in (5.8). When this prior knowledge is not available, we will need to estimate the model order in addition to the parameters for that model. In some instances, it may be possible to easily infer the model order from visual examination of the data. For example, in [Figure 5.4](#) we note that the periodogram displays two resonances for  $f > 0$ . This indicates that the number of damped sinusoids must have been two, with each one giving rise to a peak in the estimated spectrum. But even this method is not assured, especially if a resonance is of low amplitude. This will occur if the amplitude of the damped sinusoid is small. Another difficulty is that the resonance seen might be due to two closely spaced sinusoidal components. An example of the latter is shown in [Figure 5.5](#).



**Figure 5.5: Periodogram of time-sampled spike waveform with an additional damped sinusoid. The two local maxima are indicated by the vertical lines. Their locations are also indicated.**

Comparing this spectrum to that in [Figure 5.4](#) it is seen to be similar in that it exhibits two resonances. This is because the data used to generate the spectrum in [Figure 5.4](#) has two damped sinusoidal components with frequencies  $f_1 = 0.06$ ,  $f_2 = 0.19$ . However, the data used to produce [Figure 5.5](#) actually contains three damped sinusoidal components with frequencies  $f_1 = 0.06$ ,  $f_2 = 0.19$ , and  $f_3 = 0.21$ . Clearly, visual inspection of the spectrum cannot indicate the two resonances because of their close proximity to each other.

---

## Exercise 5.6 – Periodogram of damped exponentials

For a single damped exponential signal

$$s[n] = \begin{cases} Ar^n & n \geq 0 \\ 0 & n < 0 \end{cases}$$

show that the magnitude-squared of the discrete-time Fourier transform is given by

$$|S(f)|^2 = \frac{A^2}{|1 - r \exp(-j2\pi f)|^2}$$

where  $S(f) = \sum_{n=-\infty}^{\infty} s[n] \exp(-j2\pi fn)$ . Next plot  $|S(f)|^2$  for  $A = 1$ ,  $r = 0.9$  and also for  $A = 1$ ,  $r = 0.7$ . Do you think you could decide that two damped exponentials are present by viewing the periodogram of their sum, i.e., of  $s[n] = 0.7^n + 0.9^n$  for  $0 \leq n \leq N - 1$ ? Hint: In finding the Fourier transform of a damped exponential use the complex geometric sum formula

$$\sum_{n=0}^{\infty} z^n = \frac{1}{1-z} \quad \text{if } |z| < 1.$$



Clearly, we will need some other method to determine the model order. The model order selection problem will only be exacerbated when we need to choose models for other signals, say for instance, the sum of phase modulated signals. There the spectrum will not even indicate a peak for each component, nullifying the use of visual examination of the spectrum for model order determination.

A model order selection method that produces good results is the *exponentially embedded family* (EEF) approach [[Kay 2005](#)]. It can be shown to produce the correct model order for large enough data records and/or high enough SNR [[Xu and Kay 2008](#), [Ding and Kay 2011](#)]. Assume that data is available containing the signal whose order is to be estimated, say  $x[n]$  for  $n = 0, 1, \dots, N - 1$  and denoted by the  $N \times 1$  data vector  $\mathbf{x}$ . The EEF approach finds the value of  $p$  as the  $k$  that maximizes the criterion

$$\text{EEF}(k) = \begin{cases} \xi_k - n_k \left[ \ln \left( \frac{\xi_k}{n_k} \right) + 1 \right] & \text{if } \frac{\xi_k}{n_k} \geq 1 \\ 0 & \text{if } \frac{\xi_k}{n_k} < 1 \end{cases} \quad k = 1, 2, \dots, k_{\max} \quad (5.10)$$

where

$$\xi_k = \frac{\mathbf{x}^T \mathbf{P}_k \mathbf{x}}{\frac{1}{N} (\mathbf{x}^T \mathbf{x} - \mathbf{x}^T \mathbf{P}_k \mathbf{x})} \quad (5.11)$$

and

$$\mathbf{P}_k = \hat{\mathbf{H}}_k \left( \hat{\mathbf{H}}_k^T \hat{\mathbf{H}}_k \right)^{-1} \hat{\mathbf{H}}_k^T. \quad (5.12)$$

The integer  $n_k$  denotes the total number of unknown signal parameters that are being estimated to form the model having  $k$  components. The unknown parameters in  $\mathbf{H}_k$  are replaced by estimates, yielding the estimated observation matrix  $\hat{\mathbf{H}}_k$ . Also, we note that  $\mathbf{P}_k$  is the estimated projection matrix using the number of columns necessary to represent the signal for a model order of  $k$ .

As an example, consider the damped sinusoid model

$$s[n] = \begin{cases} \sum_{i=1}^k A_i r_i^n \cos(2\pi f_i n + \phi_i) & n \geq 0 \\ 0 & n < 0. \end{cases}$$

Two cases are of interest: when the damping factors  $r_i$ 's and frequencies  $f_i$ 's are known, and when they are not and must be estimated. For the known parameter case, we have the usual linear model with the  $\mathbf{H}$  matrix given for  $k = 2$  in (5.8) as an example. For an arbitrary  $k$  the matrix will have dimension  $N \times 2k$ . When the damping factors and frequencies are unknown and must be estimated to produce an estimated observation matrix for use in (5.12), the number of unknown parameters for each component is 4. Thus, for the damped sinusoid case we assume the signal is composed of  $k$  components with 4 unknown parameters per component. With 4 unknown parameters per component, we have that the total number of unknown parameters is  $n_k = 4k$ .

Continuing the damped sinusoid example, suppose that we wish to determine the number of components, either one, two, three, or four. Then, we compute (5.10) for  $k = 1, 2, 3, 4$  and choose  $p$  as the value of  $k$  that maximizes  $\text{EEF}(k)$ . To simplify the discussion we will assume that the sets of damping factors and frequencies are known for the four possible models. Only the amplitudes and phases are unknown. Thus, for  $k = 1$  we use

$$\hat{\mathbf{H}}_1 = \mathbf{H}_1 = \begin{bmatrix} 1 & 0 \\ r_1 \cos(2\pi f_1) & r_1 \sin(2\pi f_1) \\ \vdots & \vdots \\ r_1^{N-1} \cos[2\pi f_1(N-1)] & r_1^{N-1} \sin[2\pi f_1(N-1)] \end{bmatrix}$$

and for  $\hat{\mathbf{H}}_2 = \mathbf{H}_2$  we use (5.8). For the remaining models with three and four components we increase the number of columns in the  $\mathbf{H}$  matrix by two columns

for each additional component. Next consider the four possible models

$$k = 1 : r_1 = 0.95, f_1 = 0.19$$

$$k = 2 : r_1 = 0.95, f_1 = 0.19; r_2 = 0.87, f_2 = 0.06$$

$$k = 3 : r_1 = 0.95, f_1 = 0.19; r_2 = 0.87, f_2 = 0.06; r_3 = 0.85, f_3 = 0.21$$

$$k = 4 : r_1 = 0.95, f_1 = 0.19; r_2 = 0.87, f_2 = 0.06; r_3 = 0.85, f_3 = 0.21; \\ r_4 = 0.9, f_4 = 0.3.$$

Computing the EEF for the data shown in [Figure 5.3](#) (which was generated using the  $k = 2$  model) produces the values

$$\text{EEF}(1) = 147.3$$

$$\text{EEF}(2) = 1778.9$$

$$\text{EEF}(3) = 1775.7$$

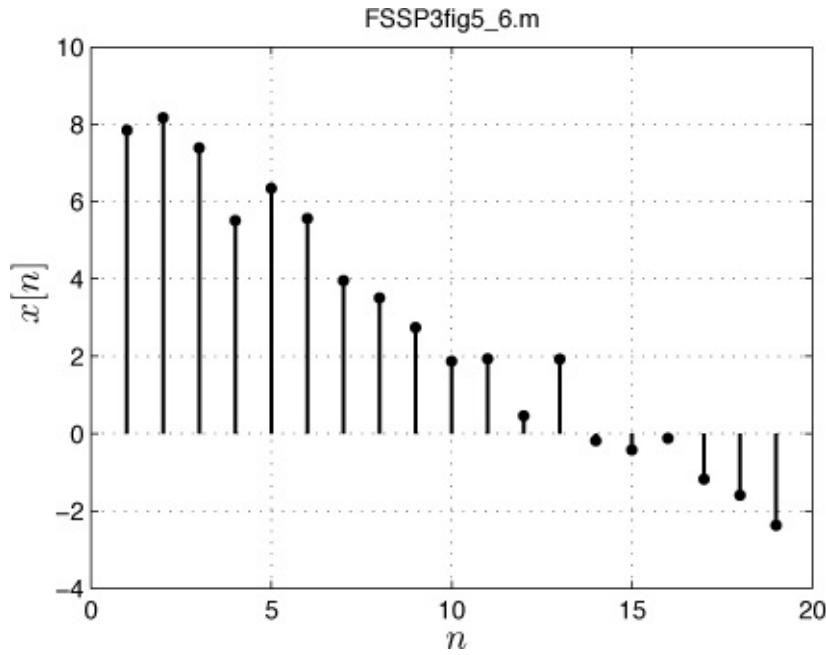
$$\text{EEF}(4) = 1767.0$$

for which the maximum is given by  $k = 2$ , the correct value. As is typical of the EEF and most model order selection approaches, underestimation of the order (see value of  $\text{EEF}(k)$  as  $k$  increases from  $k = 1$  to  $k = 2$ ) is easily recognized, but discerning overestimation of the order is much more subtle (see preceding value of  $\text{EEF}(k)$  as  $k$  increases from  $k = 2$  to  $k = 3$  to  $k = 4$ ). Overfitting is therefore of primary concern.

---

### Exercise 5.7 – Fitting a polynomial to data

Consider the data samples shown in [Figure 5.6](#). Using the EEF, estimate the model order. Is the data best described by a constant  $s[n] = A$  or by a straight line  $s[n] = A + Bn$  or by a parabola  $s[n] = A + Bn + Cn^2$  or by a cubic  $s[n] = A + Bn + Cn^2 + Dn^3$ ? The data samples can be found on the CD as `FSSP3exer5_7.mat`.



**Figure 5.6: Polynomial signal of unknown order embedded in white Gaussian noise.**

Hint: Note that there are only linear parameters for each model and for the cubic model we have

$$\mathbf{H}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 2^2 & 2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & N-1 & (N-1)^2 & (N-1)^3 \end{bmatrix}.$$

Also, to load the data into MATLAB use `load FSSP3exer5_7`.

## 5.6. Lessons Learned

- An exact signal model does not exist. The best we can hope for is a good approximation to model the observed data.
- Models are either fixed or data-adaptive. Use a fixed model if the data characteristics do not change from data set to data set. Otherwise, use a data-adaptive model that estimates the model parameters on-line.
- Be careful not to include too many parameters in the model. A simpler model is usually better when the parameters must be estimated.
- Any prior knowledge about the data characteristics should be incorporated

-----  
into the model.

- A simple method to determine the starting time of a signal is a sliding window power detector. However, better methods may exist if more signal characteristics are known.
- Signal models should be checked to make sure they are identifiable. Otherwise, it is likely that any subsequent algorithm may exhibit unpredictable behavior.
- For maximization of a function to determine a parameter estimate always use a grid search if practical.
- Visual examination of the spectrum of a signal cannot always be used to discern the number of signal components.
- Model order selection methods typically err on the side of estimating too many model parameters as opposed to too few parameters.

## References

- Ding, Q., S. Kay, "Inconsistency of the MDL: On the Performance of Model Order Selection Criteria with Increasing Signal-to-Noise Ratio", *IEEE Trans. on Signal Processing*, May 2011.
- Kay, S., "Embedded Exponential Families: A New Approach to Model Order Selection", *IEEE Trans. on Aerospace and Electronics*, Jan. 2005.
- Liu, J., W. Wang, F. Golnaraghi, K. Liu, "Wavelet Spectrum Analysis for Bearing Fault Diagnostics", *Measurement Science and Technology*, Vol. 19, pp. 1–9, 2008.
- Xu, C., S. Kay, "Source Enumeration Using the EEF", *IEEE Signal Processing Letters*, Dec. 2008.

## Appendix 5A. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand ('state', 0)` and `randn ('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

- 5.1** You should observe that the signal has damped out at about  $N = 120$ . The MATLAB program `FSSP3exer5_1.m`, which is contained on the CD, can be run to obtain the plot of  $s[n]$ . Using the given formula with  $r = 0.95$  you will obtain  $N = 134$ . The formula is derived as follows.

$$\left| \frac{s[N]}{s[0]} \right| = \left| \frac{Ar^N \cos(2\pi f_0 N)}{A} \right| \leq r^N < 0.001$$

and solving for  $N$  produces the desired rule of thumb. For a phase not equal to zero you could replace  $s[0]$  by the maximum absolute value of  $s[n]$ , which would occur near  $n = 0$  and repeat the procedure.

- 5.2** You should observe a large change in the value of  $T[n]$  whenever a spike is encountered. For example, the starting time of the spike shown in [Figure 5.3](#) can be found to be about 0.125 sec from your plot. The solution is given by the MATLAB program `FSSP3exer5_2.m`, which is contained on the CD.

- 5.3** For the first set of parameters we have

$$\begin{aligned} s[n] &= (\frac{1}{2})^n \cos(2\pi(1/10)n) \\ &= (-\frac{1}{2})^n \cos(2\pi(1/10)n - \pi n) \quad (\text{since } \cos(n\pi) = (-1)^n) \\ &= (-\frac{1}{2})^n \cos(2\pi(1/10)n - 2\pi(5/10)n) \\ &= (-\frac{1}{2})^n \cos(2\pi(4/10)n) \end{aligned}$$

yielding the same signal as for the second set of parameters.

- 5.4** The phase modulated signal can be put into the partially linear model form (which assumes  $f_0$  and  $m$  are unknown) as

$$\begin{aligned} s[n] &= A \cos[2\pi(f_0 n + \frac{1}{2}mn^2) + \phi] \\ &= A \cos[2\pi(f_0 n + \frac{1}{2}mn^2)] \cos(\phi) - A \sin[2\pi(f_0 n + \frac{1}{2}mn^2)] \sin(\phi) \\ &= \alpha_1 \cos[2\pi(f_0 n + \frac{1}{2}mn^2)] + \alpha_2 \sin[2\pi(f_0 n + \frac{1}{2}mn^2)] \end{aligned}$$

so that  $\theta = [\alpha_1 \alpha_2]^T$  and

$$\mathbf{H}(f_0, m) =$$

$$\begin{bmatrix} 1 & 0 \\ \cos[2\pi(f_0 + \frac{1}{2}m)] & \sin[2\pi(f_0 + \frac{1}{2}m)] \\ \vdots & \vdots \\ \cos[2\pi(f_0(N-1) + \frac{1}{2}m(N-1)^2)] & \sin[2\pi(f_0(N-1) + \frac{1}{2}m(N-1)^2)] \end{bmatrix}.$$

The start frequency  $f_0$  and sweep rate  $m$  are estimated by finding the location of the maximum of

$$J(f_0, m) = \mathbf{x}^T \mathbf{H}(f_0, m) [\mathbf{H}^T(f_0, m) \mathbf{H}(f_0, m)]^{-1} \mathbf{H}^T(f_0, m) \mathbf{x}.$$

Using the values found, i.e.,  $\hat{f}_0$  and  $\hat{m}$ , and substituting them into  $\mathbf{H}(f_0,$

$m$ ) to form  $\mathbf{H}(\hat{f}_0, \hat{m})$  we find the estimates of  $\alpha_1$  and  $\alpha_2$  as

$$\begin{bmatrix} \hat{\alpha}_1 \\ \hat{\alpha}_2 \end{bmatrix} = \left[ \mathbf{H}^T(\hat{f}_0, \hat{m}) \mathbf{H}(\hat{f}_0, \hat{m}) \right]^{-1} \mathbf{H}^T(\hat{f}_0, \hat{m}) \mathbf{x}$$

and finally, the amplitude and phase estimates are

$$\begin{aligned} \hat{A} &= \sqrt{\hat{\alpha}_1^2 + \hat{\alpha}_2^2} \\ \hat{\phi} &= \arctan\left(\frac{-\hat{\alpha}_2}{\hat{\alpha}_1}\right). \end{aligned}$$

Note that the arctan function is a “four-quadrant” inverse tangent function.

**5.5** The results are that for a noise variance of  $\sigma^2 = 0.001$

$$\begin{aligned} \hat{A} &= 0.9997 \\ \hat{\phi} &= 0.0061 \\ \hat{f}_0 &= 0.1000 \\ \hat{m} &= 0.0005 \end{aligned}$$

and for  $\sigma^2 = 1$

$$\begin{aligned} \hat{A} &= 1.0095 \\ \hat{\phi} &= 0.1912 \\ \hat{f}_0 &= 0.1000 \\ \hat{m} &= 0.0005. \end{aligned}$$

The MATLAB program `FSSP3exer5_5.m`, which is contained on the CD, can be run to obtain these results.

**5.6** To determine the discrete-time Fourier transform

$$\begin{aligned} S(f) &= \sum_{n=-\infty}^{\infty} s[n] \exp(-j2\pi f n) && \text{(definition)} \\ &= \sum_{n=0}^{\infty} A r^n \exp(-j2\pi f n) \\ &= A \sum_{n=0}^{\infty} [r \exp(-j2\pi f)]^n && (|z| = |r \exp(-j2\pi f)| = r < 1) \\ &= \frac{A}{1 - r \exp(-j2\pi f)}. \end{aligned}$$

If you plot  $|S(f)|^2$  for different values of  $r$ , they will overlap in frequency and hence the periodogram of the sum of two damped exponentials will reveal only a single peak. Run the MATLAB program

FSSP3exer5\_6.m, which is contained on the CD, to view the two plots for  $r = 0.7$  and for  $r = 0.9$ .

**5.7** Using the observation matrix  $\mathbf{H}_4$  given in the exercise compute the EEF for values of  $k = 1$  by taking only the first column, yielding  $\mathbf{H}_1$ , then for  $k = 2$  by taking the first two columns, yielding  $\mathbf{H}_2$ , etc. The values of the EEF should be

$$\text{EEF}(1) = 11.3$$

$$\text{EEF}(2) = 1073.0$$

$$\text{EEF}(3) = 1486.7$$

$$\text{EEF}(4) = 1484.8$$

so that a third order model, the parabola with  $s[n] = A + Bn + Cn^2$ , would be chosen. The actual signal used to generate the data was  $s[n] = 10 - n + 0.02n^2$ , a parabola. The MATLAB program FSSP3exer5\_7.m, which is contained on the CD, can be run to obtain the solution.

# Chapter 6. Noise Model Selection

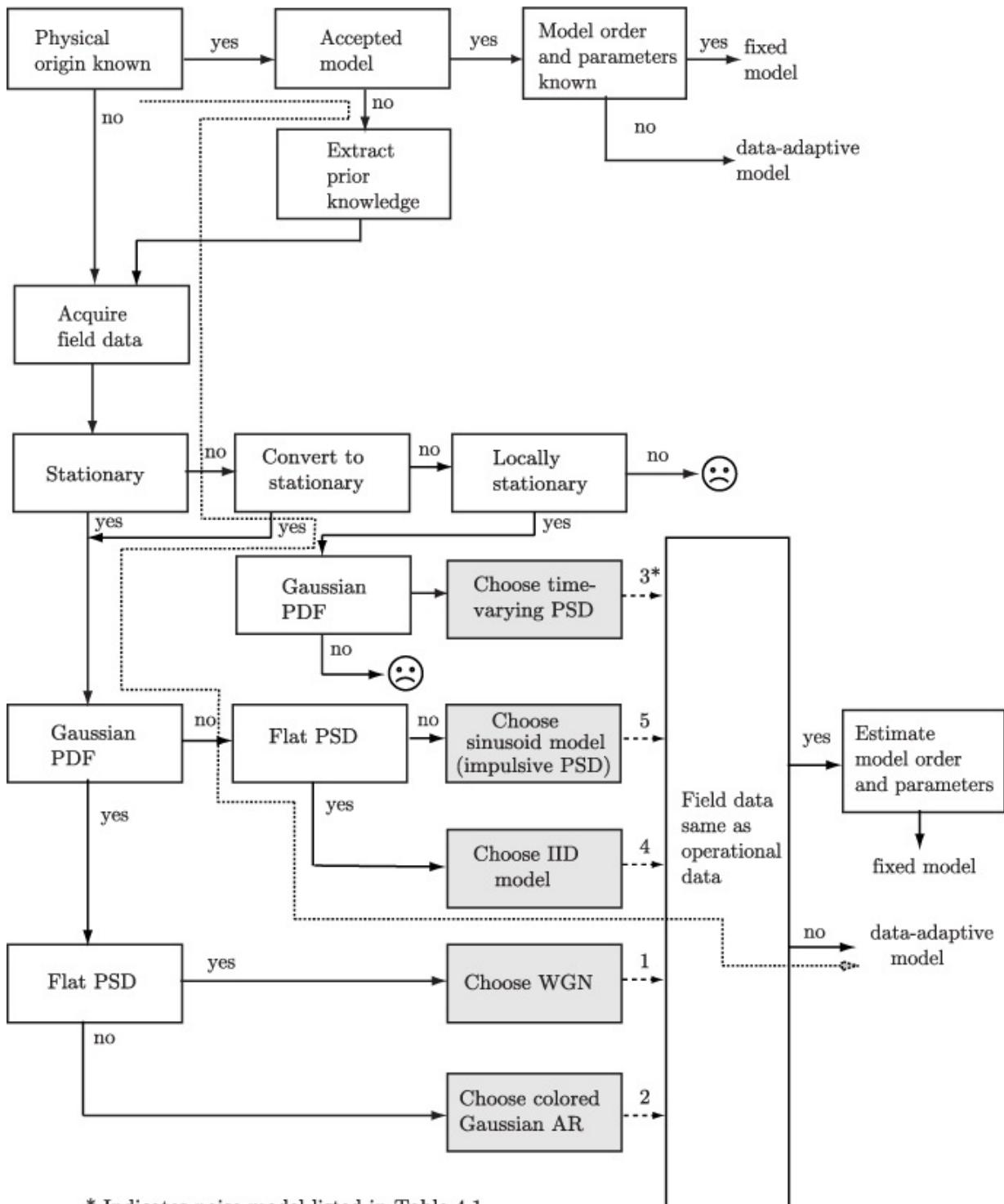
## 6.1. Introduction

We complete our discussion of model selection in this chapter by focusing on noise models. Most of the introductory comments made for signal model selection given in [Section 5.1](#) apply here as well. Rather than repeat the discussion, we just summarize the salient points.

1. A noise model is only an approximation to reality. It need not be perfect, as it is only intended to model the important features of the noise. Those features are the ones that affect the signal processing algorithm performance. Consider, for example, the modeling of the spectral characteristics of noise over bands that are not likely to contain a signal. These bands are irrelevant to the algorithm performance, and so any convenient model can be chosen for them.
2. Noise models can be fixed or data-adaptive. An example of the former is white Gaussian noise (WGN) with known variance  $\sigma^2 = 1$ , while for the latter, it is WGN with an unknown variance that will need to be estimated on-line.
3. The simpler the model the better. Very complicated models run the risk of failing in practice due to an overabundance of parameters that must be estimated. For example, the use of an overly complex autoregressive (AR) power spectral density (PSD) model, having a large number of AR filter coefficients to model very fine spectral details, would be ill advised.
4. Any physical knowledge about the noise generation mechanism should be incorporated into the model. This knowledge is translated into constraints that will improve performance by restricting parameter values to those that are physically realizable. An example might be to constrain the range of power values of an interference so that the algorithm need only perform well against likely interferer levels.

## 6.2. Noise Modeling

The procedure for selecting a noise model begins much the same as for a signal model, as described in [Section 5.2](#). A “roadmap”, which we now describe, is shown in [Figure 6.1](#).



**Figure 6.1: Roadmap for noise model selection.**

### 6.2.1. A Roadmap

We first ascertain whether the physical mechanism generating the noise is

known, and if so, is there an accepted model? If the model is known, then we determine whether its model order and parameter values are known. If so, then we are done and employ the given model, termed a *fixed model*. If not, we will need to estimate the model parameters and possibly the model order on-line, resulting in a *data-adaptive* model. If there is no accepted model, then we extract as much information as possible from the physics to help guide us in model selection. Without an acceptable model, we must acquire field data in an attempt to analyze its characteristics, which hopefully will suggest a model. At this point the noise model selection process departs from that for signal model selection.

The possible noise models have been listed in [Table 4.1](#). They are also shown in [Figure 6.1](#) in the shaded boxes. We first determine if the noise data is stationary. Some considerations in doing so have already been described in [Section 4.2](#). If the noise is nonstationary, then we can attempt to make it stationary to avail ourselves of the more easily handled stationary models. The procedure to make the data stationary might be as simple as dividing the data by the square-root of the noise power if the noise has a power that is changing in time (and of course, if this time variation is known). Another transformation is described in [Section 6.3](#) (see [\(6.1\)](#) for nonstationary mean extraction). If the noise process cannot be made stationary, then we next try to assume it is locally stationary. Such an assumption is valid if the noise process has time-varying parameters that vary slowly. In particular, if the noise is Gaussian, we should choose the time-varying general Gaussian model given in [Table 4.1](#) as Item 3. The fact that the noise is locally stationary will allow us to employ the AR spectral model and to estimate its model order and/or parameters from a block of field data. The time duration of the block must be such that the parameters *do not change significantly over this time interval*. Otherwise, for a rapid nonstationarity the data characteristics will change before we can accurately estimate them. In this case, we are at a loss to choose a noise model (see “unhappy face” in [Figure 6.1](#)!). Also, if the noise is locally stationary but not Gaussian, it is difficult to formulate an accurate model (another unhappy face).

Next, assuming a stationary noise process, we determine if the probability density function (PDF) is Gaussian. Methods to do so are described in [Section 6.4](#). If the PDF is nonGaussian and the PSD is flat, the latter indicating no correlation between samples, then we choose the nonGaussian independent and identically distributed (IID) model, which is Item 4 in [Table 4.1](#). (Of course, theoretically, lack of correlation does not imply independence, but in practice it is a reasonable approximation). If the PSD is not flat, but is impulsive, then we choose the randomly phased sinusoid model listed as Item 5 in [Table 4.1](#). If the

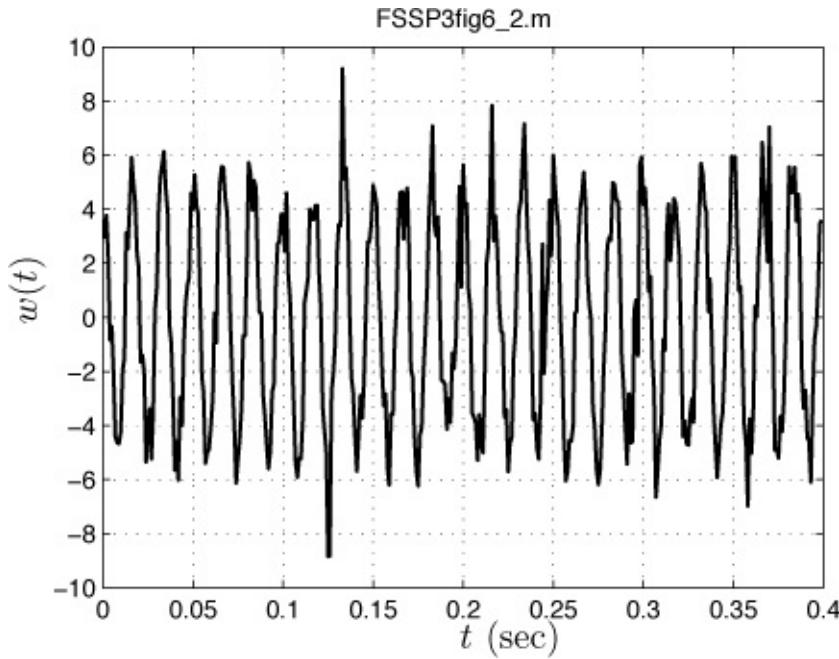
process can be assumed to be Gaussian and its PSD is flat, we choose as our model WGN, i.e., Item 1 in [Table 4.1](#). For a Gaussian process with a nonflat PSD, the appropriate model is colored Gaussian noise. This model corresponds to Item 2 in [Table 4.1](#).

Next, as is the same for signal model selection, we need to determine if the field data we have used to choose the noise model has the same characteristics as the operational data. The operational data is that data which the algorithm will actually be operating upon. If this is the case, and the model order and parameters have been estimated, then we are done. A fixed model is the result. Otherwise, we will need to estimate the model order and/or parameters on-line, producing a data-adaptive algorithm.

An example of the entire noise modeling process is given in the next section.

### 6.3. An Example

An important problem for people with heart anomalies is the detection of cardiac arrhythmias. To determine when such an event occurs, requires the patient to wear a monitor that senses the heartbeat waveform and generates an electrocardiogram (ECG) [[Ligtenberg and Kunt 1983](#)]. To complicate matters the monitoring is done over an extended period of time, typically 24 hours, during which time the person will be engaged in physical activities such as walking. This movement can generate motion artifacts in the ECG due to sensor movement. Also, it is common to have 60 Hz power line pickup. Thus, the recorded ECG is corrupted by motion noise and interference. A hypothetical example of a recorded waveform, obtained between heartbeats, is shown in [Figure 6.2](#). Our objective is to select a noise model, where we use the term “noise” to also include the interference. To do so we refer to [Figure 6.1](#) in describing the selection steps as indicated by the dashed line.



**Figure 6.2: Hypothetical noise waveform observed from Holter monitor.**

To begin, we note that the physical origin of the noise is known, being due to sensor movement and power line interference of 60 Hz (note the 6 cycles of the interference in the  $[0, 0.1]$  second interval in [Figure 6.2](#)). The latter is easily modeled by a 60 Hz sinusoid, but its amplitude and phase cannot be known in advance. This is because these parameters depend upon the exact electromagnetic coupling mechanism from the power line source, which is unknown, to the pickup point, which is also unknown. Furthermore, the amplitude and phase will vary with the person's location, i.e., his proximity to the power source, and therefore with time, as the person moves. The motion noise, which is seen in [Figure 6.2](#) as the higher frequency "hash", is even more difficult to model. It will depend upon the type of adhesive used to affix the sensors to the body, the level of physical activity, *etc.* Hence, an accepted model is not readily available. As an alternative approach, we first acquire some field data by sampling the waveform shown in [Figure 6.2](#) at a rate of  $F_s = 1/\Delta = 1000$  samples/sec, producing  $N = 400$  samples. (In practice, we would use much more data to formulate our model.) This data might be obtained by having a patient with a normal heartbeat wear the monitor and then extract his/her QRS complexes so that all that remains is the noise in between the heartbeats. To simplify our discussion, we will assume that a noise waveform, such as shown in [Figure 6.2](#), is available whose QRS signals have been perfectly extracted (a big stretch!). Note that in [Figure 6.2](#) a QRS beat is not present in the 0.4 second

interval shown. For a typical heart rate of 80 beats per minute the time interval between beats is 0.75 seconds, allowing us to extract the noise only segment.

We first ask whether the noise is stationary. Clearly, the mean changes in time due to the interference. We can, however, convert the data to that of a stationary noise process if the mean can be extracted. Assuming that the frequency of the interference is 60 Hz and it is perfectly sinusoidal, we can estimate the amplitude and phase and then subtract it from the sampled waveform  $x[n]$ . It should be emphasized that the intent of converting the original noise process to a stationary process is to allow us to model the converted process using the many possible models for a stationary process. When the entire modeling procedure is complete, however, we will then reintroduce the time-varying mean as part of the overall model. Then, subsequent algorithm development will need to account for the nonstationary mean.

---



### What do we mean by *mean*?

The *temporal* mean is the value obtained by integrating a waveform such as shown in [Figure 6.2](#) over time and dividing by the time interval. For a sinusoid, if we integrate, we will obtain a value that is less than the area of a half-cycle, and therefore, dividing by the large time interval will cause the temporal mean to converge to zero. The *probabilistic* mean (sometimes called the *ensemble* mean) is what we have been denoting as  $E[X]$  for the random variable  $X$ . It denotes the “average value” of the random variable if we *repeat the experiment* that generated the outcome of the random variable many times. Referring to [Figure 6.2](#), if we fix the time to be  $t = 0.1$ , at which time a peak occurs, then  $W(0.1) = 5$ . We know this to be true since we used a sinusoid with amplitude  $A = 5$  to generate this data. If we repeat the experiment to generate another 400 points of data, the *same sinusoid will be used* with the same value at  $t = 0.1$ . Hence the value of  $W(t)$  at  $t = 0.1$  will still be 5, the waveform only changing at  $t = 0.1$  due to the other noise component present (it is assumed that the mean of the other component of noise is zero). Thus, we have that  $E[W[0.1]] = 5$ . If instead, we were looking at the waveform at time  $t_1 = 0.1 + 0.5(1/60)$ , then the probabilistic mean would be  $E[W(t_1)] = -5$ . This is why we say the mean is changing in time. Unless otherwise stated, the *mean* always means the *probabilistic mean*.



To convert the data to a stationary process we must subtract the nonstationary mean to yield

$$y[n] = x[n] - \hat{A} \cos[2\pi(60)n\Delta + \hat{\phi}]. \quad (6.1)$$

To estimate the amplitude and phase we use  $\hat{\theta} = [\hat{\alpha}_1 \hat{\alpha}_2]^T = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$ , where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ \cos(2\pi f_0) & \sin(2\pi f_0) \\ \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \sin[2\pi f_0(N-1)] \end{bmatrix}$$

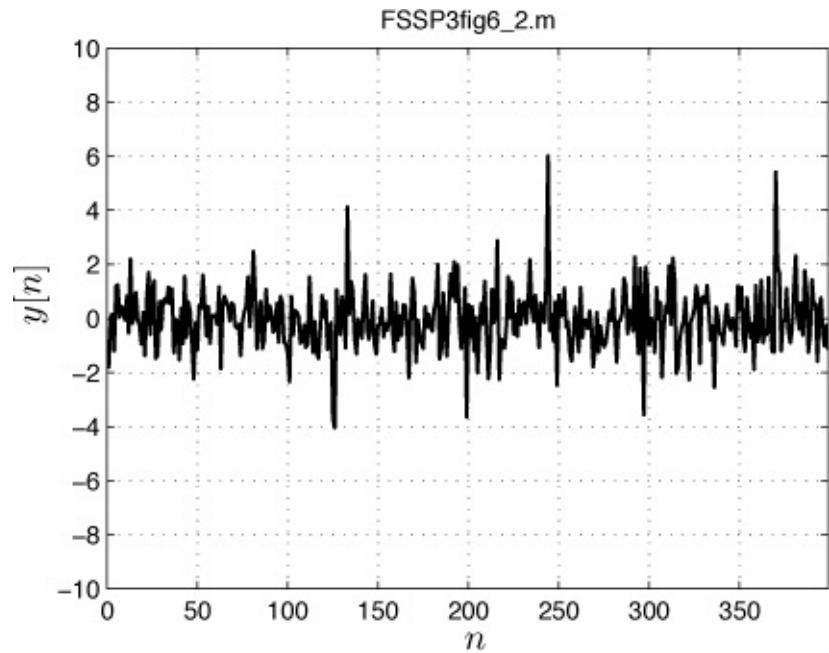
and  $f_0 = 60\Delta = 0.06$  cycles/sample to obtain the estimates of the sine and cosine amplitudes  $\alpha_1$  and  $\alpha_2$ . The estimates of amplitude and phase are then given by

$$\begin{aligned} \hat{A} &= \sqrt{\hat{\alpha}_1^2 + \hat{\alpha}_2^2} \\ \hat{\phi} &= \arctan\left(\frac{-\hat{\alpha}_2}{\hat{\alpha}_1}\right) \end{aligned}$$

using the four-quadrant inverse for the tangent function (see [Algorithm 9.2](#)). In [Figure 6.3](#) we display the *residual* data  $y[n]$ . Now it appears to be more noise-like, having extracted the time-varying mean. Another possible approach to removing the 60 Hz interference is to filter the data with a finite impulse response (FIR) filter such as

$$z[n] = x[n] + b[1]x[n-1] + b[2]x[n-2] \quad (6.2)$$

where the filter coefficients  $b[1], b[2]$  are chosen to produce a zero in the frequency response at 60 Hz.



**Figure 6.3: Residual data set after estimating 60 Hz interference and subtracting it out. The points have been connected by straight lines for easier viewing.**

---

### **Exercise 6.1 – Sinusoid removal by finite impulse response (FIR) filtering**

Let  $b[1] = -2 \cos[2\pi(0.06)]$  and  $b[2] = 1$ . Verify that the filter frequency response given by  $H(f) = 1 + b[1] \exp(-j2\pi f) + b[2] \exp(-j4\pi f)$  is equal to zero at the correct frequency. Next, filter  $x[n]$  for  $n = 0, 1, \dots, 399$ , the sampled version of  $x(t)$  shown in [Figure 6.2](#) and contained on the CD as FSSP3exer6\_1.mat, and display the output  $z[n]$ . You can use `load FSSP3exer6_1` to obtain the data. Compare your results against those shown in [Figure 6.3](#) (the data  $y[n]$  will be loaded in with  $x[n]$  so you can compare the two methods visually). Does filtering or estimation/extraction using (6.1) appear to work better? Hint: You can let  $x[-2] = x[-1] = 0$  for the initial conditions of the FIR filter.

---



---

### **Exercise 6.2 – Sinusoid removal by estimation/extraction - another viewpoint**

Note that  $y[n]$  can be equivalently expressed as

$$y[n] = x[n] - \hat{\alpha}_1 \cos(2\pi f_0 n) - \hat{\alpha}_2 \sin(2\pi f_0 n)$$

and this can be written in vector form with  $\mathbf{y} = [y[0] \ y[1] \dots y[N-1]]^T$  as

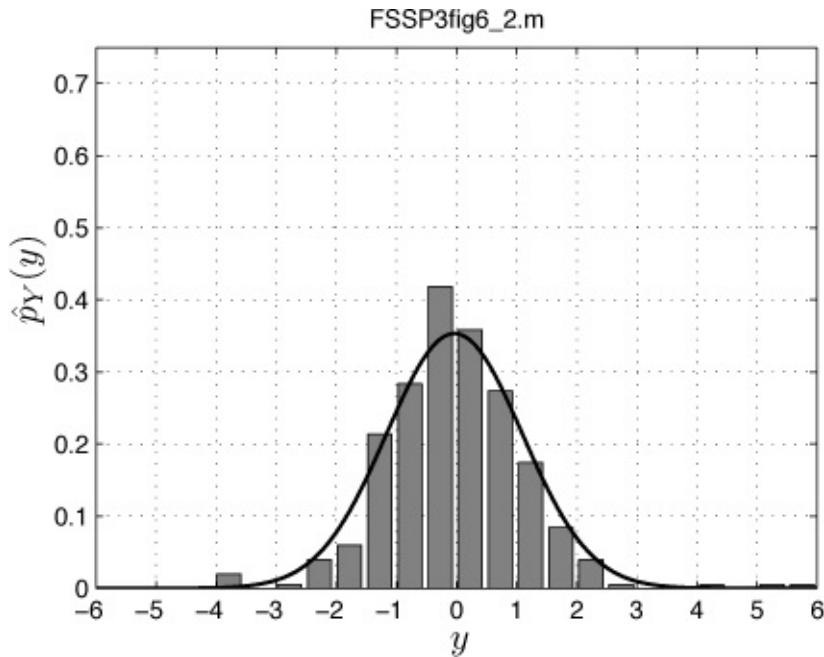
$$\begin{aligned}\mathbf{y} &= \mathbf{x} - \mathbf{H}\hat{\boldsymbol{\theta}} \\ &= \mathbf{x} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \\ &= (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T) \mathbf{x}.\end{aligned}\quad (6.3)$$

If  $\mathbf{x} = \mathbf{H}\boldsymbol{\theta} + \mathbf{w}$  (recall the linear model form of (3.21) and see [Algorithm 9.2](#) for a sinusoid in noise), what is  $\mathbf{y}$  for each value of  $\boldsymbol{\theta}$ ? Does (6.3) get rid of the interference? What happens to the noise? Has it also been altered?



We can now assume that  $y[n]$  consists of the noninterference noise, although this is clearly an approximation unless the estimates of amplitude and phase in (6.1) are perfect, which they are not. Thus, the original data  $x[n]$ , which was nonstationary, has been converted to the approximately stationary data  $y[n]$ . (We have implicitly assumed that the motion noise, which ideally is  $y[n]$ , is stationary.) Our next task is to decide whether the noise shown in [Figure 6.3](#) can be adequately modeled as Gaussian. By this we mean that each sample should have a Gaussian PDF. To even attempt such a decision we will need to assume that all the samples have the *same PDF*, sometimes referred to as *identically distributed*. Otherwise, we cannot go further. To estimate a PDF from data it is necessary to have multiple samples from the same PDF, much like to estimate the mean of a random variable we will need multiple samples with the *same mean* to average together. Estimation of the PDF will be discussed in more detail in [Section 6.4](#). For now we will assume that this can be done with the PDF estimate given by the bar plot in [Figure 6.4](#). Since the Gaussian PDF requires specification of its mean  $\mu$  and variance  $\sigma^2$  parameters, we estimate these from the data  $y[n]$  to obtain

$$\begin{aligned}\hat{\mu} &= \frac{1}{N} \sum_{n=0}^{N-1} y[n] = -0.0345 \\ \widehat{\sigma^2} &= \frac{1}{N} \sum_{n=0}^{N-1} (y[n] - \hat{\mu})^2 = 1.2758.\end{aligned}$$



**Figure 6.4:** Estimate of PDF of  $y[n]$  (assumed to be the same for each value of  $n$ ) shown as the bar chart versus a Gaussian PDF with its mean and variance matched to the estimated mean and variance of the  $y[n]$  data.

Then we use these in the theoretical Gaussian PDF to produce

$$\hat{p}_Y(y) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} \exp\left[-\frac{1}{2\hat{\sigma}^2}(y - \hat{\mu})^2\right]$$

which is shown as the solid curve in [Figure 6.4](#). The Gaussian fit does not appear to be particularly good. There are some events that exceed  $\pm 3\sigma$ . In this case  $\hat{\sigma}^2 \approx 1.28$  and so events larger than about  $3\hat{\sigma} = 3.39$  ( $\hat{\sigma} = \sqrt{\hat{\sigma}^2}$ ) in magnitude should not occur. The theoretical probability of these occurring is 0.0027 so that for  $N = 400$  samples, we expect about one occurrence. Yet as seen in [Figure 6.3](#) there are 6 occurrences. Also, referring to [Figure 6.4](#) the estimated PDF near  $y = 0$  appears to be too large relative to the Gaussian fitted PDF. With only 400 samples, however, our conclusions are only guesses so that in addition to an examination of the time data and the estimated PDF, additional measures of nonGaussianity may be helpful in making a modeling decision. One very simple test is to compute the *kurtosis*, which measures the fourth-order central moment  $E[(Y - \mu)^4]$ . Large excursions from the mean will yield large values of the kurtosis measure due to the raising of  $Y - \mu$  to the fourth power. It is defined as

$$\kappa = \frac{E[(Y - \mu)^4]}{\text{var}^2(Y)} \quad (6.4)$$

and is the normalized fourth-order central (about the mean) moment. For a Gaussian PDF with mean  $\mu$  and variance  $\sigma^2$  it is equal to 3.

---

### Exercise 6.3 – Kurtosis for a Gaussian random variable

Using the result that

$$E[(Y - \mu)^k] = 1 \cdot 3 \cdot 5 \cdots (k - 1)\sigma^k \quad (6.5)$$

if  $k$  is an even integer, find the kurtosis.




---

When the kurtosis is larger than 3, it indicates a nonGaussian PDF, and in particular, one that has large tails. To estimate the kurtosis we use

$$\hat{\kappa} = \frac{\frac{1}{N} \sum_{n=0}^{N-1} (y[n] - \bar{y})^4}{\left( \frac{1}{N} \sum_{n=0}^{N-1} (y[n] - \bar{y})^2 \right)^2} \quad (6.6)$$

where  $\bar{y} = (1/N) \sum_{n=0}^{N-1} y[n]$ . For the residual data of [Figure 6.3](#) the kurtosis is  $\hat{\kappa} = 6.58$ , indicating a strong nonGaussianity. In [Appendix 6A](#) we describe the concept of confidence intervals and in particular, the one for  $\hat{\kappa}$ . This is useful for determining how large the estimated kurtosis needs to be to declare the PDF to be nonGaussian. For example, is  $\hat{\kappa} = 4$  large enough or maybe we should require  $\hat{\kappa} = 5$  to decide that a PDF is nonGaussian? Since the 95% confidence interval for kurtosis based on a data record of length  $N = 400$  is [6.10, 7.06] (see [Exercise 6.19](#) in [Appendix 6A](#)), we conclude that the noise  $y[n]$  should be modeled as nonGaussian and thus, proceed to the next step in [Figure 6.1](#) of modeling the PSD. According to the roadmap we now need to assess whether the PSD is flat. This requires estimation of the PSD, with a more detailed discussion to be given in [Section 6.4](#). We will employ the *averaged periodogram* estimator (see [Chapter 11](#)). Since we have  $N = 400$  samples of noise data, we divide up the data record into 10 consecutive nonoverlapping blocks of 40 samples each, compute the periodogram for each block of data, and finally average the periodograms at each frequency. Before doing so, however, it is convenient to transform the data so that it is *standardized*. This means we wish to make sure the mean is zero and the variance is one.

---

## Exercise 6.4 – Standardizing a random variable

Show that if a random variable  $X$  has a mean  $E[X] = \mu$  and a variance  $\text{var}(X) = \sigma^2$ , then the new random variable

$$Y = \frac{X - \mu}{\sqrt{\sigma^2}}$$

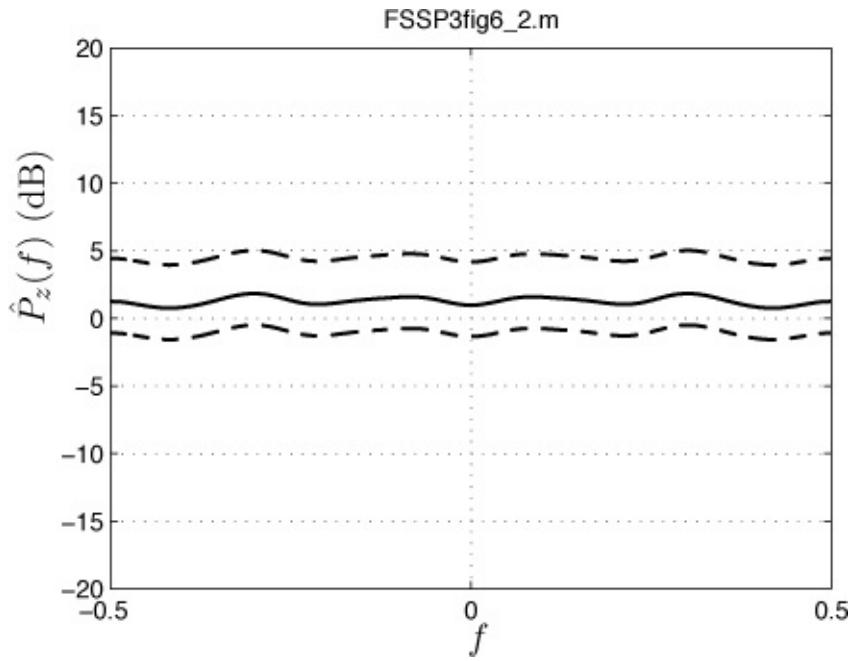
has a mean of zero and a variance of one. Hint: Use the properties  $E[aX + b] = aE[X] + b$  and  $\text{var}(aX + b) = a^2\text{var}(X)$ , for  $a$  and  $b$  constants.

---

To standardize the random variable we need to know the mean and variance of the data. The best we can do, in the absence of this knowledge, is to replace the theoretical mean and variance by their estimates. Hence, we replace the data set  $y[n]$  by

$$z[n] = \frac{y[n] - \bar{y}}{\sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (y[n] - \bar{y})^2}} \quad (6.7)$$

and then compute the averaged periodogram estimate for  $z[n]$ . Note that the standardization process will not alter the *shape* of the PSD, only its value near  $f = 0$ , since the temporal mean has been subtracted, and also the total power will be normalized to one. A decision about its flatness is then more easily made. The results are shown in [Figure 6.5](#) with the averaged periodogram plotted in dB quantities. The reason for the logarithmic plot along the vertical axis is explained in more detail in [Section 6.4.7](#).



**Figure 6.5: Estimate of PSD of standardized data  $z[n]$ , using the averaged periodogram, and the 95% confidence interval (dashed curves).**

The dashed curves are the 95% confidence intervals (given in [Section 6.4.7](#)). With the standardization a flat PSD would have the mathematical form  $P_z(f) = 1$  for  $-1/2 \leq f \leq 1/2$ , which in dB quantities is  $10 \log_{10} 1 = 0$  dB. It is seen that the estimated PSD is close to the 0 dB line, but could vary about  $\pm 3$  dB according to the confidence intervals. With these results we conclude that the PSD is indeed flat since it does not appear to exhibit any sharp resonances or nulls, which would preclude such a conclusion.

---

### Exercise 6.5 – Using the autocorrelation sequence to determine PSD flatness

Another way to determine if a PSD is flat is to examine its autocorrelation sequence (ACS) (see [Appendix 4A](#)). Since the PSD is the Fourier transform of the ACS [[Kay 2006](#)], a flat PSD corresponds to an impulsive ACS. For a flat PSD  $P_z(f) = 1$ , the inverse Fourier transform, i.e., the ACS, is  $r_z[k] = 1$  for  $k = 0$  and  $r_z[k] = 0$ , otherwise. The MATLAB program `autocorrelation_est(z, p)` estimates the ACS  $r_z[0], r_z[1], \dots, r_z[p]$  from the data  $z[n]$ . Use it to estimate the ACS of  $z[n]$ . The data  $y[n]$  shown in [Figure 6.3](#) can be accessed by the MATLAB command

load FSSP3exer6\_5 and standardized to  $z[n]$  using (6.7). Is the estimated ACS indicative of the flat PSD? In the next section we will discuss the estimation of the ACS.

---



### Using multiple approaches lends credibility

It is always a good idea to analyze the data using a variety of approaches. The evidence yielded by a single analysis method may not support a definitive model. If, however, several approaches all seem to point to the same model, then its credibility is greatly enhanced. An example of this approach is the use of spectral estimation to decide if the PSD model should be chosen to be flat along with an examination of the estimated autocorrelation sequence as was illustrated in [Exercise 6.5](#).



Referring to [Figure 6.1](#) our noise model selection process has now proceeded to choosing the IID model, which is Item 4 in [Table 4.1](#). Recall that a flat PSD implies that the data samples are uncorrelated and hence we *assume* that they are also independent. The last modeling decision we need to make is to choose an appropriate *nonGaussian* PDF.

The estimated PDF was shown in [Figure 6.4](#). Recall that the estimate of the mean was  $\hat{\mu} = -0.0345$  and since there is no reason to believe otherwise, we will assume that the mean of the PDF is zero. Also, the estimated PDF does not seem to exhibit a skew, meaning more mass for  $y > 0$  than for  $y < 0$  or vice-versa. Therefore, we will assume that the PDF is even, i.e.,  $pY(-y) = pY(y)$ . These assumptions are consistent with usual noise characteristics. Next we proceed to estimate the PDF.

There are many methods available to estimate the PDF. In [Figure 6.4](#) we have used a standard histogram (see [Section 6.4.6](#) for details). Another approach is to assume that the PDF can be modeled using an AR model, much the same as was done in [Section 4.4](#) for modeling of the PSD.

---

### Exercise 6.6 – A PDF versus a PSD

The only difference between a PDF and a PSD is the total area under the curve. Note that both these functions are nonnegative. What is the total

area for a PDF, a PSD? Give an example of a noise process whose PSD has a total area of one and also a PSD whose total area is ten.

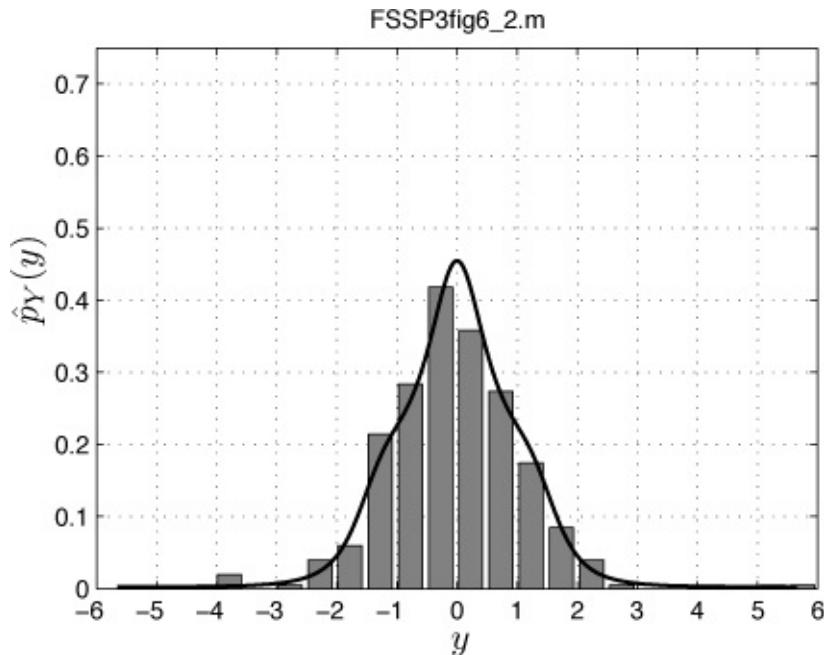
---

A description of the AR method is given in [Section 6.4](#). It is easily implemented and therefore suitable for on-line estimation of the PDF, as is required for a data-adaptive algorithm. The result of applying this estimator to the data shown in [Figure 6.3](#) is given in [Figure 6.6](#), along with the histogram estimate. The estimate is seen to produce a much better fit to the histogram than the Gaussian PDF, which was shown in [Figure 6.4](#). Assuming the operational data has the same statistical characteristics as the field data used to develop our noise model, we are now done.

To summarize, the noise model is

$$w[n] = A \cos[2\pi(60\Delta)n + \varphi] + w_{NG}[n] \quad n = 0, 1, \dots, N - 1$$

where  $A$  and  $\varphi$  will need to be estimated on-line. The  $w_{NG}[n]$  represents the nonGaussian noise, which is assumed to be IID noise with the PDF shown in [Figure 6.6](#).



**Figure 6.6:** Estimate of PDF for data given in [Figure 6.3](#). The AR PDF estimator is used to produce the solid curve. The bar plot is the histogram estimate, previously shown in [Figure 6.4](#).



## You can never be sure!

For a nominal price you can purchase a t-shirt from the American Statistical Association that says “Statistics means never having to say you’re certain!” Such is the curse and also the allure of designing statistical signal processing algorithms. The noise model we have finally chosen in our example is a reasonable one, but probably not the correct one, if such a correct model even exists. The only way to ever “be sure” is to have enough data so that any errors in the modeling procedure can be said to be negligible. Sadly, *that will never be the case in practice*. Hence, the foregoing procedure has produced a model that even with all its errors will hopefully lead to a signal processing algorithm that works “well” in practice. By *working well* we mean statistically or in the long run. All this will be made clearer in the next chapter when we discuss the performance evaluation of algorithms.



---

## 6.4. Estimation of Noise Characteristics

We now describe in more detail the methods used in the previous section to analyze the data of [Figure 6.2](#). These methods can also be used in a data-adaptive algorithm if the model parameters are not known in advance and so must be estimated online. Model order selection, determining the *total number* of model parameters, is discussed in [Section 6.5](#).

The estimation methods to be discussed include estimation of the following:

1. Moments, including mean, variance, covariance, and autocorrelation sequence
2. PDF, both first-order and multivariate Gaussian
3. PSD

### 6.4.1. Mean

The *mean* of a continuous random variable  $X$ , i.e., one whose outcomes lie in the interval  $-\infty < x < \infty$ , is defined as  $E[X] = \int_{-\infty}^{\infty} x p_X(x) dx$ . It is sometimes denoted by the symbol  $\mu$  and is estimated using the *sample mean*. If we are given the data samples  $x_1, x_2, \dots, x_N$ , which are assumed to *all have the same mean*  $E[X]$ , then the sample mean estimator is

$$\widehat{E[X]} = \frac{1}{N} \sum_{i=1}^N x_i. \quad (6.8)$$

The symbolism  $\bar{x}$  is also used to denote the sample mean.

---

### Exercise 6.7 – DC level versus mean estimator

If we observe the data  $x[n] = A + w[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $A$  is to be estimated and  $w[n]$  is noise with each sample *having a mean of zero*, how should we estimate  $A$ ? Explain your answer. What would happen if the mean of  $w[n]$  were not zero?




---

As the data record length  $N$  increases,  $\widehat{E[X]}$  will become closer to the true mean (see also [Figure 1.1](#)). For IID Gaussian data samples a 95% confidence interval for the mean is

$$\left[ \widehat{E[X]} - 1.96 \frac{\sigma}{\sqrt{N}}, \widehat{E[X]} + 1.96 \frac{\sigma}{\sqrt{N}} \right] \quad (6.9)$$

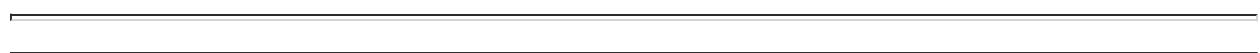
where  $\sigma = \sqrt{\text{var}(X)}$ . The correct interpretation of this interval is as follows. If, for example,  $\widehat{E[X]} = 2.5$ ,  $N = 100$ , and  $\sigma^2 = 1$ , we can say that 95% of the time, this interval, which is  $[2.5 - 1.96/\sqrt{100}, 2.5 + 1.96/\sqrt{100}] = [2.30, 2.70]$ , will *contain the true mean*. To reduce the length of the interval for a more precise statement about the true mean we must increase the data record length. Finally, if  $\sigma^2$  is unknown, as is typically the case, then it must be replaced by an estimate.

---



**If in doubt, increase the data record length  $N$ .**

Whenever the confidence interval is too large, a foolproof method of determining the value of a parameter is to increase the data record length until the estimate converges to a *fixed number*. Of course, this assumes we have access to additional data and also that the data is stationary (at least each sample should have the same mean). It is convenient to display the estimate versus  $N$  in order to assess this (as in [Figure 1.1](#)).



## Exercise 6.8 – Estimating the mean

Generate 10,000 outcomes of a  $N(1, 1)$  random variable using the MATLAB command  $x=1+randn(10000, 1)$ . Compute  $\widehat{E[X]}$  using (6.8) for each  $N$ , i.e.,  $\widehat{E[X]} = x_1, (x_1 + x_2)/2, (x_1 + x_2 + x_3)/3$ , etc. How many samples are required for the estimate to converge to 1?

---

### 6.4.2. Variance

The *variance* of a continuous random variable  $X$  is defined as  $\text{var}(X) = E[(X - E[X])^2] = \int_{-\infty}^{\infty} (x - E[X])^2 p_X(x) dx$ . It is also denoted by  $\sigma^2$ . Similar to the mean estimate, we replace the expectation operator  $E[\cdot]$  by an average over the samples of the expectation argument, which in this case is  $(x - E[X])^2$ , to produce the estimate

$$\widehat{\text{var}(X)} = \widehat{\sigma^2} = \frac{1}{N} \sum_{i=1}^N (x_i - \widehat{E[X]})^2 \quad (6.10)$$

where  $\widehat{E[X]}$  is given by (6.8). Here the approximate (we do not know  $\sigma^2$  and the confidence interval of the estimator depends on this unknown parameter) 95% confidence interval is

$$\left[ \widehat{\text{var}(X)} - 1.96 \sqrt{\frac{2(\widehat{\sigma^2})^2}{N}}, \widehat{\text{var}(X)} + 1.96 \sqrt{\frac{2(\widehat{\sigma^2})^2}{N}}, \right].$$

Note that the width of the confidence interval will be larger than for the mean (see (6.9)) if  $2\sigma^4/N > \sigma^2/N$  or if  $\sigma^2 > 1/2$ . This is because squaring the data causes the variability to increase. For example, to estimate the fourth-order moment (as we have done to estimate the kurtosis) we would use

$$\widehat{E[X^4]} = \frac{1}{N} \sum_{i=1}^N x_i^4 \quad (6.11)$$

while for the second-order moment we would use

$$\widehat{E[X^2]} = \frac{1}{N} \sum_{i=1}^N x_i^2. \quad (6.12)$$

The variability of the fourth-order moment estimator will be larger for  $\sigma^2 > 1/\sqrt{48}$ . Hence, it is *usually more difficult to estimate the higher-order*

moments.

---

### Exercise 6.9 – Estimating second-and fourth-order moments of a Gaussian random variable

Use the MATLAB command `x=randn(30, 1)` to generate 30 outcomes of a  $N(0, 1)$  random variable and compute the estimated second-and fourth-order moments using (6.11) and (6.12). Then, repeat the entire experiment 100 times to generate 100 estimates for each moment. Plot the pairs  $(\widehat{E[X^2]}, \widehat{E[X^4]})$  (called a *scatter diagram*), but don't connect the points. Which estimate has a higher variability?

•

---

#### 6.4.3. Covariance

For two continuous random variables  $X$  and  $Y$  the *covariance* is defined as [Kay 2006]

$$\begin{aligned}\text{cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - E[X])(y - E[Y]) p_{X,Y}(x, y) dx dy\end{aligned}$$

where  $p_{X, Y}(x, y)$  is the *joint PDF* of  $X$  and  $Y$ . This quantity is of immense interest in signal processing since it can be used to predict the outcome  $y$  of an *unobserved* random variable by using a linear function of the outcome  $x$  of an *observed* random variable. If the covariance is not zero, then some prediction (via a linear function) is possible and the random variables are said to be *correlated*. The reader is asked to consider the following two extremes.

---

### Exercise 6.10 – Uncorrelated versus correlated random variables

Generate 1000 outcomes of  $X$  and  $Y$  by using the MATLAB commands given by `x=rand(1000, 1)-0.5` and `y=rand(1000, 1)-0.5`, and then form the ordered pairs of numbers  $(x_i, y_i)$  for  $i = 1, 2, \dots, 1000$ . Plot  $y$  versus  $x$  using the plotting command `plot(x, y, '.' )`. Can you predict the outcome of  $Y$  if you know that  $X = x = 0$  has been observed? Repeat the experiment but instead use the commands `u=rand(1000, 1)`, `x=u-0.5`, `y=u-0.5`. Are the random variables correlated in each case?

•

---

The covariance is estimated in essentially the same way as the variance. However, now we need to have  $N$  pairs of outcomes  $(x_i, y_i)$  for  $i = 1, 2, \dots, N$ . The estimate is then given by

$$\widehat{\text{cov}(X, Y)} = \frac{1}{N} \sum_{i=1}^N (x_i - \widehat{E[X]})(y_i - \widehat{E[Y]}) \quad (6.13)$$

where  $\widehat{E[X]} = (1/N) \sum_{i=1}^N x_i$ , and  $\widehat{E[Y]} = (1/N) \sum_{i=1}^N y_i$ .

#### 6.4.4. Autocorrelation Sequence

For a stationary random process the *autocorrelation sequence* is defined as  $r_x[k] = E[X[n]X[n+k]]$  (see [Appendix 4A](#)). Assuming that we have the data samples  $x[0], x[1], \dots, x[N-1]$ , the ACS is estimated for  $k = 0, 1, \dots, p$  as

$$\hat{r}_x[k] = \frac{1}{N} \sum_{n=0}^{N-1-k} x[n]x[n+k] \quad (6.14)$$

and since the ACS is an even sequence, i.e.,  $r_x[-k] = r_x[k]$ , we use for  $k < 0$ ,  $\hat{r}_x[k] = \hat{r}_x[-k]$ , with the values of the ACS for positive  $k$  given by (6.14). Note that for a good estimate we require  $N \gg p$  so that the number of *lag products* in (6.14) that are averaged together, which is  $N - k$ , is large. A MATLAB program called `autocorrelation_est.m`, which is contained on the CD, can be used to compute the ACS estimate.

#### 6.4.5. Mean Vector and Covariance Matrix

For more than two random variables we denote them as  $X_1, X_2, \dots, X_L$  and refer to the  $L \times 1$  vector

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_L \end{bmatrix}$$

as the *random vector*. Each element is a random variable and as such has a mean  $E[X_i]$ , which when expressed as a vector is called the *mean vector*

$$E[\mathbf{X}] = \begin{bmatrix} E[X_1] \\ E[X_2] \\ \vdots \\ E[X_L] \end{bmatrix}.$$

To estimate the mean vector we need  $N$  data vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ , where each data vector has dimension  $L \times 1$ . Then, the mean vector estimate is

$$\widehat{E[\mathbf{X}]} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i. \quad (6.15)$$


---

### Exercise 6.11 – Mean vector estimate is just the vector of mean estimates

If the  $k$ th element of a vector  $\mathbf{x}$  is denoted by  $[\mathbf{x}]_k$ , show that the  $k$ th element of the mean vector estimate is

$$[\widehat{E[\mathbf{X}]}]_k = \frac{1}{N} \sum_{i=1}^N [\mathbf{x}_i]_k \quad (6.16)$$

which is just the usual sample mean for the  $k$ th element of the data vectors. Do this for the case of  $N = 2$ .




---

The *covariance matrix*, which has dimension  $L \times L$ , is defined as

$$\mathbf{C} = \begin{bmatrix} \text{var}(X_1) & \text{cov}(X_1, X_2) & \dots & \text{cov}(X_1, X_L) \\ \text{cov}(X_2, X_1) & \text{var}(X_2) & \dots & \text{cov}(X_2, X_L) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(X_L, X_1) & \text{cov}(X_L, X_2) & \dots & \text{var}(X_L) \end{bmatrix}.$$

It is estimated by estimating each element separately (note that it is symmetric so that only the elements on the main diagonal and above the main diagonal need to be estimated). We can use (6.10) for the diagonal elements and (6.13) for the off-diagonal elements. Denoting the  $[m, n]$  element of the estimated covariance matrix by  $[\hat{\mathbf{C}}]_{mn}$  we have that

$$[\hat{\mathbf{C}}]_{mn} = \frac{1}{N} \sum_{i=1}^N ([\mathbf{x}_i]_m - [\widehat{E[\mathbf{x}]}]_m)([\mathbf{x}_i]_n - [\widehat{E[\mathbf{x}]}]_n) \quad m = 1, 2, \dots, L; n = 1, 2, \dots, L.$$

It can be shown that we can express the estimate of the covariance *matrix* in more compact form as

$$\hat{\mathbf{C}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \widehat{E[\mathbf{X}]}) (\mathbf{x}_i - \widehat{E[\mathbf{X}]})^T \quad (6.17)$$

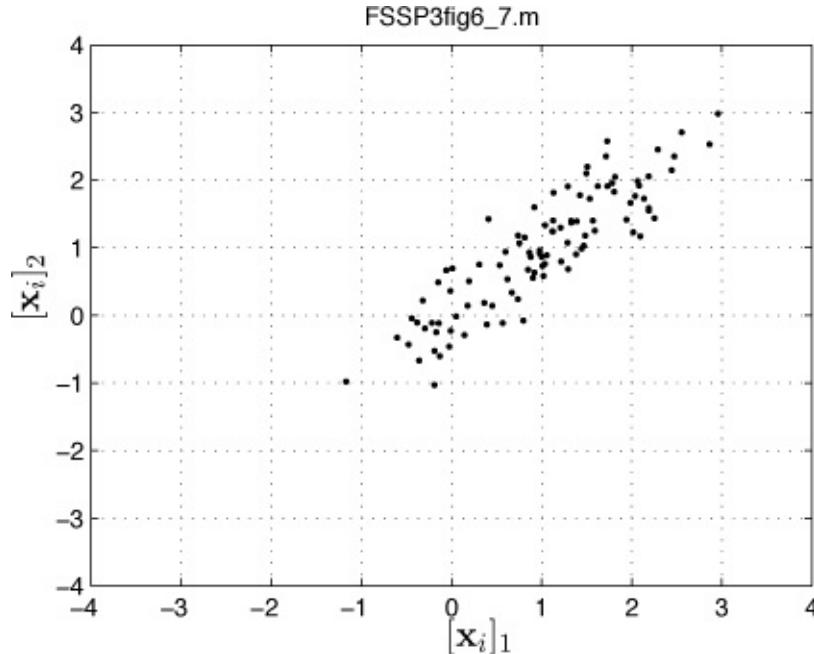
---

where  $\widehat{E[\mathbf{X}]}$  is given by (6.15). This form is useful for computation,

### Exercise 6.12 – Mean vector and covariance matrix computation

For  $L = 2$ , Gaussian random vector outcomes have been generated and are shown in [Figure 6.7](#). The  $2 \times 1$  outcomes  $\mathbf{x}_i$ , for  $i = 1, 2, \dots, 100$  are given in the data file `FSSP3exer6_12.mat` as the array  $Z$ , which is  $2 \times 100$ . It can be loaded into MATLAB using `load FSSP3exer6_12`. Use [\(6.15\)](#) and [\(6.17\)](#) to estimate the mean vector and the covariance matrix, respectively. Then compare them to the true values which are

$$E[\mathbf{X}] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}.$$



**Figure 6.7: 100 outcomes of a  $2 \times 1$  Gaussian random vector  $\mathbf{X}$ .**

The utility of being able to estimate the mean vector and covariance matrix is that when these quantities are known, the *entire* multivariate Gaussian PDF is known. That PDF for the observed data sample  $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_L]^T$  is given as

$$p_X(\mathbf{x}) = \frac{1}{(2\pi)^{L/2} \det^{1/2}(\mathbf{C})} \exp \left[ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right].$$

To specify the PDF we need only knowledge of the mean vector  $E[\mathbf{X}] = \boldsymbol{\mu}$  and the covariance matrix  $\mathbf{C}$ . These quantities are easily estimated using [\(6.15\)](#) and

(6.17). Thus, we are able to estimate the  $L$ -dimensional PDF. The multivariate Gaussian PDF is the *only one* that lends itself to such a simple procedure and hence its ubiquitous use in practice. Without the Gaussian assumption it is only practical to estimate the first-order PDF. Then, to obtain the  $L$ -dimensional PDF we are forced into assuming the noise samples are independent and each one has the same PDF (they are IID) so that we can multiply the first-order PDFs together to obtain the  $L$ -dimensional PDF. The IID assumption then implies that

$$pX(x_1, x_2, \dots, x_L) = pX(x_1)pX(x_2) \cdots pX(x_L).$$

By doing so, our problem of PDF estimation is reduced to estimating the first-order PDF  $pX(x)$ , an immense reduction in effort. Be aware, however, that the IID assumption ignores any correlation between samples. We next show how this is done, for example, as was used to obtain the PDF estimate shown in [Figure 6.4](#) as the bar plot.

#### 6.4.6. PDF

The PDF denoted by  $pX(x)$  is defined as the function, which must be nonnegative and integrate to one, and whose area between the values  $x = a$  and  $x = b > a$  gives the probability that an outcome of the random variable  $X$  will fall within the interval  $(a, b)$ . In theory, the values of  $a$  and  $b$  can be  $-\infty$  and  $\infty$ , respectively but in practice, this just means “large”. Since

$$P[a < X < b] = \int_a^b pX(x)dx$$

if  $a = x - \Delta x/2$ ,  $b = x + \Delta x/2$  and  $\Delta x$  is positive and very small, we have

$$P[a < X < b] \approx pX(x)\Delta x$$

by approximating the area under the curve by the area of a rectangle of width  $\Delta x$ . But the probability can be estimated using  $N$  outcomes as

$$P[a < X < b] \approx \frac{N((a, b))}{N}$$

where  $N((a, b))$  is the number of outcomes falling in the interval  $(a, b)$ . Therefore, we estimate the PDF at  $x$  using

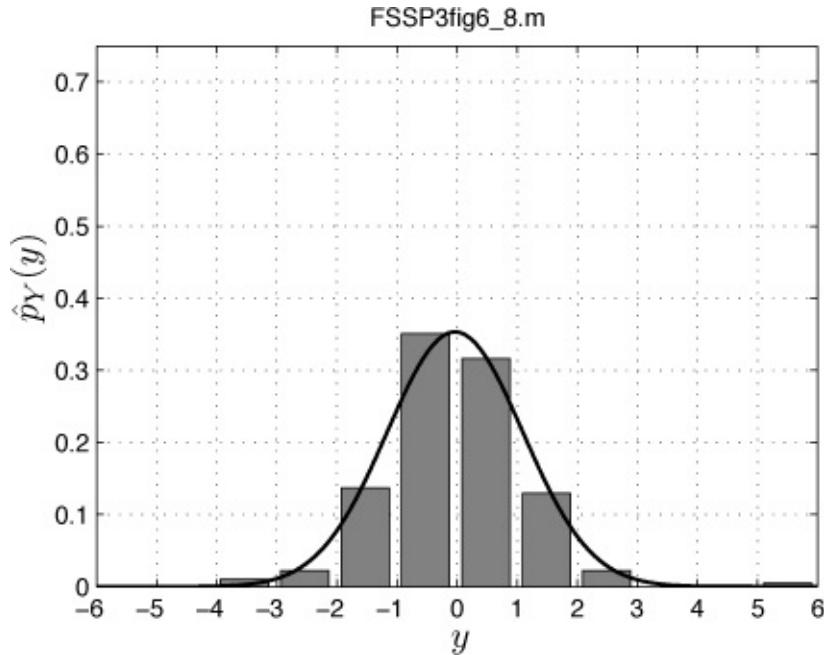
$$\hat{p}_X(x) = \frac{N((x - \Delta x/2, x + \Delta x/2))}{N\Delta x} \quad (6.18)$$

for some small  $\Delta x$ . An example has already been given in [Figure 6.4](#) as the bar plot. There we used the  $N = 400$  outcomes from [Figure 6.3](#), which ranged in

value from about  $-4.0$  to  $6.0$ . The number of points chosen to estimate the PDF, which is the number of bars shown in [Figure 6.4](#), is  $20$ . Hence, the distance between successive points, also called the *bin width*, was  $\Delta x = 0.50$ . The PDF estimate at  $x = -0.27$ , which corresponds to the bar with the largest height, was

$$\hat{p}_X(-0.27) = \frac{84}{(400)(0.50)} = 0.42$$

since  $84$  out of  $400$  outcomes lie within the interval  $(-0.27 - 0.50/2, -0.27 + 0.50/2) = (-0.52, -0.02)$ . Note that in some intervals there were no outcomes, and these are readily observed to correspond to a PDF estimate of zero. The latter occurrence is usually not reasonable in practice since the PDF should be a continuous curve. To avoid this possibility it is recommended as a “rule of thumb” that there should be at least  $10$  outcomes in each interval for which the PDF is to be estimated. One way to accomplish this is to widen the bins, such as shown in [Figure 6.8](#), in which there are now only  $10$  bins with the bin width doubled from those shown in [Figure 6.4](#).



**Figure 6.8: Estimate of PDF of  $y[n]$  shown as the bar chart versus a Gaussian PDF with its mean and variance matched to the estimated mean and variance. The bin width has been widened relative to that used in [Figure 6.4](#).**

As expected there are now fewer zero estimates. But the so-called “resolution” is reduced, meaning that the PDF estimate is smoother. The Gaussian fit even looks better now since the reduced number of bins is not able to capture the

nonGaussian nature of the PDF. This is indeed a quandary and illustrates a fundamental tradeoff when estimating a function (as opposed to a point, as for example, in the estimation of the DC level  $A$ ). For good resolution we require more bins, resulting in narrower bins. But narrower bins ( $\Delta x$  smaller) means that the probability of an outcome falling in a given bin decreases. Therefore, the variance of the PDF estimate increases and becomes more unreliable. An example follows.

---

### **Example 6.1 – Effect of bin width on PDF estimate**

Assume that we wish to estimate the PDF of the data generated by the MATLAB command `x=randn(100, 1)` at  $x = 0$  (we should already know what value to expect!). Consider the estimates

$$\hat{p}_X(0) = \frac{N((- \Delta x / 2, \Delta x / 2))}{100 \Delta x}$$

for  $\Delta x = 1, 0.5, 0.25, 0.125, 0.0625, 0.03125$ . The results are shown in [Table 6.1](#). It

**Table 6.1: Estimated PDF at  $x = 0$  for differing bin widths for a Gaussian PDF with mean zero and variance one.**

$\Delta x$	outcomes	$\hat{p}_X(0)$	$p_X(0)$
1	39	0.3900	0.3989
0.5	22	0.4400	0.3989
0.25	12	0.4800	0.3989
0.125	8	0.6400	0.3989
0.0625	3	0.4800	0.3989
0.03125	1	0.3200	0.3989

is seen that as  $\Delta x$  decreases, so does the number of outcomes in the bin centered about  $x = 0$ . We would be tempted to use a bin width as large as possible and in this example it would be the choice of  $\Delta x = 1$ , resulting in a very good estimate. However, in the *general case* a severely *biased* estimate may result by doing so. This is because we would obtain “on the average”

$$\begin{aligned}
E[\hat{p}_X(0)] &= \frac{E[N((-Δx/2, Δx/2))]}{NΔx} \\
&= \frac{NP[-Δx/2 < X < Δx/2]}{NΔx} \quad (\text{analogous to } N \text{ coin flips with} \\
&\quad \text{probability of heads } p \\
&\quad \text{- expected number of heads is } Np) \\
&= \frac{1}{Δx} \int_{-Δx/2}^{Δx/2} p_X(x) dx \quad (\text{definition of PDF})
\end{aligned}$$

which is a *smoothed version* (less resolution) of the actual PDF value.



### Exercise 6.13 – How much smoothing?

For a  $N(0, 1)$  PDF calculate  $\frac{1}{Δx} \int_{-Δx/2}^{Δx/2} p_X(x) dx$  for the values of  $Δx$  given in [Table 6.1](#). How do these smoothed values compare to the true one?

Hint: Recall the use of the  $Q$  function (see [\(2.6\)](#)) to evaluate the area under a Gaussian PDF.



Using our rule of thumb of at least 10 outcomes per bin, we would choose the narrowest bin width of  $Δx = 0.25$  in [Table 6.1](#) for an estimate of  $\hat{p}_X(0) = 0.48$ .

A confidence interval can be found for the PDF estimate. Since the variance of the PDF estimate can be shown to be approximately

$\text{var}(\hat{p}_X(x)) = p_X(x)/(NΔx)$ , the approximate 95% confidence interval is

$$\hat{p}_X(x) \pm 1.96 \sqrt{\frac{\hat{p}_X(x)}{NΔx}}. \quad (6.19)$$

For the problem at hand and using  $Δx = 0.25$  we have

$$0.48 \pm 1.96 \sqrt{\frac{0.48}{100(0.25)}} = (0.21, 0.75).$$

The method of PDF estimation just discussed is called the *histogram method*. It has been implemented in the MATLAB program `pdf_hist_est.m` and is contained on the CD.

### Exercise 6.14 – PDF estimation practice - histogram method

For the data given in [Figure 6.3](#) run the MATLAB program `pdf_hist_est.m` using 20 bins to obtain the estimate as shown in [Figure 6.4](#) (use `axis([-6 6 0 0.75])` to obtain the same axes scaling). Then, increase the number of bins to 35. Are the results any better? The data is contained in the MATLAB data file `FSSP3exer6_14.mat` as the array `y` and can be accessed by using `load FSSP3exer6_14`.

---

For the interested reader many variants of the histogram approach exist. Some of these are described in [[Webb 2002](#)] as methods for *nonparametric density estimation*.

The histogram approach to PDF estimation does not make any assumptions about the form of the PDF and is therefore called a *nonparametric estimator*. A second general approach models the PDF by a function with adjustable parameters. It assumes that the PDF is even ( $pX(-x) = pX(x)$ ), but can easily be extended to nonsymmetric PDFs [[Kay 1998](#)]. The parameters are estimated from the data to form the estimated PDF. The model chosen is our “old friend”, the AR model, although this time it is used to represent the PDF. Recall from [Section 4.4](#) that the AR PSD model is given by

$$P_x(f) = \frac{\sigma_u^2}{|1 + a[1]\exp(-j2\pi f) + \cdots + a[p]\exp(-j2\pi fp)|^2} \quad |f| \leq 1/2.$$

Noting that  $P_x(-f) = P_x(f)$  for real AR filter coefficients, we can use the model to represent a symmetric PDF if the data lie in the interval  $[-1/2, 1/2]$ . We need only replace the frequency variable  $f$  by the PDF variable  $x$ . Also, since we are representing a PDF we must make sure that  $\int_{-\frac{1}{2}}^{\frac{1}{2}} p_X(x)dx = 1$ . Hence, the AR PDF estimator is

$$p_X(x) = \frac{\sigma_u^2}{|1 + a[1]\exp(-j2\pi x) + \cdots + a[p]\exp(-j2\pi xp)|^2} \quad |x| \leq 1/2. \quad (6.20)$$

To make sure the data is in the specified interval we scale the data samples appropriately. Then, we use essentially the same procedure as for modeling a spectrum (see [Section 4.4](#)) and finally convert the estimated AR PDF to one that will represent the original unscaled data. The entire procedure is as follows, where we assume that the IID data  $\{x_1, x_2, \dots, x_N\}$  are available.

1. Scale the data to lie within the  $[-1/2, 1/2]$  interval by forming

$$y_i = \frac{x_i}{2k_s\hat{\sigma}}$$

where  $\hat{\sigma} = \sqrt{(1/N) \sum_{i=1}^N x_i^2}$  is the estimated standard deviation (recall that the PDF is symmetric and therefore  $E[X] = 0$ ) and typically we choose  $k_s = 5$  or slightly larger.

2. Numerically compute the “autocorrelation” sequence (actually the sampled characteristic function) as

$$\hat{\phi}_y[k] = \begin{cases} 1 & k = 0 \\ \frac{1}{N} \sum_{i=1}^N \cos(2\pi y_i k) & k = 1, 2, \dots, p. \end{cases}$$

3. Using the “autocorrelation” sequence solve the *Yule-Walker equations* to obtain the filter parameters the filter parameters

$$\begin{bmatrix} \hat{\phi}_y[0] & \hat{\phi}_y[1] & \dots & \hat{\phi}_y[p-1] \\ \hat{\phi}_y[1] & \hat{\phi}_y[0] & \dots & \hat{\phi}_y[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\phi}_y[p-1] & \hat{\phi}_y[p-2] & \dots & \hat{\phi}_y[0] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} \hat{\phi}_y[1] \\ \hat{\phi}_y[2] \\ \vdots \\ \hat{\phi}_y[p] \end{bmatrix}. \quad (6.21)$$

Once the equations have been solved for the AR filter parameters, the excitation noise variance is found as

$$\hat{\sigma}_u^2 = \hat{\phi}_y[0] + \hat{a}[1]\hat{\phi}_y[1] + \dots + \hat{a}[p]\hat{\phi}_y[p].$$

5. Compute the  $AR(p)$  modeled PDF by substituting the obtained AR parameters into (6.20) and note that  $X = (2k_s\hat{\sigma})Y$  to obtain (since if  $X = aY$ , then  $pX(x) = pY(x/a)(1/|a|)$ )

$$\hat{p}_X(x) =$$

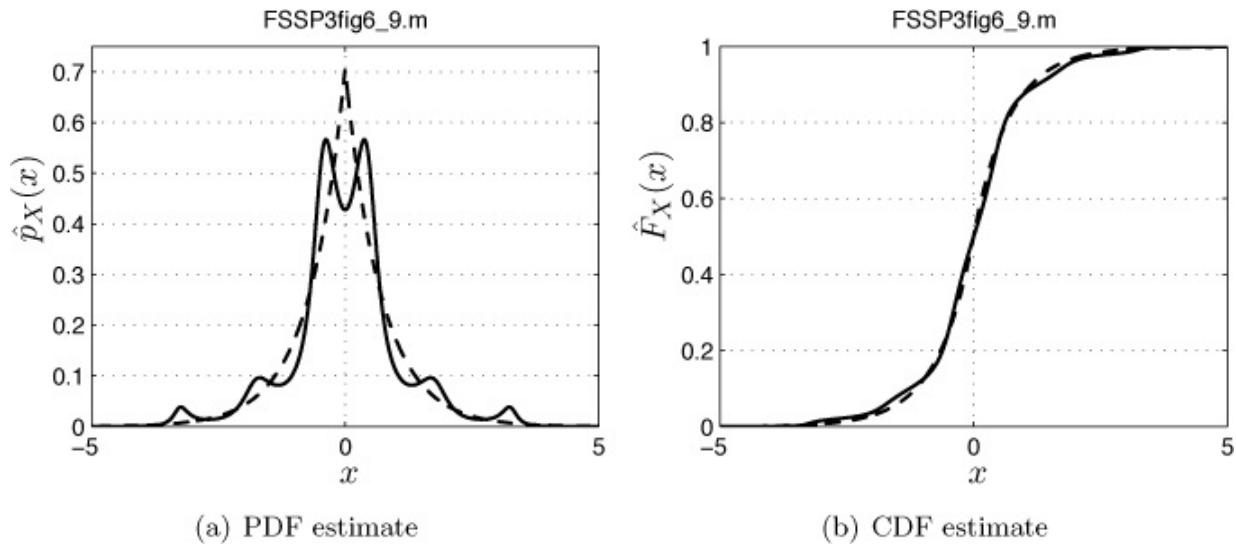
$$\frac{(\hat{\sigma}_u^2)/(2k_s\hat{\sigma})}{|1 + \hat{a}[1]\exp[-j2\pi(x/(2k_s\hat{\sigma}))] + \dots + \hat{a}[p]\exp[-j2\pi(x/(2k_s\hat{\sigma}))p]|^2} \quad (6.22)$$

for  $|x| \leq k_s\hat{\sigma}$  and zero otherwise.

This procedure has been implemented in the MATLAB program `pdf_AR_est.m`, which is contained on the CD. It is assumed that the model order  $p$  is known. If this is not the case, and the model order must be estimated, then the MATLAB program `pdf_AR_est_order.m`, also contained on the CD, can be used. The model order estimator is described in the next section. As an example, using both programs for the data shown in [Figure 6.3](#), it is found that using  $k_s = 5$ , the model order estimate is  $\hat{p} = 4$  and the estimated PDF using this order has been shown in [Figure 6.6](#).

As a second example, consider data that has the Laplacian PDF as shown in [Figure 4.16](#). Using  $N = 400$  outcomes and  $k_s = 5$  the AR PDF estimate is shown

in [Figure 6.9a](#), along with the true PDF. The typical equiripple approximation to the true PDF is exhibited by the AR model. This was previously noted in [Figure 4.9](#) for the AR PSD modeling. Since we will be ultimately interested in probabilities, this artifact may not cause any difficulties. The estimated cumulative distribution function (CDF) obtained by integrating the PDF estimate is shown in [Figure 6.9b](#) and is seen to be a close match to the true one. In this example we have chosen a nontypical set of outcomes, yielding a poor PDF estimate, to illustrate the “improvement” afforded by the CDF. The latter, because of its definition as an integral, essentially smooths the PDF estimate.



**Figure 6.9: Estimate of PDF and CDF of a Laplacian random variable using an AR PDF estimate with an estimated model order of  $\hat{p} = 6$ . The true PDF and true CDF are shown as the dashed curves.**

### Exercise 6.15 – PDF estimation practice - AR method

For the data given in [Figure 6.3](#) run the MATLAB program `pdf_AR_est.m` using  $p = 2$  and also  $p = 15$ . Use  $k_s = 5$ . Are the results any better than those shown in [Figure 6.6](#) in which  $p = 4$ ? How does the choice of model order affect the estimate? The data is contained in the MATLAB data file `FSSP3exer6_15.mat` as the array `y` and can be accessed by using `load FSSP3exer6_15`. Hint: The PDF estimate can be plotted automatically by calling the subprogram with the appropriate arguments. Also, use `axis([-6 6 0 0.75])` to obtain the same axes scaling as in [Figure 6.6](#).



---

See also [Chapter 14](#) for an application of the AR PDF method to estimation of the PDF of geomagnetic noise.

#### 6.4.7. PSD

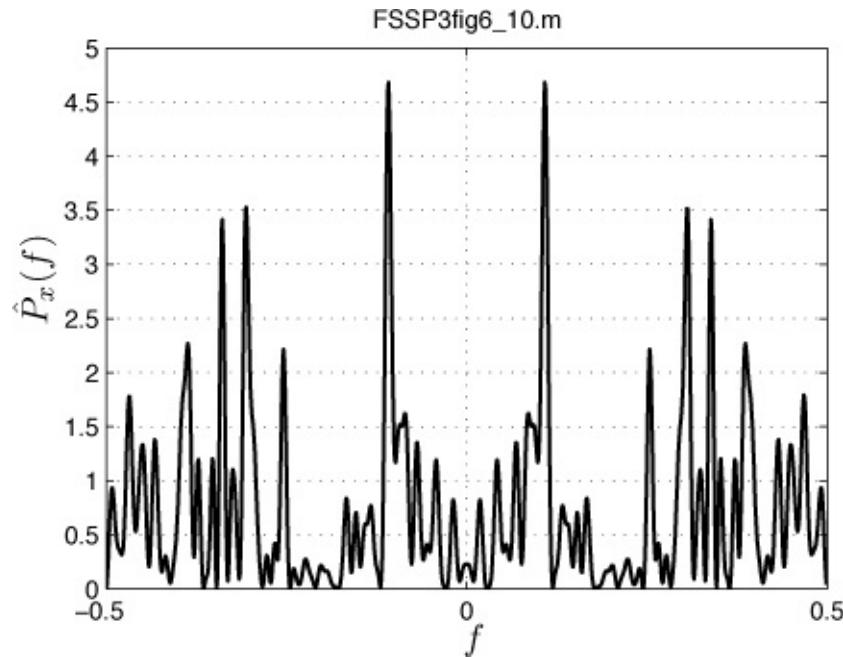
In this section we briefly describe the method for PSD estimation used to produce [Figure 6.5](#). A more complete discussion of spectral estimation is given in [Chapter 11](#). A standard estimate of the PSD is motivated by the definition (see [Appendix 4A](#))

$$P_x(f) = \lim_{M \rightarrow \infty} \frac{1}{2M+1} E \left[ \left| \sum_{n=-M}^M x[n] \exp(-j2\pi fn) \right|^2 \right].$$

If we ignore the expectation operation (since we usually only have access to a single realization) and also use all the available data (the observed portion of the realization), which is assumed to be  $x[n]$  for  $n = 0, 1, \dots, N - 1$ , then we have the *periodogram* spectral estimator given by

$$\hat{P}_x(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi fn) \right|^2. \quad (6.23)$$

In some instances the periodogram is an optimal estimator such as for the *frequency* of a single sinusoid in WGN (see [Algorithm 9.3](#)), but not so in general for an arbitrary data set. As an example, consider  $N = 100$  samples of WGN with variance  $\sigma^2 = 1$ . For this data set the true PSD is  $P_x(f) = \sigma^2 = 1$ . Yet as shown in [Figure 6.10](#) it is very noisy. It might be expected that increasing the data record length would improve the performance. Such is not the case, as is illustrated in the next exercise.



**Figure 6.10:** Estimate of PSD of WGN with  $\sigma^2 = 1$  using a periodogram for  $N = 100$  data samples. The true PSD is  $P_x(f) = 1$ .

---

### Exercise 6.16 – Inconsistency of the periodogram spectral estimator

Generate  $N = 500$  samples of WGN of variance  $\sigma^2 = 1$  by using  $x = \text{randn}(500, 1)$  and then estimate the PSD using the periodogram. Do so by running the MATLAB program `PSD_est_avper(x, 500, 1024, 1, 0)`, which is contained on the CD. Is the estimate any better? What appears to happen as the data record length  $N$  increases?




---

The only way to reduce the estimation variance is to perform some averaging at each frequency. To do so the data set  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is divided into  $I$  successive blocks of length  $L$  samples each, where it is assumed that  $N = IL$ . The data blocks are given by

$$y_i[n] = x[n + iL] \quad n = 0, 1, \dots, L - 1; i = 0, 1, \dots, I - 1.$$

Then for each block we compute the periodogram as

$$\hat{P}_x^{(i)}(f) = \frac{1}{L} \left| \sum_{n=0}^{L-1} y_i[n] \exp(-j2\pi f n) \right|^2 \quad (6.24)$$

and then average all the periodograms together to yield the *averaged periodogram* as

$$\hat{P}_x(f) = \frac{1}{I} \sum_{i=0}^{I-1} \hat{P}_x^{(i)}(f). \quad (6.25)$$

This procedure was used to produce [Figure 6.5](#) in which  $N = 400$ ,  $L = 40$  (length of each block), and  $I = 10$  (number of blocks). The MATLAB program given by `PSD_est_avper.m` implements this estimator. Many variations of this PSD estimator can be found in the literature and some are mentioned in [Chapter 11](#).

---

### **Exercise 6.17 – PSD estimation practice - averaged periodogram estimator for an AR process**

Generate  $N = 500$  successive samples of an AR random process with model order  $p = 2$  and parameters  $a[1] = -2r \cos(f_0)$ ,  $a[2] = r^2$ , and  $\sigma_u^2 = 1$  for  $r = 0.95$  and  $f_0 = 0.25$ . You can use the MATLAB program `ARgendata.m` to do this. Then, estimate the PSD using the averaged periodogram with  $L = 500$ ,  $L = 100$ , and  $L = 20$ . Plot the estimated PSDs in linear quantities and compare them to the true PSD (use `ARpsd.m` to compute the true PSD values). Explain your results.




---

As illustrated by the previous exercise, as the number of periodograms averaged together increases, the variability of the estimate decreases *but* the resolution also decreases. In statistical terms *the variance decreases but the bias increases*. This is a *fundamental tradeoff* inherent in all estimation but is particularly prevalent in spectral estimation. This phenomenon is called the *bias-variance tradeoff*.

The confidence interval for the averaged periodogram shown in [Figure 6.5](#) is based on a plot of the estimate in logarithmic, i.e., dB, quantities. The advantage in doing so is that *the length of the confidence interval does not depend on frequency*. It can be shown that [[Kay 1988](#)] the 95% confidence interval is

$$10 \log_{10} \hat{P}_x(f) \begin{cases} +10 \log_{10} \frac{2I}{F_{\chi^2_\nu}^{(-1)}(0.025)} \\ -10 \log_{10} \frac{F_{\chi^2_\nu}^{(-1)}(0.975)}{2I} \end{cases} \text{ dB} \quad (6.26)$$

where  $F_{\chi^2_\nu}^{(-1)}(x)$  is the inverse CDF for a  $\chi^2_\nu$  random variable, i.e., a chi-squared distribution with  $\nu$  degrees of freedom. For the spectral estimate given in [Figure 6.5](#) we have that  $I = 10$  and it is readily found that  $F_{\chi^2_{20}}^{(-1)}(0.025) = 9.6$  and  $F_{\chi^2_{20}}^{(-1)}(0.975) = 34.2$ . You can use the MATLAB subprogram `chipr2.m`, which is contained on the CD, to verify that  $F_{\chi^2_{20}}(9.6) = 0.025$  and  $F_{\chi^2_{20}}(34.2) = 0.975$ . Using these values in (6.26) produces

$$10 \log_{10} \hat{P}_x(f) \begin{cases} +3.19 \\ -2.33 \end{cases} \text{ dB.}$$

Another advantage of plotting the PSD estimate in logarithmic quantities is that it allows a wider dynamic range to be displayed. This is particularly useful in the detection of multiple sinusoids with widely disparate powers (see [[Kay 1988](#), pg. 70]).

In our introductory example of noise model selection we concluded that a flat PSD was a reasonable model. Hence, it is only necessary to estimate the power since  $P_x(f) = \sigma^2$ . If we had concluded that the PSD was not flat, then the next step would have been to choose a model for the PSD. The usual model for colored noise would be based on an AR PSD and so the model order and parameters would then need to be estimated. Since this problem is part of the more general problem known as *parametric spectral estimation*, which is described in [Chapter 11](#), we will defer our discussion until then.

## 6.5. Model Order Selection

The exponentially embedded family (EEF) approach, a method that works well for model order selection, was introduced in [Section 5.5](#) for signal modeling. For noise modeling it is used for determining the order of an AR PSD model and an AR PDF model. The former is discussed in [Chapter 11](#) while the latter is now described. In this chapter we have used it to determine the model order for the AR PDF estimate shown in [Figure 6.6](#). The EEF approach finds the value of  $p$  as the  $k$  that maximizes the criterion

$$\text{EEF}(k) = \begin{cases} \xi_k - k \left[ \ln \left( \frac{\xi_k}{k} \right) + 1 \right] & \text{if } \frac{\xi_k}{k} \geq 1 \\ 0 & \text{if } \frac{\xi_k}{k} < 1 \end{cases} \quad k = 1, 2, \dots, k_{\max} \quad (6.27)$$

where

$$\xi_k = 2 \ln \prod_{i=1}^N \frac{\hat{p}_x^{(k)}(x_i)}{\hat{p}_x^{(0)}(x_i)} \quad (6.28)$$

and  $\hat{p}_x^{(k)}(x)$  is the AR PDF estimate assuming a model order of  $k$  and is given by (6.22). For  $k = 0$  we use  $\hat{p}_x^{(0)}(x) = \widehat{\sigma_u^{(0)}}/(2k_s\hat{\sigma})$  and note that for the PDF to integrate to one, we must have

$$\int_{-k_s\hat{\sigma}}^{k_s\hat{\sigma}} \hat{p}_x^{(0)}(x) dx = 1$$

and therefore  $\widehat{\sigma_u^{(0)}} = 1$ . This model order estimator has been implemented in the MATLAB program `pdf_AR_est_order.m` and can be found on the CD.

## 6.6. Lessons Learned

- An exact noise model does not exist. The best we can hope for is a good approximation in modeling the observed data.
- Models are either fixed or data-adaptive. Use a fixed model if the statistical characteristics of the data do not change from data set to data set. Otherwise, use a data-adaptive model that estimates the model parameters on-line.
- Be careful not to include too many parameters into the model. A simpler model is usually better.
- Any prior knowledge about the data characteristics should be incorporated into the model.
- To remove sinusoidal interference, estimate amplitude and phase and then subtract the estimated interference from the data.
- Use a confidence interval to assess the accuracy of an estimated parameter and whether statistical error could have caused a large deviation of the estimated parameter from the true value.
- If possible, use multiple analysis methods to determine a model, appealing to consistency to suggest a good model.
- If possible, always increase the data record length to ensure that the estimate of a model parameter has converged to the true value.

- Higher order moments are more difficult to estimate and therefore require more data.
- The only multivariate PDF that can be reliably estimated in practice is the Gaussian due to its sole dependence on the mean and covariance.
- In estimating a PDF you should have at least 10 outcomes per bin.
- Estimation of the CDF always produces a smoother estimate than for the PDF.
- The periodogram produces an unreliable spectral estimate. Some averaging is required.
- A fundamental tradeoff in spectral estimation is that as the estimator yields higher resolution (less bias), it also exhibits more variability (more variance).
- Plot PSD estimates in dB quantities to allow the use of constant length confidence intervals and also to display wide dynamic range spectra.
- Model order selection methods typically err on the side of estimating too many model parameters as opposed to too few parameters.

## References

- Cox, D.R., D.V. Hinkley, *Theoretical Statistics*, Chapman and Hall, NY, 1974.
- Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Kay, S., “Model-Based Probability Density Function Estimation”, *IEEE Signal Processing Letters*, Dec. 1998.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- Ligtenberg, A., M. Kunt, “A Robust-Digital QRS-Detection Algorithm for Arrhythmia Monitoring”, *Computers and Biomedical Research*, pp. 273–286, 1983.
- Webb, A., *Statistical Pattern Recognition, Second Ed.*, J. Wiley, NY, 2002.

## Appendix 6A. Confidence Intervals

In deciding against the assumption of a Gaussian PDF to model the data shown in [Figure 6.3](#), we implicitly relied on a confidence interval. Recall that a Gaussian random variable with a mean of zero and a variance of  $\sigma^2$  should *seldom* exceed  $\pm 3\sigma$ , but a visual examination of the data reveals many such

exceedances. Loosely speaking, we are *confident* that the outcome of the random variable should lie in the *interval*  $[-3\sigma, 3\sigma]$ . If it does not, then it is reasonable to reject our initial premise of a  $N(0, \sigma^2)$  PDF. Formalizing this notion, we have mathematically that for a Gaussian random variable  $Y[n]$  with zero mean and variance  $\sigma^2$ , the probability of an outcome being in the interval is

$$P[-3\sigma < Y[n] < 3\sigma] = Q(-3) - Q(3) = 0.9973.$$

More generally, it can be shown that if a random variable  $T$  has the PDF  $N(\mu, \sigma^2)$  then [Kay 2006]

$$P[\mu - k_s\sigma < T < \mu + k_s\sigma] = 1 - 2Q(k_s). \quad (6A.1)$$

For example, if  $k_s = 3$  (our usual value), then  $T$  should be observed to fall in the interval  $[\mu - 3\sigma, \mu + 3\sigma]$  with probability  $1 - 2Q(3) = 0.9973$ . If it does not, then we expect something is wrong with our assumption that  $T \sim N(\mu, \sigma^2)$ . Either  $T$  is not a Gaussian random variable and/or the true values of  $\mu, \sigma^2$  are not those that we assumed. This provides a powerful method of assessing the validity of our assumptions. Since in practice, the mean and variance are seldom known in advance, we replace them by estimates to yield a *confidence interval* of  $[\hat{\mu} - 3\hat{\sigma}, \hat{\mu} + 3\hat{\sigma}]$ . Of course, with estimates of the mean and variances replacing the true values, the confidence interval is only approximate. A typical confidence interval replaces the probability of 0.9973 by 0.95 to produce an approximate 95% *confidence interval* of  $[\hat{\mu} - 1.96\hat{\sigma}, \hat{\mu} + 1.96\hat{\sigma}]$ . Finally, note that  $\hat{\mu}$  is an estimate of  $\mu = E[T]$ , where  $\mu$  is typically the quantity we are attempting to estimate. Since we usually have only a single observed value of  $T$ , we use as our estimate  $\hat{\mu}$ , the value of  $T$ . This produces the 95% confidence interval for the mean of  $T$ , i.e.,  $\mu$ , of

$$[T - 1.96\hat{\sigma}, T + 1.96\hat{\sigma}]. \quad (6A.2)$$

$\hat{\sigma}$  typically will depend on  $T$ , as is the case in the following example.

Consider the problem of estimating the DC level  $A$  for the observed data  $x[n] = A + w[n]$ , where  $w[n]$  is WGN with variance  $\sigma_w^2$ . This problem was described in [Chapter 1](#). There we saw that a good estimator is

$\hat{A} = T = (1/N) \sum_{n=0}^{N-1} x[n]$ , which is the sample mean. Furthermore, it was found that  $\text{var}(T) = \sigma^2 = \sigma_w^2/N$ . Hence, an approximate 95% confidence interval for  $A$  is from [\(6A.2\)](#)

$$\left[ \hat{A} - 1.96 \sqrt{\frac{\widehat{\sigma}_w^2}{N}}, \hat{A} + 1.96 \sqrt{\frac{\widehat{\sigma}_w^2}{N}} \right] \quad (6A.3)$$

where

$$\widehat{\sigma}_w^2 = \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \hat{A})^2$$

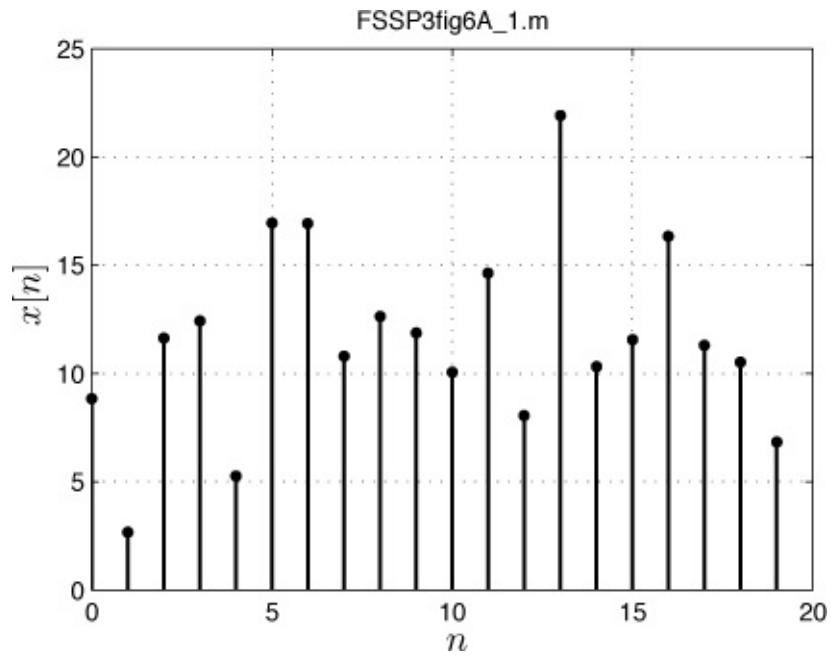
and is also commonly written as

$$\hat{A} \pm \frac{1.96}{\sqrt{N}} \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \hat{A})^2}. \quad (6A.4)$$

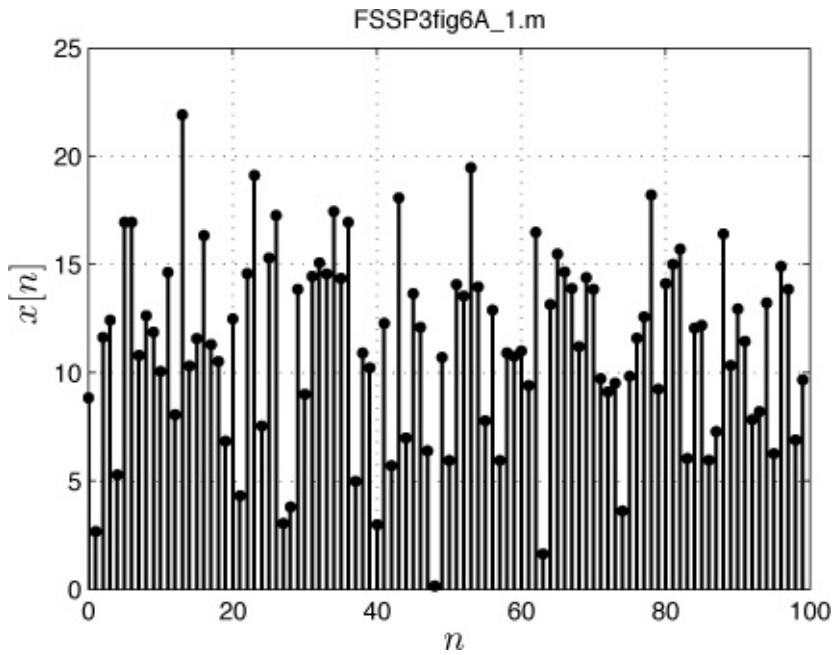
In this case the width of the confidence interval, which is  $2(1.96)\hat{\sigma}$ , depends on  $\hat{A}$ , i.e.,  $T$ .

As a numerical example, consider the data shown in [Figure 6A.1](#). Suppose we think that  $A = 10$ . It is found from the data that

$\hat{A} = T = (1/N) \sum_{n=0}^{N-1} x[n] = 11.58$ . Also, from [\(6A.3\)](#) the approximate 95% confidence interval for  $A$  is found to be  $[9.70, 13.45]$ , which *covers* the postulated value of  $A = 10$ . Hence, this result would prompt us to accept our initial assumption. The discrepancy between our assumed value of  $A = 10$  and the estimated value of  $\hat{A} = 11.58$  would then be attributed to the perturbation due to the noise  $w[n]$ . The only way to be “more certain” is to use a longer data record for which the confidence interval will be shorter in length. Such a data record is shown in [Figure 6A.2](#).



**Figure 6A.1:** Observed data for a DC level in WGN with variance  $\sigma_w^2 = 25$ .



**Figure 6A.2:** Observed data for a DC level in WGN with variance  $\sigma_w^2 = 25$ .  
The data record length is now  $N = 100$ .

### Exercise 6.18 – Accepting or rejecting the assumption that $A = 10$

For the data record shown in [Figure 6A.2](#) and which is given on the CD as

FSSP3exer6\_18.mat, compute the approximate 95% confidence interval using (6A.3). Would you now accept the hypothesis that  $A = 10$ ? You can load in the data by using `load FSSP3exer6_18`.

---

In summary, given an unknown parameter  $\theta$ , which we wish to estimate using  $\hat{\theta}$ , the approximate 95% confidence interval is

$$\hat{\theta} \pm 1.96 \sqrt{\text{var}(\hat{\theta})}. \quad (6A.5)$$


---

### Exercise 6.19 – Confidence interval for kurtosis

We now wish to estimate the kurtosis  $\kappa$ , which is our unknown parameter  $\theta$ , since it will indicate whether our data (that shown in Figure 6.3) can be modeled as Gaussian, for which  $\kappa = 3$ , or not. It can be shown that if (6.6) is used as our estimate, then [Cox and Hinkley 1974]

$$\text{var}(\hat{\kappa}) \approx \frac{24}{N}$$

for large  $N$  (say  $N > 100$ ). Determine the approximate 95% confidence interval for the data shown in Figure 6.3 and contained on the CD as FSSP3exer6\_19.mat as  $y$ . You can load the data by using `load_FSSP3exer6_19`. Would you model the data shown in Figure 6.3 as Gaussian or nonGaussian?

---

## Appendix 6B. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**6.1** Let  $f_0 = 0.06$ . Then,

$$\begin{aligned} H(f_0) &= 1 - 2 \cos(2\pi f_0) \exp(-j2\pi f_0) + \exp(-j4\pi f_0) \\ &= 1 - [\exp(j2\pi f_0) + \exp(-j2\pi f_0)] \exp(-j2\pi f_0) + \exp(-j4\pi f_0) \\ &= 0. \end{aligned}$$

The estimation/extraction approach appears to work better as there is less noise in  $y[n]$ . These results can be obtained by running

FSSP3exer6\_1.m, which is contained on the CD.

**6.2** To determine  $\mathbf{y}$  we have

$$\begin{aligned}\mathbf{y} &= (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)(\mathbf{H}\boldsymbol{\theta} + \mathbf{w}) \\ &= \mathbf{H}\boldsymbol{\theta} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H}\boldsymbol{\theta} + (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)\mathbf{w} \\ &= (\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)\mathbf{w}.\end{aligned}$$

Note that for any value of  $\boldsymbol{\theta}$  (any sinusoidal amplitude and phase) the interference component out of this operation is zero. However, the noise at the output is also altered by the factor  $(\mathbf{I} - \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)$ . The filtering approach of [Exercise 6.1](#) only *approximately* nulls out the interference (due to the zero initial and final conditions of the filter), and affects the noise as well. From your results in [Exercise 6.1](#) you should have observed that the filtered output is noisier.

**6.3** To find the kurtosis for a Gaussian PDF note from [\(6.5\)](#) that

$$\begin{aligned}E[(Y - \mu)^2] &= 1 \cdot \sigma^2 = \sigma^2 \\ E[(Y - \mu)^4] &= 1 \cdot 3\sigma^4 = 3\sigma^4\end{aligned}$$

which yields  $\kappa = 3$ .

**6.4** Using the hint, we have that

$$\begin{aligned}E[Y] &= E\left[\frac{X - \mu}{\sqrt{\sigma^2}}\right] \\ &= E\left[\frac{X}{\sqrt{\sigma^2}} - \frac{\mu}{\sqrt{\sigma^2}}\right] \\ &= \frac{1}{\sqrt{\sigma^2}}\mu - \frac{1}{\sqrt{\sigma^2}}\mu \quad (\text{use hint here for expectation}) \\ &= 0\end{aligned}$$

and

$$\begin{aligned}\text{var}(Y) &= \text{var}\left(\frac{X - \mu}{\sqrt{\sigma^2}}\right) \\ &= \frac{1}{\sigma^2}\text{var}(X) \quad (\text{use hint here for variance}) \\ &= \frac{\sigma^2}{\sigma^2} = 1.\end{aligned}$$

**6.5** The estimated ACS should be one at  $k = 0$  and very close to zero for the

other values. Hence, the results lend credibility to the assumption of a flat PSD. Run `FSSP3exer6_5.m`, which is contained on the CD, to see the results.

**6.6** For a PDF the total area is one since the probability is given by the area under the curve. Hence,  $P[-\infty < X < \infty] = \int_{-\infty}^{\infty} p_X(x)dx = 1$  since the outcome of the random variable must be finite. For a PSD there is no such restriction. Considering a WGN noise process with variance  $\sigma^2$ , the PSD is  $P_x(f) = \sigma^2$  for  $|f| \leq 1/2$  so that the total area is  $\sigma^2$ . This can take on any value, and in particular for  $\sigma^2 = 1$  and  $\sigma^2 = 10$ , the total area is one and ten, respectively.

**6.7** We should use our usual sample mean estimator  $\hat{A} = (1/N) \sum_{n=0}^{N-1} x[n]$ . This is because the mean of each  $x[n]$  is  $E[x[n]] = E[A + w[n]] = E[A] + E[w[n]] = A + 0 = A$ . However, if the mean of  $w[n]$  were  $\mu \neq 0$ , then

$$E[\hat{A}] = \frac{1}{N} \sum_{n=0}^{N-1} E[x[n]] = \frac{1}{N} \sum_{n=0}^{N-1} (A + \mu) = A + \mu$$

and on the average we would obtain the wrong value.

**6.8** By plotting the estimates versus  $N$  you should see convergence at about  $N > 6000$  samples. Of course, the choice of 6000 is somewhat subjective and depends on how you define convergence. The number could have also been chosen as  $N > 2000$ , depending upon how close the estimate appears to be to the true value. To make statements like this more precise requires one to adopt a probabilistic definition of convergence [Kay 2006], which unfortunately, requires more sophisticated mathematical machinery. The results can be obtained by running `FSSPexer6_8.m`, which is contained on the CD.

**6.9** The spread in the vertical direction (fourth-order moment) should be much greater than in the horizontal direction (second-order moment). This indicates the increased variability of the higher order moment. The results can be obtained by running `FSSPexer6_9.m`, which is contained on the CD.

**6.10** For the first part of the problem you should see a scatter diagram that has points nearly uniformly distributed within the square  $[-0.5, 0.5] \times [-0.5, 0.5]$ . There is zero correlation between  $X$  and  $Y$  since they are independent. Thus, the outcome of  $Y$  cannot be predicted even if we know that  $x = 0$  since for  $x = 0$  the values of  $y$  range over the whole

interval  $(-0.5, 0.5)$  in a uniform way. However, for the second part you should see a line  $y = x$  since the random variables are perfectly correlated. Hence, a perfect prediction of  $Y$  is just  $\hat{Y} = x$ . The results can be obtained by running `FSSPexer6_10.m`, which is contained on the CD.

**6.11** By (6.15) for  $N = 2$  we have that

$$\widehat{E[\mathbf{X}]} = \frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2).$$

Since  $\mathbf{x}$  is  $L \times 1$ , we can write  $\mathbf{x}_1$  and  $\mathbf{x}_2$  as

$$\mathbf{x}_1 = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_L^{(1)} \end{bmatrix}$$

where the superscript indicates the number of the outcome of the random vector and similarly for  $\mathbf{x}_2$ . Thus, we have for the sum of the two data vectors using the usual element-wise addition and then element-wise scaling

$$\begin{aligned} \widehat{E[\mathbf{X}]} &= \frac{1}{2} \begin{bmatrix} x_1^{(1)} + x_1^{(2)} \\ x_2^{(1)} + x_2^{(2)} \\ \vdots \\ x_L^{(1)} + x_L^{(2)} \end{bmatrix} \\ &= \begin{bmatrix} (x_1^{(1)} + x_1^{(2)})/2 \\ (x_2^{(1)} + x_2^{(2)})/2 \\ \vdots \\ (x_L^{(1)} + x_L^{(2)})/2 \end{bmatrix} \end{aligned}$$

so that the  $k$ th element, for  $k = 1, 2, \dots, L$ , is

$$[\widehat{E[\mathbf{X}]})_k] = (x_k^{(1)} + x_k^{(2)})/2 = ([\mathbf{x}_1]_k + [\mathbf{x}_2]_k)/2$$

and the more general result follows similarly.

**6.12** You should obtain the mean vector and covariance matrix estimates of

$$\widehat{E[\mathbf{X}]} = \begin{bmatrix} 1.0023 \\ 0.9666 \end{bmatrix} \quad \hat{\mathbf{C}} = \begin{bmatrix} 0.7679 & 0.6928 \\ 0.6928 & 0.7912 \end{bmatrix}.$$

Note that the mean vector estimate is very near the true mean but the covariance matrix estimate is very inaccurate. This is another manifestation of the difficulty in estimating higher order moments. For a good estimate of the covariance matrix you will need at least  $N = 1000$ . Run the program `FSSP3exer6_12.m`, which is contained on the CD, for these results.

- 6.13** If you use the MATLAB program `Q.m` to find the area under the  $N(0, 1)$  PDF curve as  $P[a < X < b] = Q(b) - Q(a)$ , then you should obtain the results shown in [Table 6B.1](#). The most accurate value (*on the average*) is obtained for the narrowest bin, i.e., the narrowest bin produces an estimate with the least amount of bias. To obtain these results run the MATLAB program `FSSP3exer6_13.m`, which is contained on the CD.

**Table 6B.1: Expected value of estimated PDF at  $x = 0$  for differing bin widths for a Gaussian PDF with mean zero and variance one.**

$\Delta x$	$E[\hat{p}_X(0)]$
1	0.3829
0.5	0.3948
0.25	0.3979
0.125	0.3987
0.0625	0.3989
0.03125	0.3989

- 6.14** You should observe that the histogram estimate with more and hence narrower bins gives a better estimate for the PDF where there is more mass (at  $x = 0$ ), i.e., better resolution, but a poorer estimate in the “tails” where there is little mass and hence a low probability of having outcomes lie within these bins. Note the increased number of bins with zero outcomes. You can run the program `FSSP3exer6_14.m`, which is contained on the CD, to obtain these results.

- 6.15** For  $p = 2$  you should observe a *smoothed* PDF estimate, while for  $p = 15$  the estimate will exhibit many peaks or will be too *noisy*. Clearly, the

appropriate order is somewhere in between these values. You can run the program `FSSP3exer6_15.m`, which is contained on the CD, to obtain the results.

- 6.16** The estimate does not improve. In fact it appears to become noisier! To obtain these results run the MATLAB program `FSSP3exer6_16.m`, which is contained on the CD.
- 6.17** You should notice that the variability of the spectral estimate decreases as more blocks ( $L$  becomes smaller) are used. However, the estimate also becomes more “smeared” since there is a loss of resolution due to the smaller block size. To obtain these results run the MATLAB program `FSSP3exer6_17.m`, which is contained on the CD.
- 6.18** You should obtain the estimate of  $\hat{A} = 11.23$  and a 95% confidence interval of  $[10.39, 12.08]$ . Since the confidence interval does not cover the hypothesized value of  $A = 10$ , we would reject the hypothesis that  $A = 10$ . The actual value of  $A$  used to produce the data in [Figures 6A.2](#) and [6A.3](#) was  $A = 11$ . You can run the program `FSSP3exer6_18.m`, which is contained on the CD, to obtain the results.
- 6.19** You should obtain the confidence interval of  $[6.10, 7.06]$ . Since it does not cover the value of  $\kappa = 3$ , which is the theoretical value for a Gaussian PDF, we would decide the data in [Figure 6.3](#) is nonGaussian. The actual PDF used to generate the noise in [Figure 6.2](#) was a Gaussian mixture PDF, which is nonGaussian (see [\(4.19\)](#)). You can run the program `FSSP3exer6_19.m`, which is contained on the CD, to obtain the results.

# Chapter 7. Performance Evaluation, Testing, and Documentation

## 7.1. Introduction

In [Chapter 2](#) we presented a general methodology for algorithm development and testing. [Chapters 3](#) through [6](#) described mathematical models that are used to either derive (to be explained more fully in [Chapter 8](#)) or to motivate an algorithm. In this chapter we assume that an algorithm has been proposed and therefore, wish to determine its performance. This requires a choice of a performance criterion and subsequent testing to determine the performance with respect to the chosen criterion. Usually, the performance is assessed by encoding the algorithm in computer software, which can then be tested with computer-generated data. The final step is testing in the field. We explore only the former, with the latter left to the practitioner. Hence, referring to [Figure 2.1](#) we now describe Step 9 – *performance analysis*, Step 10 – *specification assessment*, and Step 11 – *sensitivity analysis*. The importance of a realistic and accurate performance evaluation cannot be overemphasized. *The difference between a working algorithm and one that is discarded balances precariously on this step!* Lastly, we address the vital but often overlooked task of algorithm documentation. If others are to understand and implement the algorithm successfully, then it is essential that this task be addressed.

## 7.2. Why Use a Computer Simulation Evaluation?

It is commonplace to evaluate the performance of an algorithm via a Monte Carlo computer simulation. We have already used this approach on numerous occasions, with examples given in [Figures 2.5](#) and [2.6](#). It is usually the preferred approach since modern statistical signal processing algorithms can be quite complex and therefore, defy an analytical performance evaluation. Furthermore, using computer simulated data as the input to an algorithm that is implemented on a digital computer allows the practitioner to:

1. Utilize a meaningful statistical performance criterion, often succinctly referred to as a *performance metric*. This metric may be too difficult to work with analytically, and so a computer simulation is required.
2. Obtain exact results as opposed to asymptotic ones. Asymptotic results are only valid for large data record lengths and/or high signal-to-noise ratios

(SNRs), and are the usual output of an analytical evaluation.

3. Quickly assess performance and study algorithm robustness to the variation of the design parameters.
  4. Easily compare algorithm performance against any competitors, assuming competitive algorithm software implementations are available.
  5. Foster a more complete understanding of the algorithm by having to “build” it in software.
  6. Assess the computational complexity of the algorithm and therefore its suitability for real-time operation, if desired, or more generally to determine computer sizing required for implementation.
  7. Have the algorithm validated by others by sharing the software implementation.
- 



### **Always use a computer simulation first to determine performance.**

For all the reasons listed above it is not only advantageous but also *essential* to perform a computer simulation of the algorithm’s performance. Sadly, experience tells the story that the most carefully designed algorithm will never perform as well in the field as under the carefully controlled conditions available within a digital computer. The performance obtained via a computer simulation should therefore be considered the best possible performance that the algorithm will produce. If the algorithm does not perform as desired when implemented and tested via a computer simulation, then it will never do so in the field!



---

We next discuss in more detail some of the important issues in performance evaluation.

### **7.3. Statistically Meaningful Performance Metrics**

One cannot determine if a pair of dice have been “loaded” by tossing them once. However, if the dice are tossed 1000 times and “snake-eyes”, which is an outcome of 1 and 1, comes up 52 times, we may conclude with reasonable certainty that the dice are loaded (since the probability of snake-eyes is only 1/36, so we would expect about 28 of these outcomes). Statistical signal

processing algorithms operate on data that is inherently random, and so a *statistical measure of performance*, i.e., a *statistical metric*, is required. The typical ones are given in [Table 7.1](#), but others are in common use, depending upon the problem of interest. All the metrics involve expectations or probabilities. In either case *they measure performance “in the long run” or as the experiment is repeated many times under identical conditions.*

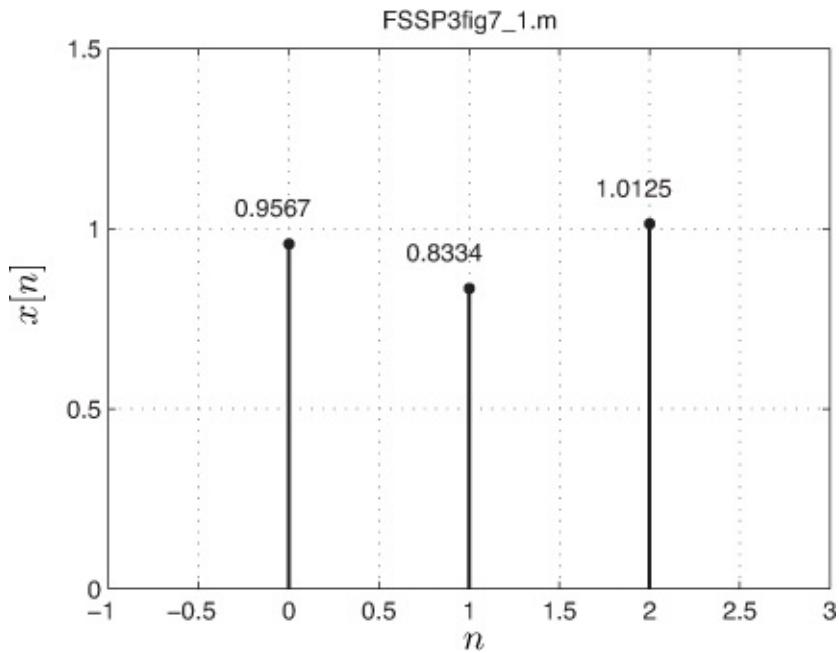
**Table 7.1: Typical statistical signal processing performance metrics. ( $P_D$  = probability of detection,  $P_{FA}$  = probability of false alarm, ROC = receiver operating characteristics.)**

Problem	Performance Metric
Parameter estimation	bias/variance or mean square error
Signal detection	$P_{FA}/P_D$ (ROC) or $P_D$ versus SNR ( $P_{FA}$ fixed)
Classification	probability of error $P_e$ or probability of correct classification $P_c = 1 - P_e$

---

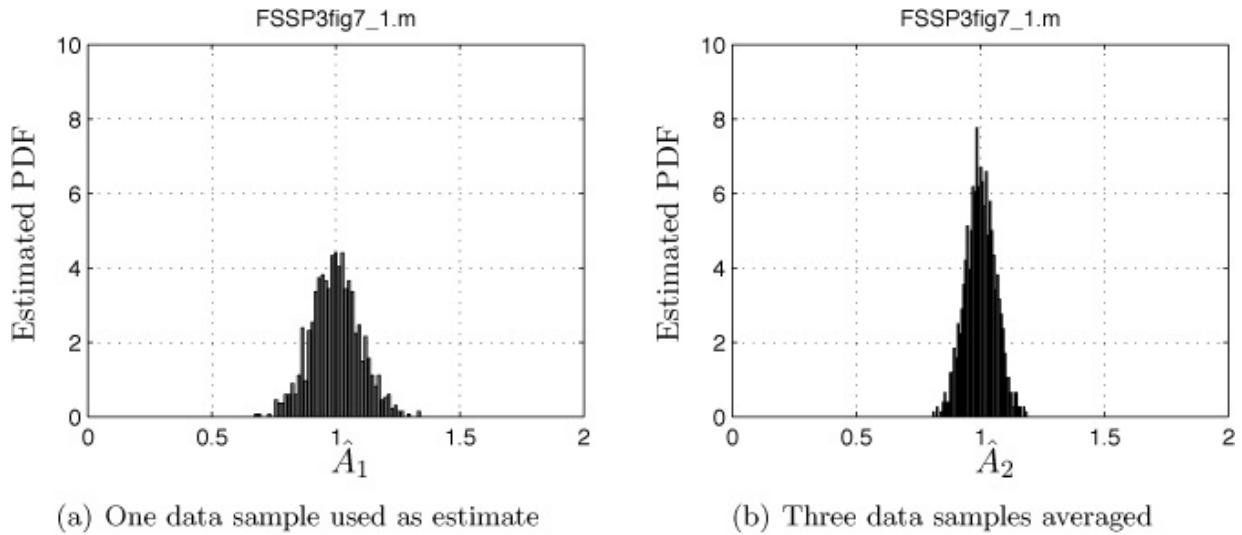
### Example 7.1 – Importance of “in the long run” performance

A DC level  $A$  is embedded in WGN  $w[n]$  to form the data set  $x[n] = A + w[n]$  for  $n = 0, 1, 2$ . It is desired to estimate the DC level from the data. Two estimators are proposed. They are  $\hat{A}_1 = x[0]$  and  $\hat{A}_2 = (x[0] + x[1] + x[2])/3$ . The true value of  $A$  is 1. A single data set is shown in [Figure 7.1](#).



**Figure 7.1: Single data set.**

The estimates for this experimental outcome are  $\hat{A}_1 = 0.9567$  and  $\hat{A}_2 = (0.9567 + 0.8334 + 1.0125)/3 = 0.9342$ . Hence, it might appear that the first estimator is better since its value is closer to the true value of  $A = 1$ . However, a second outcome of  $x[0], x[1], x[2]$  might indicate just the opposite result, i.e., that  $\hat{A}_2$  is closer to one than  $\hat{A}_1$ . The only way to assess the long run performance of the two estimators is to determine their probability density functions (PDFs). Repeating the entire experiment 1000 times, calculating the 1000 estimates,  $\hat{A}_1$  and  $\hat{A}_2$ , and estimating the PDF (see [Section 6.4.6](#)), produces the PDF estimates shown in [Figure 7.2](#). It is apparent from observing the estimated PDFs that both estimators have a mean at the true value of  $A = 1$ , but the second estimator has a lower variance and so is preferred. Note that this conclusion was only possible *by repeating the experiment many times*.



**Figure 7.2: Estimated PDF for the two DC level estimators.**



### Exercise 7.1 – Another DC level estimator

A DC level in WGN consists of IID samples with each sample having the PDF  $x[n] \sim N(A, \sigma^2)$ . As such the *median*, which is the value of  $x[n]$  at which the probability of being less than this value is 0.5, is given by  $A$ . Someone proposes that a better estimator of  $A$  is to use the observed median. As an example, for the data set shown in [Figure 7.1](#) the three values when arranged in ascending order are  $\{0.8334, 0.9567, 1.0125\}$ , and so the observed median *estimate* is given by the value in the middle or  $\hat{A}_{\text{med}} = 0.9567$ . Determine whether this estimator is better than  $\hat{A}_2$ . You can use `pdf_hist_est.m` to obtain the estimated PDF and use `load FSSP3exer7_1` to load in the 1000 data sets that were used to produce [Figure 7.2](#). You will obtain a  $3 \times 1000$  array called  $x$ . Hint: You may want to compute the estimated variance for both  $\hat{A}_2$  and  $\hat{A}_{\text{med}}$  for a numerical comparison of the performance (see [Section 6.4](#) for how this is done).



The performance metrics listed in [Table 7.1](#) are the standard ones used in signal processing. We next define and illustrate each one.

#### 7.3.1. Parameter Estimation Performance Metrics

As you should have seen in the previous exercise, visual comparison of estimated PDFs is sometimes difficult. A simpler metric to work with in practice employs just two estimator properties, namely its bias and its variance. The *bias* is defined as the difference “on the average” between the estimate and the true value. It is given by

$$b(\theta) = E[\hat{\theta}] - \theta \quad -\infty < \theta < \infty$$

and may depend on the true value of the unknown parameter. Clearly, we would like the bias to be zero. However, zero bias only assures us that the estimate will yield the correct value “on the average”. Any individual experiment may produce an estimate that is far from the true value. To make sure that *each* experiment will produce a value close to the true value, we also employ the *variance* metric. The variance is defined as the usual variance of the *random variable*  $\hat{\theta}$ , and is denoted by  $\text{var}(\hat{\theta})$ . It should ideally be small so that the PDF is concentrated about its mean (compare [Figure 7.2a](#) to [Figure 7.2b](#)), which, if the bias is zero, will be the true value. It is sometimes possible to determine these metrics analytically, but most often in practice, it is done via a computer simulation, just as was done in [Exercise 7.1](#) for the variance.

It is also possible to combine these two metrics into a single definitive metric called the *mean squared error* (MSE). It is defined as

$$\text{mse}(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] \tag{7.1}$$

and measures the average squared deviation from the true parameter value. It can be shown to be related to the bias and variance as [[Kay 1993](#), pg. 19]

$$\text{mse}(\hat{\theta}) = b^2(\theta) + \text{var}(\hat{\theta}). \tag{7.2}$$

Most good estimators are unbiased (meaning that  $b(\theta) = 0$ ) so that the variance is then equivalent to the MSE. If one is not sure if the estimator is unbiased, then the MSE should be the metric of choice.

---

### Exercise 7.2 – Bias and variance of DC level estimators

Prove that both  $\hat{A}_1$  and  $\hat{A}_2$  in [Example 7.1](#) are unbiased estimators. Next compute their variance. Does this help explain the results seen in [Figure 7.2](#)? Hint: Show that  $E[\hat{A}_1] = A$  and  $E[\hat{A}_2] = A$  for all  $A$  for the first part of the exercise. Also, recall that  $\text{var}(aX) = a^2 \text{var}(X)$  for  $a$  constant, and  $\text{var}(X + Y) = \text{var}(X) + \text{var}(Y)$  for  $X$  and  $Y$  uncorrelated random variables.



As usual, in order to determine these metrics, we resort to a computer simulation to produce many outcomes of the estimator and then estimate the metrics. The estimate of any metric is produced by replacing the definition by its sample mean ( $E[\cdot]$ ) is replaced by  $(1/N) \sum_{i=1}^N (\cdot)$ ). As an example, if we observe  $N$  outcomes of the estimator  $\hat{\theta}$ , which we denote by  $\{\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_N\}$ , then in the case of variance we use the estimate

$$\widehat{\text{var}}(\hat{\theta}) = \frac{1}{N} \sum_{i=1}^N \left[ \hat{\theta}_i - \left( \frac{1}{N} \sum_{j=1}^N \hat{\theta}_j \right) \right]^2$$

(see also [Section 6.4](#) for a further discussion).

---

### Exercise 7.3 – Evaluation of parameter estimators

Using a computer simulation compare the sample mean estimator  $\hat{A} = \bar{x} = (1/N) \sum_{n=0}^{N-1} x[n]$  and the median estimator  $\hat{A}_{\text{med}}$  for a DC level with  $A = 1$  in WGN. Assume that the WGN variance is  $\sigma^2 = 1$  and the data record length is  $N = 11$ . Compute the bias, variance, and MSE for both estimators.

#### 7.3.2. Detection Performance Metrics

There are two common methods for evaluation of detectors. They are the receiver operating characteristic (ROC), and the detection probability versus signal-to-noise ratio (SNR) with the false alarm probability fixed. To describe these metrics consider the problem of detecting a DC level  $A > 0$  in WGN of variance  $\sigma^2$ , based on observing the data samples  $x[n] = A + w[n]$  for  $n = 0, 1, \dots, N - 1$ . The optimal Neyman-Pearson detector decides that a signal is present if (see [Section 8.2.2](#) and [\[Kay 1998, pg. 96\]](#))

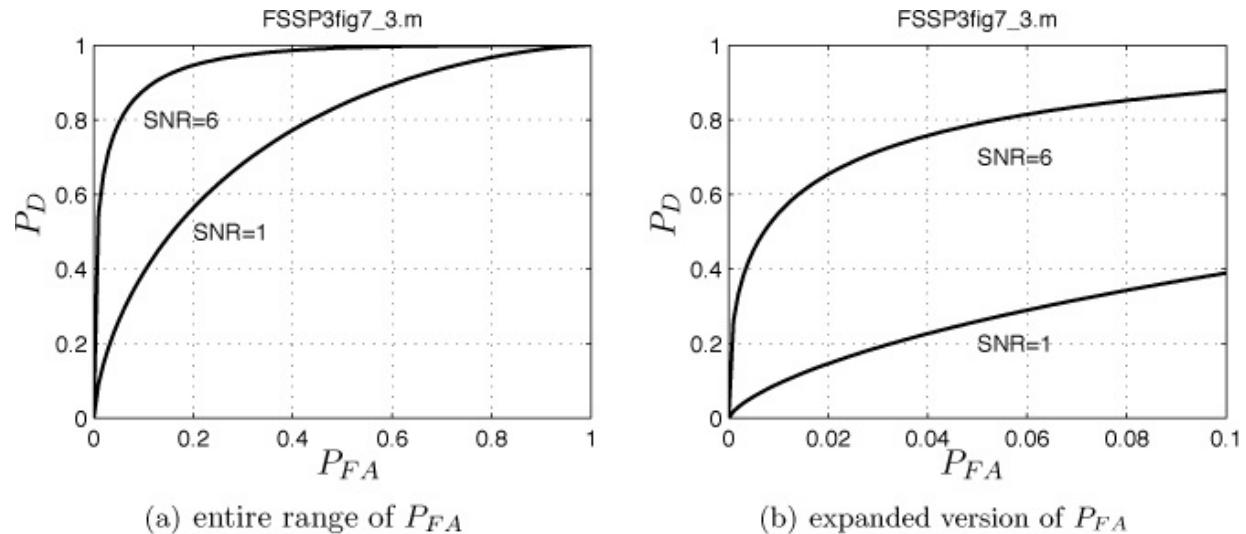
$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \gamma \tag{7.3}$$

where  $\gamma$  is called the threshold. We define the probability of detection  $P_D$  as the probability that  $\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \gamma$  when *there is a signal present*, and the probability of false alarm  $P_{FA}$  as the probability that  $\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \gamma$  when *there is only noise present*. In either case, because of the decision rule, we are

declaring a signal present because the statistic, i.e., the sample mean, has exceeded the threshold. The performance can be found analytically and is given by

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{NA^2}{\sigma^2}} \right) \quad (7.4)$$

where the functions  $Q(\cdot)$  and  $Q^{-1}(\cdot)$  have been defined in [Section 4.3](#). (The MATLAB subprograms `Q.m` and `Qinv.m`, which are contained on the CD, can be used to evaluate  $P_D$ ). The ROC, which plots  $P_D$  versus  $P_{FA}$  with the  $\text{SNR} = N A^2 / \sigma^2$  (actually an energy-to-noise ratio) fixed, is shown in [Figure 7.3a](#) for different values of SNR. Note that the curve must be concave and above the  $45^\circ$  line. Otherwise, it can be shown that a better detector can be based on the outcome of the toss of a coin! The value of  $(P_{FA}, P_D)$  is sometimes called the *operating point*. Although not obvious from examination of (7.4), for a given SNR, the operating point can be changed by adjusting the threshold  $\gamma$ . As the threshold increases, the operating point moves along the ROC from the point  $(1, 1)$  for  $\gamma = -\infty$  to  $(0, 0)$  for  $\gamma = \infty$ . In light of the decision rule given in (7.3) the reader may wish to ponder why this is so.



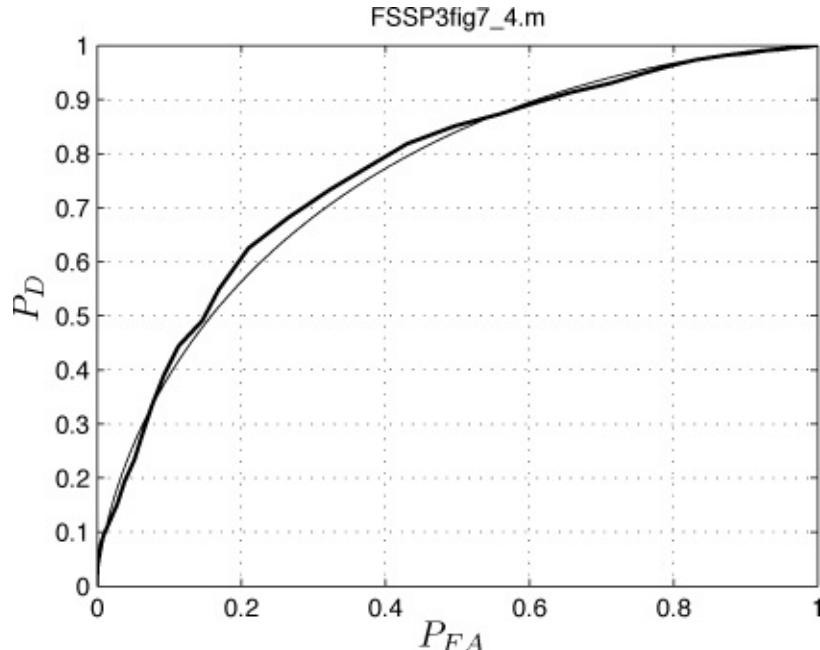
**Figure 7.3: Receiver operating characteristics for a DC level in WGN.**

A good detector should have a large  $P_D$  even for a very small  $P_{FA}$ . Thus, it is sometimes advantageous to expand the plot so that the performance at low  $P_{FA}$ 's is more apparent, as shown in [Figure 7.3b](#). In practice, analytical determination of  $P_{FA}$  and  $P_D$  can be difficult and so the ROC is estimated using a computer simulation. Estimating probabilities is straightforward but may be time

consuming. The procedure for generating an ROC is as follows:

1. Generate a large number of data sets consisting of noise only, i.e.,  $x[n] = w[n]$  for  $n = 0, 1, \dots, N - 1$ . This produces, say for  $N = 4$  and 1000 data sets: data set 1 = {1.1, -3.5, 5.0, -3.4}, data set 2 = {0.3, 0.8, -2.4, 1.1}, ..., data set 1000 = {0.3, -0.1, 3.1, 2.7}, as an example.
2. Compute the sample mean given by (7.3) for each data set, and which is denoted by  $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{1000}$ .
3. Choose a set of equally spaced values for  $\gamma$  over the range  $\bar{x}_{\min} \leq \gamma \leq \bar{x}_{\max}$ .
4. For each value of  $\gamma$  count the number of times  $\bar{x}$  exceeds  $\gamma$  and divide this by 1000. This then yields the estimate of  $P_{FA}$  or more explicitly denoted by  $P_{FA}(\gamma)$  to indicate its dependence on the threshold.
5. Repeat all the steps 1–4 but replace the data by  $x[n] = A + w[n]$ . Use the same set of  $\gamma$ 's as in Step 4. This then yields the estimate of  $P_D$ , or more explicitly denoted by  $P_D(\gamma)$  to indicate its dependence on the threshold.
6. For each  $\gamma$  plot the pair  $(P_{FA}(\gamma), P_D(\gamma))$ .

As an example, for an SNR of 1, the estimated ROC using 1000 data sets is shown in [Figure 7.4a](#), along with the theoretical ROC previously given in [Figure 7.3a](#) for SNR = 1. The MATLAB program `roccurve.m`, which is contained on the CD, has been used to obtain the  $(P_{FA}, P_D)$  pairs from the sample mean detector statistics for the two cases of noise only and a signal plus noise.



**Figure 7.4: Estimated receiver operating characteristics for a DC level in WGN. The computer simulation estimate is shown as the heavy line, and the theoretical curve given by (7.4) is shown as the light line.**

---

### Exercise 7.4 – Generating an ROC

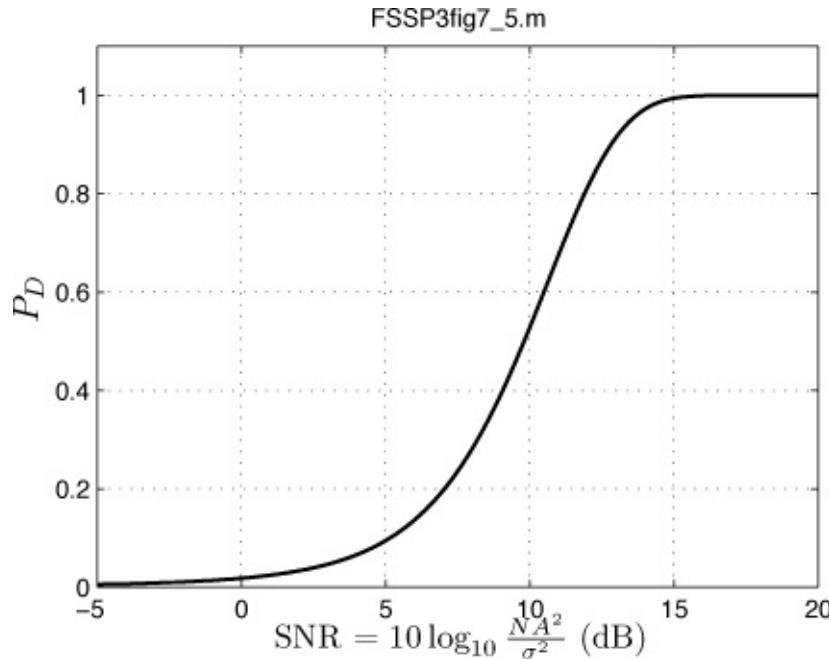
For the DC level in WGN consider the following two sets of conditions:

$A = 1/\sqrt{10}$ ,  $N = 10$ ,  $\sigma^2 = 1$  and  $A = 0.1$ ,  $N = 100$ ,  $\sigma^2 = 1$ . Generate 1000 data sets for the noise only condition and another 1000 sets for the signal plus noise condition for each of the two different sets of signal parameters. Then use `roccurve.m` to obtain the  $(P_{FA}, P_D)$  pairs and finally plot the ROCs. What do you notice?

•

---

The alternative method used to describe detection performance is to plot the probability of detection versus SNR. From (7.4) it is seen that if we fix  $P_{FA}$ , then  $P_D$  depends only on the SNR. This type of plot is used heavily in practice since it indicates how much SNR is necessary to meet the specification of a given probability of detection for a fixed probability of false alarm. As an example, if we wish to maintain a  $P_{FA}$  of  $10^{-3}$ , then using (7.4) with  $P_{FA} = 10^{-3}$  and plotting  $P_D$  versus SNR, we obtain [Figure 7.5](#). Note that for near perfect detection we require an SNR of about 15 dB. For lower  $P_{FA}$ 's the curve shifts to the right so that we will need a higher SNR.



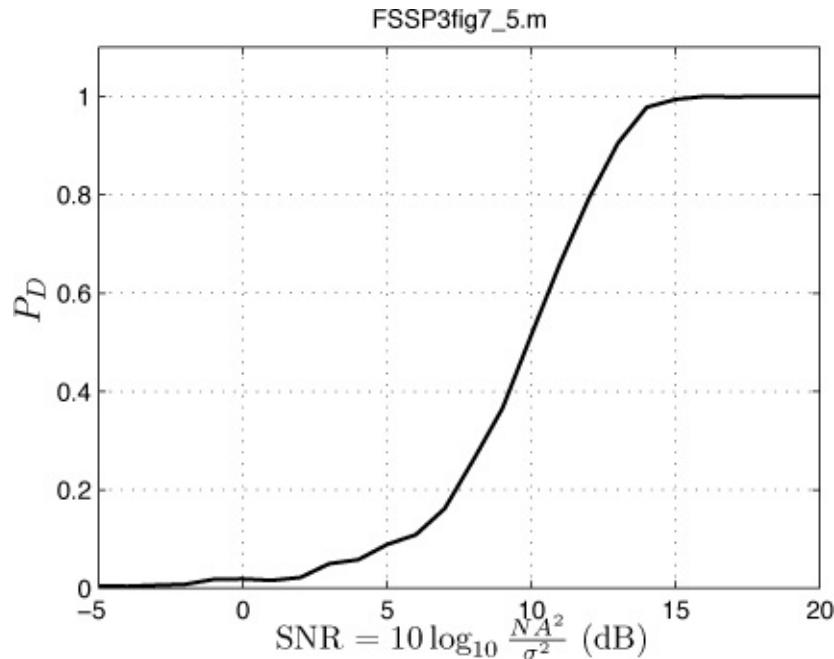
**Figure 7.5: Theoretical probability of detection versus SNR in dB quantities.**

If we cannot derive the theoretical performance, and indeed this task can be quite difficult, then we can again resort to a computer simulation. As an example, assume that  $N = 10$ ,  $\sigma^2 = 5$  and  $A$  varies to obtain a range of SNRs according to  $\text{SNR} = NA^2/\sigma^2$ .  $P_{FA}$  is to be fixed at  $10^{-3}$ . (A cautionary note is that detection performance will often depend upon the signal parameters in a different fashion than the simple dependence on  $NA^2/\sigma^2$  as given in (7.4). Thus, we might wish to plot  $P_D$  versus  $A^2/\sigma^2$  with  $N$  as a parameter.) The steps to obtain  $P_D$  versus SNR are as follows:

1. Generate a large number, say 100,000, outcomes of the detection statistic assuming noise only. In this case we generate outcomes of  $T_0(\mathbf{x}) = (1/N) \sum_{n=0}^{N-1} w[n]$ , where  $w[n]$  is WGN with variance  $\sigma^2 = 5$ .
2. For the given  $P_{FA} = 10^{-3}$  determine the threshold  $\gamma$  so that only 100 of the 100,000 outcomes of  $T_0(\mathbf{x})$  exceed  $\gamma$ .
3. Generate a large number, say 1000, outcomes of the detection statistic assuming signal plus noise. In this case we generate outcomes of  $T_1(\mathbf{x}) = (1/N) \sum_{n=0}^{N-1} (A + w[n])$ , where  $w[n]$  is WGN with variance  $\sigma^2 = 5$ . The amplitude of the DC level  $A$  is chosen to yield the SNR desired.

4. Count the number of  $T_1(\mathbf{x})$ 's that exceed the threshold and divide by 1000. This yields an estimate of  $P_D$  for the given SNR.
5. Repeat steps 3 and 4 for all desired SNRs.

Note that the number of outcomes of  $x[n] = w[n]$  is usually required to be much larger for the noise only case since the  $P_{FA}$  that we are trying to estimate typically is very small, in this case, 0.001. We should therefore choose the number of outcomes so that there is a high probability of many threshold exceedances. Consider what might happen if  $P_{FA} = 0.001$  but we generated only 1000 outcomes of  $T_0(\mathbf{x})$ . An example of an estimated detection curve for  $N = 10$  and  $\sigma^2 = 5$  is shown in [Figure 7.6](#) and compares favorably with the theoretical curve of [Figure 7.5](#).



**Figure 7.6: Estimated probability of detection versus SNR in dB quantities.**

### Exercise 7.5 – Generating a $P_D$ versus SNR curve

Repeat the previous example to produce a curve similar to that shown in [Figure 7.6](#). Now, however, let  $P_{FA} = 10^{-4}$ . You will need to run many experiments for the noise only case! Do the results match the theoretical prediction of (7.4)? Hint: You may want to get a cup of coffee while you are waiting!



---

### 7.3.3. Classification Performance Metrics

Classification, also known as pattern recognition and discrimination, is a slightly more general version of the hypothesis testing problem encountered in signal detection. There may be more than two hypotheses and typically each hypothesis concerns the presence of a different signal. A typical application is in deciding which digit of the possible 10 digits has been written on a postal envelope.

Another important application is in digital communications in which we wish to decide whether a bit that was sent is either a “0” or a “1”. Consider this latter application in which a positive DC level  $A$  is transmitted to communicate a one bit and a negative DC level  $-A$  is transmitted to communicate a zero bit. Also, as is typical, the two possible bits are assumed to be equally likely in occurring. Hence, for a receiver that senses the voltage level but is subject to WGN the optimal decision rule is to choose whichever bit is more likely [[Kay 1998](#), pg. 112]. This translates into a decision rule that decides bit 1 if

$(1/N) \sum_{n=0}^{N-1} x[n] \geq 0$  and bit 0 if  $(1/N) \sum_{n=0}^{N-1} x[n] < 0$ , assuming that  $N$  samples are received. The usual performance metric is the probability of error  $P_e$ . It is evaluated using

$$P_e = P[\text{decide } 1 | 0 \text{ sent}]P[0 \text{ sent}] + P[\text{decide } 0 | 1 \text{ sent}]P[1 \text{ sent}]$$

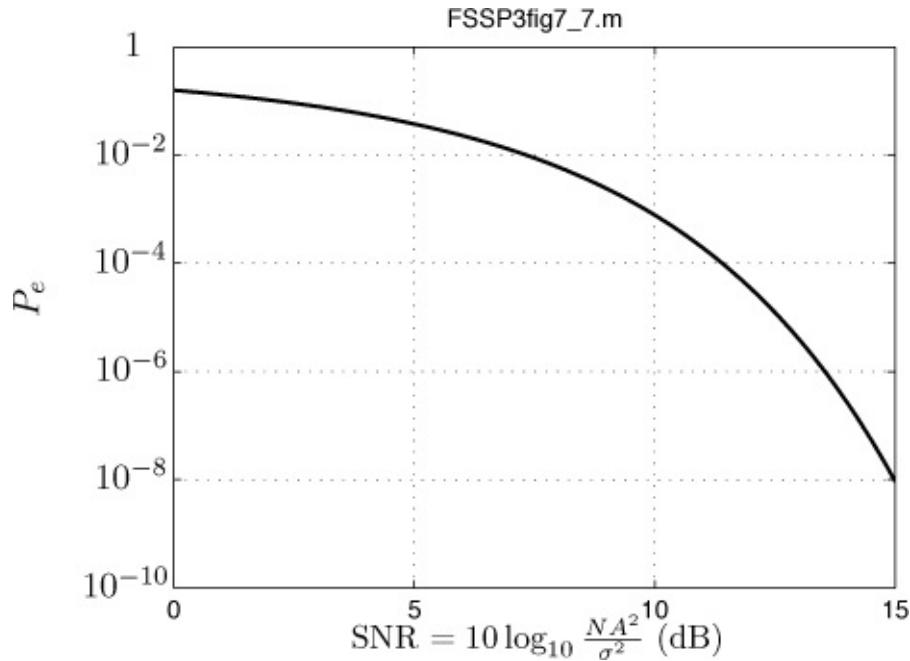
where  $P[C|D]$  denotes the probability of event  $C$  occurring when it is known that event  $D$  has occurred. Because we assume that the bits are equally likely to be transmitted, and using the foregoing decision rule, the  $P_e$  becomes

$$P_e = P\left[\frac{1}{N} \sum_{n=0}^{N-1} x[n] \geq 0 \middle| 0 \text{ sent}\right] \frac{1}{2} + P\left[\frac{1}{N} \sum_{n=0}^{N-1} x[n] < 0 \middle| 1 \text{ sent}\right] \frac{1}{2}.$$

This can be shown to be given by the theoretical expression [[Kay 1998](#), pg. 117]

$$P_e = Q\left(\sqrt{\frac{NA^2}{\sigma^2}}\right) \quad (7.5)$$

and is seen to depend only on the SNR (actually the energy-to-noise ratio). The probability of error versus SNR in dB is plotted in [Figure 7.7](#).



**Figure 7.7: Theoretical probability of error versus SNR in dB quantities.**

Note that typically very low  $P_e$ 's are desired, on the order of  $10^{-7}$ . Once again in the absence of an analytical expression for  $P_e$  such as (7.5) it is always possible to determine the performance using a computer simulation. Note, however, that for very low  $P_e$ 's this will require more sophisticated techniques of Monte Carlo simulation than used here. The interested reader should consult [Rubinstein 1981].

To determine the performance via a computer simulation the procedure is to:

1. Generate either a 0 or 1 with equal probability and then construct the signal as either  $s[n] = -A$  or  $s[n] = A$  for  $n = 0, 1, \dots, N - 1$ , respectively.  
The MATLAB code segment

```
bit=floor(rand(1,1)+0.5);
s=A*(2*bit-1)*ones(N,1);
```

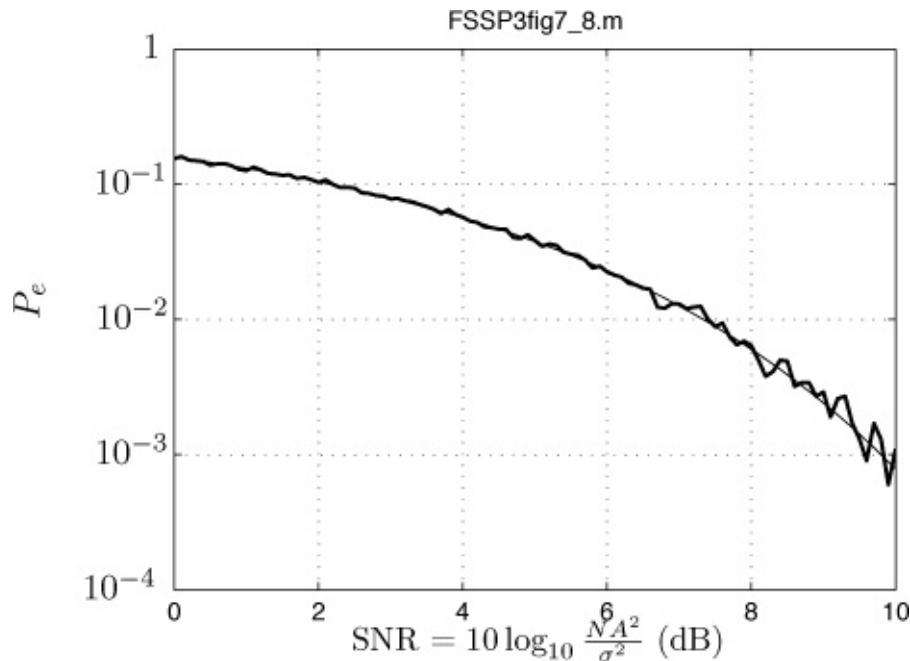
can be used to do this. Repeat this procedure to generate a large number, say 10,000 signals (each one is  $N$  samples in length).

2. For each  $s[n]$  add WGN to generate 10,000 noisy data sets, with each one being  $x[n] = A + w[n]$  for a 1 or  $x[n] = -A + w[n]$  for a 0.
3. Decide a bit 1 if  $(1/N) \sum_{n=0}^{N-1} x[n] \geq 0$  and a 0 otherwise.
4. Count the number of bits that were decoded incorrectly and divide by 10,000.

This will yield the estimate of  $P_e$  for the given value of  $A$  chosen.

5. Repeat steps 1–4 for all SNRs of interest by changing the value of  $A$ , i.e., let  $A = \sqrt{\text{SNR}(\sigma^2/N)}$  (SNR is given in linear quantities).

As an example, let  $N = 10$  and  $\sigma^2 = 1$ , and adjust  $A$  to yield SNRs from 0 dB to 10 dB. Then, a MATLAB simulation produces the  $P_e$  versus SNR in dB curve shown as the dark line in [Figure 7.8](#). Also, the theoretical performance prediction as given by (7.5) is shown as the light line. Note that we have only performed the simulation for a maximum SNR of 10 dB since we have used 10,000 bits. For higher SNRs and therefore lower  $P_e$ 's, the number of bits would need to be much larger. Also, the effect of fewer bit errors for an SNR of 10 dB is manifested in [Figure 7.8](#) as an estimate of  $P_e$  that is less accurate, hence, the noticeable departure from the theoretical curve.



**Figure 7.8: Estimated probability of error versus SNR in dB quantities. Estimated  $P_e$  is shown as the dark line and the theoretical  $P_e$  is shown as the light line.**

### Exercise 7.6 – Estimating $P_e$

A phase-shift keyed (PSK) digital communication system transmits either

the signal  $s_1[n] = A \cos(2\pi(0.25)n)$  for a 1 bit or  $s_0[n] = A \cos(2\pi(0.25)n + \pi)$  for a 0 bit. At the receiver the data is  $x[n] = s_i[n] + w[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $w[n]$  is WGN with variance  $\sigma^2$ . An optimal receiver can be shown to decide a 1 if (for the bits being equally probable)

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] \cos(2\pi(0.25)n) \geq 0$$

and a 0 otherwise. (See also [Section 8.2.3](#) for the maximum likelihood (ML) rule and [[Kay 1998](#), pg. 117].) The theoretical  $P_e$  is given by (for  $N$  even)

$$P_e = Q\left(\sqrt{\frac{NA^2}{2\sigma^2}}\right). \quad (7.6)$$

If  $N = 10$  and  $\sigma^2 = 1$ , use a computer simulation with 100,000 trials to estimate  $P_e$  for SNR = 5 dB, where the SNR is now defined as

$$\text{SNR} = 10 \log_{10} \frac{NA^2}{2\sigma^2}.$$

Compare your results to the theoretical prediction of (7.6).



## 7.4. Performance Bounds

Bounds on the performance of an algorithm are valuable in that (see also the discussion in Feasibility Study of [Section 2.2](#)):

1. They establish feasibility of the specifications for algorithm performance.
2. They can measure the loss in performance of the algorithm from the theoretical upper limit. If the algorithm nearly attains the upper bound, then there is little motivation to look for a better performing algorithm.
3. They provide a “sanity check” on the results from a computer or analytical performance evaluation. Exceeding the bound indicates an analytical error or an invalid assumption in the algorithm development or a logical/coding error in the software implementation of the algorithm.

### Example 7.2 – A suboptimal detector

Consider the detection of a DC level in WGN. Then, when a signal is

present, we observe  $x[n] = A + w[n]$ , where  $A > 0$  and  $w[n]$  is WGN with variance  $\sigma^2$ . When noise only is present, we observe  $x[n] = w[n]$ . It is well known (see [Section 8.3](#)) that the optimal detector in terms of maximizing the probability of detection  $P_D$  for a fixed probability of false alarm  $P_{FA}$  is to decide the signal is present if the condition  $(1/N) \sum_{n=0}^{N-1} x[n] \geq \gamma$  is satisfied. If  $P_{FA} = 1/2$ , then it can be shown that we should choose  $\gamma = 0$ . The performance of this detector (called the Neyman-Pearson detector) is given by [\(7.4\)](#) and is repeated here for convenience

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{NA^2}{\sigma^2}} \right).$$

This performance is the best that can be achieved, i.e., is an upper bound, for the stated problem. Hence, as an example, if  $P_{FA} = 1/2$ ,  $N = 100$ ,  $A = 0.1$ , and  $\sigma^2 = 1$ , it is given by  $P_D = Q(0 - 1) = 0.8413$ .

An alternative detector is proposed that averages the cube roots of the data samples. This detector decides a signal is present if  $(1/N) \sum_{n=0}^{N-1} (x[n])^{1/3} \geq 0$ . It can be shown to also produce  $P_{FA} = 1/2$ . To find its detection performance we use the code

[Click here to view code image](#)

```
clear all % clear out all variables
randn('state',0) % initialize random number generator
count=0; % initialize threshold exceedance counter
for i=1:10000
    x=0.1+randn(100,1); % generate data set
    for j=1:100 % take the cube root
        if x(j)>=0
            y(j,1)=x(j)^(1/3);
        else
            y(j,1)=-((-x(j))^(1/3)); % this avoids the complex cube root
        end
    end
    T=mean(y); % compute detection statistic
    if T>=0 % determine if threshold exceeded
        count=count+1;
    end
end
P_Dest=count/10000 % estimate of P_D
```

When this code is run, the result is an estimated  $P_D$  of 0.8228, which as expected is less than the NP bound of 0.8413. As a sidenote, even though the performance is poorer, the cube root sample mean detector can be

shown to be more robust in its performance to nonGaussian noise [[Kay 1998](#), pg. 402]. Hence, accepting the slight degradation in performance may be a good practical alternative to designing a detector that must operate in many different types of noise environments.



### Exercise 7.7 – Practice in using the NP bound

Repeat the previous experiment but choose the data record length to be  $N = 10$ . What is the detection performance of the cube root sample mean detector? Is it below the bound? Significantly so?



The usual bounds that are employed for the various statistical signal processing problems are listed in [Table 7.2](#). Further descriptions can be found in [Chapter 8](#).

**Table 7.2: Typical statistical signal processing performance bounds.**

Problem	Bound
Parameter estimation	Cramer-Rao lower bound on variance, $\text{var}(\hat{\theta}) \geq \text{CRLB}$
Signal detection	Neyman-Pearson, $P_{D_{NP}} \geq P_D$
Classification	Maximum a posteriori decision rule, $P_{c_{MAP}} \geq P_c$

## 7.5. Exact versus Asymptotic Performance

Computer simulations yield “exact” performance evaluations for any data record length (exact as long as the number of trials is adequate—see discussion in [Section 6.4](#)). A disadvantage is that each new set of conditions requires a new computer simulation. An analytical performance evaluation, on the other hand, summarizes the performance under all conditions, lending more insight into the algorithm. Analytical performance evaluations, however, are frequently only available for the asymptotic case, which means that *they are only valid for large data records and/or large SNR*. The lack of exact analytical results is due to the mathematical difficulty involved in deriving them. The conditions under which the asymptotic results can be used as good approximations to the exact results are in general difficult to determine. We can usually only assert that they are valid for a *large enough* data record and/or a *large enough* SNR. Hence, to

determine when they apply one typically must resort again to a computer simulation to determine the necessary values of data record length and/or SNR. The one case in which exact performance results are readily available is for the linear model (see [Section 8.3](#)). The expressions obtained for the linear model apply to any data record length and any SNR. Suffice it to say that the use of analytical asymptotic results for exact performance evaluation is in general risky! Furthermore, *using them to compare the performance of competing algorithms is fraught with danger since the data record lengths and/or SNRs necessary for each algorithm's performance to be accurately given by the asymptotic expression may be different.*

---

### **Example 7.3 – Asymptotic variance of kurtosis estimator**

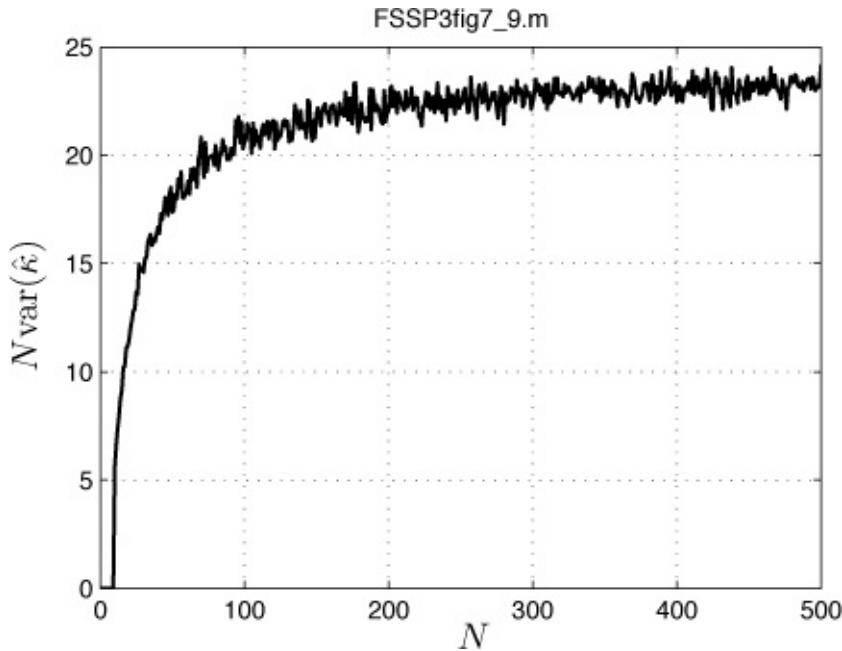
The estimator of the kurtosis was given in [\(6.6\)](#) as

$$\hat{\kappa} = \frac{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^4}{\left(\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2\right)^2}$$

and its asymptotic (as  $N \rightarrow \infty$ ) variance for  $x[n]$  being WGN (see [Appendix 6A](#)) as

$$\text{var}(\hat{\kappa}) = \frac{24}{N}.$$

For finite data records this is only an approximation. To determine how good an approximation it is, one can simulate WGN, estimate the kurtosis and then estimate the variance of the kurtosis estimator. It is more convenient in doing so to consider the *normalized variance*  $N \text{ var}(\hat{\kappa})$ , as  $N$  increases, to see how close this quantity becomes to 24. A computer simulation, in which we generate 10,000 estimates of the kurtosis for a WGN data record of length  $N$ , and then plot the normalized variance versus  $N$  is shown in [Figure 7.9](#). The asymptotic value of 24 is seen to be approximate, with the approximation appearing to be adequate only for data record lengths of 400 or more. Thus, in assessing nonGaussianity of a WGN data record using the kurtosis estimate, at least  $N = 400$  samples are required.



**Figure 7.9: Normalized variance versus data record length for kurtosis estimator.**

◊

---



---

### Exercise 7.8 – Another asymptotic expression

In [Section 6.4](#) we studied how to estimate the autocorrelation sequence (ACS) for a stationary Gaussian random process. The usual estimator is

$$\hat{r}_x[k] = \frac{1}{N} \sum_{k=0}^{N-k-1} x[n]x[n+k] \quad k = 0, 1, \dots, N-1.$$

This estimator is slightly biased since  $E[\hat{r}_x[k]] = (1 - k/N)r_x[k]$ , but the bias is claimed to be an acceptable tradeoff in making sure the estimator has the same properties as the theoretical autocorrelation sequence [[Jenkins and Watts 1968](#)]. For large  $N$  this bias is negligible. A formula for the asymptotic variance (as  $N \rightarrow \infty$ ) is often used. It is

$$\text{var}(\hat{r}_x[k]) = \frac{1}{N} \sum_{m=-\infty}^{\infty} (r_x^2[m] + r_x[m+k]r_x[m-k]). \quad (7.7)$$

Now assume that  $x[n]$  is an autoregressive process of order 1 (see [Section 4.4](#)), so that its true ACS is

$$r_x[k] = \frac{\sigma_u^2}{1 - a^2[1]} (-a[1])^{|k|}.$$

The asymptotic variance of  $\hat{r}_x[2]$  for  $a[1] = -0.7$  and  $\sigma_u^2 = 1$  can be found to be  $N\text{var}(\hat{r}_x[2]) = 17.62$  by using a computer evaluation of (7.7).

Perform a computer simulation to determine how large  $N$  must be for the asymptotic variance to be a good approximation to the actual variance of  $\hat{r}_x[2]$ . Recall that outcomes of an AR process can be generated using the MATLAB program ARgendata.m. Use at least 100,000 trials and estimate the variance for  $N = 100, 200, 300, 400$ , and 500.

## 7.6. Sensitivity

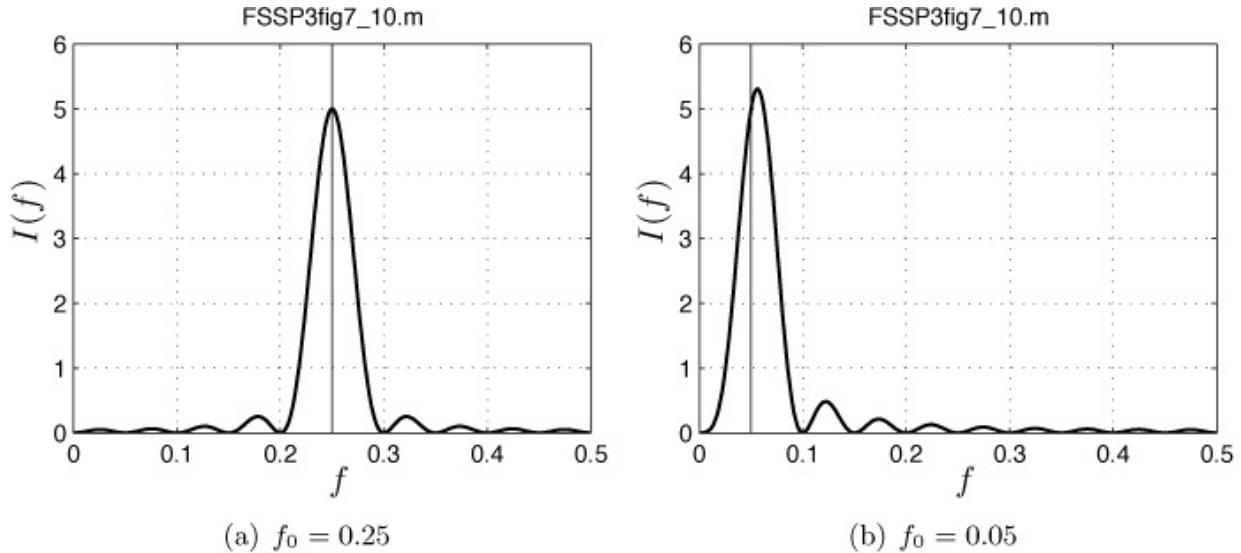
It is important to assess the sensitivity of an algorithm to the underlying assumptions, sometimes called the *robustness* of the algorithm (see also the solution to [Exercise 1.2](#) for an example). This is because in an operational environment the data assumptions will seldom be satisfied. Furthermore, in the implementation of a device there are usually manufacturing tolerances that will cause the device to perform differently than predicted by the ideal design. Finally, if the algorithm has been derived from “first principles” such as maximum likelihood, then in the course of the derivation certain approximations may have been made for mathematical tractability. If these approximations are not satisfied in practice, then the algorithm performance will degrade. An example of this is the use of a periodogram to estimate frequency. It can be shown to be a maximum likelihood estimator if the frequency is in the range  $2/N \leq f \leq 1/2 - 2/N$  (see [Algorithm 9.3](#)). Otherwise, for frequencies that do not satisfy this assumption the positive and negative components of a real sinusoid will interact (see also Figure 1.A.1) to cause a shift in the peak location from the true frequency. To see how sensitive the periodogram estimator is to this assumption it is necessary to employ a computer evaluation. As already noted, it is difficult enough to analytically derive the performance of an algorithm when the assumptions are valid, let alone when they are not.

### Example 7.4 – Sensitivity of periodogram frequency estimator to assumption of possible frequencies

Assume that we have a sinusoid embedded in WGN  $x[n] = \cos(2\pi f_0 n) + w[n]$  for  $n = 0, 1, \dots, N - 1$ . The periodogram defined as

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f n) \right|^2 \quad 0 \leq f \leq 1/2 \quad (7.8)$$

is an approximate maximum likelihood estimator, with the approximation a good one if the frequency satisfies the restriction  $2/N \leq f_0 \leq 1/2 - 2/N$  (see [Algorithm 9.3](#)). To determine the sensitivity to this assumption we plot the periodogram in [Figure 7.10a](#) for  $f_0 = 0.25$  and  $N = 20$ . For sake of clarity we assume that there is no noise present so that we are essentially assessing the sensitivity of the bias of the estimator. Note that for this value of  $N$  the restricted range is  $0.1 \leq f_0 \leq 0.4$ , and thus as seen in [Figure 7.10a](#), the location of the peak of the periodogram indicates the correct frequency. If, however,  $f_0 = 0.05$ , then the true frequency does not satisfy the assumption used to derive the periodogram estimator. As seen in [Figure 7.10b](#), the location of the peak is now displaced from the true frequency.



**Figure 7.10: Sensitivity of the periodogram frequency estimator to true frequency (shown as the light vertical line) for  $N = 20$ . The additive noise is assumed to be zero.**



### Exercise 7.9 – Quantifying the sensitivity of periodogram frequency estimator

Using the location of the maximum of (7.8) for  $x[n] = \cos(2\pi f_0 n)$ , first determine the estimate for each value of  $f_0$  in the interval  $0 < f_0 < 1/2$  for  $N = 20$ . Then, plot the absolute error  $|\hat{f}_0 - f_0|$  as a function of the true frequency  $f_0$ . If the maximum absolute error that can be tolerated is 0.01 what is the range of frequencies, i.e.,  $f_{\min} \leq f_0 \leq f_{\max}$ , for which the estimator can be used?

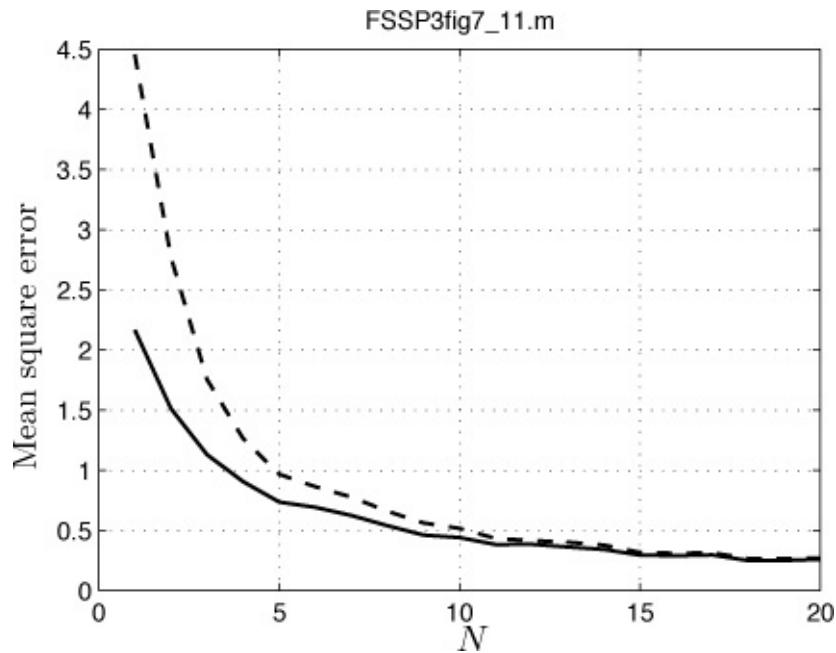
---

## 7.7. Valid Performance Comparisons

Much of statistical signal processing is devoted to designing new algorithms to improve performance. To do so it is *imperative* that a *benchmark* be available. A benchmark is a standard algorithm whose performance can be considered the “one to beat”. Of course, if the benchmark corresponds to an optimal algorithm, with respect to a set of underlying assumptions, then proceeding further is futile. When this is not the case, a benchmark can be quite valuable in developing and assessing new algorithms. It is also usually true that the current benchmark is the result of the efforts of many researchers, so that outperforming it will not be easy.

In comparing algorithms fairly, it is essential that we do not “compare apples to oranges”. The characteristics of the test data and conditions under which the benchmark is assumed to operate under must be adhered to. It is not fair to change either of these. For example, to estimate a DC level in WGN, we use the sample mean. A competitor may indicate that the MSE of a new estimator versus  $N$ , as shown in [Figure 7.11](#) as the solid curve, outperforms the sample mean. According to the figure the claim appears to be true, especially for shorter data records, i.e.,  $N < 10$ . The comparison shown in [Figure 7.11](#) is for a DC level  $A = 1$  embedded in WGN with variance  $\sigma^2 = 5$ . It seems curious that the new estimator should outperform the sample mean since the latter is known to be a minimum variance unbiased estimator [[Kay 1993](#), pg. 31] (although not necessarily a minimum MSE estimator). On closer inspection of the new estimator it is found that it is given explicitly as

$$\begin{aligned}\hat{A}_{\text{new}} &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] && \text{if } \left| \frac{1}{N} \sum_{n=0}^{N-1} x[n] \right| \leq 2 \\ &= 2 && \text{if } \frac{1}{N} \sum_{n=0}^{N-1} x[n] > 2 \\ &= -2 && \text{if } \frac{1}{N} \sum_{n=0}^{N-1} x[n] < -2.\end{aligned}$$



**Figure 7.11: Sample mean estimator (shown as the dashed line) and a competing estimator (shown as the solid line) for a DC level in WGN.**

The algorithm designer has implicitly assumed that the possible range of  $A$  is  $-2 \leq A \leq 2$  in formulating the new estimator. The sample mean estimator, however, makes no such assumption on the range of values of  $A$ . It has been designed to produce a good estimate for *any value of A*. The improvement in performance of the new estimator can therefore be attributed to the replacement of the values of the sample mean that exceed 2 in magnitude by 2. The estimated value is now closer to the true value of  $A = 1$ . This is an example of an unfair comparison. The two competing estimators operate under a different set of assumptions on the data—no fair!

---

### Exercise 7.10 – Effect of a faulty assumption

Determine the MSE for the two estimators, the sample mean and  $\hat{A}_{\text{new}}$ , for the range of values  $2 \leq A \leq 4$  and  $N = 5$ . As before let  $\sigma^2 = 5$ . Do so by fixing a value of  $A$ , generating 1000 outcomes of the data set and then 1000 estimates for each estimator. Finally, compute the MSE as

$$\text{MSE}(A) = \frac{1}{1000} \sum_{i=1}^{1000} (\hat{A}_i - A)^2$$

for each estimator. What can you say about the utility of each estimator?





### There is no “free lunch”.

As illustrated by the previous example and exercise, an algorithm that seems to perform better than a competitor under one set of conditions may not do so under a different set of conditions. In effect, *there is no free lunch*. Improved performance of an algorithm must first be quantified and then must be *explained*. Improved performance is almost always due to implicit information that has been utilized for one algorithm but not the other. If it seems too good to be true, it probably isn’t!



In the same vein, in comparing algorithms it is essential to use the same test case for the data. Such a test case should be chosen to realistically model the operating conditions to be found in practice. Doing so avoids the invalid and unfair comparison of algorithms.

## 7.8. Performance/Complexity Tradeoffs

Optimal algorithms usually come with a high computational cost. Other than the linear model, we are usually required to maximize or integrate a multidimensional function. Doing so in the implementation of a real-time algorithm may not be possible. Hence, we are faced with either a “paring down” of the optimal algorithm or its replacement by a suboptimal one in an effort to reduce the computation. It is always the case that the computational complexity can be traded off against performance. An example of this tradeoff is illustrated by the example of [Section 5.4](#) in which the amplitudes, phases, damping factors, and frequencies of the signal

$$s[n] = A_1 r_1^n \cos(2\pi f_1 n + \phi_1) + A_2 r_2^n \cos(2\pi f_2 n + \phi_2) \quad n = 0, 1, \dots, N - 1$$

are to be estimated. An optimal estimator must maximize a four-dimensional function of the parameters  $\{r_1, f_1, r_2, f_2\}$ . To reduce the computation one can instead use the covariance method (see [Algorithm 11.3](#) and [[Kay 1988](#), pg. 225]), which only requires the solution of a set of linear equations. Of course, the tradeoff is poorer performance, especially when the SNR is low. Whether this tradeoff is a reasonable one will depend upon how much performance degradation can be tolerated.

In testing algorithms via a computer simulation it is necessary to develop code, typically in MATLAB. We now consider code development for performance evaluation only and *not* a software implementation for an actual system, which might be a computer program encoded in C<sup>++</sup> for a specific hardware platform. Since the code is for performance testing purposes only, a preliminary version should mimic the mathematics as closely as possible. It is of no concern at this stage how “pretty” the code is or how efficient it is in terms of computation time. *Of foremost concern is accuracy.* (The author is frequently guilty of computing constants in loops—luckily the computer doesn’t object to this waste of computer resources!)

Coding of an algorithm must be done in *small steps*, with each code segment validated before the next step proceeds. Otherwise, if the entire algorithm and performance evaluation is coded, don’t expect it to work—it never does! Each person will have his/her own methods for developing code. For the purposes of signal processing algorithm coding the author has found the following strategy to be useful.

1. Write the program in as simple a form as possible using small segments of code.
2. Validate each step. To do so test with no noise first and then also with a large data record.
3. Once a segment of code used for a particular purpose is coded and validated, it is helpful to recode it as a *function subprogram*.
4. Test against a case for which the known theoretical result is available.

An example follows.

---

### Example 7.5 – Performance of sample mean estimator in Gaussian mixture noise

Assume we wish to determine the MSE of the sample mean estimator of  $A$  for the data model  $x[n] = A + w[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $w[n]$  is IID noise with PDF given as the Gaussian mixture

$$p_W(w) = (1 - \epsilon) \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2\sigma_1^2}w^2\right) + \epsilon \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_2^2}w^2\right).$$

This PDF was described in [Section 4.6](#). It might be desired to produce a plot of the MSE versus the value of  $A$  for a given data record length  $N$  and

for given values of the noise parameters  $\epsilon$ ,  $\sigma_1^2$ ,  $\sigma_2^2$ . Assume that  $1 \leq A \leq 5$ ,  $N = 10$ , and  $\epsilon = 0.1$ ,  $\sigma_1^2, \sigma_2^2 = 100$ . Then the first step is to be able to generate IID samples of the noise. This requires the use of rand to be able to choose the Gaussian components with probabilities  $1 - \epsilon$  and  $\epsilon$ . Once the Gaussian component is chosen, a Gaussian random variable outcome is generated using randn. Since these outcomes will be random from computer run to computer run, it is advantageous to fix these “random numbers” using the calls `rand('state', 0)` and `randn('state', 0)`, which sets each random number generator to its initial state. Otherwise, each time the program is run, a different set of random numbers will be generated, greatly confusing the code debugging. When the program is operational, these calls may be omitted, if desired. The code necessary to generate a single realization of  $N$  samples of Gaussian mixture noise with the parameters specified is given below.

[Click here to view code image](#)

```

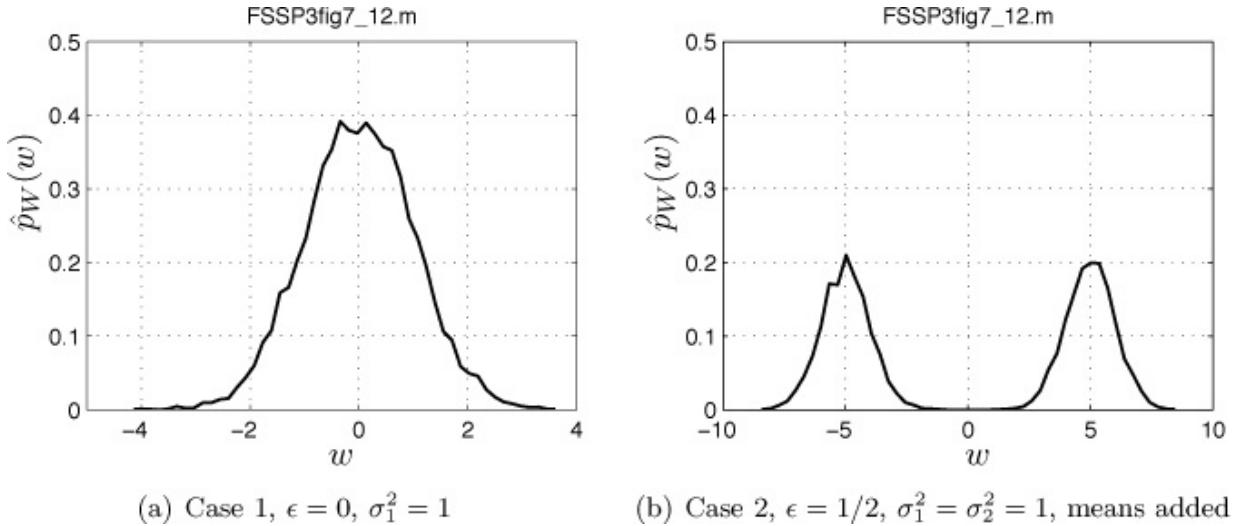
clear all          % always clear out all "stale" data values
rand('state',0)   % initialize uniform random number generator
randn('state',0)  % initialize Gaussian random number generator
epsilon=0.1;       % set noise parameters
sig21=1;
sig22=100;
N=10;             % set data record length
u=rand(N,1);      % uniform random variable outcomes used
                  % to choose Gaussian components
for i=1:N
    if u(i)<1-epsilon % choose first Gaussian component
        % with probability 1-epsilon
        w(i,1)=sqrt(sig21)*randn(1,1); % scale to obtain
                                         % correct variance
    else % choose second Gaussian component with probability epsilon
        w(i,1)=sqrt(sig22)*randn(1,1); % scale to obtain
                                         % correct variance
    end
end

```

How should we test this segment of code? A known but special test case is that of a Gaussian PDF with mean zero and variance one. We can obtain this special case by letting `epsilon=0`. Doing so for  $N = 10,000$  and using `pdf_hist_est.m` to estimate the PDF (see [Section 6.4.6](#) for more details) produces the PDF estimate shown in [Figure 7.12a](#). (Recall that the samples are IID and so  $N = 10,000$  can be considered as 10,000 outcomes of the single random variable  $W$ .) Comparing this to the

theoretical PDF, which is

$$p_W(w) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}w^2\right)$$



(a) Case 1,  $\epsilon = 0$ ,  $\sigma_1^2 = 1$

(b) Case 2,  $\epsilon = 1/2$ ,  $\sigma_1^2 = \sigma_2^2 = 1$ , means added

**Figure 7.12: Estimated Gaussian mixture PDF for two test cases.**

indicates a very close fit. One could also check the mean and the variance to see if these estimates were close to the true values of 0 and 1, respectively. Since we have only tested the case when there is a single Gaussian mode, we also must check to see if the code produces the mixture PDF. To easily discern this case we can add a constant to each Gaussian mode, a +5 and a -5, to make the modes separate. Also, we set  $\epsilon = 0.5$  and  $\sigma_1^2 = \sigma_2^2 = 1$  so that the modes are of equal height. In this case the theoretical PDF is

$$p_W(w) = \frac{1}{2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(w-5)^2\right) + \frac{1}{2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(w+5)^2\right).$$

We should see two versions of [Figure 7.12a](#), one centered at  $w = 5$  and the other centered at  $w = -5$  and each one should have height 1/2 of that seen in [Figure 7.12a](#). The code used is

[Click here to view code image](#)

```
clear all
rand('state',0)
randn('state',0)
epsilon=0.5;
sig21=1;
sig22=1;
N=10000;
u=rand(N,1);
```

```

for i=1:N
    if u(i)<1-epsilon
        w(i,1)=sqrt(sig21)*randn(1,1)-5; % now add -5 for
                                         % validation purposes only
    else
        w(i,1)=sqrt(sig22)*randn(1,1)+5; % now add +5 for
                                         % validation purposes only
    end
end
[xx,pdfest]=pdf_hist_est(w,50,1,-10,10,0.5);

```

with the result shown in [Figure 7.12b](#).

Next it is convenient to code the Gaussian mixture data generation into a function subprogram. This is given below and is contained on the CD.

[Click here to view code image](#)

```

% Gaussmix_gendata.m
%
% This program generates a realization of IID Gaussian mixture noise
% given the mixture parameters and the noise variances.
%
% Input Parameters:
%
%     sig21      - variance of first component of Gaussian mixture
%     sig22      - variance of second component of Gaussian mixture
%     epsilon    - mixing probability
%     N          - number of data points desired
%
% Output Parameters:
%
%     x          - array of dimension Nx1 of data samples
%
function x=Gaussmix_gendata(sig21,sig22,epsilon,N)
    u=rand(N,1);
    for i=1:N
        if u(i)<1-epsilon
            x(i,1)=sqrt(sig21)*randn(1,1);
        else
            x(i,1)=sqrt(sig22)*randn(1,1);
        end
    end

```

This then allows the successive code that is written to be easier to follow. It will have fewer interactions between variables and provides a subprogram for all future work. Note that this function subprogram should be tested against the code in the original program to make sure the outputs are identical (don't forget though to use `rand ('state', 0)` and `randn ('state', 0)` before calling the function subprogram).

Next we simulate the DC level in Gaussian mixture noise for a data record of  $N = 10$  and 10,000 trials using

[Click here to view code image](#)

```
clear all
rand('state',0)
randn('state',0)
epsilon=0.1;
sig21=1;
sig22=100;
N=10;
A=1;
for i=1:10000
x=A+Gaussmix_gendata(sig21,sig22,epsilon,N);
Ahat(i,1)=mean(x);
end
```

This should generate 10,000 estimates of  $A$  using the sample mean  $\hat{A} = (1/N) \sum_{n=0}^{N-1} x[n]$ . Since the sample mean is known to be unbiased, the MSE should equal the variance. The variance of  $\hat{A}$  should be

$$\text{var}(\hat{A}) = \text{var}(W)/N = \frac{(1 - \epsilon)\sigma_1^2 + \epsilon\sigma_2^2}{N} = 1.09$$

for our example. Hence, adding the code

[Click here to view code image](#)

```
mse=sum((Ahat-A).^2)/10000
var_theory=((1-epsilon)*sig21+epsilon*sig22)/N
```

will produce  $\text{mse} = 1.1018$  and  $\text{var\_theory} = 1.0900$ , which appear to be in close agreement. Finally, the code can be augmented to loop over different values of  $A$ .



---

## 7.10. Algorithm Documentation

Proper documentation of an algorithm can be a painful process. It is necessary though, if the algorithm is to be reliably used by others. If the algorithm is to be compared to existing algorithms, implemented in a system, or used as a basis for further development, a document fully describing the “inner workings” is always the first step to success. In [Appendix 7A](#) a suggested listing of the information to be included in such a document is given. Working from this checklist a sample document description is given in [Appendix 7B](#).

## 7.11. Lessons Learned

- The first step in algorithm development is to implement the algorithm using a computer simulation. It allows a quick study of performance, robustness to design parameters, and computational complexity, and also lends insight into the inner workings of the algorithm.
- Test the algorithm on computer-generated data before trying actual field data since the former can be easily controlled, not so the latter.
- Always employ a statistical measure of performance.
- In a computer simulation performance evaluation, make sure to use enough trials. Keep increasing the number of trials until the performance metric gives a consistent result.
- Use performance bounds to compare the algorithm performance against. This can sometimes point out analytical and/or software implementation errors and also serves as an indication of the loss in performance against an optimal algorithm.
- Never use an asymptotic analytical result to determine the performance of an algorithm without first establishing that the approximation is a good one. Similarly, do not use asymptotic results to compare the performances of competing algorithms, unless the asymptotic approximations are valid for the operating conditions of *both* algorithms.
- Once the algorithm has been developed and tested with ideal computer data, be sure to test its robustness by varying the design parameters.
- Use benchmarks in assessing algorithm performance.
- Be careful to compare algorithms using the same data assumptions.
- If the performance of an algorithm is too good to be true, it probably isn't (no free lunch).
- In coding an algorithm in software follow the mathematics as closely as possible and sacrifice code efficiency for accuracy.
- Always document the algorithm in detail for future reference and for distribution to others.

## References

Jenkins, G.M., D.G. Watts, *Spectral Analysis and Its Applications*, Holden-Day, San Francisco, 1968.

Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall,

Englewood Cliffs, NJ, 1988.

Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory, Vol. I*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.

Rubinstein, R.Y., *Simulation and the Monte Carlo Method*, Wiley, NY, 1981.

## **Appendix 7A. A Checklist of Information to Be Included in Algorithm Description Document**

### **Problem and Goals**

- detection, estimation, classification, etc.
- metric for performance criterion (ROC, MSE, etc.)

### **History**

- approach is standard, adapted from previous approaches, new approach
- optimal approach known
- proposed approach used in other fields – successes and difficulties
- provided words for all acronyms used

### **Assumptions**

- prior knowledge necessary
- what is assumed known and what is assumed unknown and must be estimated (either in-situ or from other data sources)
- statistical assumptions made

### **Mathematical model**

- description of model with equations
- signal and noise models
- probability density functions

### **Algorithm description**

- word description – how does algorithm operate and why it should work (no equations)
- block diagram with pseudocode and equation number references (also cross reference equation numbers to MATLAB code)
- Computational complexity

## **Algorithm implementation**

- departures from equations or “tricks” used (for example, FFT used to implement Fourier transform)

## **MATLAB implementation**

- code should be easy to follow!! Use function subprograms wherever possible.
- documented code with equation number references
- flow chart
- description of variables (use variable names that match equation symbols)
- computational tricks used
- toolboxes and supporting programs required
- typical run time duration
- MATLAB version number
- verification test case

## **Performance for computer simulated data**

- reproducibility – set random number generators to initial state
- table of parameter values used
- results obtained on computer simulated data
- compare to theoretical performance, if possible
- compare to bounds such as matched filter for detection or maximum a posteriori decision rule for classification

## **Performance for field data**

- table of parameter values used
- results obtained on field data
- comparison to benchmark performance

## **Strong/weak links**

- when does algorithm work well, and when does it not?
- other evidence of possible success of proposed algorithm based on similar results in literature
- sensitivity analysis to assumptions and parameter values
- if iterative in nature, convergence issues

---

## References

- open literature
- proprietary materials

## Appendices

- mathematical derivations
- listing of all MATLAB code (code must be documented)

## Appendix 7B. Example of Algorithm Description Document

### A Computationally Efficient Sinusoid Detector Abstract

An algorithm to detect a sinusoid of unknown frequency in white Gaussian noise is proposed. It is shown to have less computational complexity than a generalized likelihood ratio test, and outperforms an energy detector.

#### 7B.1. Problem and Goals

The problem is to detect a *single* sinusoidal signal of unknown frequency that is embedded in noise. It is assumed that the metric to be used to assess performance is the receiver operating characteristic (ROC). The goal is to reduce the computation that would be needed to implement a generalized likelihood ratio test, which must first estimate the frequency.

#### 7B.2. History

The optimal approach to this problem would be a generalized likelihood ratio test (GLRT) [1]. The approach described herein is similar to one proposed in [2] for detection of a complex sinusoid in complex white Gaussian noise (CWGN). Here the data is real and so the previous approach does not apply. There it was shown to produce performance nearly as good as the GLRT but with less computation.

#### 7B.3. Assumptions

The frequency of the sinusoid is assumed to be unknown but within certain limits. The amplitude and phase of the sinusoid are assumed known. The noise is assumed to be white Gaussian noise whose variance need not be known if only an ROC is desired. To set a threshold for a fixed false alarm rate, however, the variance would need to be known a priori. The time duration over which the signal may be present is known.

#### 7B.4. Mathematical Model

The signal is given by

$$s[n] = \cos(2\pi f_0 n) \quad n = 0, 1, \dots, N - 1$$

where  $f_0$  is the unknown frequency, which is known to be within the interval  $[f_{0\min}, f_{0\max}]$ . The noise is assumed to be WGN with variance  $\sigma_w^2$  [3, pg. 528]. The overall received data is assumed to be  $x[n] = w[n]$  when the signal is absent and  $x[n] = s[n] + w[n]$  when the signal is present for  $n = 0, 1, \dots, N - 1$ .

### 7B.5. Algorithm Description

The proposed detection algorithm is to decide that a signal is present if the “autocorrelation” of the data produces a large value exceeding zero. This is based on the premise that the autocorrelation sequence of a sinusoid is some positive value (for a given range of frequencies) while that for WGN is theoretically zero for a lag greater than zero. To see this note that the autocorrelation for a lag of one of a data set is defined as

$$T_{ac}(\mathbf{x}) = \sum_{n=0}^{N-1} x[n]x[n+1] \quad (7B.1)$$

where  $\mathbf{x} = [x[0] \ x[1] \ \dots \ x[N-1]]^T$ . If noise only is present, this will be on the average

$$\begin{aligned} E[T_{ac}(\mathbf{x})] &= E \left[ \sum_{n=0}^{N-1} w[n]w[n+1] \right] \\ &= \sum_{n=0}^{N-1} E[w[n]w[n+1]] \\ &= \sum_{n=0}^{N-1} E[w[n]]E[w[n+1]] = 0 \end{aligned}$$

where  $E[\cdot]$  denotes the expected value. The last step results from the assumption that the noise is WGN and hence is zero mean and is uncorrelated [3]. The autocorrelation of a sinusoid, on the other hand, is

$$\begin{aligned}
T_{ac}(\mathbf{x}) &= \sum_{n=0}^{N-1} \cos(2\pi f_0 n) \cos(2\pi f_0(n+1)) \\
&= \sum_{n=0}^{N-1} \left[ \frac{1}{2} \cos(2\pi f_0) + \frac{1}{2} \cos(2\pi f_0(2n+1)) \right] \\
&\approx \frac{N}{2} \cos(2\pi f_0) + 0
\end{aligned}$$

with the double-frequency term summing to a small value relative to the first term. Note that  $f_0$  should be between 0 and 1/4 or else  $\cos(2\pi f_0)$  could be zero (for example, if  $f_0 = 1/4$ ) or even negative. This places a limit on the allowable frequencies.

The proposed algorithm has a modest computational complexity. This is because from (7B.1) the number of multiply/adds is on the order of  $N$ . This compares favorably with even an FFT, which requires on the order of  $N \log_2 N$  multiply/adds.

Other detectors that can be used as benchmarks on performance are the matched filter as an upper bound, which assumes that  $f_0$  is known, and the energy detector, which is typically used in the absence of much signal information. These two other detectors are given by

$$T_{mf}(\mathbf{x}) = \sum_{n=0}^{N-1} x[n] \cos(2\pi f_0 n) \quad (7B.2)$$

for the matched filter and

$$T_{ed}(\mathbf{x}) = \sum_{n=0}^{N-1} x^2[n] \quad (7B.3)$$

for the energy detector. The performance of the autocorrelation detector is expected to be poorer than the matched filter but better than the energy detector.

No pseudocode is provided for the algorithm since it is a very simple algorithm.

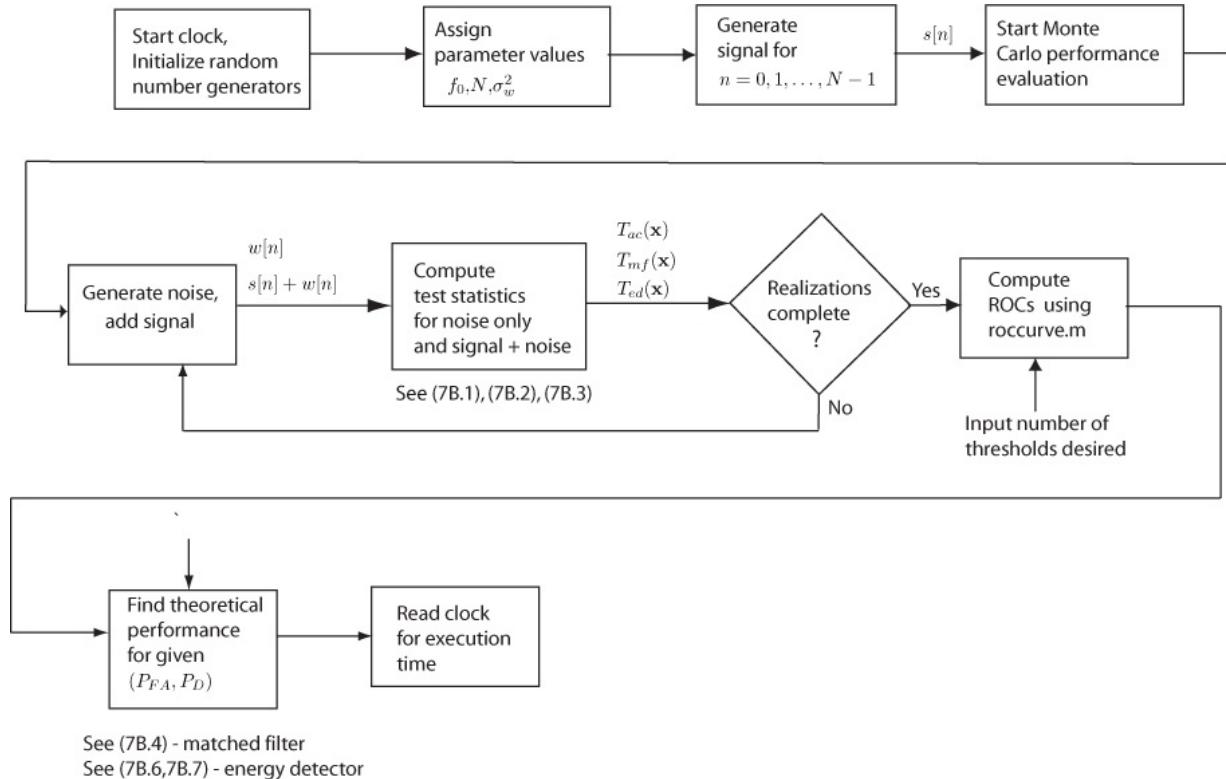
## 7B.6. Algorithm Implementation

Straightforward implementation in MATLAB.

## 7B.7. MATLAB Implementation

See MATLAB code for `detection_demo.m` in [Section 7B.12](#), as well as

supporting programs. A flow chart of the program `detection_demo.m` is given in [Figure 7B.1](#).



**Figure 7B.1: Flow chart of computer code, `detection_demo.m`.**

The version of MATLAB used is R2006A. No toolboxes have been used. The cross-reference table of the variables used is given in [Table 7B.1](#). A verification test case is given in the code. The run time for the code given was about 10 sec. No computational tricks are used.

**Table 7B.1: Translation of mathematical variables to MATLAB.**

Symbol	MATLAB variable	Description
$f_0$	<code>f0</code>	frequency
$N$	<code>N</code>	data record length
$s[n]$	<code>s</code>	signal
$w[n]$	<code>w</code>	noise
$n$	<code>n</code>	time index
$\sigma_w^2$	<code>varw</code>	noise variance
$T_{ac}(\mathbf{x})$	<code>Tac_0, Tac_1</code>	autocorrelator
$T_{mf}(\mathbf{x})$	<code>Tmf_0, Tmf_1</code>	matched filter
$T_{ed}(\mathbf{x})$	<code>Ted_0, Ted_1</code>	energy detector
$d^2$	<code>d2</code>	deflection coefficient
$P_{FA}$	<code>Pfa</code>	probability of false alarm
$P_D$	<code>Pd</code>	probability of detection
$\gamma$	<code>gamma</code>	threshold

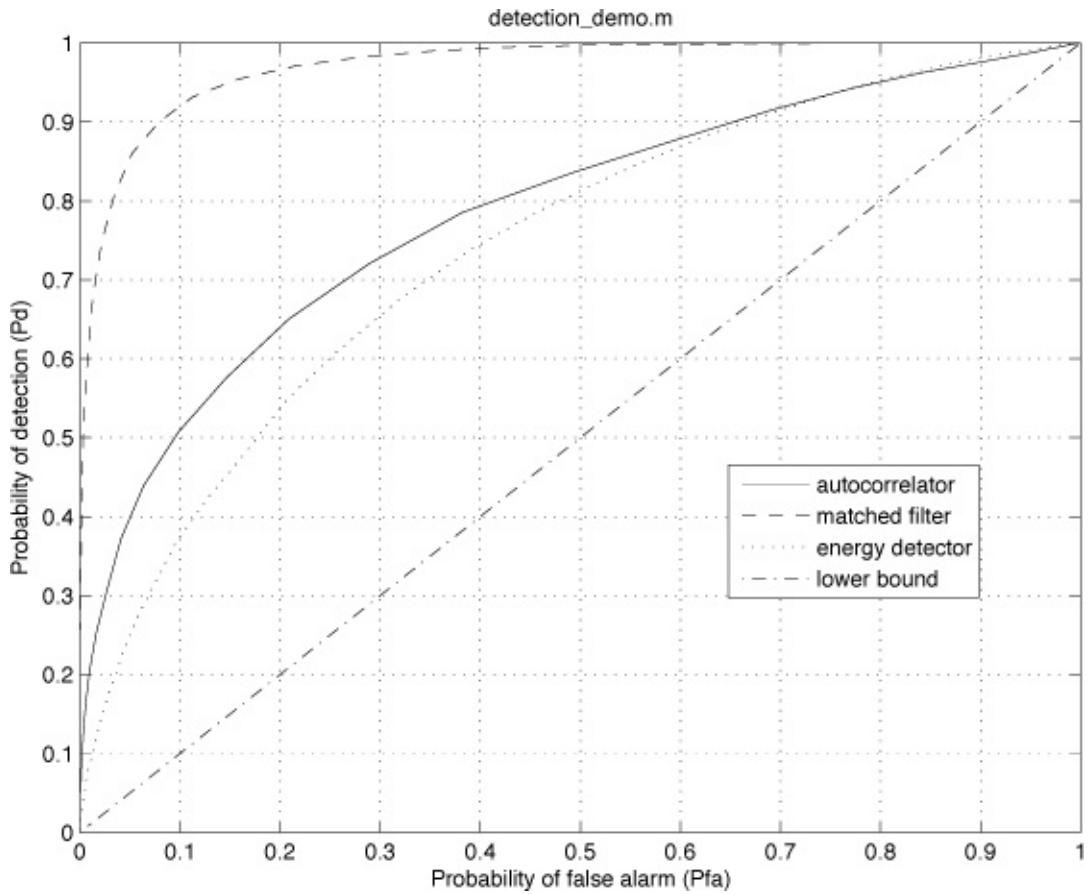
### 7B.8. Performance for Computer-Generated Data

The data parameter values used in the simulation are listed in [Table 7B.2](#). The results are shown in [Figure 7B.2](#), where the ROC (probability of detection  $P_D$  versus probability of false alarm  $P_{FA}$ ) is plotted. As expected the matched filter does the best, the energy detector the worst for most of the  $P_{FA}$  range and the autocorrelation detector performance is somewhere in the middle. The analytical performance prediction for the matched filter and the energy detector can be obtained. For the matched filter it is [1, pg. 103]

$$P_D = Q(Q^{-1}(P_{FA}) - \sqrt{d^2}) \quad (7B.4)$$

**Table 7B.2: Values used in simulation.**

Symbol	MATLAB variable	Value	Description
$f_0$	<code>f0</code>	0.025	frequency
$N$	<code>N</code>	25	data record length
$\sigma_w^2$	<code>varw</code>	2	noise variance



**Figure 7B.2: ROC for computer-generated data.**

where the  $Q$  function is defined as

$$Q(x) = \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}t^2\right] dt$$

and the inverse  $Q$  function, i.e.,  $Q^{-1}$  is the value of  $x$  needed to obtain a particular value of  $Q(x)$ . The  $d^2$  is the deflection coefficient defined as

$$d^2 = \frac{\sum_{n=0}^{N-1} s^2[n]}{\sigma_w^2}. \quad (7B.5)$$

The theoretical performance for  $P_{FA} = 0.1$  and  $d^2 = 7.1642$  is calculated in the MATLAB program `detection_demo.m` and is found to be  $P_D = 0.9185$ , in good agreement with the results shown in [Figure 7B.2](#). The performance for the energy detector is given as (see [Section 7B.12](#))

$$P_{FA} = Q_{\chi_N^2} \left( \frac{\gamma}{\sigma_w^2} \right) \quad (7B.6)$$

$$P_D = Q_{\chi'^2_N(d^2)} \left( \frac{\gamma}{\sigma_w^2} \right) \quad (7B.7)$$

where  $Q_{\chi_N^2}(x)$  is the right-tail probability of a central chi-squared random variable with  $N$  degrees of freedom and  $Q_{\chi'^2_N(d^2)}(x)$  is the right-tail probability of a noncentral chi-squared random variable with  $N$  degrees of freedom and noncentrality parameter  $d^2$  [1, pg. 26]. The theoretical performance is calculated in the MATLAB program `detection_demo.m` for  $\gamma = 61$ , and is found to be  $P_{FA} = 0.2061$  and  $P_D = 0.5365$ , in good agreement with the results shown in [Figure 7B.2](#).

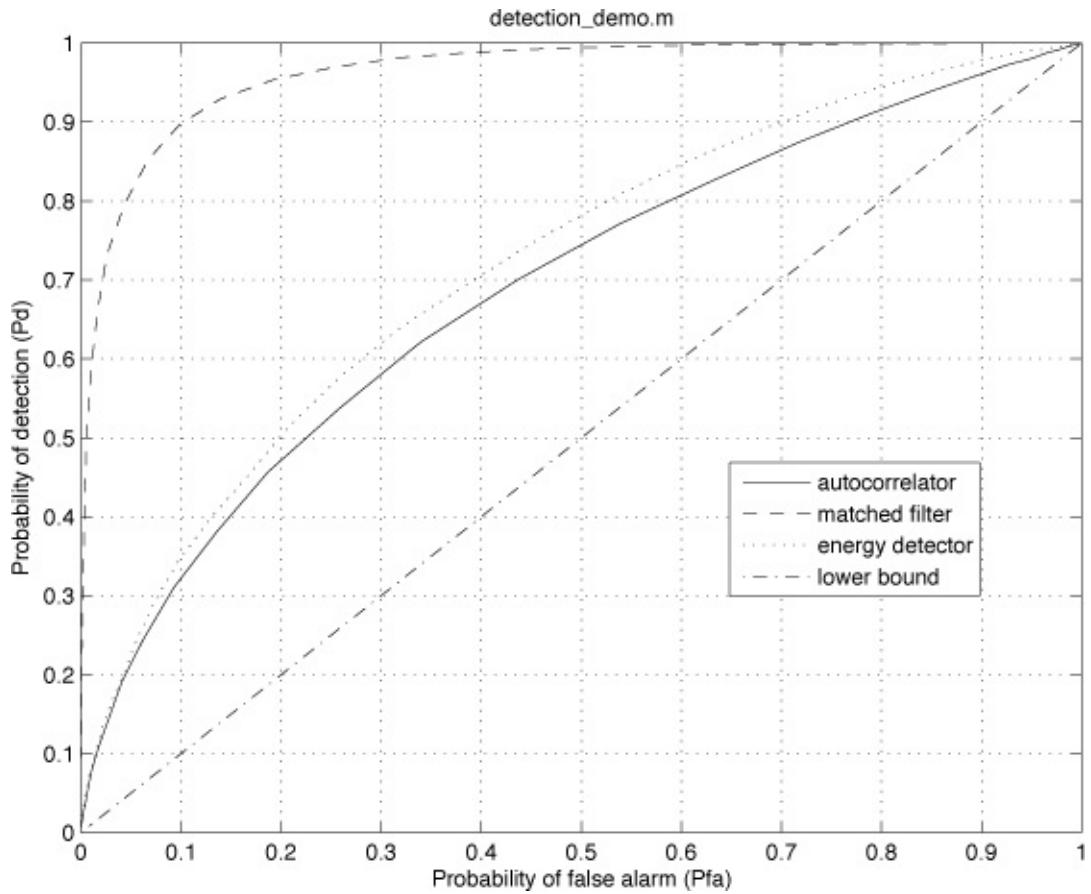
The theoretical performance of the autocorrelation detector is difficult to find.

### 7B.9. Performance for Field Data

1. List the parameter values used
2. Show results of proposed algorithm on test data provided
3. Compare proposed algorithm performance to benchmark performance

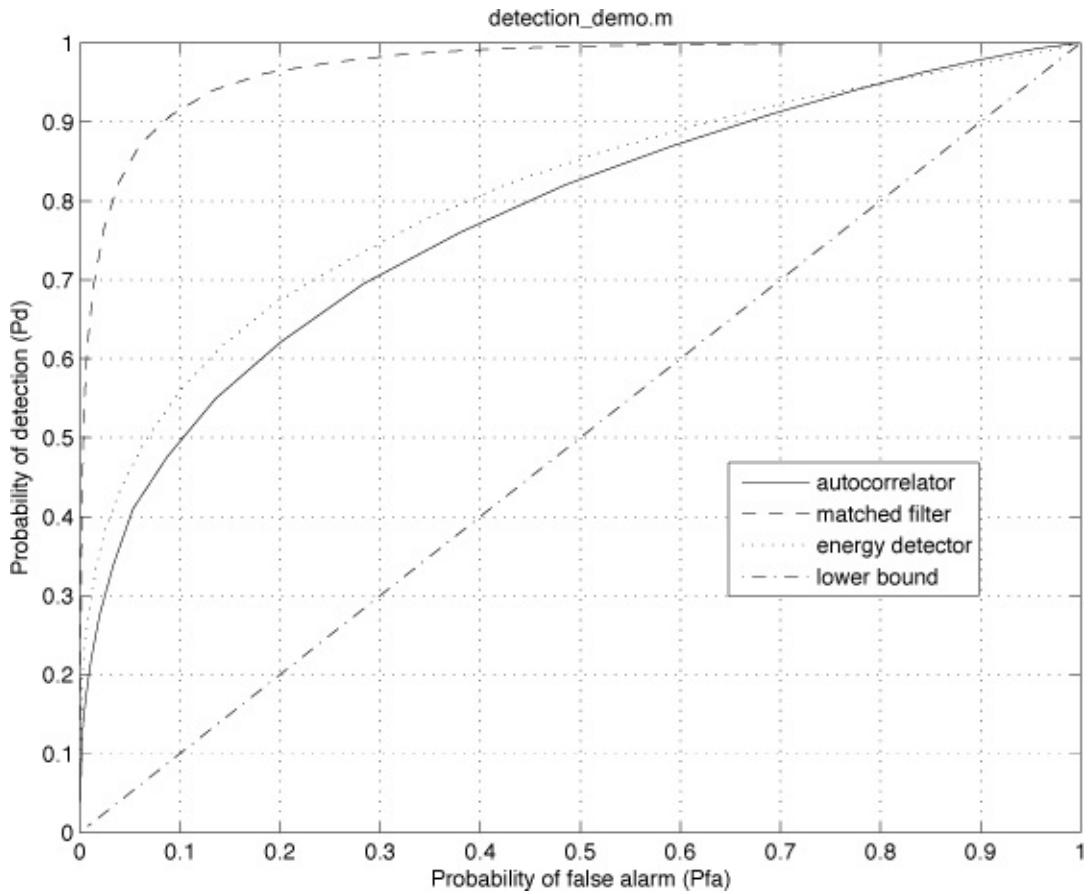
### 7B.10. Strong/Weak Links

As mentioned previously, the frequency needs to be in the specified range. Its performance will depend critically on the frequency. An example is given in [Figure 7B.3](#), where  $f_0 = 0.15$  and should be compared to [Figure 7B.2](#). Now the energy detector outperforms the proposed detector.



**Figure 7B.3: ROC for computer-generated data – effect of frequency.**

The assumption of white *Gaussian* noise is critical since the replacement of WGN by white *uniform noise* also produces poorer results. White uniform noise is defined as independent and identically distributed noise samples with probability density function being uniform on the interval  $[-\sqrt{3\sigma_w^2}, \sqrt{3\sigma_w^2}]$ . The results for the computer simulated data are shown in [Figure 7B.4](#) and should be compared to [Figure 7B.2](#). From [Figure 7B.4](#) it is seen that the energy detector now outperforms the proposed detector.



**Figure 7B.4: ROC for computer-generated data – effect of noise statistics.**

## 7B.11. References

1. S. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Prentice-Hall, Upper Saddle, NJ, 1998.
2. S. Kay, “Robust Detection via Autoregressive Spectrum Analysis”, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pp. 256–269, April 1982.
3. S. Kay, *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.

## 7B.12. Supporting Materials

### Evidence for Improved Performance

In [2] it was shown that for complex data the proposed detector (when normalized to one in magnitude) had only a 2.5 dB loss relative to a GLRT. The frequency was assumed to be unknown and could take on any value, not necessarily restricted to some interval.

### Derivation of Energy Detector Performance

The energy detector is defined as one that decides a signal is present if

$$T_{ed}(\mathbf{x}) = \sum_{n=0}^{N-1} x^2[n] > \gamma.$$

When noise only is present,  $x[n] = w[n]$  for  $w[n]$  being WGN with variance  $\sigma_w^2$ . Hence,

$$\begin{aligned} P_{FA} &= P[T_{ed}(\mathbf{x}) > \gamma] = P\left[\frac{T_{ed}(\mathbf{x})}{\sigma_w^2} > \frac{\gamma}{\sigma_w^2}\right] \\ &= P\left[\sum_{n=0}^{N-1} \left(\frac{x[n]}{\sigma_w}\right)^2 > \gamma/\sigma_w^2\right] \\ &= P[\chi_N^2 > \gamma/\sigma_w^2] \\ &= Q_{\chi_N^2}\left(\frac{\gamma}{\sigma_w^2}\right). \end{aligned}$$

When there is a signal present, we have by the same argument that

$$P_D = P\left[\sum_{n=0}^{N-1} \left(\frac{x[n]}{\sigma_w}\right)^2 > \gamma/\sigma_w^2\right]$$

but now each  $x[n]/\sigma_w$  is Gaussian with a nonzero mean of  $s[n]/\sigma_w$  and a variance of one. Thus,

$$\sum_{n=0}^{N-1} \left(\frac{x[n]}{\sigma_w}\right)^2$$

is a noncentral chi-squared random variable with noncentrality parameter

$$\sum_{n=0}^{N-1} \left(\frac{s[n]}{\sigma_w}\right)^2$$

and the results given in (7B.6) and (7B.7) follow.

### MATLAB Code

This program as well as the external programs required are contained on the CD.

[Click here to view code image](#)

`detection_demo.m`

```

% detection_demo.m
%
% This program compares the performance of an autocorrelator detector
% with that of a matched filter and an energy detector by computing a
% receiver operating characteristic (ROC). The signal is
% a sinusoid of unknown frequency embedded in white Gaussian noise.
%
%
% Input parameters:
%
%   f0 - frequency of sinusoidal signal (0<f0<0.25)
%   N   - number of samples in data record
%   varw - variance of white Gaussian noise
%   nreal - number of realizations used to generate ROC
%
%
% Output parameters:
%
%   Pfa - array of dimension ngam x 1 containing Pfa values
%   Pd   - array of dimension ngam x 1 containing Pd values
%
%
% Verification of program:
%
% The exact values for the various detectors should be:
% for the autocorrelator, Pfa_ac(25)=0.0981, Pd_ac(25)=0.5069
% for the matched filter, Pfa_mf(25)=0.1575, Pd_mf(25)=0.9540
% for the energy detector, Pfa_ed(25)= 0.0428, Pd_ed(25)= 0.2270
%
%
% External programs required:
%
% roccurve.m - used to compute the ROC
% Q.m         - used to compute Q function (see [1, pg. 50])
% Qinvm.m     - used to compute inverse Q function (see [1, pg. 51])
% Qchipr2     - used to compute right-tail-probability for
%                 central and noncentral chi-squared probability
density
%
%                         function (see [1, pg. 55])
%
clear all
close all
timestart=clock; % clock time when program begins
randn('state',0) % set random number generator to fixed initial state
to
% generate same set of noise samples each time this program is run
rand('state',0)
f0=0.025;
% f0=0.15; % change to this frequency to show sensitivity to
frequency
N=25;
n=[0:N-1]'; % integer time samples
s=cos(2*pi*f0*n); % generate signal
varw=2;
nreal=10000;

```

```

for i=1:nreal % compute realizations of detector test statistics
    w=sqrt(varw)*randn(N,1); % generate white Gaussian noise
%    w=sqrt(12*varw)*(rand(N,1)-0.5); % generate white uniform noise
to show
                                % effect of noise statistics
x=s+w;                         % generate signal plus noise
Tac_0(i,1)=sum(w(1:N-1).*w(2:N)); % autocorrelation test
statistic for
                                % noise only - see (7B.1)
Tac_1(i,1)=sum(x(1:N-1).*x(2:N)); % autocorrelation test
statistic for
                                % signal plus noise
Tmf_0(i,1)=w'*s; % matched filter test statistic - see (7B.2)
Tmf_1(i,1)=x'*s;
Ted_0(i,1)=w'*w; % energy detector test statistic - see (7B.3)
Ted_1(i,1)=x'*x;
end
ngam=50; % number of thresholds used to determine (Pfa, Pd) pairs
[Pfa_ac,Pd_ac,gam]=roccurve(Tac_0,Tac_1,ngam);
[Pfa_mf,Pd_mf,gam]=roccurve(Tmf_0,Tmf_1,ngam);
[Pfa_ed,Pd_ed,gam]=roccurve(Ted_0,Ted_1,ngam);
plot(Pfa_ac,Pd_ac,Pfa_mf,Pd_mf,Pfa_ed,Pd_ed,[0:0.01:1]',[0:0.01:1]')
xlabel('Probability of false alarm (Pfa)')
ylabel('Probability of detection (Pd)')
legend('autocorrelator','matched filter','energy detector', 'lower
bound')
title('detection\_demo.m')
grid
d2=s'*s/varw; % This is deflection coefficient - see (7B.5)
Pfa_mft=0.1; Pd_mft=Q(Qinv(Pfa_mft)-sqrt(d2)) % matched filter
theoretical
                                % Pd - see (7B.4)
gamma=61;epsilon=0.01;
Pfa_edt=Qchipr2(N,0,gamma/varw,epsilon) % energy detector theoretical
                                % Pfa - see (7B.6)
Pd_edt=Qchipr2(N,d2,gamma/varw,epsilon) % energy detector
theoretical
                                % Pd - see (7B.7)
runtime=clock-timestamp % total run time

```

## Appendix 7C. Solutions to Exercises

To obtain the results described below you should initialize the random number generators to `rand ('state', 0)` and `randn ('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**7.1** The estimated PDF for the observed median looks very similar to that shown in [Figure 7.2b](#). If the variance is estimated from the computer simulation, you should find that the variance of the sample mean  $\hat{A}_2$  is

0.0037 and that for the median estimator  $\hat{A}_{\text{med}}$  is 0.0047, slightly larger.

The program `FSSP3exer7_1.m`, which is contained on the CD, produces these results.

**7.2** To show that both estimators are unbiased we have

$$\begin{aligned} E[\hat{A}_1] &= E[x[0]] = E[A + w[0]] \\ &= E[A] + E[w[0]] \quad (\text{linearity of expectation}) \\ &= A + E[w[0]] \quad (\text{expected value of constant is the constant}) \\ &= A \quad (\text{noise has zero mean}) \end{aligned}$$

and similarly for  $\hat{A}_2$

$$\begin{aligned} E[\hat{A}_2] &= E\left[\frac{1}{3}(x[0] + x[1] + x[2])\right] \\ &= \frac{1}{3}(E[x[0]] + E[x[1]] + E[x[2]]) \quad (\text{linearity of expectation}) \\ &= \frac{1}{3}(A + A + A) = A. \end{aligned}$$

The variances are

$$\begin{aligned} \text{var}(\hat{A}_1) &= \text{var}(x[0]) \\ &= \text{var}(A + w[0]) \\ &= \text{var}(w[0]) \quad (\text{addition of a constant does not change variance}) \\ &= \sigma^2 \end{aligned}$$

and

$$\begin{aligned} \text{var}(\hat{A}_2) &= \text{var}\left(\frac{1}{3}(x[0] + x[1] + x[2])\right) \\ &= \frac{1}{9}(\text{var}(x[0]) + \text{var}(x[1]) + \text{var}(x[2])) \quad (\text{using hints}) \\ &= \frac{1}{9}(3\sigma^2) = \frac{\sigma^2}{3}. \end{aligned}$$

Thus, both estimators are unbiased but  $\text{var}(\hat{A}_2) < \text{var}(\hat{A}_1)$ . The estimates of the PDFs in [Figure 7.2](#) verifies the smaller variance, i.e., width, of the PDF for  $\hat{A}_2$ .

**7.3** We have the following results for 1000 trials:

$$\begin{aligned} b(A) &= 0.0054 && \text{for the sample mean} \\ &= 0.0110 && \text{for the median} \end{aligned}$$

$$\begin{aligned} \text{var}(\hat{A}) &= 0.0949 && \text{for the sample mean} \\ &= 0.1378 && \text{for the median} \end{aligned}$$

$$\begin{aligned} \text{mse}(\hat{A}) &= 0.0949 && \text{for the sample mean} \\ &= 0.1379 && \text{for the median.} \end{aligned}$$

Note that both MSEs are approximately the same as the variances since the squared-biases are nearly zero and are small relative to the variance. The MATLAB program `FSSP3exer7_3.m`, which is contained on the CD, produces these results.

**7.4** You should see two nearly identical ROCs since the SNR is the same in both cases, and they should match the curve shown in [Figure 7.4](#). The program `FSSP3exer7_4.m`, which is contained on the CD, produces these results.

**7.5** The computer simulation results should match the theoretical results. Note, however, that the run time will be very long, possibly hours. The program `FSSP3exer7_5.m`, which is contained on the CD, produces these results.

**7.6** The theoretical  $P_e$  is 0.0377 while the computer simulation result is 0.0376. The program `FSSP3exer7_6.m`, which is contained on the CD, produces these results.

**7.7** The upper bound is given by the theoretical NP value of  $P_{DNP} = 0.6241$  while the computer simulation result for the cube root detector is 0.6176. The degradation in performance is very minimal. The program `FSSP3exer7_7.m`, which is contained on the CD, produces these results.

**7.8** The results for the various values of  $N$  are

$$N = 100 \quad N\text{var}(\hat{r}_x[2]) = 17.1264$$

$$N = 200 \quad N\text{var}(\hat{r}_x[2]) = 17.3687$$

$$N = 300 \quad N\text{var}(\hat{r}_x[2]) = 17.4578$$

$$N = 400 \quad N\text{var}(\hat{r}_x[2]) = 17.4531$$

$$N = 500 \quad N\text{var}(\hat{r}_x[2]) = 17.4484$$

It is seen that about  $N = 300$  is sufficient (assuming an error of no more than  $0.17/N$ ) for the asymptotic results to hold. The program FSSP3exer7\_8.m, which is contained on the CD, produces these results.

**7.9** The absolute error will be less than 0.01 for  $0.025 \leq f_0 \leq 0.475$ , which is about  $1/(2N) \leq f_0 \leq 0.5 - 1/(2N)$ . The program FSSP3exer7\_9.m, which is contained on the CD, produces these results.

**7.10** For  $A$  greater than about 2.8 the new estimator, being biased, will produce a larger MSE. The program FSSP3exer7\_10.m, which is contained on the CD, produces these results.

# Chapter 8. Optimal Approaches Using the Big Theorems

## 8.1. Introduction

In this chapter we summarize the main theorems of mathematical statistics that are used to *derive* optimal statistical signal processing algorithms. They will be referred to in subsequent chapters as the theoretical underpinnings of the algorithms to be described. *These theorems all rely on knowledge of the probability density function (PDF) of the data.* As an example, for a signal in noise problem, such as encountered in the estimation of the amplitude of a signal embedded in noise, we will need to explicitly write down the PDF. To do so we will need to choose a model for the signal, as discussed in [Chapter 3](#), as well as a model for the noise, as discussed in [Chapter 4](#). For example, the DC level in WGN assumes that the signal is given by  $s[n] = A$ , where  $A$  is to be estimated, and the noise is assumed to be WGN with known variance  $\sigma^2$ . Then the observed data becomes

$$x[n] = A + w[n] \quad n = 0, 1, \dots, N - 1$$

and letting the data set be given by  $\mathbf{x} = [x[0] \ x[1] \dots x[N - 1]]^T$ , the PDF is

$$\begin{aligned} p(\mathbf{x}) &= \prod_{n=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x[n] - A)^2\right] \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2\right]. \end{aligned} \quad (8.1)$$

This is because each data sample is independent of all the others, and the signal  $A$  becomes the mean of each sample since we have assumed a deterministic signal. At this point we have the PDF of the data and can apply the appropriate theorem to determine the optimal estimator of  $A$ . As we have already discussed in [Chapter 1](#), the optimal estimator is the sample mean given by

$\hat{A} = (1/N) \sum_{n=0}^{N-1} x[n]$ . We will soon describe in some detail how this estimator was obtained. For now, note that knowledge of the PDF is essential in doing so. Without it, we cannot utilize any of the optimality theorems, but would have to resort to a suboptimally performing estimator. In summary, we have that

**signal/noise models + theorem = optimal algorithm.**

To actually determine the optimal algorithm requires one to apply the theorem to the particular PDF that describes the data. This will generally involve an analytical derivation. For some classes of problems, these derivations have already been carried out, principally for the *linear data model* as already discussed in [Chapter 3](#), and so the optimal algorithm is well known. Later in the chapter these linear model results will be described.

Before proceeding further some comments on the importance of the knowledge of the PDF are in order. For the DC level in WGN we observe a data set  $\{x[0], x[1], \dots, x[N - 1]\}$ , whose PDF is known except for the value of  $A$ . To remind us that the PDF depends on  $A$ , it is convenient to change the notation slightly from  $p(\mathbf{x})$  to  $p(\mathbf{x}; A)$ . The true value of  $A$  clearly influences the outcome of the data set. For example, if  $A = 5$  and  $\sigma^2 = 1$ , then we expect few, if any of the observed data samples to be outside the range  $5 \pm 3\sigma = 5 \pm 3 = [2, 8]$ . If we do, then inferring that  $A = 5$  would probably not be a good choice. The optimality theorems to be described rely on how probable the observed data values are, making use of probabilities computed from the known PDF. In effect, the theorems optimally infer parameter values, the truth of hypotheses, *etc.* from the data by comparing the frequency of occurrence of the data values to those calculated from the known PDF. Hence, *the PDF is the basis for any optimal algorithm*. In practice, the choice of signal and noise models to yield Gaussian PDFs is made for mathematical tractability.

In addition to providing the methodology for obtaining optimal algorithms, when they exist, the theorems provide:

1. insight into the workings of the algorithms. This insight can be extremely valuable for proposing new algorithms when the exact conditions for the theorems to be valid are violated, and so the theorems cannot be used.
2. upper bounds on performance, which as was previously mentioned in [Section 7.4](#), is useful for feasibility studies.
3. enhanced understanding through study of simple analytical examples (our archetypical example being the DC level in WGN).
4. the basis for extensions to the theory.

## 8.2. The Big Theorems

We now state and illustrate by example the big theorems. They are summarized in [Table 8.1](#). A full discussion is contained in [[Kay 1993](#), [1998](#)], and the reader

is referred there for the theoretical details as well as numerous examples. A particular case of importance, the linear model (see [Chapter 3](#)), allows easy application of these theorems to yield explicit results. Some of these results are summarized in [Section 8.3](#).

**Table 8.1: Summary of the big theorems.**

Problem	Assumptions	Performance metric	Theorem	Procedure
Parameter estimation	parameter constant	MVU	8.2.1	If exists, $\hat{\theta}$ satisfies $\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = I(\theta)(\hat{\theta} - \theta)$
Parameter estimation	outcome of random variable	MMSE	8.2.2	$\hat{\theta} = E[\theta   \mathbf{x}] = \frac{\int_{-\infty}^{\infty} \theta p(\mathbf{x} \theta) p(\theta) d\theta}{\int_{-\infty}^{\infty} p(\mathbf{x} \theta) p(\theta) d\theta}$
Detection	hypotheses fixed	Maximize $P_D$ , $P_{FA}$ fixed	8.2.3	decide $\mathcal{H}_1$ if $\frac{p(\mathbf{x}; \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} > \gamma_{NP}$
Classification	hypotheses random, $M = 2$	minimize $P_e$	8.2.4	decide $\mathcal{H}_1$ if $\frac{p(\mathbf{x}; \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} > \frac{P[\mathcal{H}_0]}{P[\mathcal{H}_1]}$
Classification	hypotheses random, any $M$	minimize $P_e$	8.2.4	decide $\mathcal{H}_k$ if $P[\mathcal{H}_k   \mathbf{x}] = \max_{i=0,1,\dots,M-1} P[\mathcal{H}_i   \mathbf{x}]$ or maximize $p(\mathbf{x}   \mathcal{H}_i) P[\mathcal{H}_i]$

### 8.2.1. Parameter Estimation

Consider estimation of a parameter  $\theta$  (in the DC level in WGN example,  $\theta = A$ ). The known PDF of the data is  $p(\mathbf{x}; \theta)$ . An optimal estimator  $\hat{\theta}$  is defined to be one that is *unbiased* ( $E[\hat{\theta}] = \theta$ , see also [Section 7.3.1](#)) and minimizes the variance ( $\text{var}(\hat{\theta})$  is minimum among all unbiased estimators). This estimator is called the *minimum variance unbiased* (MVU) estimator. In practice, an MVU estimator will have a variance given by the lower bound, termed the *Cramer-Rao lower bound* (CRLB). The CRLB is calculated as

$$\text{var}(\hat{\theta}) \geq \text{CRLB} = \frac{1}{E \left[ -\frac{\partial^2 \ln p(\mathbf{x}; \theta)}{\partial \theta^2} \right]} \quad (8.2)$$

and is satisfied with equality if for a given function of the data, which is  $\hat{\theta}$ , the relationship

$$\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = I(\theta)(\hat{\theta} - \theta) \quad (8.3)$$

holds for some  $I(\theta)$ , a positive function of  $\theta$ . In this case, the  $\hat{\theta}$  that appears in [\(8.3\)](#) is the MVU estimator and its variance is given by  $\text{var}(\hat{\theta}) = 1/I(\theta)$ . As a result, the optimal estimator and its minimum variance are found by inspection. See also [Appendix 8A](#) for some insights into the CRLB.

---

### Example 8.1 – DC level in WGN

We now answer the question posed in [Chapter 1](#) as to why the sample mean estimator is an optimal one. Since the PDF is given by [\(8.1\)](#), we have

$$\begin{aligned}\ln p(\mathbf{x}; A) &= \ln \left[ \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left( -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2 \right) \right] \\ &= -\frac{N}{2} \ln(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2\end{aligned}$$

and therefore

$$\begin{aligned}\frac{\partial \ln p(\mathbf{x}; A)}{\partial A} &= \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (x[n] - A) \\ &= \underbrace{\frac{N}{\sigma^2}}_{I(A)} \left( \underbrace{\frac{1}{N} \sum_{n=0}^{N-1} x[n]}_{\hat{A}} - A \right)\end{aligned}\tag{8.4}$$

which is of the form of [\(8.3\)](#) with

$$\begin{aligned}\hat{A} &= \frac{1}{N} \sum_{n=0}^{N-1} x[n] \\ \text{var}(\hat{A}) &= \frac{1}{I(A)} = \frac{\sigma^2}{N}.\end{aligned}$$

Therefore, we can conclude that the sample mean is the MVU estimator and has the minimum variance of  $\sigma^2/N$ .



### Exercise 8.1 – Evaluating the CRLB

Using [\(8.2\)](#) evaluate the CRLB. Do you get the same result for the variance?



This example is a special case of the linear model, which is discussed further

in [Section 8.3](#). Also, note that an alternative means of identifying  $\hat{A}$  is to set  $\partial \ln p(\mathbf{x}; A)/\partial A$  to zero and solve for  $A$ . Doing so we have from [\(8.4\)](#)

$$\frac{1}{\sigma^2} \sum_{n=0}^{N-1} (x[n] - A) = 0$$

which yields

$$\hat{A} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

as the MVU estimator. In general, we can find the MVU estimator  $\hat{\theta}$  (when it exists) by solving the equation

$$\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = 0 \quad (8.5)$$

for  $\theta$ . This says that the function  $\ln p(\mathbf{x}; \theta)$ , which is called the *log-likelihood* function since it is considered to be a function of  $\theta$  with  $\mathbf{x}$  considered fixed, must have a local minimum or maximum or inflection point at  $\hat{\theta}$ . If it turns out to be a global maximum, then the MVU estimator  $\hat{\theta}$  is the location of the point that maximizes  $\ln p(\mathbf{x}; \theta)$  over  $\theta$  or equivalently maximizes  $p(\mathbf{x}; \theta)$  over  $\theta$ . An estimator with this property is called the *maximum likelihood estimator* (MLE), and is one of the most important estimators since it can be used even when the MVU estimator *does not exist*. As a consequence, the MLE is used for almost all practical problems. We will have more to say about the MLE in [Section 8.5.1](#). See also [Appendix 8A](#).

---

## Exercise 8.2 – Signal amplitude estimation

Assume the data consists of a signal  $A s[n]$  that is known except for its amplitude  $A$  ( $-\infty < A < \infty$ ). If the signal is embedded in WGN, the data model becomes

$$x[n] = A s[n] + w[n] \quad n = 0, 1, \dots, N - 1.$$

Note that if  $s[n] = 1$ , we have our old friend the DC level in WGN. Find the optimal estimator of  $A$  and its variance by using [\(8.3\)](#). Also, show that it is unbiased. To do so note that the PDF is given by a slight modification of [\(8.1\)](#) as

$$p(\mathbf{x}; A) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - As[n])^2 \right].$$

We summarize the previous results in a theorem.

---



---

### Theorem 8.2.1 (Determination of the MVU estimator)

*The MVU estimator is the unbiased estimator ( $E[\hat{\theta}] = \theta$ ) that has the minimum variance. If it exists, then (in practice) the variance must attain the Cramer-Rao lower bound*

$$\text{var}(\hat{\theta}) = \frac{1}{I(\theta)} = \frac{1}{E \left[ -\frac{\partial^2 \ln p(\mathbf{x}; \theta)}{\partial \theta^2} \right]}$$

and can be found by identifying  $\hat{\theta}$  in the expression

$$\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = I(\theta)(\hat{\theta} - \theta)$$

or equivalently by solving

$$\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = 0$$

for  $\theta$ , the value obtained becoming  $\hat{\theta}$ .

---

Note that we have added the parenthetical remark “in practice” since it can be shown that there are MVU estimators that do not satisfy the CRLB. However, for most practical problems, they do not arise. Also, even if the CRLB is not satisfied, its evaluation is still a valuable guide to the potential improvement possible of a suboptimal estimator as illustrated by the next exercise.

---

### Exercise 8.3 – Using the CRLB to assess potential performance improvements

For the data model of [Exercise 8.2](#) let  $s[n] = (1/2)^n$  for  $n = 0, 1, 2$ . The unbiased estimator of  $A$ ,  $\hat{A} = x[0]$  is proposed. Determine its variance.

Next evaluate the CRLB for  $\hat{A}$ . How much potential improvement is possible with a better estimator?

---

The foregoing discussion on the method for optimal parameter estimation is based on the assumption that  $\theta$  is an unknown *constant*. This means that if the experiment for a DC level in WGN were to be repeated many times to generate multiple data sets, then the value of  $A$  would be the same in each data set (although of course the noise samples  $w[n]$  for  $n = 0, 1, \dots, N - 1$  would vary in a random manner from data set to data set). But what if nature were to change  $A$  for each experiment? How would this affect our choice of an estimator? As a concrete physical example, consider the problem of estimating the daytime temperature at exactly 12 PM (noon) for a given day. Also, in doing so, assume the temperature reading is subject to errors. Assuming the temperature is the same for a few minutes before noon to a few minutes afterwards, we could take multiple temperature readings at times 11:58, 11:59, 12:00, 12:01, and 12:02 and average them together in an effort to reduce the sensor errors. Presumably, nature will change the noon temperature slightly each day, but we know that, for example, in August in Rhode Island the temperature cannot be  $0^\circ$  F. It is reasonable then that even before we begin to take our readings, we acknowledge this by enforcing the temperature to be within certain limits. We can utilize this *prior knowledge* by *assuming* the noon temperature is the *outcome of a random variable with a certain PDF*. For example, we might assume that the temperature at noon is a Gaussian random variable with mean  $70^\circ$  F and standard deviation  $5^\circ$  F. In this way, by assuming a *prior PDF* we will weight the temperature readings near  $70^\circ$  F more heavily and therefore, hopefully produce a better estimate through our use of prior knowledge. To do so we need to assume a *distinctly different data model*, one in which the parameter of interest is the *outcome of a random variable*. Termed the *Bayesian approach*, it utilizes Bayes theorem to arrive at the optimal estimator. Our previous data modeling assumption of a constant but unknown temperature is termed *the classical or frequentist approach*.

Using the Bayesian approach, it is possible to estimate the *outcome* of a DC level in WGN if a prior PDF for  $A$  is available. The performance metric (criterion of optimality) is now defined to be the Bayesian mean square error (MSE) and is given by

$$\begin{aligned} \text{Bmse}(\hat{A}) &= E[(A - \hat{A})^2] \\ &= \int_{-\infty}^{\infty} (A - \hat{A})^2 p(\mathbf{x}, A) d\mathbf{x} dA. \end{aligned} \tag{8.6}$$

Note that the use of the term *Bayesian MSE* is meant to highlight the fact that the expected value in (8.6) is with respect to the *joint PDF* of  $\mathbf{x}$  and  $A$ , since  $A$  is now assumed to be a random variable. Using the concept of a conditional PDF

[[Kay 2006](#)], this joint PDF can be written as

$$p(\mathbf{x}, A) = p(\mathbf{x}|A)p(A). \quad (8.7)$$

It consists of  $p(\mathbf{x}|A)$ , the conditional PDF of the data *given* the value of  $A$ , and  $p(A)$ , the prior PDF of  $A$ . The Bayesian MSE can be shown to be minimized by choosing the mean of the *posterior PDF*  $p(A|\mathbf{x})$ , and is computed as follows.

$$\hat{A} = E[A|\mathbf{x}] \quad (\text{symbolism for mean of } p(A|\mathbf{x})) \quad (8.8)$$

$$= \int_{-\infty}^{\infty} Ap(A|\mathbf{x})dA \quad (\text{definition of mean})$$

$$= \int_{-\infty}^{\infty} A \frac{p(\mathbf{x}|A)p(A)}{p(\mathbf{x})} dA \quad (\text{using Bayes' rule})$$

$$= \int_{-\infty}^{\infty} A \frac{p(\mathbf{x}|A)p(A)}{\int_{-\infty}^{\infty} p(\mathbf{x}, A')dA'} dA \quad (\text{definition of marginal PDF})$$

$$= \frac{\int_{-\infty}^{\infty} Ap(\mathbf{x}|A)p(A)dA}{\int_{-\infty}^{\infty} p(\mathbf{x}|A')p(A')dA'}. \quad (\text{using (8.7)}) \quad (8.9)$$

Finding the Bayesian minimum mean square error (MMSE) estimator as given by (8.9) (also called the *conditional mean estimator*) requires knowledge of the prior PDF  $p(A)$  as well as some integrations. For multiple parameters the integrations become multidimensional, exacerbating the computational problem. Recent progress in the use of Monte Carlo methods for computation of the Bayesian MMSE estimator has produced more practical approaches to implementing this estimator [[O'Hagan and Forster 2004](#)]. An example of the Bayesian MMSE estimator is given next.

---

### Example 8.2 – Estimation of outcome of random DC level in WGN

Assume that the data set consists of  $x[n] = A + w[n]$  for  $n = 0, 1, \dots, N - 1$  but now the DC level is modeled as the outcome of  $A \sim \mathcal{N}(\mu_A, \sigma_A^2)$ . As usual,  $w[n]$  is WGN with known variance  $\sigma^2$ . Since  $A$  is now modeled as a random variable, we require the additional assumption that  $A$  and  $w[n]$  are independent. Then, it can be shown that the Bayesian MMSE estimator is [[Kay 1993](#), pg. 319]

$$\hat{A} = \alpha \bar{x} + (1 - \alpha)\mu_A \quad (8.10)$$

where  $\bar{x}$  is the sample mean and

$$\alpha = \frac{\sigma_A^2}{\sigma_A^2 + \sigma^2/N}.$$

The latter is a weighting factor ( $0 < \alpha < 1$ ) that weights the sample mean and the prior estimate of  $A$ , i.e.,  $\mu_A$  (before the data is observed). Clearly, the estimator cannot be implemented without knowledge of  $\mu_A$ ,  $\sigma_A^2$ , and  $\sigma^2$ . An explanation is given in [Appendix 8A](#) as to why  $\mu_A$  is called the prior estimate as well as some insights as to why  $E[A|\mathbf{x}]$  is the Bayesian MMSE estimator.

---



### Exercise 8.4 – Making sense of the random DC level estimator

Referring to [\(8.10\)](#) find the estimator and explain its significance under the following conditions:

1. We have perfect prior knowledge, which is imparted to the estimator by letting  $\sigma_A^2 = 0$ .
  2. We have no prior knowledge, which is imparted to the estimator by letting  $\sigma_A^2 \rightarrow \infty$ .
  3. The data record is very long so that  $N \rightarrow \infty$ .
- 



The Bayesian MMSE estimator is attractive in that it always exists; hence, the optimal estimator is known and its explicit form is known. We need only evaluate some integrals. The only downside is that we require an additional assumption, i.e., knowledge of the prior PDF of  $A$ , which requires us to have some knowledge about the possible values of  $A$ . And if we are wrong in our assumptions, then the estimator will not perform well—possibly poorer than the classical estimator. If we wish to play it safe by assuming  $\sigma_A^2 \rightarrow \infty$  to indicate no prior knowledge, then the Bayesian MMSE estimator and the classical sample mean estimator become one and the same. This, in itself, is somewhat comforting! Suffice it to say that the choice of either a classical estimator or a Bayesian estimator is a controversial one (see [[Efron 1986](#)] for arguments). Prior knowledge, when available, should always be used, pointing to a Bayesian approach, but when not, the classical approach should be used. We summarize the previous results in a theorem.

---

### Theorem 8.2.2 (Determination of the Bayesian MMSE estimator)

*The Bayesian MMSE estimator of a parameter  $\theta$ , assumed to be the outcome of a random variable with prior PDF  $p(\theta)$ , is found by computing the mean of the posterior PDF  $p(\theta|\mathbf{x})$  as*

$$\hat{\theta} = \frac{\int_{-\infty}^{\infty} \theta p(\mathbf{x}|\theta) p(\theta) d\theta}{\int_{-\infty}^{\infty} p(\mathbf{x}|\theta') p(\theta') d\theta'}. \quad (8.11)$$

*The minimum Bayesian MSE is*

$$Bmse(\hat{\theta}) = \int \text{var}(\theta|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \quad (8.12)$$

*where  $\text{var}(\theta|\mathbf{x})$  denotes the variance of the posterior PDF  $p(\theta|\mathbf{x})$  and the integration is over all values of  $\mathbf{x}$ .*

---



---



### Bayesian versus classical estimator performance comparisons

Since the Bayesian approach and the classical approach employ different assumptions about the unknown parameter, it makes no sense to compare their performance. Doing so leads to the problem of which model to choose for  $\theta$ , either as a constant, in the classical case, or as the outcome of a random variable, in the Bayesian case. The performance metrics, the minimum variance of all unbiased estimators in the classical case and the minimum mean square error in the Bayesian case, are fundamentally different and reflect the difference in modeling assumptions. Trying to compare the estimators is a case of comparing “apples to oranges”, as was cautioned against in [Chapter 7](#).




---

We have not discussed estimation of multiple parameters since the theory becomes more difficult, except in the case of the linear model (see [Section 8.3](#)). The extensions of the theorems can be found in [[Kay 1993](#), [1998](#)].

#### 8.2.2. Detection

The detection problem first introduced in [Section 2.3](#) is the problem of choosing between two competing hypotheses,  $H_1$  and  $H_0$ . Typically,  $H_1$  denotes the hypothesis of a signal embedded in noise, whereas  $H_0$  denotes the hypothesis of noise only. It is desired to choose between the hypotheses so as to maximize the probability of detection  $P_D$  while at the same time constraining the probability of

false alarm  $P_{FA}$  to some small value, say  $P_{FA} = \alpha$ . This means that we wish to decide  $H_1$  when it is false ( $H_0$  true) with a small probability  $\alpha$  and among those decision rules guaranteeing this constraint, choose the one that maximizes the probability of deciding  $H_1$  when it is true, which is  $P_D$ . As discussed in [Chapter 7](#) the criterion of maximizing  $P_D$  subject to a constraint on  $P_{FA}$  is called the Neyman-Pearson criterion. In [Section 7.3](#) we stated that for the detection of a DC level  $A > 0$  in WGN, the optimal decision rule is to decide a signal is present if

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \gamma$$

where  $\gamma$  is a threshold that is chosen to ensure  $P_{FA} = \alpha$ . This optimal detector follows from the Neyman-Pearson (NP) theorem.

The Neyman-Pearson theorem assumes that the PDFs under each hypothesis,  $p(\mathbf{x}; H_1)$  when  $H_1$  is true, and  $p(\mathbf{x}; H_0)$  when  $H_0$  is true, are *known*. Then, the optimal rule is to decide  $H_1$  is true (a signal is present) if

$$L(\mathbf{x}) = \frac{p(\mathbf{x}; H_1)}{p(\mathbf{x}; H_0)} > \gamma_{NP} \quad (8.13)$$

where  $\gamma_{NP}$  is chosen to satisfy the  $P_{FA} = \alpha$  constraint. Note that  $L(\mathbf{x})$  is called the *likelihood ratio* and measures how probable it is that  $H_1$  is true relative to the probability that  $H_0$  is true for the given observed data set  $\mathbf{x}$ . The set of  $\mathbf{x}$ 's that satisfy the inequality of [\(8.13\)](#) form a *decision region* whereby if the observed data falls within that region, then we decide that a signal is present, and if not, then we decide there is no signal present. This decision region is unaltered if we modify the inequality expression by performing any operation that does not change the values satisfying it. For example, multiplying both sides by a positive constant will not alter this region since the basic inequality is unchanged. Such is the case if we have the inequality  $x^2 + y^2 > 1$ , which can be replaced by  $2x^2 + 2y^2 > 2$ . Another such example follows as an exercise. We will have occasion to make use of this manipulation in simplifying the form of the detector.

---

### Exercise 8.5 – Monotonically increasing transformations

If  $x > 1$ , show that  $\ln(x) > \ln(1)$  and conversely by plotting  $\ln(x)$  versus  $x$  for  $x > 0$ . Is it true that if  $x > 1$ , then for any function  $g$  that  $g(x) > g(1)$ ? If

not, give an example of when it is not true.

The optimal detector for a DC level in WGN is obtained by utilizing the NP theorem as given next.

### Example 8.3 – DC level in WGN

Assume we have our usual model  $x[n] = A + w[n]$ , where  $A$  is unknown with  $A > 0$ , and  $w[n]$  is WGN with known variance  $\sigma^2$ . We wish to decide, based on the data set  $x[n]$  for  $n = 0, 1, \dots, N - 1$ , whether there is a signal present, which has a known level and is assumed to be positive ( $A > 0$ ), or not present ( $A = 0$ ). The required PDFs needed to compute the likelihood ratio are given by (8.1) as

$$p(\mathbf{x}; \mathcal{H}_1) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2 \right]$$

and by letting  $A = 0$  in (8.1)

$$p(\mathbf{x}; \mathcal{H}_0) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} x^2[n] \right]$$

so that from (8.13) the likelihood ratio and decision rule are

$$L(\mathbf{x}) = \frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - A)^2 \right]}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} x^2[n] \right]} > \gamma_{NP}. \quad (8.14)$$

Taking the natural logarithm of both sides, which according to [Exercise 8.5](#) will not change the inequality, we have

$$-\frac{1}{2\sigma^2} \left[ \sum_{n=0}^{N-1} (x[n] - A)^2 - \sum_{n=0}^{N-1} x^2[n] \right] > \ln \gamma_{NP}$$

which simplifies to

$$\frac{A}{\sigma^2} \sum_{n=0}^{N-1} x[n] - \frac{NA^2}{2\sigma^2} > \ln \gamma_{NP}. \quad (8.15)$$

Since we have assumed that  $A > 0$ , we can multiply both sides by  $\sigma^2/(NA)$  to obtain

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \underbrace{\frac{A}{2} + \frac{\sigma^2}{NA} \ln \gamma_{NP}}_{\gamma} \quad (8.16)$$

which is the result previously given in [Chapter 7](#). It can also be shown that the new threshold  $\gamma$  is given by [[Kay 1998](#), pg. 68]

$$\gamma = \sqrt{\frac{\sigma^2}{N}} Q^{-1}(P_{FA})$$

where the inverse  $Q$  function, i.e.,  $Q^{-1}$ , is the inverse function of

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right) dt.$$

The final optimal decision rule decides  $H_1$  if

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \sqrt{\frac{\sigma^2}{N}} Q^{-1}(P_{FA}).$$

◊

---

For this relatively simple example the detection performance has already been given in [Chapter 7](#). Also, it should be noted that the same detector results from using either (8.14) or (8.16) since these inequalities are equivalent. The form expressed in (8.16), however, lends more insight into the operation of the detector and furthermore, allows the threshold to be determined analytically. It is much more difficult to derive the PDF of  $L(\mathbf{x})$  than that of the sample mean *detection statistic*, a necessary step in finding the threshold. We now summarize these optimal detector results.

### **Theorem 8.2.3 (Determining the optimal detector – Neyman-Pearson)**

*To maximize the probability of detection  $P_D$  for a constrained false alarm probability  $P_{FA} = \alpha$ , the optimal detector decides a signal is present if*

$$L(\mathbf{x}) = \frac{p(\mathbf{x}; \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} > \gamma_{NP}.$$

*The threshold  $\gamma_{NP}$  is chosen to satisfy the  $P_{FA}$  constraint*

$$P_{FA} = P[L(\mathbf{x}) > \gamma_{NP}; H_0] = \alpha.$$

In order to implement the NP detector we require knowledge of the PDFs under  $H_0$  and  $H_1$ . If we relax this assumption ever so slightly by assuming that the value of  $A$  is unknown and can therefore, be positive or negative, then the detector cannot be realized, an example of which is given in the next exercise. In this case, no optimal detector exists. The only general results available for a signal that contains unknown parameters is for the linear model. This case will be described in [Section 8.3.2](#).

---

### Exercise 8.6 – Effect of imperfect signal knowledge

For the previous example consider the simple case of  $N = 1$  so that (8.15) becomes

$$\frac{A}{\sigma^2}x[0] - \frac{A^2}{2\sigma^2} > \ln \gamma_{NP}.$$

Letting  $\sigma^2 = 1$  and  $\gamma_{NP} = e$ , find all the values of  $x[0]$  that satisfy the inequality if  $A = 1$  (this is called the the *decision region* for deciding  $H_1$ ). Repeat for  $A = -1$ . What can you say about the need to know the value of  $A$  in applying the NP theorem?




---

### 8.2.3. Classification

In the detection problem our goal was to decide between two competing hypotheses, either  $H_1$  for a signal present, or  $H_0$  for noise only present. Essentially, the same problem occurs when we wish to classify a set of data into one of two possible classes. In digital communications (see also [Section 7.3.3](#)) the goal is to decide if a “0” or a “1” was transmitted. Therefore, there are again two competing hypotheses, but now the two possible data sets each contain a signal, although of course the signals will be different. In this example, an error in deciding a 1 if a 0 is actually transmitted or an error in deciding a 0 if a 1 is actually transmitted are both equally undesirable. Hence, it makes more sense to combine the errors into one overall performance metric. In contrast to this philosophy, in the detection problem we purposely constrained  $P_{FA}$ , which is the probability of deciding  $H_1$  when  $H_0$  is true, and maximized  $PD$ , which is equivalent to minimizing  $1 - PD$ . The latter is the probability of deciding  $H_0$  when  $H_1$  is true. Hence in the detection problem the two errors are considered separately, with special importance assigned to the false alarm error. (In radar,

for example, it would be most unwise to launch a missile in response to a perceived enemy aircraft, which later was noted to be a false alarm!) A more reasonable metric for the classification problem, as already described in [Chapter 7](#), is the *overall* probability of error  $P_e$ , which is defined as

$$P_e = P[\text{decide } \mathcal{H}_1 | \mathcal{H}_0 \text{ is true}]P[\mathcal{H}_0 \text{ is true}] + P[\text{decide } \mathcal{H}_0 | \mathcal{H}_1 \text{ is true}]P[\mathcal{H}_1 \text{ is true}]. \quad (8.17)$$

We will denote the *a priori* probabilities of each hypothesis occurring succinctly as  $P[H_0] = P[H_0 \text{ is true}]$  and  $P[H_1] = P[H_1 \text{ is true}] = 1 - P[H_0]$ .

---



## Performance metrics and assumptions for detection and classification

The reader should note the different assumptions concerning the hypotheses in detection and classification. In detection problems the NP criterion assumes that either  $H_0$  is *true* or  $H_1$  is *true*. This means that repeated trials of an experiment would fix one of the two hypotheses. And in fact, in [Chapter 7](#) for which a computer simulation was described to estimate  $P_{FA}$  and  $P_D$ , we considered two fixed conditions in generating the data. This is also reflected in the data PDF notation  $p(\mathbf{x}; H_0)$ , for example, in which  $H_0$  is assumed to be in effect. But for classification it is assumed that each hypothesis is chosen as the outcome of a random event and hence the need to assign a prior probability, such as  $P[H_0]$  for example. This is also reflected in expressing the data PDF as the *conditional PDF*  $p(\mathbf{x}|H_0)$ , for example. Also, in simulating the performance via a computer it was seen in [Chapter 7](#) that for each trial of the experiment either  $H_0$  was chosen or  $H_1$  was chosen according to their prior probabilities,  $P[H_0]$  and  $P[H_1]$ , respectively. The latter assumption in the classification problem also allowed the use of  $P_e$  as the optimality criterion, as expressed by [\(8.17\)](#). One should always keep in mind these assumptions and that they are *fundamentally different*. The distinction is analogous to that made between a classical estimator and a Bayesian estimator.




---

To minimize  $P_e$  the optimal decision rule is to decide  $H_1$  if

$$L(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{H}_1)}{p(\mathbf{x}|\mathcal{H}_0)} > \frac{P[\mathcal{H}_0]}{P[\mathcal{H}_1]} = \gamma_{MAP} \quad (8.18)$$

where the threshold is now explicitly given by the ratio of the a priori probabilities. The subscript on  $\gamma$  is meant to indicate *maximum a posteriori probability*, the reason for which will be described shortly. The decision rule is nearly identical to the NP detector except for the explicit threshold. As a special case, if  $P[H_0] = P[H_1] = 1/2$ , then the threshold is unity and we decide  $H_1$  if

$$p(\mathbf{x}|\mathcal{H}_1) > p(\mathbf{x}|\mathcal{H}_0) \quad (8.19)$$

or we choose the hypothesis with the larger value of the PDF. This is called the *maximum likelihood (ML) decision rule*. An example follows.

---

#### Example 8.4 – Two-level communication system

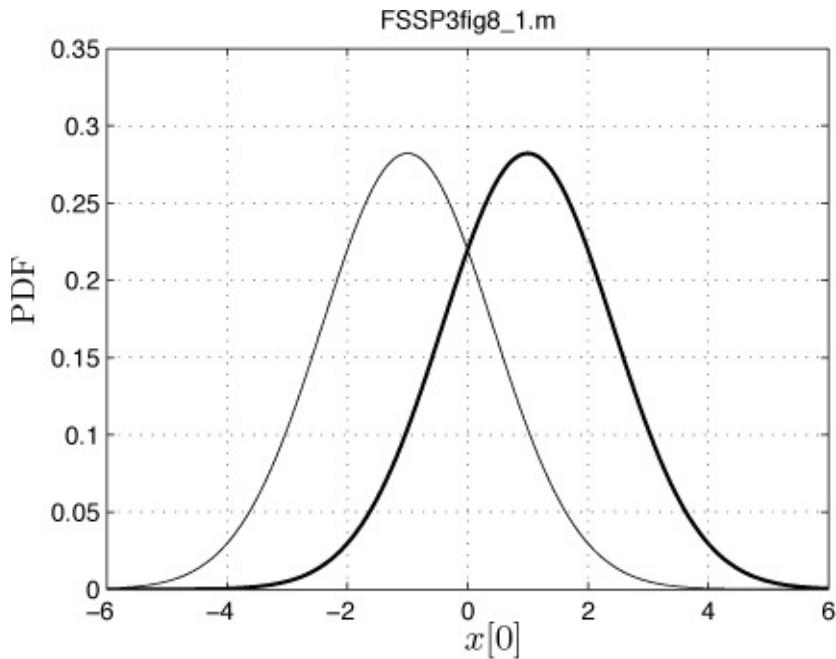
Assume that we receive the single sample  $x[0] = 1 + w[0]$  if a 1 is transmitted and  $x[0] = -1 + w[0]$  if a 0 is transmitted. As usual  $w[0]$  is a sample of WGN with variance  $\sigma^2$ . Each bit is assumed to have the same probability of being transmitted, meaning that  $P[H_0] = P[H_1] = 1/2$ . The optimal decision rule is the ML rule given by (8.19) and hence we decide  $H_1$  if

$$p(x[0]|H_1) > p(x[0]|H_0)$$

which is

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x[0] - 1)^2\right] > \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x[0] + 1)^2\right] \quad (8.20)$$

and  $H_0$  otherwise. Both PDFs are shown in [Figure 8.1](#). It is seen that the inequality is satisfied if  $x[0] > 0$ , which should make sense in light of the signals that were chosen to represent the two different bits, either a +1 for a 1 bit or a -1 for a 0 bit.



**Figure 8.1: Two PDFs for the received data sample  $x[0]$  for a DC level in WGN with variance  $\sigma^2 = 2$ . The heavy line indicates  $p(x[0]|H_1)$  and the lighter line indicates  $p(x[0]|H_0)$ .**

◊

### Exercise 8.7 – Two-level digital communication system

Prove that for the inequality of (8.20) to hold that  $x[0] > 0$ . Hint: Take the natural logarithm of both sides and simplify.

•

The decision rule in (8.18) can also be equivalently written by dividing both sides by the *unconditional* PDF (which is positive)

$$p(\mathbf{x}) = p(\mathbf{x}|H_0)P[H_0] + p(\mathbf{x}|H_1)P[H_1]$$

to yield

$$\frac{p(\mathbf{x}|H_1)P[H_1]}{p(\mathbf{x})} > \frac{p(\mathbf{x}|H_0)P[H_0]}{p(\mathbf{x})}.$$

But by a form of Bayes' rule this is equivalent to

$$P[\mathcal{H}_1|\mathbf{x}] > P[\mathcal{H}_0|\mathbf{x}]. \quad (8.21)$$

The probabilities  $P[H_1|x]$  and  $P[H_0|x]$  are called the a posteriori probabilities, and are the counterparts to the *a priori probabilities*  $P[H_1]$  and  $P[H_0]$ , respectively, except they summarize all the information known about the hypotheses *after the data is received*. In this form it is clear why this decision rule as given by (8.18) is called the *maximum a posteriori probability* or MAP rule.

It is of great practical importance that the MAP rule easily extends to the case of more than two hypotheses. If we have  $M$  hypotheses  $H_0, H_1, \dots, H_{M-1}$ , which is the case commonly encountered in classification, then to minimize  $P_e$  we should decide  $H_k$  if

$$P[H_k|x] > P[H_i|x] \quad \text{for all } i \neq k$$

where  $i = 0, 1, \dots, M - 1$ . In other words, we should choose the hypothesis with the maximum a posteriori probability. We now summarize these results.

---

#### **Theorem 8.2.4 (Determination of optimal classifier)**

An optimal classifier minimizes the probability of decision error  $P_e$ . To optimally decide among the  $M$  hypotheses  $H_0, H_1, \dots, H_{M-1}$  with a priori probabilities  $P[H_0], P[H_1], \dots, P[H_{M-1}]$  of occurring, where

$\sum_{i=0}^{M-1} P[\mathcal{H}_i] = 1$ , we should choose the hypothesis with the maximum a posteriori probability  $P[H_i|x]$ . This is termed the maximum a posteriori (MAP) rule. The probability is equivalently written as

$$P[\mathcal{H}_i|x] = \frac{p(\mathbf{x}|\mathcal{H}_i)P[\mathcal{H}_i]}{p(\mathbf{x})}$$

and so we can maximize  $p(\mathbf{x}|H_i)P[H_i]$  over  $i$ . If the hypotheses are equally probable, i.e.,  $P[H_i] = 1/M$  for all  $i$ , then we can equivalently maximize just  $p(\mathbf{x}|H_i)$  and this is termed the maximum likelihood (ML) rule.

For the special case of two hypotheses  $H_0$  and  $H_1$  we should decide  $H_1$  if

$$\frac{p(\mathbf{x}|\mathcal{H}_1)}{p(\mathbf{x}|\mathcal{H}_0)} > \frac{P[\mathcal{H}_0]}{P[\mathcal{H}_1]}$$

and  $H_0$  otherwise. If the hypotheses are equally probable, we should

choose the hypotheses with the larger  $p(\mathbf{x}|H_i)$ .

---

---

### Exercise 8.8 – Binary ( $M = 2$ ) classification

Consider the same problem as in [Example 8.4](#) but now let  $\sigma^2 = 1$  for simplicity and also assume we use *two samples* of +1 to represent a 1 bit and two samples of -1 to represent a 0 bit. The two hypotheses are equally probable with data PDFs

$$\begin{aligned} p(x[0], x[1]|\mathcal{H}_1) &= \frac{1}{2\pi} \exp \left[ -\frac{1}{2}(x[0] - 1)^2 - \frac{1}{2}(x[1] - 1)^2 \right] \\ p(x[0], x[1]|\mathcal{H}_0) &= \frac{1}{2\pi} \exp \left[ -\frac{1}{2}(x[0] + 1)^2 - \frac{1}{2}(x[1] + 1)^2 \right]. \end{aligned} \quad (8.22)$$

Determine the ML rule. Then, perform a computer simulation (see [Section 7.3.3](#)) to generate 100 outcomes of  $(x[0], x[1])$  and plot the outcomes as points in the  $x$ - $y$  plane. Use an “x” to represent the outcome if  $H_1$  was chosen for a particular trial and an “o” if  $H_0$  was chosen. Then, draw the decision boundary, which divides the plane into two parts with the points to be classified as  $H_0$  on one side and the points to be classified at  $H_1$  on the other side. Hint: You should be able to show that the decision boundary is a line.

•

---

## 8.3. Optimal Algorithms for the Linear Model

Recall from [Sections 3.4](#) and [3.5](#) that the linear model is given by

$$\mathbf{x} = \mathbf{H}\boldsymbol{\theta} + \mathbf{w} \quad (8.23)$$

where  $\mathbf{x} = [x[0] \ x[1] \dots, x[N - 1]]^T$  is the data vector,  $\mathbf{H}$  is a known  $N \times p$  observation matrix with  $N > p$ ,  $\boldsymbol{\theta}$  is a  $p \times 1$  vector of parameters, and  $\mathbf{w}$  is an  $N \times 1$  random vector whose elements are samples of WGN with known variance  $\sigma^2$ . The component  $\mathbf{H}\boldsymbol{\theta}$  represents a deterministic signal  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$  and typically the elements of  $\boldsymbol{\theta}$  are the amplitudes of the signal components. For example, such signals as polynomials (with the DC level being a special case), sinusoids, damped exponentials, and FM signals are all described by the linear model. Its utility is that the PDF is easily written down and manipulated so that the big theorems can be applied to yield actual algorithms. Furthermore, it easily

generalizes our previous case of a single unknown parameter to multiple parameters,  $\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \dots \ \theta_p]^T$ .

The PDF of the linear model is a special case of the multivariate Gaussian PDF, for which a multitude of analytical properties and results have been obtained. See, for example, [[Graybill 1976](#)] for a comprehensive description. The exact form of the PDF will depend upon whether it is more appropriate to model the signal as deterministic or as the outcome of a random event (see [Chapter 3](#)). Hence, in describing the algorithms obtainable with the linear model we will need to consider these two cases separately. The first case, termed the *classical linear model*, assumes that  $\boldsymbol{\theta}$  is a deterministic parameter, and the second case, termed the *Bayesian linear model*, assumes that  $\boldsymbol{\theta}$  is the outcome of a random vector. In the latter case, we need to assign a prior PDF to  $\boldsymbol{\theta}$  and the one used is the multivariate Gaussian PDF with mean vector  $\boldsymbol{\mu}_{\boldsymbol{\theta}}$  and covariance matrix  $\mathbf{C}_{\boldsymbol{\theta}}$ . Additionally, it is assumed that  $\boldsymbol{\theta}$  and  $\mathbf{w}$  are independent.

We next summarize the algorithms for each model. Further details are provided in [[Kay 1993, 1998](#)].

### 8.3.1. Parameter Estimation

For the classical linear model the PDF of  $\mathbf{x}$  is

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\theta}) \right]. \quad (8.24)$$

The CRLB for  $\boldsymbol{\theta}$  can be shown to be satisfied and therefore the MVU estimator is given by

$$\hat{\boldsymbol{\theta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}. \quad (8.25)$$

As a side note, the form of the estimator given by (8.25) is identical to the least squares estimator obtained by minimizing  $J(\boldsymbol{\theta}) = (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})$  over  $\boldsymbol{\theta}$  as was shown by example in [Section 3.5](#). This follows from the form of the linear model PDF given in (8.24). Hence, in the case of the classical linear model the *least squares estimator is optimal* but not otherwise. Additionally, the variance of each element of  $\hat{\boldsymbol{\theta}}$  is

$$\text{var}(\hat{\theta}_i) = \sigma^2 [(\mathbf{H}^T \mathbf{H})^{-1}]_{ii} \quad i = 1, 2, \dots, p$$

where  $[\mathbf{A}]_{ii}$  denotes the  $[i, i]$  element of  $\mathbf{A}$ . Examples of the estimators obtained for the classical linear model have been given in [Section 3.5](#).

For the Bayesian linear model the posterior PDF is given by

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{1}{(2\pi)^{N/2} \det^{1/2}(\mathbf{C}_{\theta|x})} \exp \left[ -\frac{1}{2} (\boldsymbol{\theta} - E[\boldsymbol{\theta}|\mathbf{x}])^T \mathbf{C}_{\theta|x}^{-1} (\boldsymbol{\theta} - E[\boldsymbol{\theta}|\mathbf{x}]) \right] \quad (8.26)$$

where the mean is

$$E[\boldsymbol{\theta}|\mathbf{x}] = \boldsymbol{\mu}_\theta + \left( \mathbf{C}_\theta^{-1} + \frac{1}{\sigma^2} \mathbf{H}^T \mathbf{H} \right)^{-1} \frac{\mathbf{H}^T}{\sigma^2} (\mathbf{x} - \mathbf{H}\boldsymbol{\mu}_\theta) \quad (8.27)$$

and the covariance matrix is

$$\mathbf{C}_{\theta|x} = \left( \mathbf{C}_\theta^{-1} + \frac{1}{\sigma^2} \mathbf{H}^T \mathbf{H} \right)^{-1}.$$

The MMSE estimator, which is given by the mean of the posterior PDF, is  
 $\hat{\boldsymbol{\theta}} = E[\boldsymbol{\theta}|\mathbf{x}]$

---

### Exercise 8.9 – MMSE estimator for no prior information

If there is no prior knowledge about  $\boldsymbol{\theta}$ , which can be expressed by letting  $\mathbf{C}_\theta = \sigma_\theta^2 \mathbf{I}$ , where  $\sigma_\theta^2 \rightarrow \infty$ , or equivalently  $\mathbf{C}_\theta^{-1} = \mathbf{0}$ , what does the MMSE estimator given by (8.27) reduce to?




---

An example follows.

---

### Example 8.5 – Random DC level in WGN

As a special case of the Bayesian linear model we let  $\boldsymbol{\theta} = A$  ( $p = 1$ ),  $\mathbf{H} = [1 \ 1 \dots 1]^T = \mathbf{1}^T$ , an  $N \times 1$  vector,  $\boldsymbol{\mu}_\theta = \mu_A$ , and  $\mathbf{C}_\theta = \sigma_A^2 \mathbf{I}$ . This is referred to as the random DC level in WGN, already encountered in [Example 8.2](#). Using (8.27) we have

$$\begin{aligned} \hat{A} &= E[A|\mathbf{x}] \\ &= \mu_A + \left( \frac{1}{\sigma_A^2} + \frac{1}{\sigma^2} \mathbf{1}^T \mathbf{1} \right)^{-1} \frac{\mathbf{1}^T}{\sigma^2} (\mathbf{x} - \mathbf{1}\mu_A) \\ &= \mu_A + \frac{N/\sigma^2}{1/\sigma_A^2 + N/\sigma^2} \frac{1}{N} (\mathbf{1}^T \mathbf{x} - \mu_A \mathbf{1}^T \mathbf{1}) \\ &= \mu_A + \frac{\sigma_A^2}{\sigma_A^2 + \sigma^2/N} (\bar{x} - \mu_A) \end{aligned}$$

which can also be written as in (8.10). This expression for  $\hat{A}$  has the

*predictor-corrector* form, with the first term the predictor due to a priori knowledge, and the second term the correction due to the data knowledge.



### 8.3.2. Detection

Here again we wish to distinguish between two possible cases. First consider the classical linear model. For detection we wish to decide if a known signal  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$  is present, in which case  $\boldsymbol{\theta} \neq \mathbf{0}$ , or if noise only is present, for which  $\boldsymbol{\theta} = \mathbf{0}$ . Since the PDFs are given by (8.24), we can easily find the NP detector by computing the likelihood ratio

$$L(\mathbf{x}) = \frac{p(\mathbf{x}; \boldsymbol{\theta})}{p(\mathbf{x}; \mathbf{0})}.$$

It can be shown that this reduces to the decision rule that decides a signal is present if

$$\mathbf{x}^T \mathbf{s} = \sum_{n=0}^{N-1} x[n] s[n] > \sqrt{\sigma^2 \sum_{n=0}^{N-1} s^2[n]} Q^{-1}(P_{FA}) \quad (8.28)$$

where the signal vector  $\mathbf{s}$  is given as  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ . The detection performance is given by

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{\sum_{n=0}^{N-1} s^2[n]}{\sigma^2}} \right). \quad (8.29)$$

#### Example 8.6 – DC level in WGN

For  $s[n] = A > 0$ , we have from (8.28) that

$$\sum_{n=0}^{N-1} x[n] A > \sqrt{NA^2\sigma^2} Q^{-1}(P_{FA}).$$

Noting that  $A > 0$  and multiplying both sides by  $1/(NA)$  yields

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > \sqrt{\frac{\sigma^2}{N}} Q^{-1}(P_{FA})$$

which agrees with our results in [Example 8.3](#). The detection performance is from (8.29)  $P_D = Q(Q^{-1}(P_{FA}) - \sqrt{(NA^2)/\sigma^2})$ .



---

For the Bayesian linear model the optimal detector can be shown to decide  $H_1$  if

$$\mathbf{x}^T \mathbf{H} \mathbf{C}_\theta \mathbf{H}^T (\mathbf{H} \mathbf{C}_\theta \mathbf{H}^T + \sigma^2 \mathbf{I})^{-1} \mathbf{x} > \gamma \quad (8.30)$$

assuming that  $\mu_\theta = \mathbf{0}$ , which is a common assumption. The determination of the threshold  $\gamma$  and also the detection performance is much more complicated than for the classical linear model. This is because the test statistic is a *quadratic* function of the data  $\mathbf{x}$ . In contrast, the test statistic for the classical linear model is linear in  $\mathbf{x}$  and so the PDF of the detection statistic is Gaussian. The reader is referred to [Kay 1998, pg. 154] for some analytical results.

### 8.3.3. Classification

We next consider only the classical linear model with equal prior probabilities due to its predominant use in practice. According to the ML rule, the optimal classifier maximizes  $p(\mathbf{x}|H_i)$ . Consider  $M$  possible hypotheses

$$H_i : \mathbf{x} = \mathbf{H}_i \boldsymbol{\theta}_i + \mathbf{w} \quad i = 0, 1, \dots, M - 1.$$

The signal is assumed to be different under each hypothesis, either in amplitude ( $\boldsymbol{\theta}_i$ ) and/or form ( $\mathbf{H}_i$ ). An example is given next.

---

#### Example 8.7 – Binary hypotheses

Consider [Exercise 8.8](#) in which we have two hypotheses so that  $M = 2$ , and also in which we have access to two samples so that  $N = 2$ . We can write the two possible signals as

$$\mathbf{s}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{s}_0 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

so that

$$\mathbf{s}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} 1 \quad \mathbf{s}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} (-1).$$

Then, we have that  $\mathbf{H}_0 = \mathbf{H}_1 = [1 \ 1]^T$ ,  $\boldsymbol{\theta}_1 = \theta_1 = 1$ , and  $\boldsymbol{\theta}_0 = \theta_0 = -1$  and the classical linear model applies to this problem. The general ML rule is to decide  $H_k$  if

$$(\mathbf{x} - \mathbf{H}_i \boldsymbol{\theta}_i)^T (\mathbf{x} - \mathbf{H}_i \boldsymbol{\theta}_i)$$

is minimum for all  $i \neq k$  since this equivalently maximizes the PDF (see (8.24)). Since  $\mathbf{s}_i = \mathbf{H}_i \boldsymbol{\theta}_i$ , we are computing  $(\mathbf{x} - \mathbf{s}_i)^T (\mathbf{x} - \mathbf{s}_i)$ , which is the squared distance from each signal vector to the data vector. Because of this, the decision rule is called a *minimum distance classifier*. For the example encountered in [Exercise 8.8](#) we have that this squared distance is

$$\begin{aligned} (\mathbf{x} - \mathbf{H}_i \boldsymbol{\theta}_i)^T (\mathbf{x} - \mathbf{H}_i \boldsymbol{\theta}_i) &= \left( \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \theta_i \right)^T \left( \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} \theta_i \right) \\ &= (x[0] - \theta_i)^2 + (x[1] - \theta_i)^2 \\ &= \begin{cases} (x[0] - 1)^2 + (x[1] - 1)^2 & i = 1 \\ (x[0] + 1)^2 + (x[1] + 1)^2 & i = 0 \end{cases} \end{aligned}$$

since  $\theta_1 = 1$  and  $\theta_0 = -1$ . Choosing the minimum of these two values is the same as choosing the maximum of the PDFs in (8.22). Note that we choose  $H_1$  if  $x[1] > -x[0]$  and  $H_0$  otherwise (see also the solution to [Exercise 8.8](#).)



## 8.4. Using the Theorems to Derive a New Result

The aforementioned theorems provide the approach to be followed if an optimal signal processing algorithm is to be found for a particular problem. Often, problems that are encountered in practice depart somewhat from results published in the open literature. In such a case one can either guess at a possible solution, and then compare its performance against some benchmark algorithm's performance or "start from first principles" *to derive* the optimal algorithm for the problem of interest. A derivation at times can seem to be a daunting task, but at least we have the guidance provided by the big theorems. To illustrate this approach we consider a binary classification problem for which the solution is not obvious. However, by applying the ML rule, the optimal algorithm is easily found. Furthermore, examination of the solution provides much insight into the operation of the algorithm and suggests extensions that might be needed for more complicated problems.

We consider the problem of classifying a data set into one of two classes. Each class will represent a distinctly different type of data. For class 1 the PDF is

$$p(\mathbf{x}|\mathcal{H}_0) = \frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 1)^2 \right] \quad (8.31)$$

which is nothing more than  $N$  independent samples of a Gaussian random variable with mean of one and a variance of one (a DC level of level  $A = 1$  in WGN with variance  $\sigma^2 = 1$ ). The second class has a Gaussian mixture PDF (see also [Section 4.6](#))

$$p(\mathbf{x}|\mathcal{H}_1) =$$

$$\frac{1}{2} \frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 1)^2 \right] + \frac{1}{2} \frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 5)^2 \right]. \quad (8.32)$$

This PDF may be thought of as one whose underlying random mechanism generates  $N$  samples, which have a mean of 1 half of the time, and  $N$  samples, which have a mean of 5, the other half of the time. Clearly, this is a problem that is not commonly encountered in the literature. Using the ML rule, however, it is relatively easy to arrive at the optimal decision rule. Using the PDFs given by (8.31) and (8.32), the ML rule decides  $H_1$  if

$$\frac{\frac{1}{2} \frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 1)^2 \right] + \frac{1}{2} \frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 5)^2 \right]}{\frac{1}{(2\pi)^{N/2}} \exp \left[ -\frac{1}{2} \sum_{n=0}^{N-1} (x[n] - 1)^2 \right]} > 1.$$

This reduces to

$$\frac{1}{2} + \frac{1}{2} \exp \left[ -\frac{1}{2} \left( \sum_{n=0}^{N-1} (x[n] - 5)^2 - \sum_{n=0}^{N-1} (x[n] - 1)^2 \right) \right] > 1.$$

By simplifying and then taking the natural logarithm of both sides, we have

$$\begin{aligned} -\frac{1}{2} \left( \sum_{n=0}^{N-1} (x[n] - 5)^2 - \sum_{n=0}^{N-1} (x[n] - 1)^2 \right) &> 0 \\ -\frac{1}{2} \sum_{n=0}^{N-1} (-10x[n] + 25 + 2x[n] - 1) &> 0 \end{aligned}$$

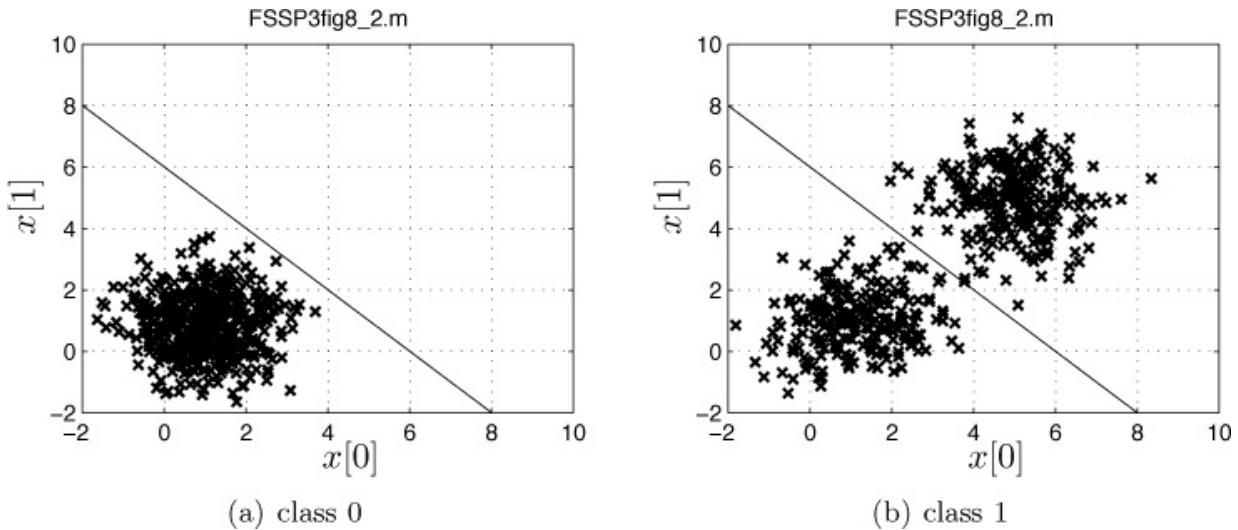
or finally we decide  $H_1$  if

$$\frac{1}{N} \sum_{n=0}^{N-1} x[n] > 3.$$

In retrospect the solution admits a very intuitive explanation. When  $H_0$  is true the sample mean has an expected value of 1, but when  $H_1$  is true the sample

mean will have an expected value of 1 for half of the time and an expected value of 5 for the other half. Hence, if  $H_1$  is true, we expect the sample mean to exceed 1 half of the time. The threshold of 3 is midway between these two possible expected values of the sample mean. We also might anticipate that the result would change if the prior probabilities were not equal, with the threshold of 3 changing. This simple example should serve to illustrate the advantage of applying the optimality theorems whenever possible. Guessing at a solution seldom produces good results!

To add to our intuition we can also perform a computer simulation. This has the advantage of validating our previous analysis, in case there is a logical or algebraic error present. The results are shown in [Figure 8.2](#) for  $N = 2$  and 500 outcomes of each class. Also, shown is the decision boundary, which is  $(x[0] + x[1])/2 > 3$ . It is also interesting to speculate that because of the form of the PDFs, it may not be possible to drive  $P_e$  to zero, even if the data record length  $N$  is increased. The reader may wish to determine if this is so and also what  $P_e$  is for the case shown in [Figure 8.2](#).



**Figure 8.2: 500 Outcomes of  $(x[0], x[1])$  for the two classes. The decision boundary, which is a line, is also shown.**

## 8.5. Practically Optimal Approaches

We have discussed the theorems that provide optimal approaches to the design of signal processing algorithms. Unfortunately, in practice the assumptions required to apply them may not be satisfied. Such is the case in parameter estimation for which an MVU estimator does not exist. Thus, we may not know

how to proceed to find a good estimator. A similar situation arises in detection and classification, when the required PDFs are not completely known. For example, we may wish to detect a sinusoid  $\cos(2\pi f_0 n)$  in WGN, but the frequency  $f_0$  may not be known. As a result, the PDF under  $H_1$  cannot be specified exactly, invalidating the use of the NP theorem. We next briefly describe some general approaches to these problems *that work well in practice*, although there may exist other approaches that produce better performing algorithms. In the experience of many researchers and practitioners, however, the approaches to be described are *practically optimal*. Additionally, there is considerable justification for using these approaches in that they can be shown to be asymptotically (as  $N \rightarrow \infty$  and/or the SNR becomes large) optimal. Of course, asymptotic optimality is no assurance that the algorithm will perform well for finite and sometimes very short data records, and/or low SNRs.

### 8.5.1. Parameter Estimation: The MLE

Consider the problem of estimating a single deterministic parameter, i.e., a constant. It was shown in [Section 8.2.1](#) that if an MVU estimator exists that attains the CRLB, then it is given by the location of the maximum of  $p(\mathbf{x}; \theta)$ . The latter is considered to be a function of  $\theta$  since the observed data samples  $\mathbf{x}$  are fixed. Such a function, viewed in this light, is termed the *likelihood function*. Hence, as we noted earlier the estimate obtained by maximizing the likelihood function is termed the *maximum likelihood estimator* (MLE). It easily extends to multiple parameter estimation so that if we wish to estimate  $\boldsymbol{\theta} = [\theta_1 \theta_2 \dots \theta_p]^T$ , then we need only maximize  $p(\mathbf{x}; \boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$ . The MLE is said to be a “turn the crank” procedure, with the only issue being a means for carrying out the maximization.

It can be shown that for most practical problems of interest, the MLE will attain the CRLB asymptotically. By asymptotically we mean for a large data record or a high SNR or both. Hence, the MLE is *practically optimal*. An example follows to illustrate the high SNR property.

---

#### Example 8.8 – Sinusoidal frequency estimation

Consider a sinusoidal signal  $s[n] = \cos(2\pi f_0 n)$  for  $n = 0, 1, \dots, N - 1$ , where  $N = 20$ , embedded in WGN with variance  $\sigma^2$ . We wish to estimate the frequency, where  $0 < f_0 < 1/2$ . The CRLB for  $\hat{f}_0$  can be shown to be [[Kay 1993](#), pg. 57]

$$\text{var}(\hat{f}_0) \geq \frac{\sigma^2}{4\pi^2 \sum_{n=0}^{N-1} n^2 \sin^2(2\pi f_0 n)}. \quad (8.33)$$

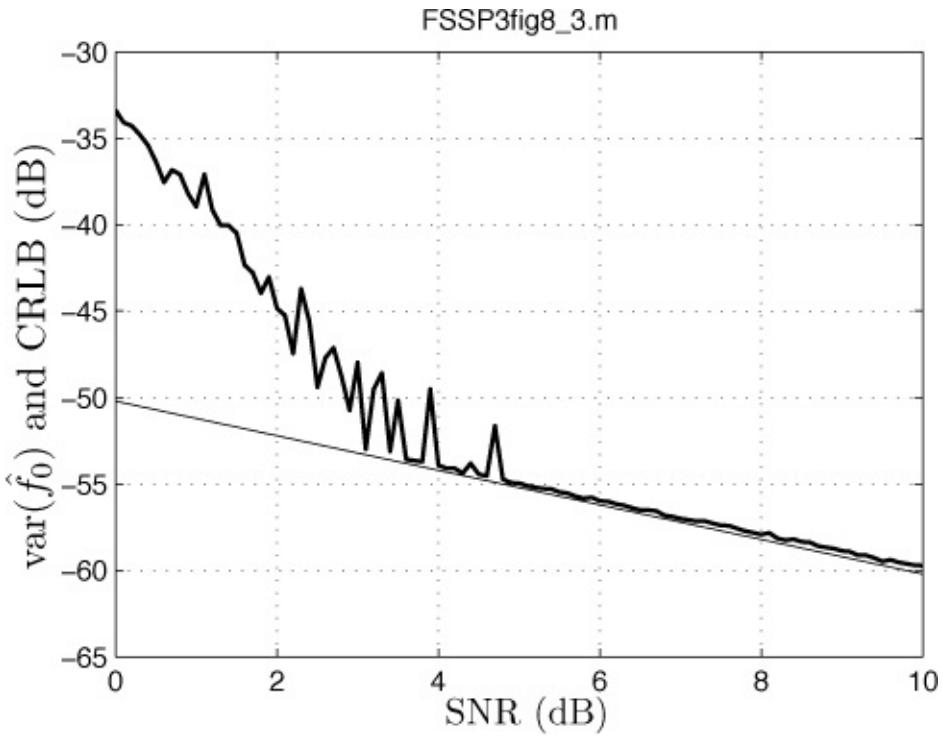
The MLE of  $f_0$  is found by maximizing the likelihood function

$$p(\mathbf{x}; f_0) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp \left[ -\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - \cos(2\pi f_0 n))^2 \right]$$

over  $0 < f_0 < 1/2$  or equivalently by minimizing

$$J(f_0) = \sum_{n=0}^{N-1} (x[n] - \cos(2\pi f_0 n))^2$$

which is a very nonlinear function of  $f_0$ . To do so we evaluate  $J(f_0)$  for a densely spaced grid of frequency points and then choose the frequency point yielding the maximum value (see also [Section 5.4](#)). Note that an analytical expression cannot be found for the location of the maximum. Using a computer simulation we have generated 10,000 data sets, estimated the frequency for each data set, and then found the variance of the estimator. Repeating this entire procedure for different SNRs, we generate a curve of estimator variance versus SNR. It is convenient to plot this as  $10 \log_{10} \text{var}(\hat{f}_0)$  versus the SNR in dB. This is because by also plotting the CRLB in this way for comparison, the CRLB becomes linear in SNR due to its form given by (8.33) ( $\text{SNR} = 1/(2\sigma^2)$ ). The results are shown in [Figure 8.3](#). As expected for high enough SNR, the CRLB is attained. Hence, above an SNR of about 5 dB, the MLE can be said to be optimal. (The MLE can also be shown to be unbiased above this threshold SNR.) Similar results are observed for large enough data records.



**Figure 8.3: Variance of the MLE of sinusoidal frequency and the CRLB (shown as the straight light line).**



### Computing the MLE via a grid search

In determining the minimum of  $J(f_0)$  for the previous example, it was necessary to compute the function at frequency points  $f_0 = 0.001, 0.002, \dots, 0.499$ . If the spacing is too wide, then there will be a frequency “quantization error” and the performance of the MLE will not attain the CRLB at the higher SNRs. Consider what would happen if the frequency spacing was only 0.01, and the true frequency was midway between two of these grid points. Then, even without noise, the maximum absolute error would be 0.01/2 in frequency. Thus, this error when squared to compute the variance would become  $10 \log_{10} 0.005^2 = -46$  dB. This error, being larger than the error due to noise of -56 dB at an SNR of about 6 dB, for example, would dominate, and the CRLB would not be attained. The choice of frequency spacing of 0.001 produces a quantization error of  $10 \log_{10} 0.0005^2 = -66$  dB and so is negligible compared to the noise error, since the minimum noise error causes the variance to be -60 dB at

the maximum SNR shown in [Figure 8.3](#) of 10 dB. Hence, as the SNR increases we will need to take a finer and finer grid of frequency points to attain the CRLB. We should choose in linear quantities

$(\Delta f/2)^2 < \text{CRLB}$ , where  $\Delta f$  is the frequency spacing. Note that it is even possible to have computer simulation results that appear to show an estimator variance being less than the CRLB. This occurs when the spacing of the frequency points is too wide, and also the true frequency is at one of these grid points. Then, once the SNR is high enough so that the variability of the likelihood peak location is very small, we will always choose the same grid point. Assume that the chosen grid point is the true frequency (although this is not the true maximum of the likelihood function, which is located between two grid points). Then, by choosing the same grid point, we will fortuitously (or not) obtain the true frequency without error! The adage “if it is too good to be true, then it probably isn’t” applies here.



### 8.5.2. Detection

When there are unknown parameters under either  $H_0$  or  $H_1$  or both, the NP approach cannot be implemented. An alternative method replaces the unknown parameters by their MLEs and then computes the likelihood ratio as before. This method is called the *generalized likelihood ratio test* (GLRT). To describe the method, explicitly assume that under  $H_1$  there are unknown parameters  $\theta_1$ , and under  $H_0$  there are unknown parameters  $\theta_0$ . The GLRT decides  $H_1$  if

$$L_{GLRT}(\mathbf{x}) = \frac{p(\mathbf{x}; \hat{\theta}_1, \mathcal{H}_1)}{p(\mathbf{x}; \hat{\theta}_0, \mathcal{H}_0)} > \gamma_{GLRT}. \quad (8.34)$$

It is important to observe that the MLEs required are different, with each one tailored to the hypothesis under consideration. Also, note that by definition of the MLE,  $\hat{\theta}_1$  is the value of  $\theta_1$  that maximizes  $p(\mathbf{x}; \theta_1, H_1)$  and  $\hat{\theta}_0$  is the value of  $\theta_0$  that maximizes  $p(\mathbf{x}; \theta_0, H_0)$ . The threshold  $\gamma_{GLRT}$  is chosen to constrain  $P_{FA}$  and is found by solving the equation

$$P_{FA} = P [L_{GLRT}(\mathbf{x}) > \gamma_{GLRT}; H_0] = \alpha$$

which is usually not an easy problem. Some general results that allow the

threshold to be determined analytically and the detection performance to be evaluated analytically are available for the linear model. In this case, the PDF under  $H_0$  is assumed known, and the PDF under  $H_1$  is known except for the classical linear model signal parameters, i.e., the  $\theta$  in  $\mathbf{H}\theta$ . The reader is referred to [[Kay 1993](#), pg. 274] for this important theorem. A further extension is available for the case when  $\sigma^2$  is also unknown in the linear model [[Kay 1998](#), pg. 345]. The detection performance of the GLRT will always be poorer than that of the NP detector, which assumes perfect knowledge of  $\theta_0$  and  $\theta_1$ , but in most practical cases, not significantly so. Typically, the degradation in performance is only 1 to 2 dB in SNR, meaning that the GLRT detector will require 1 to 2 dB more signal power to attain the same performance as the upper bound given by the NP detector.

### 8.5.3. Classification

For classification using PDFs with unknown parameters the situation described for detection only worsens. In practice, there does not appear to be any clear winner. Numerous books are devoted to this problem, and it remains an active area of research. Therefore, we do not address this problem here but refer the reader to the many excellent books describing various approaches [[Duda et al. 2001](#), and [Webb 2002](#)].

## 8.6. Lessons Learned

- An optimally performing algorithm can be obtained by applying one of the big theorems to data described by an analytical signal and noise model.
- Perfect knowledge of the PDF is necessary to apply one of the big theorems.
- When evaluating an estimator that is not optimal, always compute the CRLB to determine how much performance improvement is possible.
- When accurate prior knowledge about a parameter is available, use the Bayesian MMSE estimator, but if not use the MLE.
- Don't compare the performance of the Bayesian MMSE estimator and the MLE. Their underlying assumptions are different.
- In deriving detectors from the likelihood ratio, reduce the inequality to as simple a form as possible.
- The metrics for detection: the probability of detection, and for classification: the probability of error employ different assumptions on the prior probabilities of the hypotheses. Hence, they cannot be compared.

- For ease of modeling and implementation always first consider the linear model, either the classical or Bayesian version.
- The least squares estimator is the optimal parameter estimator for the linear model, but not otherwise.
- When possible, always try to solve a problem by appealing to “first principles”, meaning that you should start with an accurate but mathematically tractable model, and derive the algorithm based on one of the big theorems. The derived algorithm will have superior performance to any “guessed” solution.
- For parameter estimation the MLE is the most important estimator in practice, and for detection the GLRT is the most important approach. Both can be said to be asymptotically optimal (for large data records and/or SNRs).
- In computing the MLE via a grid search of the likelihood function, it is necessary to choose a grid spacing of points to ensure that the grid quantization error is much smaller than the estimation error due to noise.

## References

- Duda, R.O., P.E. Hart, D.G. Stark, *Pattern Classification, Second Ed.*, J. Wiley, NY, 2001.
- Efron, B., “Why Isn’t Everyone a Bayesian?” and subsequent comment papers, *American Statistician*, p. 1–5, 1986.
- Graybill, F., *Theory and Application of the Linear Model*, Duxbury, MA, 1976.
- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory*, Vol. I, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory*, Vol. II, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- O’Hagan, A., J. Forster, *Kendall’s Advanced Theory of Statistics*, Vol. 2B, *Bayesian Inference*, Oxford Univ. Press, NY, 2004.
- Webb, A. *Statistical Pattern Recognition*, Second Ed., J. Wiley, NY, 2002.

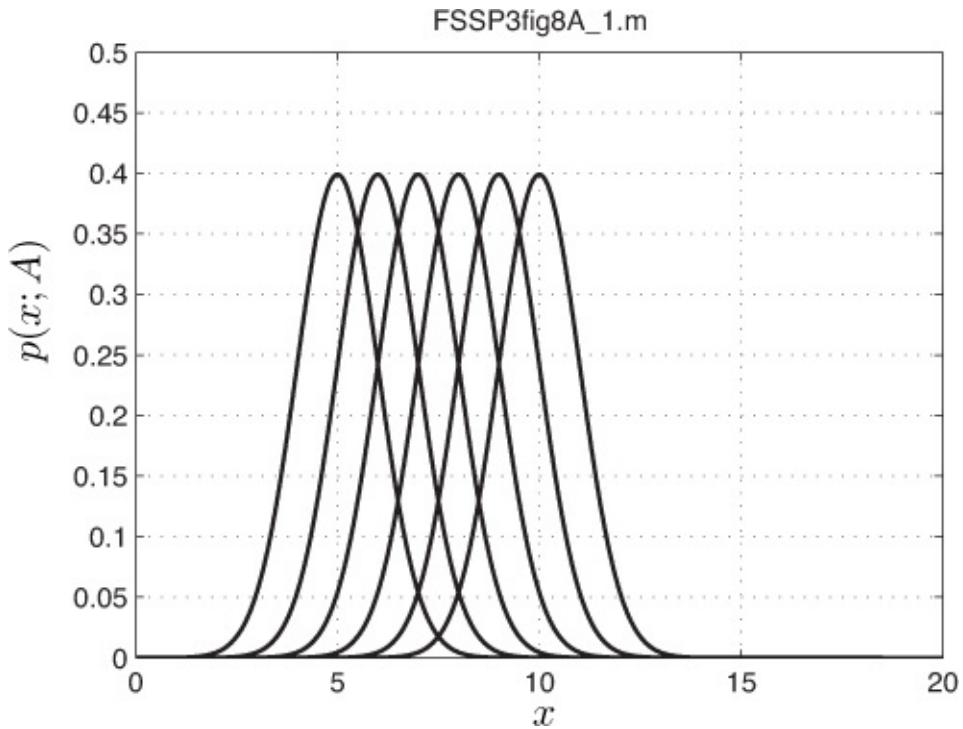
## Appendix 8A. Some Insights into Parameter Estimation

### 8A.1. Classical Approach

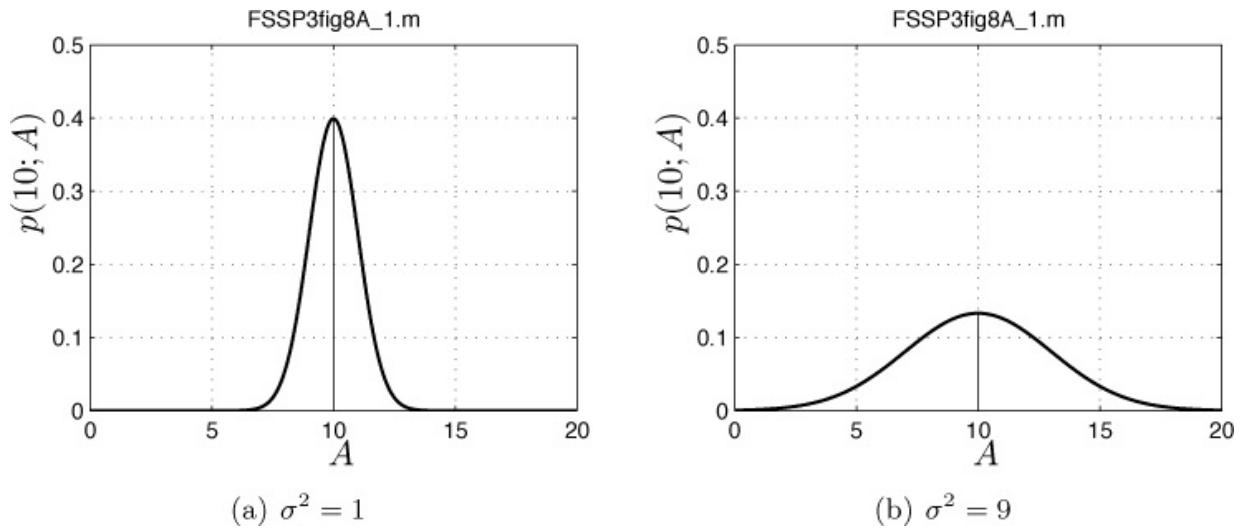
We consider the problem of estimating the mean  $A$  of a Gaussian random variable based on the observation of a single outcome, which we denote by  $x$ . We can also write the outcome in the form  $x = A + w$ , where  $w$  is the outcome of a zero mean Gaussian random variable with variance  $\sigma^2$ . This is because  $x = A + w$  is the outcome of a Gaussian random variable with mean  $E[x] = E[A + w] = A$  and variance  $\text{var}(x) = \text{var}(A + w) = \text{var}(w) = \sigma^2$ . As such, the equivalent problem is that of estimating the DC level signal embedded in WGN based on one sample. The PDF is

$$p(x; A) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x - A)^2\right]. \quad (8A.1)$$

The inclusion in the PDF notation  $p(x; A)$  of  $A$  reminds us that the PDF is dependent on  $A$  (which then influences the outcome, and provides the information to infer the value of  $A$  from our observed value of  $x$ ). Some of the possible PDFs given by (8A.1) are shown in [Figure 8A.1](#) for values of  $A$  being  $A = 5, 6, \dots, 10$  and  $\sigma^2 = 1$ . Now for a given observed data sample value  $x = x_0$ , we can plot  $p(x_0; A)$  versus  $A$  to see how the PDF and hence the probability of observing  $x_0$  will depend upon the true value of  $A$ . Shown in [Figure 8A.2a](#) is the function  $p(x_0; A)$  plotted versus  $A$  for  $x_0 = 10$  and  $\sigma^2 = 1$ . This function is called the *likelihood function* since it indicates how likely each value of  $A$  is, given that  $x_0$  has been observed. The curve is identical to that for  $p(x; A = 10)$  because of the form of (8A.1) (*not true* in general).



**Figure 8A.1: Plots of the PDF  $p(x; A)$  for  $A = 5, 6, \dots, 10$ .**



**Figure 8A.2: The likelihood functions for two different values of  $\sigma^2$ .**

Note that the values of  $A$  outside the interval shown are very unlikely to be the true value since for these values of  $A$ , we have that  $p(10; A) \approx 0$ . By increasing the variance to  $\sigma^2 = 9$ , we obtain the plot shown in [Figure 8A.2b](#). It is observed now that the range of values for which  $A$  is likely is increased significantly. Hence, it will be much more difficult to estimate  $A$  since the value of  $x_0 = 10$  has a nonzero probability of having been observed for all of these additional  $A$ .

values. This indicates that the variance of any estimator of  $A$  will be critically dependent upon  $\sigma^2$ .

Referring to [Figure 8A.2](#), if we had indeed observed  $x_0 = 10$ , then it seems reasonable to estimate  $A$  by the most likely value. In this case, it is  $\hat{A} = x_0 = 10$  or the value of  $A$  that maximizes the likelihood function. This is called the *maximum likelihood estimator* (MLE). Note that the MLE, which is  $\hat{A} = x$ , is unbiased, meaning  $E[\hat{A}] = E[x] = A$ , and its variance is

$$\text{var}(\hat{A}) = \text{var}(x) = \sigma^2$$

and is related to the width of the likelihood functions shown in [Figure 8A.2](#). In fact, the distance in  $A$  from the point at which the likelihood function is maximum and the point to the right that is an inflection point on the curve can be shown to be given by  $\sigma$ . A related measure of width is the curvature of a function evaluated at the maximum location of the function. The *relative* curvature is found as the curvature of the *logarithm* of this function. It is found as the negative of the second derivative and from [\(8A.1\)](#) is given by differentiating

$$\frac{\partial \ln p(x; A)}{\partial A} = \frac{1}{\sigma^2}(x - A) \quad (8A.2)$$

to yield

$$\frac{\partial^2 \ln p(x; A)}{\partial A^2} = -\frac{1}{\sigma^2}$$

and finally

$$-\frac{\partial^2 \ln p(x; A)}{\partial A^2} = \frac{1}{\sigma^2}$$

since the second derivative is negative at a maximum. Since the variance of the MLE  $\hat{A} = x$  is  $\sigma^2$ , we see that

$$\text{var}(\hat{A}) = \frac{1}{-\frac{\partial^2 \ln p(x; A)}{\partial A^2}}.$$

It is usually the case that the curvature of the likelihood function will depend upon  $x$ . In this case it did not since the form of  $p(x; A)$  resulted in the same type of likelihood function for all  $x$ , just shifted in location to be centered about  $x$ . To accommodate the general case we have that

$$\text{var}(\hat{A}) = \frac{1}{-E\left[\frac{\partial^2 \ln p(x; A)}{\partial A^2}\right]}. \quad (8A.3)$$

Note that the variance can also be written as

$$\text{var}(\hat{A}) = \frac{1}{E \left[ \left( \frac{\partial \ln p(x; A)}{\partial A} \right)^2 \right]} \quad (8A.4)$$

which is easily verified using (8A.2).

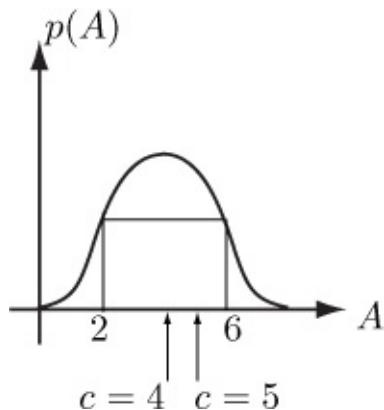
Why (8A.3) should also be a lower bound on the variance of an unbiased estimator, and is the *Cramer-Rao lower bound*, is the result of a fortuitous identity. It states that

$$E \left[ \frac{\partial \ln p(x; \theta)}{\partial \theta} (\hat{\theta} - \theta) \right] = 1 \quad (8A.5)$$

if  $\hat{\theta}$  is an unbiased estimator of  $\theta$ . Using (8A.2) with  $x = \hat{\theta}$  and  $A = \theta$ , we see that this is true. The CRLB as given by the right-hand-side of (8A.4) results by applying the Cauchy-Schwarz inequality to (8A.5), i.e.,  $E^2 [uv] \leq E[u^2] E[v^2]$ , with  $u = \partial \ln p(x; \theta) / \partial \theta$ ,  $v = \hat{\theta} - \theta$ , and recognizing that  $E[uv] = 1$  from (8A.5).

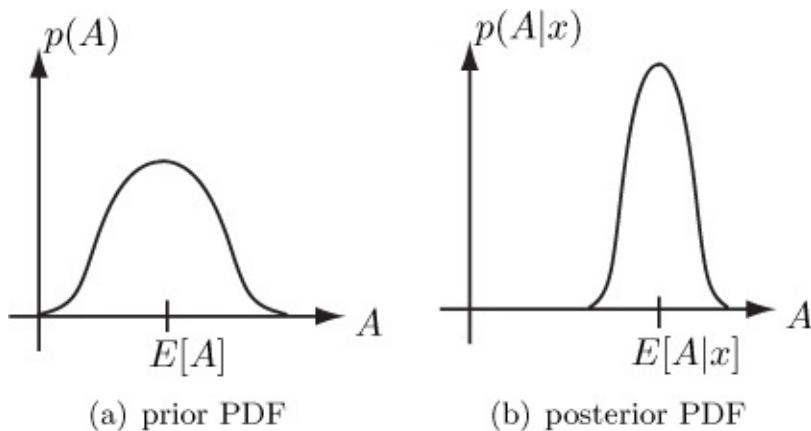
## 8A.2. Bayesian Approach

Consider the Bayesian MMSE estimator for the outcome of a random variable  $A$  based on a single outcome. The MMSE estimator follows from the simple result that the value of  $c$ , a constant, that minimizes the MSE  $E[(A - c)^2]$  is just the mean of  $A$ , i.e.,  $E[A]$ . For example, consider the PDF of  $A$  as shown in Figure 8A.3. Then, if the points  $A = 2$  and  $A = 6$  occur and we choose  $c = 5$ , the contribution to the MSE is  $(2 - 5)^2 + (6 - 5)^2 = 10$ . But if the center is chosen as  $c = 4$ , then the contribution is only  $(2 - 4)^2 + (6 - 4)^2 = 8$ . The center is just the mean, so that we should choose  $\hat{A} = c = E[A]$  to minimize the MSE. This result is easily verified by writing the MSE as  $E[(x - c)^2] = E[x^2] - 2cE[x] + c^2$ , differentiating with respect to  $c$  and setting the result equal to zero. Also, note that if the estimator is the mean, then the minimum MSE is  $E[(A - \hat{A})^2] = E[(A - c)^2] = E[(A - E[A])^2] = \text{var}(A)$ .



**Figure 8A.3: Prior PDF used to explain MMSE estimator as the mean of the PDF.**

The same argument applies after the data is observed except that in this case the PDF is the posterior PDF  $p(A|x)$  and is found using Bayes' rule. As before the MSE is minimized by letting the estimator be the mean. Thus,  $\hat{A} = E[A|x]$  and it can be shown that the minimum MSE is the average variance of the posterior PDF. Typically, we see the prior PDF and posterior PDF as depicted in [Figure 8A.4](#). The narrower width of the posterior PDF indicates that the estimator, being the mean  $E[A|x]$ , will be less in error as measured by the MSE, than that for the prior PDF mean estimator  $E[A]$ .



**Figure 8A.4: PDFs of the unknown parameter before and after data is observed.**

## Appendix 8B. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators,

respectively.

**8.1** We have from (8.4) that

$$\frac{\partial \ln p(\mathbf{x}; A)}{\partial A} = \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (x[n] - A).$$

If we differentiate a second time, we obtain

$$\frac{\partial^2 \ln p(\mathbf{x}; A)}{\partial A^2} = \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (-1) = -\frac{N}{\sigma^2}$$

producing

$$-\frac{\partial^2 \ln p(\mathbf{x}; A)}{\partial A^2} = \frac{N}{\sigma^2}.$$

Since this does not depend on  $\mathbf{x}$  (generally it will), and the expected value of a constant is the constant, we have finally

$$\frac{1}{E\left[-\frac{\partial^2 \ln p(\mathbf{x}; A)}{\partial A^2}\right]} = \frac{\sigma^2}{N}.$$

Thus, the CRLB is  $\sigma^2/N$  and agrees with  $\text{var}(A) = \sigma^2/N$ .

**8.2** Using

$$p(\mathbf{x}; A) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left[-\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - As[n])^2\right]$$

and taking the natural logarithm we have

$$\ln p(\mathbf{x}; A) = \ln \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} - \frac{1}{2\sigma^2} \sum_{n=0}^{N-1} (x[n] - As[n])^2.$$

Differentiating yields

$$\begin{aligned} \frac{\partial \ln p(\mathbf{x}; A)}{\partial A} &= \frac{1}{\sigma^2} \sum_{n=0}^{N-1} (x[n] - As[n])s[n] \\ &= \frac{1}{\sigma^2} \left( \sum_{n=0}^{N-1} x[n]s[n] - A \sum_{n=0}^{N-1} s^2[n] \right) \\ &= \frac{\sum_{n=0}^{N-1} s^2[n]}{\sigma^2} \left( \frac{\sum_{n=0}^{N-1} x[n]s[n]}{\sum_{n=0}^{N-1} s^2[n]} - A \right). \end{aligned} \quad (8B.1)$$

Using (8.3) produces the MVU estimator

$$\hat{A} = \frac{\sum_{n=0}^{N-1} x[n]s[n]}{\sum_{n=0}^{N-1} s^2[n]}$$

whose variance is the reciprocal of the constant before the parenthesis

$$\text{var}(\hat{A}) = \frac{\sigma^2}{\sum_{n=0}^{N-1} s^2[n]}.$$

To show that it is unbiased

$$\begin{aligned} E[\hat{A}] &= E\left[\frac{\sum_{n=0}^{N-1} x[n]s[n]}{\sum_{n=0}^{N-1} s^2[n]}\right] \\ &= \frac{\sum_{n=0}^{N-1} E[x[n]]s[n]}{\sum_{n=0}^{N-1} s^2[n]} \\ &= \frac{\sum_{n=0}^{N-1} As[n]s[n]}{\sum_{n=0}^{N-1} s^2[n]} = A. \end{aligned}$$

**8.3** First we have that

$$\text{var}(\hat{A}) = \text{var}(x[0]) = \text{var}(s[0] + w[0]) = \text{var}(w[0]) = \sigma^2.$$

From [Exercise 8.2](#), the CRLB is given by

$$\begin{aligned} \text{CRLB} &= \text{var}(\hat{A}) \\ &= \frac{\sigma^2}{\sum_{n=0}^{N-1} s^2[n]} \\ &= \frac{\sigma^2}{s^2[0] + s^2[1] + s^2[2]} \\ &= \frac{\sigma^2}{1 + (1/2)^2 + (1/4)^2} = \frac{16}{21}\sigma^2. \end{aligned}$$

The possible improvement is a factor of 21/16.

**8.4** If  $\sigma_A^2 = 0$ , we have that  $\alpha = 0$ , yielding  $\hat{A} = \mu_A$ . In this case the prior knowledge is asserted to be perfect, and hence the data is ignored. If  $\sigma_A^2 \rightarrow \infty$ , just the opposite happens. There is no prior knowledge and so the estimator is the sample mean, uninfluenced by the prior PDF of  $A$ . Finally, if  $N \rightarrow \infty$ , the knowledge contributed by the data “swamps out” the prior knowledge, producing again the sample mean estimator.

**8.5** A standard graph of  $\ln(x)$  shows that it rises above the abscissa at  $x = 1$  so that  $\ln(x) > 0 = \ln(1)$  for  $x > 1$ . This monotonicity property only holds for certain operations. A counterexample is if  $x > 1$ , and we multiply

both sides by  $-1$  to yield  $-x > -1$ . The region for which the latter inequality is satisfied is for  $x < 1$ , clearly a different region.

**8.6** Evaluating  $\frac{A}{\sigma^2}x[0] - \frac{A^2}{2\sigma^2} > \ln \gamma_{NP}$  for  $\sigma^2 = 1$  and  $\gamma_{NP} = e$ , we have that  $Ax[0] - A^2/2 > 1$ . For  $A = 1$ , this becomes  $x[0] > 3/2$  while for  $A = -1$ , it is  $x[0] < -3/2$ . Since the decision rules are different, it is not possible to implement the detector without knowledge of  $A$  (actually we just need the sign of  $A$ ).

**8.7** We have that

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x[0] - 1)^2\right] > \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x[0] + 1)^2\right]$$

but by taking the natural logarithm of both sides, this becomes

$$-\frac{1}{2\sigma^2}(x[0] - 1)^2 > -\frac{1}{2\sigma^2}(x[0] + 1)^2$$

and multiplying by  $2\sigma^2$  and expanding we have

$$-(x^2[0] - 2x[0] + 1) > -(x^2[0] + 2x[0] + 1)$$

which reduces to  $4x[0] > 0$  or  $x[0] > 0$ .

**8.8** The ML rule chooses the hypothesis with the larger PDF. Thus, from (8.22) we choose  $H_1$  if

$$(x[0] + 1)^2 + (x[1] + 1)^2 > (x[0] - 1)^2 + (x[1] - 1)^2$$

which simplifies to choosing  $H_1$  if  $2(x[0]+x[1]) > -2(x[0]+x[1])$  or  $x[0]+x[1] > 0$ . This is the region  $y > -x$  in the  $x-y$  plane. The boundary of the decision regions is therefore  $y = -x$ . By performing a computer simulation you should see that the line divides the two sets of outcomes correctly with only a few errors. You can run the program FSSP3exer8\_8.m, contained on the CD, to obtain the results.

**8.9** The MMSE estimator is

$$\hat{\theta} = E[\theta|\mathbf{x}] = \mu_\theta + \left(\mathbf{C}_\theta^{-1} + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H}\right)^{-1} \frac{\mathbf{H}^T}{\sigma^2} (\mathbf{x} - \mathbf{H}\mu_\theta).$$

If we let  $\mathbf{C}_\theta^{-1} = \mathbf{0}$ , then we have

$$\begin{aligned}
E[\boldsymbol{\theta}|\mathbf{x}] &= \boldsymbol{\mu}_\theta + \left( \frac{1}{\sigma^2} \mathbf{H}^T \mathbf{H} \right)^{-1} \frac{\mathbf{H}^T}{\sigma^2} (\mathbf{x} - \mathbf{H}\boldsymbol{\mu}_\theta) \\
&= \boldsymbol{\mu}_\theta + \sigma^2 (\mathbf{H}^T \mathbf{H})^{-1} \frac{\mathbf{H}^T}{\sigma^2} (\mathbf{x} - \mathbf{H}\boldsymbol{\mu}_\theta) \\
&= \boldsymbol{\mu}_\theta + (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} - (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H}\boldsymbol{\mu}_\theta \\
&= (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}
\end{aligned}$$

which has *the same form* as the classical linear model estimator (the performance will be different, however, due to the different assumptions on  $\boldsymbol{\theta}$  and the different performance metrics!)

## **Part II. Specific Algorithms**

# Chapter 9. Algorithms for Estimation

## 9.1. Introduction

Previous chapters have described the general methodology and approaches used to develop practical statistical signal processing algorithms. We now turn our attention to describing common problems of interest and the algorithms used to solve them. Most of the algorithms to be described have been derived from the optimal and suboptimal approaches that were discussed in [Chapter 8](#) for the signal/noise models presented in [Chapters 3](#) and [4](#). Many of the algorithms have been extracted from [[Kay 1993](#), [1998](#)], and so the reader is referred there for the derivations. Our intent in this chapter is to summarize many of the commonly used algorithms that “have stood the test of time”, and hence, provide good solutions to various signal processing problems. The range of applications is immense, including radar, sonar, communications, speech/audio, biomedical signal processing, pattern recognition, and data analysis, among others. Appropriate references to these application areas are also given.

We will describe estimation algorithms in this chapter with detection algorithms in [Chapter 10](#) and spectral estimation algorithms in [Chapter 11](#). In an attempt to organize the estimation algorithms in an orderly fashion, we have chosen to categorize them as solving problems of: extracting signal information, and enhancing signals corrupted by noise and/or interference. Both these problems involve an estimation procedure for a signal embedded in noise. To extract signal information we use parameter estimation. To enhance a signal corrupted by noise and/or interference we use filtering concepts, which may be viewed as signal estimation with the signal comprising the parameters of interest. Note that this categorization has been chosen for convenience and that algorithms included in one category are frequently used in the other.

It should also be observed that the algorithm solutions depend heavily on whether the signal is assumed to be deterministic or the outcome of a random process. The reader at this point may wish to review the different signal models described in [Chapter 3](#). These differences in modeling translate directly into the resultant algorithms employed in practice.

In keeping with our exposition we will describe algorithms for real data. In practice, particularly for radar and sonar, the algorithms must operate on complex data. Some of the extensions of the algorithms for complex data are given in [Chapter 12](#). [Table 9.1](#) summarizes the algorithms to be described in this

chapter.

**Table 9.1: Summary of estimation algorithms.**

Algorithm	Description
9.1	Deterministic signal amplitude estimation
9.2	Sinusoidal amplitude and phase estimation
9.3	Sinusoidal amplitude, phase, and frequency estimation
9.4	Estimation of time delay
9.5	Bearing estimation
9.6	Estimation of power of WGN
9.7	Estimation of random signal power
9.8	Random signal center frequency estimation
9.9	Multiple sinusoids in WGN - optimal solution
9.10	Multiple sinusoids in WGN - suboptimal solution
9.11	Extraction of random periodic signal in noise
9.12	Extraction of random signal in noise
9.13	Extraction of signal in noise and interference

## 9.2. Extracting Signal Information

---

### Algorithm 9.1 – Deterministic signal amplitude estimation

#### 1. Problem

Estimate the amplitude of a deterministic signal in white Gaussian noise (WGN).

#### 2. Application area example

In a digital communication system a pilot signal  $s[n]$  is transmitted prior to data transmission to determine the attenuation of the channel [[Haykin 1994](#)]. The amplitude of the received signal must be estimated to determine the necessary gain to be used to compensate for the channel attenuation.

#### 3. Data model/assumptions

$$x[n] = As[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $A$  is the amplitude to be estimated ( $-\infty < A < \infty$ ),  $s[n]$  is the known signal, and  $w[n]$  is WGN with variance  $\sigma^2$  (need not be known to implement estimator but must be known to determine performance).

#### 4. Estimator

$$\hat{A} = \frac{\sum_{n=0}^{N-1} x[n]s[n]}{\sum_{n=0}^{N-1} s^2[n]} \quad (9.1)$$

The MATLAB subprogram `FSSP3alg9_1_sub.m` can be used to implement the estimator and is contained on the CD.

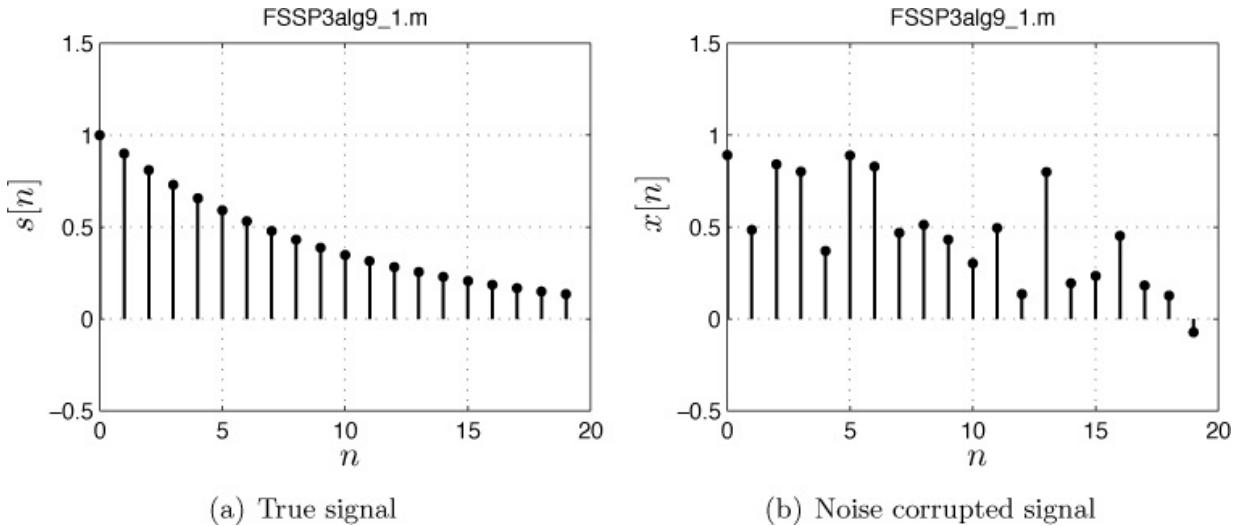
#### 5. Performance

Estimator is optimal in that it attains the Cramer-Rao lower bound (CRLB). The probability density function (PDF) of  $\hat{A}$  is Gaussian with

$$\hat{A} \sim \mathcal{N}\left(A, \frac{\sigma^2}{\sum_{n=0}^{N-1} s^2[n]}\right).$$

#### 6. Example

Consider a damped exponential signal given by  $s[n] = r^n$ , with  $A = 1$ ,  $r = 0.9$ , and  $N = 20$ . The true signal is shown in [Figure 9.1a](#) and the noisy signal in [Figure 9.1b](#). The WGN variance is  $\sigma^2 = 1/16$ . The estimated value of  $A$  for this data set is  $\hat{A} = 0.9893$ .



**Figure 9.1: Signal and data for amplitude estimation.**

#### 7. Explanation

When no noise is present, it is readily seen that  $\hat{A} = A$  (let  $x[n] = As[n]$  in (9.1)). When noise is present, the weighting by  $s[n]$  followed by summing

in the numerator will help to reduce the noise effects.

## 8. Comments/References

For multiple known signals with unknown amplitudes the MLE is obtained by noting that we have the linear model as described in [Section 3.5](#). Also, see [Exercise 8.2](#) and [[Kay 1998](#), pg. 254].



---

### Exercise 9.1 – More heavily damped exponential

Repeat the example shown in [Figure 9.1](#) but let  $r = 0.6$  and estimate the amplitude. Compare your results to those in the example and explain the difference. Be sure to initialize the random number generator using `randn ('state', 0)` to generate the same noise realization. This allows a more valid comparison since then only the signal is being changed from the example.



---

### Algorithm 9.2 – Sinusoidal amplitude and phase estimation

#### 1. Problem

Estimate the amplitude and phase of a sinusoid embedded in WGN.

#### 2. Application area example

Vowel-like speech sounds may be synthesized by adding together harmonically related sinusoidal components. To do so the amplitude and phase of each component must be estimated from an actual speech sound [[McAulay and Quartieri, 1986](#)].

#### 3. Data model/assumptions

$$x[n] = A \cos(2\pi f_0 n + \varphi) + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $A$  ( $A > 0$  for identifiability), and  $\varphi$  ( $-\pi \leq \varphi < \pi$ ) are the amplitude and phase parameters, respectively to be estimated. The frequency  $f_0$  ( $0 < f_0 < 1/2$ ) is assumed known.  $w[n]$  is WGN with variance  $\sigma^2$  (need not be known to implement estimator but must be known to determine performance).

#### 4. Estimator

$$\begin{aligned}\hat{A} &= \sqrt{\hat{\alpha}_1^2 + \hat{\alpha}_2^2} \\ \hat{\phi} &= \arctan\left(\frac{-\hat{\alpha}_2}{\hat{\alpha}_1}\right)\end{aligned}\quad (9.2)$$

where

$$\begin{bmatrix} \hat{\alpha}_1 \\ \hat{\alpha}_2 \end{bmatrix} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x} \quad (9.3)$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ \cos 2\pi f_0 & \sin 2\pi f_0 \\ \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \sin[2\pi f_0(N-1)] \end{bmatrix}.$$

The inverse tangent function output is chosen so that  $\hat{\phi}$  is the angle lying in the same quadrant as  $(\hat{\alpha}_1, -\hat{\alpha}_2)$ . If  $2/N < f_0 < 1/2 - 2/N$ , then the estimator is approximately given by

$$\begin{aligned}\hat{A} &= \frac{2}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n) \right| \\ \hat{\phi} &= \arctan\left(\frac{-\sum_{n=0}^{N-1} x[n] \sin(2\pi f_0 n)}{\sum_{n=0}^{N-1} x[n] \cos(2\pi f_0 n)}\right)\end{aligned}\quad (9.4)$$

where the arctan function is the four quadrant inverse, interpreted as producing an angle in the quadrant associated with the point  $(\sum_{n=0}^{N-1} x[n] \cos(2\pi f_0 n), -\sum_{n=0}^{N-1} x[n] \sin(2\pi f_0 n))$ . The MATLAB subprogram FSSP3alg9\_2\_.m can be used to implement the estimator and is contained on the CD.

#### 5. Performance

The exact form given by (9.2) is the MLE and hence is asymptotically optimal in that it attains the CRLB. Its asymptotic PDF (as  $N \rightarrow \infty$  and/or a high) is

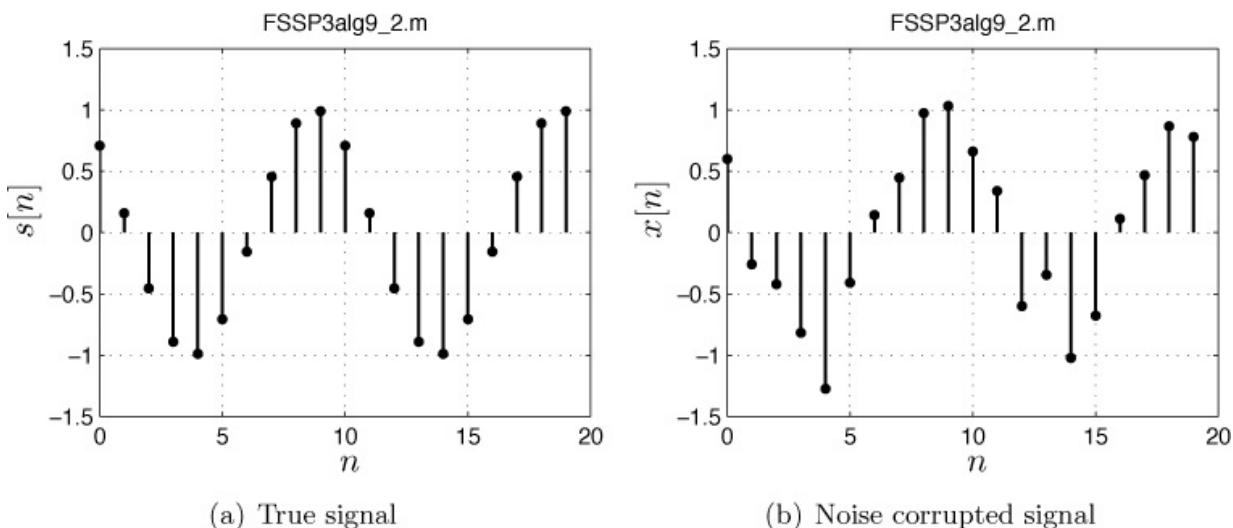
$$\hat{A} \sim \mathcal{N}\left(A, \frac{2\sigma^2}{N}\right)$$

$$\hat{\phi} \sim \mathcal{N}\left(\phi, \frac{1}{N\eta}\right)$$

where  $\eta = A^2/(2\sigma^2)$  is the signal-to-noise ratio (SNR), and  $\hat{A}$  and  $\hat{\phi}$  are independent.

## 6. Example

Consider a sinusoid with  $A = 1$ ,  $f_0 = 0.1$ ,  $\phi = \pi/4 = 0.7854$ ,  $N = 20$  embedded in WGN with variance  $\sigma^2 = 1/16$ . The true signal is shown in [Figure 9.2a](#) and the noisy signal in [Figure 9.2b](#). The estimated amplitude is  $\hat{A} = 0.9295$  and the estimated phase is  $\hat{\phi} = 0.8887$ .



**Figure 9.2: Sinusoidal signal and data for amplitude and phase estimation.**

## 7. Explanation

We are first converting to the equivalent linear signal model (see [Section 1.3](#))

$$s[n] = \alpha_1 \cos(2\pi f_0 n) + \alpha_2 \sin(2\pi f_0 n)$$

estimating  $\alpha_1$  and  $\alpha_2$ , and then converting back to amplitude and phase. The  $\alpha_1$  and  $\alpha_2$  parameters are estimated using a least squares minimization of

$$\sum_{n=0}^{N-1} (x[n] - \alpha_1 \cos(2\pi f_0 n) - \alpha_2 \sin(2\pi f_0 n))^2$$

to produce (9.3). Note that for the approximation of (9.4), if we convert to a *complex amplitude*, the estimator becomes

$$\hat{A} \exp(j\hat{\phi}) = \frac{2}{N} \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n)$$

which is recognized as the Fourier transform of the data (scaled by  $2/N$ ) evaluated at the true frequency  $f_0$ .

## **8. Comments/References**

See [Section 1.3](#) and [[Kay 1993](#), pp. 56–57, 193–195].

---



---



## **Exercise 9.2 – Lower SNR sinusoid**

Repeat the example shown in [Figure 9.2](#) but change the noise variance to  $\sigma^2 = 10/16$  (a 10 dB decrease in SNR) and estimate the amplitude and phase. Compare your results to that in the example.

---



---



## **Algorithm 9.3 – Sinusoidal amplitude, phase, and frequency estimation**

### **1. Problem**

Estimate the amplitude, phase, and frequency of a sinusoid embedded in WGN.

### **2. Application area example**

In radar and active sonar a sinusoidal pulse is transmitted. The received echo is changed in amplitude and phase due to propagation and scattering effects. The frequency is changed due to the motion of a target, which reflects the signal, i.e., due to the Doppler effect [[Skolnik 1980](#), [Burdic 1984](#)]. All these parameters need to be estimated. Some modifications must be made to accommodate complex data as described in [Chapter 12](#). See also [Section 2.3](#) for an ultrasound example.

### **3. Data model/assumptions**

$$x[n] = A \cos(2\pi f_0 n + \phi) + w[n] \quad n = 0, 1, \dots, N - 1 \quad (9.5)$$

where  $A$  ( $A > 0$  for identifiability), and  $\phi$  ( $-\pi \leq \phi < \pi$ ), and  $f_0$  ( $0 < f_0 < 1/2$ ) are the amplitude, phase, and frequency parameters, respectively, to be estimated.  $w[n]$  is WGN with variance  $\sigma^2$  (need not be known to implement the estimator but must be known to determine performance).

### **4. Estimator**

The frequency is first estimated by maximizing the function

$$J(f_0) = \mathbf{x}^T \mathbf{H}(f_0) (\mathbf{H}^T(f_0) \mathbf{H}(f_0))^{-1} \mathbf{H}^T(f_0) \mathbf{x} \quad (9.6)$$

over the interval  $0 < f_0 < 1/2$ , where

$$\mathbf{H}(f_0) = \begin{bmatrix} 1 & 0 \\ \cos 2\pi f_0 & \sin 2\pi f_0 \\ \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \sin[2\pi f_0(N-1)] \end{bmatrix}.$$

If it is known that the frequency is in the interval  $2/N < f_0 < 1/2 - 2/N$ , then the function to be maximized can be approximated by

$$J'(f_0) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n) \right|^2$$

and is recognized as the periodogram. Once the estimate is found as  $\hat{f}_0$ , then (9.2) and (9.3) can be used to find the estimates of the amplitude and phase if  $\mathbf{H}$  is replaced by  $\mathbf{H}(\hat{f}_0)$ . If it is known that  $2/N < f_0 < 1/2 - 2/N$ , then the amplitude and phase estimates can be approximated using (9.4). The MATLAB subprogram FSSP3alg9\_3\_sub.m can be used to implement the exact estimator and is contained on the CD.

### **5. Performance**

The exact form given by maximizing (9.6) and using (9.2) is the MLE and hence is asymptotically optimal in that it attains the CRLB. Its asymptotic performance (as  $N \rightarrow \infty$  and/or a high SNR) is

$$\begin{aligned}\hat{A} &\sim \mathcal{N}\left(A, \frac{2\sigma^2}{N}\right) \\ \hat{\phi} &\sim \mathcal{N}\left(\phi, \frac{2(2N-1)}{\eta N(N+1)}\right) \\ \hat{f}_0 &\sim \mathcal{N}\left(f_0, \frac{12}{(2\pi)^2 \eta N(N^2-1)}\right)\end{aligned}$$

where  $\eta = A^2/(2\sigma^2)$  is the SNR, and  $\hat{A}$  is independent of  $\hat{\phi}$  and  $\hat{f}_0$ .

## 6. Example

Consider the same signal as shown in [Figure 9.2a](#) and the data set shown in [Figure 9.2b](#). Now assume the frequency is unknown. The estimates obtained using (9.6) with (9.2) and (9.3) are  $\hat{A} = 0.9353$  (true value is  $A = 1$ ),  $\hat{\phi} = 0.9943$  (true value is  $\phi = 0.7854$ ), and  $\hat{f}_0 = 0.0980$  (true value is  $f_0 = 0.1$ ). Note that the frequency is estimated very accurately since its variance is approximately proportional to  $1/N^3$ .

## 7. Explanation

The only difference between this estimator and the one when the frequency is known is the need to first estimate the frequency. This is done essentially by finding the frequency where the periodogram, an estimate of the power spectral density, is maximized. The need for the matrix implementation of (9.6) is to avoid the “interference” for the complex components of the real sinusoid at frequencies near 0 or 1/2 (see [Figure 1A.1](#)).

## 8. Comments/References

See [Section 3.5.4](#) and [[Kay 1993](#), pp. 56–57, 193–195].

### Exercise 9.3 – Noiseless sinusoid

Repeat the example for the data shown in [Figure 9.2b](#) but let the noise variance be zero. Do you estimate the sinusoidal parameters without error?

### Algorithm 9.4 – Estimation of time delay

## **1. Problem**

Estimate the time delay of a known deterministic signal in WGN.

## **2. Application area example**

Laser range finders called Light Detection and Ranging (LIDAR) are utilized to determine the range of an object. They do so by transmitting a signal and measuring the time it takes to propagate to the object and return to the receiver. Then the range is given by  $R = c\tau_0/2$ , where  $\tau_0$  is the round trip propagation time, i.e., the *delay time*, and  $c$  is the speed of propagation [[Adams 2000](#)].

## **3. Data model/assumptions**

$$x[n] = s[n - n_0] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the known signal,  $n_0$  is the delay to be estimated, and  $w[n]$  is WGN with variance  $\sigma^2$  (need not be known to implement estimator). It is assumed that  $s[n]$  is  $M$  samples in length, and is nonzero for  $n = 0, 1, \dots, M - 1$ . Also, for all possible values of  $n_0$ , the delayed signal  $s[n - n_0]$  is contained in the observation interval  $0 \leq n \leq N - 1$ .

## **4. Estimator**

The estimator is a “running correlator”, which correlates each possible received signal with the data as

$$J(n_0) = \sum_{n=n_0}^{n_0+M-1} x[n]s[n - n_0] \quad 0 \leq n_0 \leq N - M \quad (9.7)$$

and chooses the value of  $n_0$  that produces a maximum. This is the estimate  $\hat{n}_0$ . Note that the estimate of the delay in seconds is  $\hat{\tau}_0 = \hat{n}_0\Delta$ , where  $\Delta$  is the time interval between samples. The MATLAB subprogram `FSSP3alg9_4_sub.m` can be used to implement the estimator and is contained on the CD.

## **5. Performance**

The value of  $n_0$  that maximizes (9.7) is the MLE and hence is asymptotically optimal in that it attains the CRLB. Considering the delay time estimate in seconds given by  $\hat{\tau}_0$ , its asymptotic performance (as  $N \rightarrow \infty$  and/or a high SNR) is

$$\hat{\tau}_0 \sim \mathcal{N} \left( \tau_0, \frac{1}{\frac{\mathcal{E}}{N_0/2} \bar{F}^2} \right)$$

where the variance of the estimator is determined by the properties of the continuous-time signal  $s(t)$ . Observe that the sampled version is  $s(n\Delta) = s[n]$ . The quantity  $\frac{\mathcal{E}}{N_0/2}$  is the energy-to-noise ratio, where the energy is

$$\mathcal{E} = \int_0^{T_s} s^2(t) dt$$

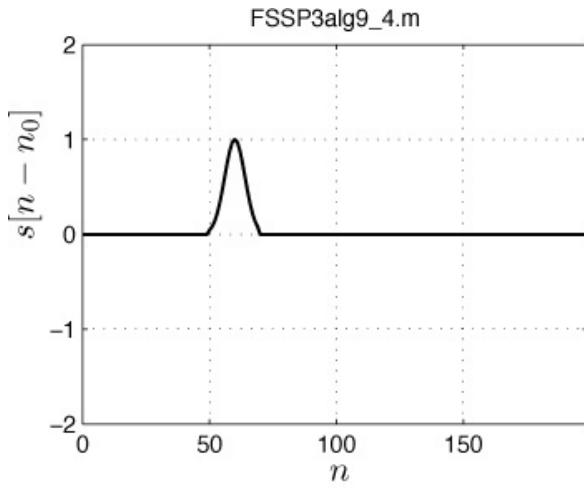
and  $T_s$  is the signal length in seconds ( $T_s = M\Delta$ ). Also,  $N_0/2$  is the power spectral density height of the continuous-time WGN. Assuming that  $s(t)$  is a lowpass signal bandlimited to  $W$  Hz, after sampling at  $2W$  samples/sec, the WGN noise variance of  $w[n]$  is  $\sigma^2 = N_0 W$  [[Kay 2006](#), pg. 583–586]. The quantity  $\bar{F}^2$  is the *mean square bandwidth* of the continuous-time signal and is defined as

$$\bar{F}^2 = \frac{\int_{-\infty}^{\infty} (2\pi F)^2 |S(F)|^2 dF}{\int_{-\infty}^{\infty} |S(F)|^2 dF}$$

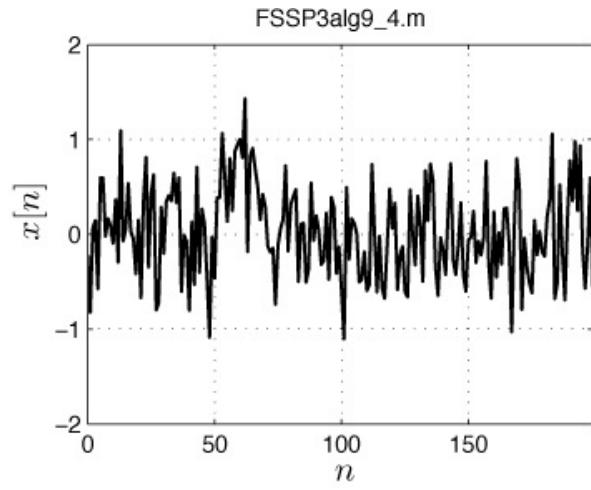
where  $F$  is the frequency in Hz and  $S(F)$  is the continuous-time Fourier transform of  $s(t)$ .

## 6. Example

Consider a signal that is a Gaussian pulse whose time delay is  $n_0 = 50$  as shown in [Figure 9.3a](#). To this signal is added WGN with variance  $\sigma^2 = 0.25$  resulting in the data shown in [Figure 9.3b](#). After implementing the running correlator, the function  $J(n_0)$  to be maximized is shown in [Figure 9.4](#) with the true time delay shown as the vertical line. In this case the maximum occurs exactly at the correct delay of  $n_0 = 50$ .

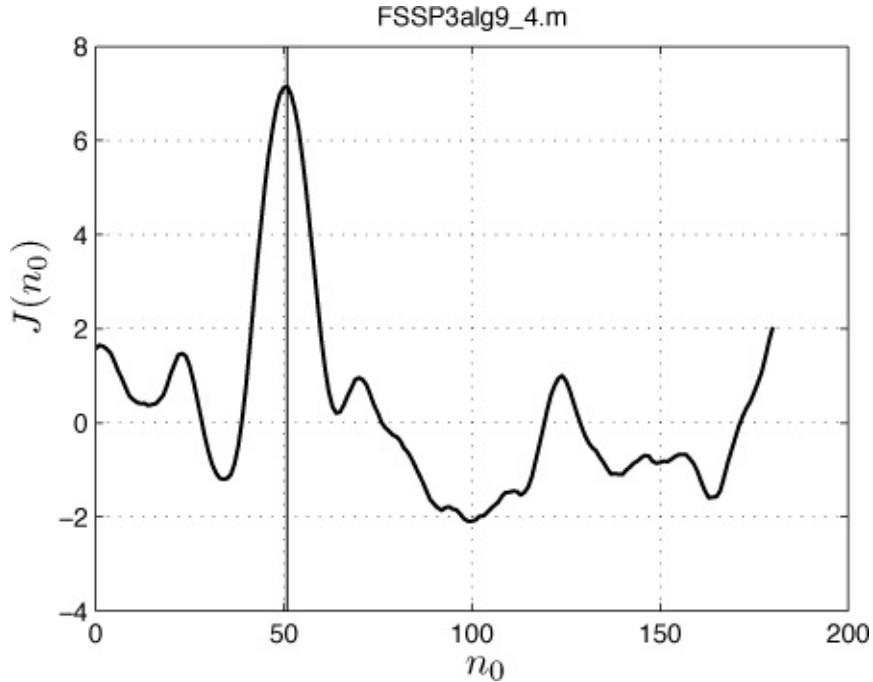


(a) True signal



(b) Noise corrupted signal

**Figure 9.3: Signal and data for time delay estimation example. The points have been connected by straight lines for easier viewing.**



**Figure 9.4: Function to be maximized to obtain MLE of time delay in samples.**

## 7. Explanation

Since  $x[n] = s[n - n_{0_{\text{true}}}] + w[n]$ , the function (9.7) becomes

$$J(n_0) = \sum_{n=n_0}^{n_0+M-1} (s[n - n_{0\text{true}}]s[n - n_0]) + \sum_{n=n_0}^{n_0+M-1} w[n]s[n - n_0]$$

so that when  $n_0 = n_{0\text{true}}$  the contribution from the first sum, which is an autocorrelation of the signal, becomes

$$\sum_{n=n_{0\text{true}}}^{n_{0\text{true}}+M-1} s^2[n - n_{0\text{true}}] = \mathcal{E}$$

which is the total signal energy. The autocorrelation sequence is known to peak at *zero lag* (when the two signals are aligned) and to be smaller otherwise. Note that “on the average” the value of the peak is the energy. This also accounts for the dependence of the CRLB on the signal energy.

## **8. Comments/References**

Using this method the accuracy of the time delay estimator cannot be better than one sample due to the time sampling quantization. Either a higher sampling rate is required or some form of interpolation is called for. Typical approaches are in [[Lai and Torp 1999](#)]. See also [[Kay 1993](#), pp. 53–56, pg. 192] for further details.



### **Exercise 9.4 – Different pulse shape**

Repeat the example shown in [Figure 9.3](#) but change the pulse to a square one of height one and width  $M = 20$ . Recall that  $N = 200$ ,  $n_0 = 50$ , and  $\sigma^2 = 0.25$ . What is the estimate?



### **Algorithm 9.5 – Bearing estimation**

#### **1. Problem**

Determine the bearing of an arriving signal using the output of a uniformly spaced line array of sensors.

#### **2. Application area example**

Acoustic underwater passive surveillance systems typically uses long linear arrays of sensors [[Urick 1975](#)]. Once a target is detected its bearing

can be estimated.

### **3. Data model/assumptions**

It is assumed that the sensors form a line array of  $M$  sensors with each sensor laid out along the  $x$ -axis in a two-dimensional space. The sensor positions are at  $x = 0, d, 2d, \dots, (M - 1)d$ . The arrival angle  $\beta$ , where  $0 < \beta < \pi/2$  and measured from the  $x$ -axis, is the arrival angle or “bearing” to the target. If a sinusoid is radiated by the target, the received signal at sensor  $n$  can be shown to be

$$x[n] = A \cos[2\pi(F_0(d/c) \cos \beta)n + \phi] + w[n] \quad n = 0, 1, \dots, M - 1 \quad (9.8)$$

where  $F_0$  is the frequency of the radiated signal in Hz,  $c$  is the propagation speed of the wavefront (with the wavefront assumed to be planar, which is valid if the signal is in the “far field”), and  $w[n]$  is WGN with variance  $\sigma^2$  (need not be known to implement estimator but must be known to determine performance). It is assumed that  $A$ ,  $\beta$ , and  $\phi$  are all unknown. It is desired to estimate  $\beta$ .

### **4. Estimator**

The estimator is analogous to that for the estimation of frequency of a sinusoid with the model given by (9.5), where  $f_0$  is replaced by the normalized spatial frequency  $F_0(d/c) \cos \beta$ . It computes the bearing estimate as the value of  $\beta$  ( $0 < \beta < \pi/2$ ) that maximizes the *spatial periodogram*

$$I_s(\beta) = \frac{1}{M} \left| \sum_{n=0}^{M-1} x[n] \exp \left[ -j2\pi \left( F_0 \frac{d}{c} \cos \beta \right) n \right] \right|^2. \quad (9.9)$$

It is assumed that  $\beta$  is not close to 0 or  $\pi/2$ . The MATLAB subprogram `FSSP3alg9_5_sub.m` can be used to implement the estimator and is contained on the CD.

### **5. Performance**

The estimator is the approximate MLE and hence is asymptotically optimal in that it attains the CRLB. Its asymptotic PDF (as  $M \rightarrow \infty$  and/or a high SNR) is

$$\hat{\beta} \sim \mathcal{N}(\beta, \text{var}(\hat{\beta})) \quad (9.10)$$

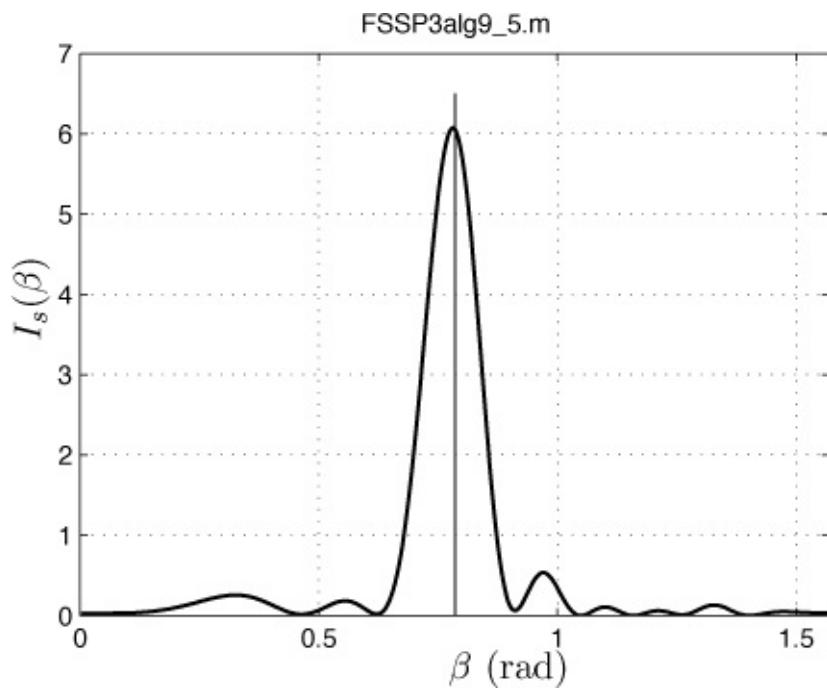
where

$$\text{var}(\hat{\beta}) = \frac{12}{(2\pi)^2 \eta M \frac{M+1}{M-1} \left(\frac{L}{\lambda}\right)^2 \sin^2 \beta} \quad (9.11)$$

and  $\eta = A^2/(2\sigma^2)$  is the SNR,  $L$  is the length of the array, and  $\lambda = c/F_0$  is the wavelength.

## 6. Example

Consider  $A = 1$ ,  $F_0 = 10,000$  Hz,  $d = \lambda/2$  m,  $c = 1500$  m/s (for underwater sound propagation), and  $\varphi = \pi/8$  for a line array with  $M = 20$  equally spaced sensors. The WGN variance is  $\sigma^2 = 1/16$ . For an arrival angle of  $\beta = \pi/4$  the spatial periodogram of the data  $x[n]$  is shown in [Figure 9.5](#). The estimate of the arrival angle is  $\hat{\beta} = 0.7823$  with the true value being  $\pi/4 = 0.7854$ . The true value is shown in [Figure 9.5](#) as the vertical line.



**Figure 9.5: Function to be maximized to obtain approximate MLE of arrival angle.**

## 7. Explanation

The bearing angle is estimated in a similar fashion to that of the frequency of a sinusoid embedded in noise. In this case the analogous frequency is given by

$$f_s = F_0 \frac{d}{c} \cos(\beta) \quad (9.12)$$

and is termed the spatial frequency. Note that it must be in the interval  $(0, 1/2)$  for identifiability. Thus, for  $d = \lambda/2$ ,  $f_s = (1/2) \cos(\beta)$  and so the arrival angle must be in the interval  $0 < \beta < \pi/2$ . For this range of angles (but for  $\beta$  not too close to 0 or  $\pi/2$ ) the frequency  $f_s$  of the “spatial sinusoid” can be found by taking the magnitude-squared Fourier transform, finding its peak location  $\hat{f}_s$ , and converting this to an angle via (9.12).

## 8. Comments/References

The model of (9.8) describes the output of the sensors at a single time instant or for a “snapshot”. In practice, there will always be multiple snapshots and so the estimator will need to accommodate these. The simplest approach is to average the periodograms from each snapshot. The estimator is an approximate MLE if  $\beta$  is not near 0 or  $\pi/2$ . It is also typically the case that  $d = \lambda/2$ , where  $\lambda$  is the wavelength, so that *spatial aliasing* is avoided. Finally, in practice the data is complex, allowing one to estimate  $\beta$  for  $0 < \beta < \pi$ .

See also [[Kay 1993](#), pp. 57–59, 195–196] and [[Johnson and Dudgeon 1993](#)].



### **Exercise 9.5 – Spatial periodogram dependence on bearing angle**

For  $\beta = \pi/8$  and  $\beta = \pi/16$  plot the spatial periodogram. Pay particular attention to the width of function to be maximized. Is it wider or narrower than that for  $\beta = \pi/4$ ? Will this affect the accuracy? You can plot the spatial periodogram by calling [betaahat beta  
Is]=FSSP3alg9\_5\_sub(x, F0, c, d); and then using  
plot(beta, Is) over the interval  $0 < \beta < \pi/2$ . Use  $F_0 = 10,000$ ,  $c = 1500$ ,  $d = \lambda/2 = c/(2F_0)$ . For the input x, use (9.8) with  $w[n] = 0$ .



### **Algorithm 9.6 – Estimation of power of WGN**

#### **1. Problem**

Estimate the power of WGN.

#### **2. Application area example**

In diagnosing the severity of speech impediments it is necessary to measure the SNR of an utterance [[Yingyong et al. 1999](#)]. This requires an estimate of the noise power (in addition to the signal power).

### **3. Data model/assumptions**

The data model is assumed to be  $x[n] = w[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $w[n]$  is WGN with variance, i.e., power, of  $\sigma^2$ .

### **4. Estimator**

The noise power is estimated using

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=0}^{N-1} x^2[n]. \quad (9.13)$$

The MATLAB subprogram `FSSP3alg9_6_sub.m` can be used to implement the estimator and is contained on the CD.

### **5. Performance**

The estimator, which is an MLE, is optimal in that it attains the CRLB. Its exact mean and variance are

$$\begin{aligned} E[\hat{\sigma}^2] &= \sigma^2 \\ \text{var}(\hat{\sigma}^2) &= \frac{2(\sigma^2)^2}{N}. \end{aligned}$$

Its PDF is a scaled (by  $a = \sigma^2/N$ ) chi-squared with  $N$  degrees of freedom given by

$$p(\hat{\sigma}^2) = \frac{1}{a^{N/2}\Gamma(N/2)} \left( \frac{\hat{\sigma}^2}{a} \right)^{N/2-1} \exp \left[ -\frac{1}{2} \left( \frac{\hat{\sigma}^2}{a} \right) \right] \quad \hat{\sigma}^2 > 0.$$

### **6. Example**

Consider WGN with variance  $\sigma^2 = 1$  for a data record length of  $N = 20$ . The estimate for this case is  $\hat{\sigma}^2 = 0.7447$ , which is far from the true value.

However, it is not unexpected since the variance of the estimator is  $2(\sigma^2)^2/N = 2/20 = 0.1$ . The standard deviation is therefore  $\sqrt{0.1} = 0.316$  so that this estimate is about one standard deviation away from the mean.

### **7. Explanation**

The estimator is a temporal average of the squares with each square being an unbiased estimator of  $\sigma^2$  and the averaging used to reduce the estimation error or variance of the estimator.

## **8. Comments/References**

See [Exercise 4.2](#) and [Section 6.4.2](#). For estimation of more general noise processes see [Chapter 11](#) on PSD estimation.



---

---

### **Exercise 9.6 – Longer data record**

Repeat the example but choose  $N = 2000$  so that the variance of the estimator is 0.001. Compare your results to that in the example.



---

---

### **Algorithm 9.7 – Random signal power estimation**

#### **1. Problem**

For a zero mean Gaussian random signal, i.e., a random process, with a known PSD, estimate the power.

#### **2. Application area example**

These types of estimators for power have applications in determining the magnitudes of weather echoes [[Zrnic 1979](#)].

#### **3. Data model/assumptions**

It is assumed that the zero mean random signal  $s[n]$  that is observed for  $n = 0, 1, \dots, N - 1$  has the PSD

$$P_s(f) = P_0 Q(f) \quad -1/2 \leq f \leq 1/2$$

where  $P_0 > 0$ ,  $Q(-f) = Q(f)$ ,  $Q(f) > 0$ , and  $\int_{-\frac{1}{2}}^{\frac{1}{2}} Q(f) df = 1$  so that  $P_0$  is the total power in the signal process.

#### **4. Estimator**

The estimator is

$$\hat{P}_0 = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{I(f)}{Q(f)} df \quad (9.14)$$

where

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \right|^2$$

is the periodogram of the signal process. The MATLAB subprogram entitled `FSSP3alg9_7_sub.m` can be used to implement the estimator and is contained on the CD.

## 5. Performance

The estimator is an approximate MLE (it assumes  $N$  is large) and so for large data records it attains the CRLB. Also, for large data records its PDF is

$$\hat{P}_0 \sim \mathcal{N}\left(P_0, \frac{2P_0^2}{N}\right).$$

## 6. Example

Consider an autoregressive (AR) random signal with two parameters  $a[1] = -2r \cos(2\pi f_0) = 0$ ,  $a[2] = r^2 = 0.9^2$ , and  $\sigma_u^2 = 1$ . The true PSD and a typical realization have been shown in [Figure 4.7](#). The total power is given by [\[Kay 1988\]](#), pg. 119]

$$P_0 = \frac{\sigma_u^2}{1 - r^4} = 2.9078.$$

For the data set shown in [Figure 4.7](#) the estimate is  $\hat{P}_0 = 3.4205$ . Note that if we had used [\(9.13\)](#), the estimate would have been  $(1/N) \sum_{n=0}^{N-1} s^2[n] = 1.9353$ , considerably poorer.

## 7. Explanation

The periodogram is an estimate of the PSD (see [Chapter 11](#)) and therefore, if  $I(f) \approx P_s(f) = P_0 Q(f)$ , the estimator will yield  $P_0$  (let  $I(f) = P_0 Q(f)$  in [\(9.14\)](#)).

## 8. Comments/References

The power of a random signal, whose PSD is not known, can be estimated using [\(9.13\)](#). The estimator given by [\(9.14\)](#) should perform better since it uses the shape of the PSD as a priori information. See also [Section 4.4](#) and [\[Kay 1993, Problems 3.16, 7.23\]](#).



## **Exercise 9.7 – Wider bandwidth PSD**

Repeat the example but change the pole radius to  $r = 0.7$  and let  $\sigma_u^2 = 0.76$  so that  $P_0 = 1$ . Then, use (9.14) as well as (9.13) to estimate the total power. You can use ARgenda.m to generate the data, and ARpsd.m with  $L=4096$  to compute the PSD values of  $P_s(f) = Q(f)$ . Compare the results of the two different estimators.

---



---

### **Algorithm 9.8 – Random signal center frequency estimation**

#### **1. Problem**

Estimate the center frequency of a zero mean Gaussian random signal, i.e., a random process, embedded in WGN.

#### **2. Application area example**

Estimation of the Doppler shift of radar returns from the atmosphere are routinely used to indicate weather conditions [[Narayanan and Dawood 2000](#)].

#### **3. Data model/assumptions**

It is assumed that the zero mean random process  $x[n]$  is observed for  $n = 0, 1, \dots, N - 1$ , and has the PSD

$$P_x(f) = \begin{cases} P_s(f - f_0) + \sigma^2 & 0 \leq f \leq 1/2 \\ P_x(-f) & -1/2 \leq f < 0 \end{cases}$$

where  $P_s(f)$  is a lowpass signal PSD centered at  $f = 0$  with its peak value at  $f = 0$  and also  $P_s(-f) = P_s(f)$ . The bandwidth of  $P_s(f)$  is  $B/2$ . Also, the center frequency  $f_0$  is assumed to satisfy  $B/2 < f_0 < 1/2 - B/2$  so that the entire bandpass signal has a PSD  $P_s(f - f_0)$  that is in the interval  $0 < f < 1/2$ . The lowpass signal PSD  $P_s(f)$  is assumed to be known, as is the white Gaussian noise variance  $\sigma^2$ .

#### **4. Estimator**

The estimator is the value of  $f_0$  that minimizes

$$J(f_0) = \int_0^{\frac{1}{2}} \frac{I(f)}{P_s(f - f_0) + \sigma^2} df \quad (9.15)$$

over the center frequencies  $B/2 < f_0 < 1/2 - B$ , where  $I(f)$  is the periodogram. As an approximation, if the overall signal power is large

relative to the noise power, the function  $J'(f_0)$  can be *maximized*, where

$$J'(f_0) = \int_0^{\frac{1}{2}} I(f) P_s(f - f_0) df \quad (9.16)$$

to obtain the estimate. This alleviates the need to know  $\sigma^2$ . For an AR signal PSD the MATLAB subprogram `FSSP3alg9_8_sub.m` can be used to implement the estimator (9.15) and is contained on the CD.

## 5. Performance

The estimator is an approximate MLE (it assumes  $N$  is large), and so for large data records it attains the CRLB. Also, for large data records its PDF is

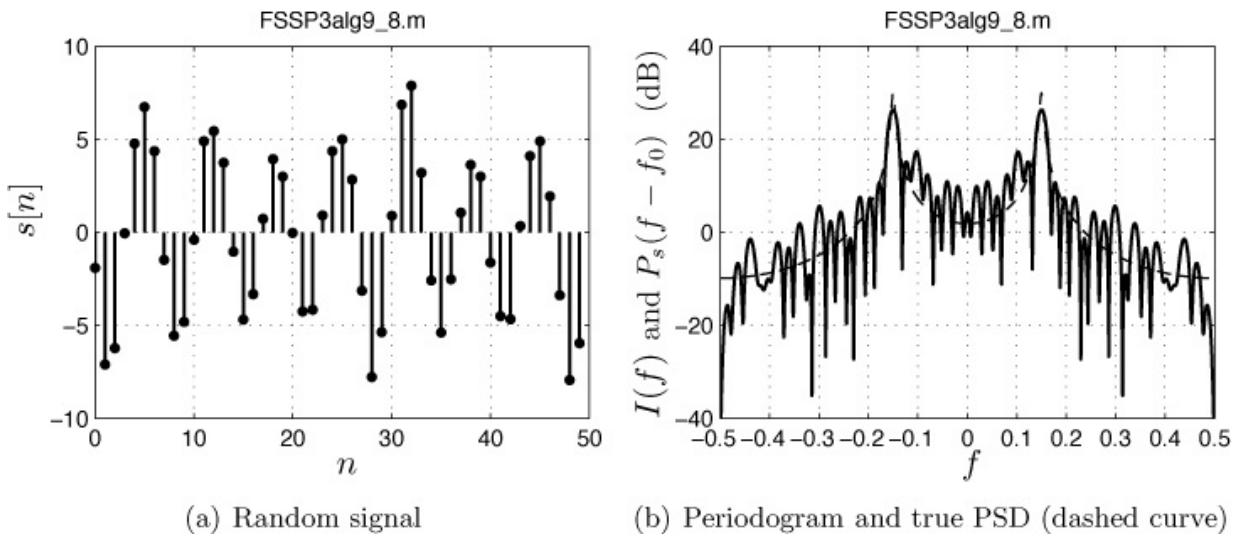
$$\hat{f}_0 \sim \mathcal{N} \left( f_0, \text{var}(\hat{f}_0) \right)$$

where

$$\text{var}(\hat{f}_0) = \frac{N}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} \left( \frac{d \ln P_x(f)}{df} \right)^2 df.$$

## 6. Example

Consider an AR random signal with two parameters  $a[1] = -2r \cos(2\pi f_0)$ ,  $a[2] = r^2$ , where  $r = 0.98$  and  $f_0 = 0.15$ , and  $\sigma_u^2 = 1$ . The PSD has a center frequency of  $f_0 = 0.15$ . We let  $\sigma^2 = 0$ . For  $N = 50$  the data set is shown in [Figure 9.6a](#). In [Figure 9.6b](#) the periodogram is displayed along with the true PSD, and is plotted in dB quantities. From the dashed curve, which is the true PSD, it is found that the bandwidth is about  $B = 0.1$ . This has been obtained by noting where the PSD is down about 20 dB from its peak value at  $f = f_0$ . This frequency is found to be  $f = f_0 + B/2 = 0.2$  so that  $B = 2(0.2 - 0.15) = 0.1$ . Using this value of  $B$  for the minimization of  $J(f_0)$  over the frequency interval  $B/2 = 0.05 < f_0 < 0.45 = 1/2 - B/2$  produces an estimate of the center frequency for this data set of  $\hat{f}_0 = 0.1501$ . A grid search of (9.15) using a frequency spacing of  $1/4096$  was used to obtain this result.



**Figure 9.6: Observed bandpass random signal (no additive WGN) and its periodogram.**

## 7. Explanation

The approximate MLE compares the periodogram spectral estimate to the theoretical PSD for all possible values of  $f_0$ . When the two spectra overlap as much as possible, i.e., when the assumed value of  $f_0$  used in the denominator of (9.15) is near the true value, then this assumed value yields a minimum of  $J(f_0)$ . Otherwise, for an incorrect value of  $f_0$ , it will occur that for some frequency bands the values of  $P_s(f - f_0)$  will be much smaller than  $I(f)$ , giving rise to a large value of  $J(f_0)$ . Also, if we were to use the approximate function of (9.16) we see that the estimate is obtained by maximizing the “spectral correlation” between the periodogram and the signal PSD as it is shifted in frequency.

## 8. Comments/References

See [[Levin 1965](#)] and [[Kay 1993](#), pp. 51–53] for further details.

---

### Exercise 9.8 – Bandpass random signal corrupted by noise

Repeat the example for the data shown in [Figure 9.6a](#) but now add WGN to the random signal. This can be done by letting  $a[1] = -2r \cos(2\pi f_0)$ ,  $a[2] = r^2$ , and  $\sigma_u^2 = 1$  to generate the bandpass signal. As before let  $f_0 = 0.15$  and  $r = 0.98$ . Use  $B = 0.1$  for the bandwidth. Finally, let the WGN

variance be  $\sigma^2 = 100$ . Then, to carry out the exercise:

1. Generate  $N = 50$  samples of the bandpass signal using `s=ARgenda(a,sig2u,N)` (don't forget to use `randn('state', 0)`). Next generate samples of WGN with  $\sigma^2 = 100$  and add to the signal to form  $x[n]$ .
2. For  $P_s(f)$  use  $a_{LP}[1] = -2r$ ,  $a_{LP}[2] = r^2$  so that  $P_s(f)$  is a lowpass PSD centered about  $f = 0$  and has a peak at  $f = 0$ .
3. Run `f0hat=FSSP3alg9_8_sub(x,alp,sig2u,B,sig2)`, where  $x$  is the noise corrupted signal you have generated,  $alp$  are the lowpass AR parameters,  $sig2u$  is  $\sigma_u^2 = 1$ ,  $B$  is the bandwidth, which should be chosen to be 0.1, and  $sig2$  is  $\sigma^2 = 100$ .

Compare your results to that in the example.

---



---

### **Algorithm 9.9 – Multiple sinusoids in WGN - optimal solution**

#### **1. Problem**

Estimate the amplitudes, phases, and frequencies of multiple sinusoids. For a practical implementation a maximum of five sinusoids embedded in WGN can be accommodated.

#### **2. Application area example**

In vibration analysis of machines it is important to determine any predominant resonances since this can lead to catastrophic machine failure [[Santamarida et al. 2000](#)].

#### **3. Data model/assumptions**

$$x[n] = \sum_{i=1}^p A_i \cos(2\pi f_i n + \phi_i) + w[n] \quad n = 0, 1, \dots, N-1 \quad (9.17)$$

#### **4. Estimator**

The frequencies are first estimated by maximizing the function

$$J(\mathbf{f}) = \mathbf{x}^T \mathbf{H}(\mathbf{f}) (\mathbf{H}^T(\mathbf{f}) \mathbf{H}(\mathbf{f}))^{-1} \mathbf{H}^T(\mathbf{f}) \mathbf{x} \quad (9.18)$$

where  $\mathbf{f} = [f_1 \ f_2 \ \dots \ f_p]^T$ , over the region  $0 < f_1 < f_2 < \dots < f_p < 1/2$ . The  $N \times 2p$  matrix  $\mathbf{H}(\mathbf{f})$  is defined as

$$\mathbf{H}(\mathbf{f}) = [\mathbf{c}_1 \ \mathbf{s}_1 \ \mathbf{c}_2 \ \mathbf{s}_2 \dots \mathbf{c}_p \ \mathbf{s}_p]$$

where

$$\mathbf{c}_i = \begin{bmatrix} 1 \\ \cos(2\pi f_i) \\ \vdots \\ \cos[2\pi f_i(N-1)] \end{bmatrix} \quad \mathbf{s}_i = \begin{bmatrix} 0 \\ \sin(2\pi f_i) \\ \vdots \\ \sin[2\pi f_i(N-1)] \end{bmatrix}$$

for  $i = 1, 2, \dots, p$ . Once the estimates of the frequencies, denoted by  $\hat{\mathbf{f}}$ , have been found, the amplitudes and phases are estimated by first computing

$$\hat{\boldsymbol{\alpha}} = \left( \mathbf{H}^T(\hat{\mathbf{f}}) \mathbf{H}(\hat{\mathbf{f}}) \right)^{-1} \mathbf{H}^T(\hat{\mathbf{f}}) \mathbf{x}$$

where  $\hat{\mathbf{f}}$  is a  $p \times 1$  vector and

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \hat{\alpha}_{11} \\ \hat{\alpha}_{21} \\ \vdots \\ \hat{\alpha}_{1p} \\ \hat{\alpha}_{2p} \end{bmatrix}$$

and then

$$\begin{aligned} \hat{A}_i &= \sqrt{\hat{\alpha}_{1i}^2 + \hat{\alpha}_{2i}^2} \\ \hat{\phi}_i &= \arctan \left( \frac{-\hat{\alpha}_{2i}}{\hat{\alpha}_{1i}} \right) \end{aligned}$$

where the arctan function is interpreted as producing an angle in the quadrant associated with the point  $(\hat{\alpha}_{1i}, -\hat{\alpha}_{2i})$ . The MATLAB subprogram entitled `FSSP3alg9_9_sub.m` can be used to implement the estimator for two sinusoids and is contained on the CD.

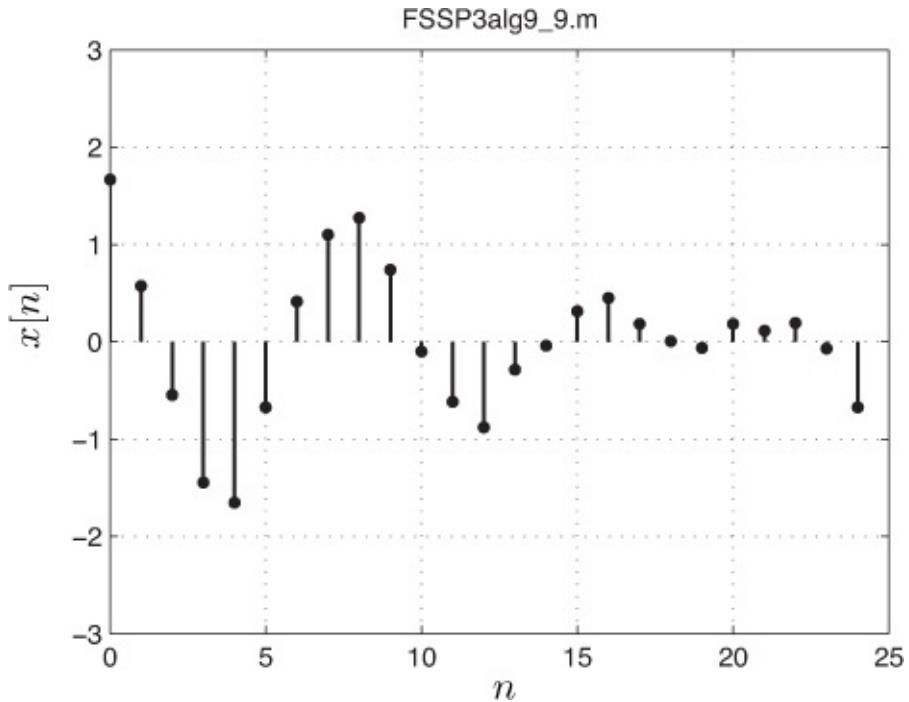
## 5. Performance

The estimator is the MLE and hence is asymptotically optimal in that it attains the CRLB. Its asymptotic performance (as  $N \rightarrow \infty$  and/or a high SNR) is such that the estimator has a multivariate Gaussian PDF and is unbiased. The variances, however, are complicated functions of all the parameters. See [[Kay 1988](#), pg. 414] for the CRLB for complex sinusoids.

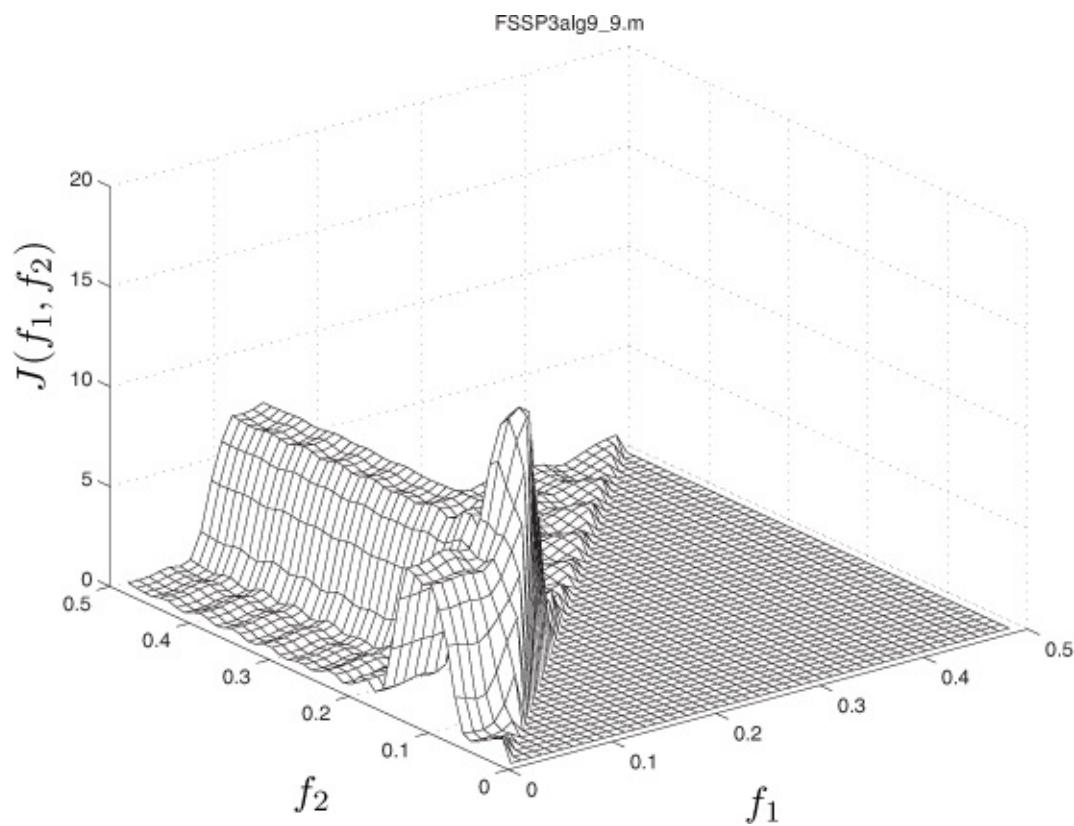
## 6. Example

Consider two sinusoids with the parameters  $A_1 = 1$ ,  $f_1 = 0.11$ ,  $\varphi_1 = 0$ , and

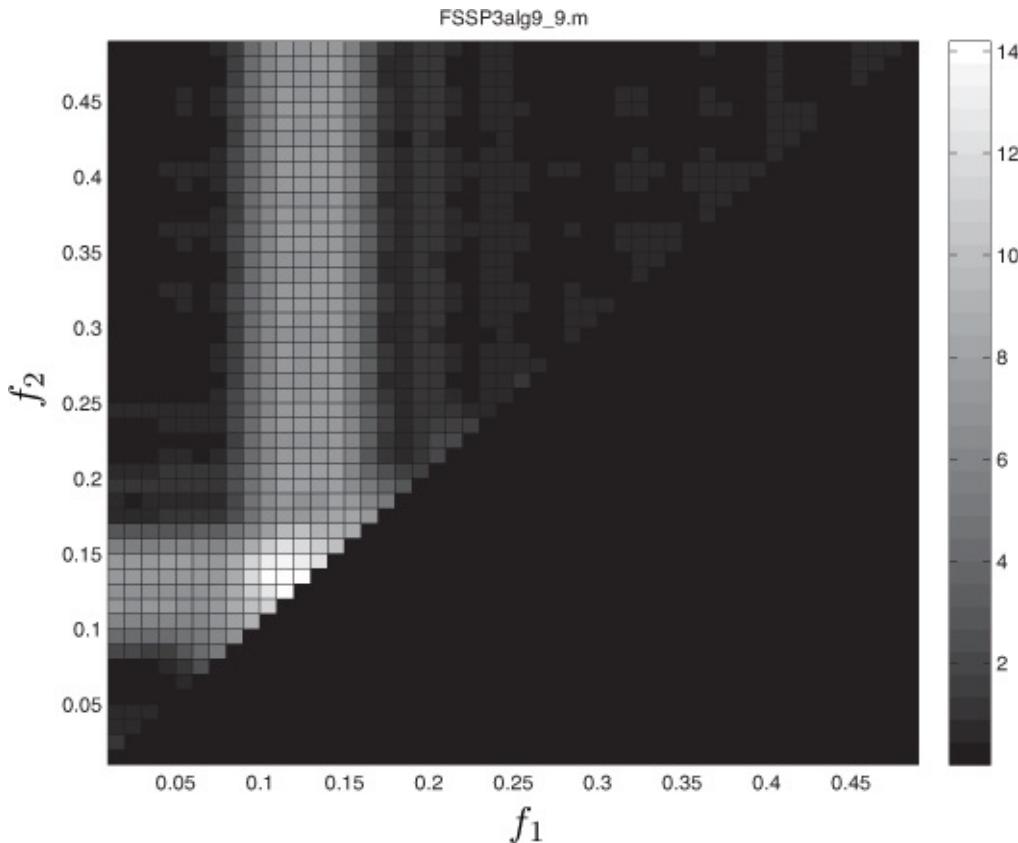
$A_2 = 1$ ,  $f_2 = 0.13$ ,  $\phi_2 = \pi/4$ , and a data record length of  $N = 25$ . The sinusoidal components are unresolvable using a periodogram (see also [Chapter 11](#) for a further description of resolution). The WGN variance is  $\sigma^2 = 0.01$ . The data set is shown in [Figure 9.7](#). The function to be maximized is shown in [Figures 9.8](#) and [9.9](#), where  $J(f_1, f_2)$  has been computed over a frequency grid with spacing of 0.01. This grid is too coarse to yield an accurate estimate but has been used for the rendering of the figures. Using a finer grid spacing of 0.0001 to actually compute the MLE, the frequency estimates are  $\hat{f}_1 = 0.1091$  (true value is  $f_1 = 0.11$ ) and  $\hat{f}_2 = 0.1295$  (true value is  $f_2 = 0.13$ ). The amplitude estimates are  $\hat{A}_1 = 0.9550$  (true value is  $A_1 = 1$ ) and  $\hat{A}_2 = 0.9946$  (true value is  $A_2 = 1$ ). The phase estimates are  $\hat{\phi}_1 = 0.0753$  (true value is  $\phi_1 = 0$ ) and  $\hat{\phi}_2 = 0.7963$  (true value is  $\phi_2 = 0.7854$ ). Note that for the coarser grid spacing used to produce [Figures 9.8](#) and [9.9](#), the frequency estimates are exactly correct due to the fortuitous choice of the sinusoidal frequencies (0.11 and 0.13) and the grid points (... , 0.10, 0.11, 0.12, 0.13,...). See also [Section 8.5](#) for a further discussion of this possible pitfall.



**Figure 9.7: Data for estimation of parameters of two sinusoids in WGN.**



**Figure 9.8: Function to be maximized over  $(f_1, f_2)$  for  $f_1 < f_2$  to find MLE of sinusoidal frequencies – 3D plot. The grid spacing has been chosen to be 0.01 for easier viewing.**



**Figure 9.9: Function to be maximized to find MLE of sinusoidal frequencies – using a gray scale. The grid spacing has been chosen to be 0.01 for easier viewing.**

## 7. Explanation

The MLE fits the sinusoidal signal components to the data using a least squares procedure (see Explanation of [Algorithm 9.2](#)). The same type of procedure as described in [Section 5.4](#) was used to reduce the maximization to that over only the nonlinear frequency parameters.

## 8. Comments/References

For more than about 5 sinusoids it is difficult to maximize  $J(\mathbf{f})$  since this requires a grid search (see also [Section 5.4](#) for the estimation of the parameters of two damped sinusoids). This has led to numerous techniques to obtain MLE type performance without the computational burden [[Kay 1988, Chapter 13](#)]. One of these is given next. Also, see [[Kay and Saha 2000](#)] for one way to reduce the computation of the MLE.



## Exercise 9.9 – A lower SNR

Repeat the example for the data shown in [Figure 9.7](#) but use a noise variance of  $\sigma^2 = 0.25$ . For the grid size use a spacing of 0.0001 between points and be sure to precede the code with `randn('state', 0)`. Compare your frequency estimation results to those in the example. The MATLAB subprogram `FSSP3alg9_9_sub.m` can be used to implement the estimator.

---

---

## Algorithm 9.10 – Multiple sinusoids in WGN - suboptimal solution

### 1. Problem

Estimate the amplitudes, phases, and frequencies of multiple sinusoids (any number of sinusoids) embedded in WGN.

### 2. Application area example

Same as [Algorithm 9.9](#).

### 3. Data model/assumptions

Same as [Algorithm 9.9](#).

### 4. Estimator

The estimator, termed the *principal component AR* estimator, proceeds as follows:

Compute the forward  $\mathbf{H}_f$  and backward  $\mathbf{H}_b$  observation matrices, both of dimension  $(N - L) \times L$ , where  $L$  is the largest integer less than or equal to  $(3/4)N$ , and

$$\mathbf{H}_f = \begin{bmatrix} x[L-1] & x[L-2] & \dots & x[0] \\ x[L] & x[L-1] & \dots & x[1] \\ \vdots & \vdots & \vdots & \vdots \\ x[N-2] & x[N-3] & \dots & x[N-L-1] \end{bmatrix}$$
$$\mathbf{H}_b = \begin{bmatrix} x[1] & x[2] & \dots & x[L] \\ x[2] & x[3] & \dots & x[L+1] \\ \vdots & \vdots & \vdots & \vdots \\ x[N-L] & x[N-L+1] & \dots & x[N-1] \end{bmatrix}$$

and the forward  $\mathbf{h}_f$  and backward  $\mathbf{h}_b$  data vectors, both of dimension  $(N -$

$L) \times 1$  are

$$\mathbf{h}_f = \begin{bmatrix} x[L] \\ x[L+1] \\ \vdots \\ x[N-1] \end{bmatrix} \quad \mathbf{h}_b = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-L-1] \end{bmatrix}$$

Next form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_f \\ \mathbf{H}_b \end{bmatrix}$$

which has dimension  $2(N-L) \times L$  and find the eigenvectors and eigenvalues of the  $L \times L$  matrix  $\mathbf{H}^T \mathbf{H}$ . Arrange the eigenvalues in descending order as  $\lambda_1 > \lambda_2 > \dots > \lambda_{2p} > \lambda_{2p+1} > \dots > \lambda_L$ . Retain the  $2p$  eigenvectors with eigenvalues  $\lambda_1, \dots, \lambda_{2p}$ , which is equivalent to finding the pseudoinverse of the matrix  $\mathbf{H}$ , assumed to have a rank of  $2p$ . Call this  $\mathbf{H}^\#$ . Then, the principal component AR filter parameter solution of length  $L \times 1$  is

$$\hat{\mathbf{a}} = -\mathbf{H}^\# \mathbf{h}$$

where  $\hat{\mathbf{a}} = [\hat{a}[1] \ \hat{a}[2] \ \dots \ \hat{a}[L]]^T$ . Finally, solve for the roots of the polynomial

$$\hat{\mathcal{A}}(z) = 1 + \hat{a}[1]z^{-1} + \hat{a}[2]z^{-2} + \dots + \hat{a}[L]z^{-L}$$

and choose the  $p$  roots that are in the upper half of the  $z$ -plane and are closest to the unit circle, i.e., the roots whose magnitudes are closest to one. Calling these  $z_1, z_2, \dots, z_p$ , the final frequency estimates are

$$\hat{f}_i = \frac{1}{2\pi} \angle z_i$$

for  $i = 1, 2, \dots, p$ . The amplitude and phase estimates are found as in [Algorithm 9.9](#) once the frequency estimates are found. The MATLAB subprogram `FSSP3alg9_10_sub.m` can be used to implement the estimator and is contained on the CD.

## 5. Performance

There are no analytical results for the performance. The estimator has been found empirically to produce good estimates and attains the CRLB for high enough SNRs.

## 6. Example

Consider the case of 5 sinusoids with  $N = 100$  and  $\sigma^2 = 0.01$ . The true

parameters and the estimated ones are shown in [Table 9.2](#). The estimates have been arranged in ascending order of frequencies. Note that the frequency estimates are good except for the last sinusoid, whose estimated frequency is 0.3081 and is not near any of the true frequencies. Its amplitude is only 0.0367, which leads one to suspect it might be due to noise. Also, it appears that the sinusoids with true frequencies of 0.1300 and 0.1305 have not been “resolved” in that the amplitude associated with the frequency estimate of 0.1304 is 2.0132. Hence, the sinusoidal component estimates, with frequencies at 0.1300 and 0.1305, have been combined into one at 0.1304. This also explains why there is a spurious frequency at 0.3081 since the estimator has only “seen” 4 frequencies, yet we have asked for 5 frequency components.

**Table 9.2: True and estimated sinusoidal parameters using principal component AR estimator.**

True $A$	$\hat{A}$	True $\phi$	$\hat{\phi}$	True $f$	$\hat{f}$
1	0.9038	0	0.1190	0.1000	0.0996
1	0.9290	0	-0.2120	0.1100	0.1108
1	2.0132	0	-0.0415	0.1300	0.1304
1	1.0346	0	-0.0325	0.1305	0.1401
1	0.0367	0	0.1413	0.1400	0.3081

## 7. Explanation

The method described leverages the good resolution properties of the AR spectral estimator (see [Chapter 11](#)). For  $p$  sinusoids in WGN an AR PSD model of order  $2p$  yields sharp lines at the  $p$  sinusoidal frequencies. When noise is added, however, the resolution is severely impacted. The principal component solution attempts to mitigate the noise effects by increasing the model order to  $L \gg 2_p$  (in this example,  $L = 75 \gg 10 = 2_p$ ) to capture the spectral lines while at the same time reducing the spurious noise peaks by retaining only the important principal components.

## 8. Comments/References

See [[Tufts and Kumaresan 1982](#), [Kay 1988](#), pp. 426–428].



## Exercise 9.10 – Comparison to MLE

Repeat [Exercise 9.9](#) which estimated the parameters for two sinusoids at a low SNR. Are the estimates using this approach as good? Be sure to precede the code with `randn ('state', 0)`. The MATLAB subprogram `FSSP3alg9_10_sub.m` can be used to implement the estimator.

---

## 9.3. Enhancing Signals Corrupted by Noise/Interference

We next summarize algorithms that can be used to enhance, i.e., extract, signals from noise and/or interference. The principal difference between the approaches hinges upon whether the signals are deterministic or random, and whether there is available an extra “copy” of the noise and/or interference. By the latter we mean the availability of noise and/or interference (but without the signal) that is highly correlated with the noise and/or interference within the data set that contains the signal. Also, we distinguish between noise corruption, for example, WGN and interference corruption, for example, a 60 Hz interferer.

---

### Algorithm 9.11 – Estimation of periodic random signal in noise

#### 1. Problem

Extract a periodic Gaussian random signal from WGN.

#### 2. Application area example

Voiced speech sounds that are periodic can be enhanced using a variant of comb filtering [[Wen et al. 2010](#)].

#### 3. Data model/assumptions

The data model is

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the periodic Gaussian random signal to be enhanced and  $w[n]$  is WGN with known variance  $\sigma^2$ . The signal has a period of  $M$  samples, which is unknown, but is constrained to be  $M_{\min} \leq M \leq M_{\max}$ . No other a priori knowledge about the signal (such as its covariance) is assumed known. See also [Section 3.5.3](#) for a discussion of *deterministic periodic*

*signals.*

#### 4. Estimator

We first estimate the period  $M$  and denote this estimate by  $\hat{M}$ . Then, assuming that the signal over the basic period is  $s[n] = g[n]$  for  $n = 0, 1, \dots, \hat{M} - 1$ , the estimator for the first period is

$$\hat{s}[n] = \hat{g}[n] = \frac{1}{K} \sum_{r=0}^{K-1} x[n + r\hat{M}] \quad n = 0, 1, \dots, \hat{M} - 1 \quad (9.19)$$

where  $K$  is the largest integer  $\leq N/\hat{M}$ . (We can improve the estimate slightly by averaging over all the samples available in the data record if  $N/\hat{M}$  is not an integer.) The remaining periods are estimated by replicating  $\hat{g}[n]$  so that  $\hat{s}[\hat{M}] = \hat{g}[0]$ ,  $\hat{s}[\hat{M} + 1] = \hat{g}[1]$ , etc. The period is first estimated by maximizing the following function

$$J(M) = \sum_{\substack{i=1 \\ I(i/M)/\sigma^2 > 1}}^{M/2-1} \left( \frac{I(i/M)}{\sigma^2} - \ln \frac{I(i/M)}{\sigma^2} - 1 \right) - \frac{M/2 - 1}{2} \ln N \quad (9.20)$$

over  $M_{\min} \leq M \leq M_{\max}$ , where  $I(f)$  is the periodogram given as

$I(f) = (1/N) |\sum_{n=0}^{N-1} x[n] \exp(-j2\pi f n)|^2$ . The MATLAB subprogram titled `FSSP3alg9_11_.m` can be used to implement the estimator and is contained on the CD.

#### 5. Performance

The estimator is an MLE and therefore is asymptotically (as  $N \rightarrow \infty$ ) optimal in that it attains the CRLB.

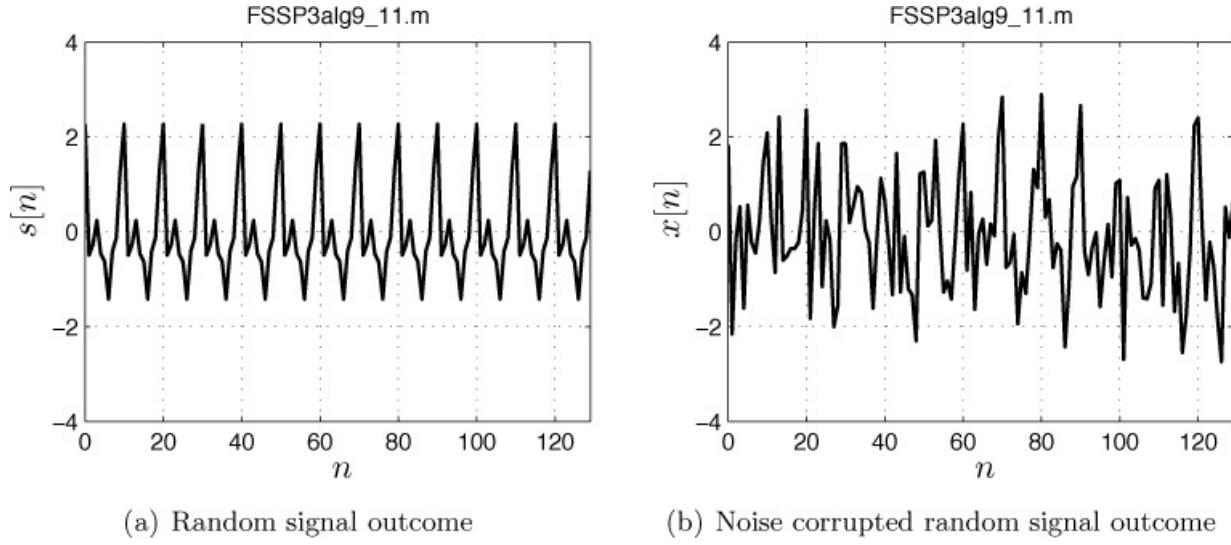
#### 6. Example

Consider the data consisting of 4 harmonically related sinusoids (a Fourier expansion of a periodic signal of period  $1/f_0$ ) embedded in WGN with variance  $\sigma^2 = 1$ . An outcome of the periodic random signal embedded in noise is given by

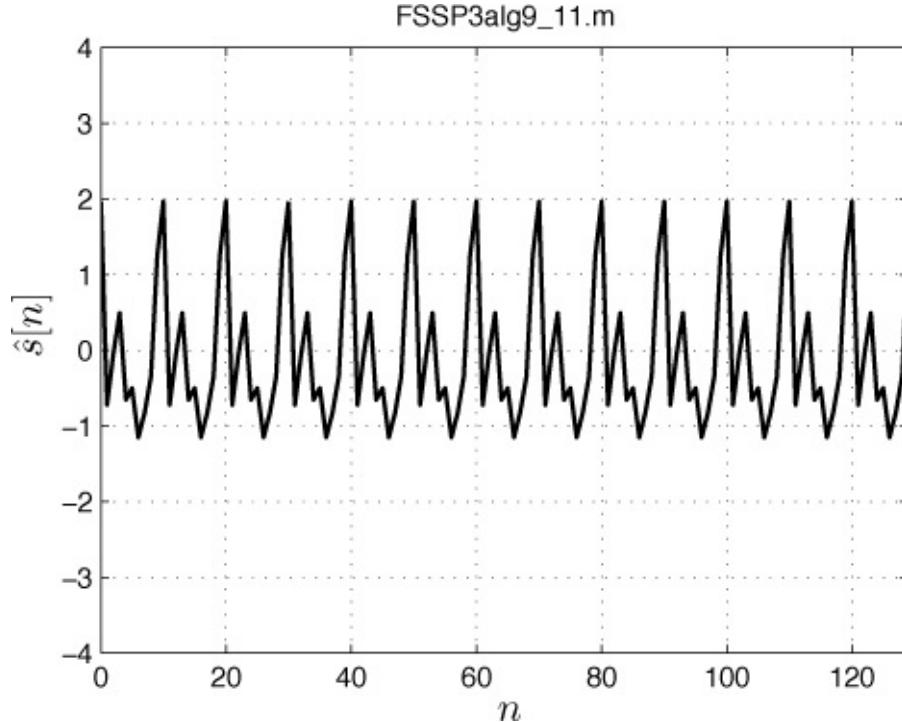
$$\begin{aligned} x[n] = & \cos(2\pi f_0 n) + \frac{1}{\sqrt{2}} \cos[2\pi(2f_0)n + \pi/3] + \frac{1}{2} \cos[2\pi(3f_0)n + \pi/7] \\ & + \frac{1}{2} \cos[2\pi(4f_0)n + \pi/9] + w[n] \quad n = 0, 1, \dots, N - 1 \end{aligned}$$

for  $f_0 = 1/M = 0.1$  and  $N = 130$  as shown in [Figure 9.10](#). Computing  $J(M)$  using  $M_{\min} = 5$  and  $M_{\max} = 50$  yields the maximum value at  $\hat{M} = 10$  so

that the period estimate is perfect. The reconstructed signal is shown in [Figure 9.11](#) and is seen to be nearly identical to the true signal.



**Figure 9.10: Outcome of random periodic signal and observed noisy data. The points have been connected by straight lines for easier viewing.**



**Figure 9.11: Estimated signal. The points have been connected by straight lines for easier viewing.**

## 7. Explanation

The estimator can be viewed as a *comb filter* in that its frequency response is a set of narrow bandpass filters with center frequencies  $f = 0, 1/M, 2/M, \dots, M/2 - 1$  for  $M$  even. This is because the signal is periodic and has energy only at these harmonic frequencies. The bandpass filters attempt to pass the signal components but filter out the white noise, which has power at all frequencies. The greater the number of periods of the signal that are in  $x[n]$ , i.e.,  $K$ , the narrower these bandpass filters become and therefore the more noise that is filtered out.

## 8. Comments/References

See [Section 3.5.3](#) for the MLE of a periodic deterministic signal with a known period. More details about this estimator can be found in [[Kay 1998](#), pp. 316–327]. If the PSD of the noise is not white, this estimator can still be used but the performance may be slightly degraded.



---

### Exercise 9.11 – Another periodic signal

Assume that  $s[n] = 1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, \dots, -1, -1, -1, -1$ , which is a periodic pulse train with period  $M = 10$ . For  $N = 130$  samples embed  $s[n]$  in WGN with variance  $\sigma^2 = 1$  and use `FSSP3alg9_11_sub.m` to extract the signal from the noise. In using the subprogram let  $M_{\min} = 5$  and  $M_{\max} = 50$ . Plot the true signal along with the estimated signal.



---

### Algorithm 9.12 – Estimation of random signal in noise

#### 1. Problem

Extract a random signal from noise. The signal is assumed to be the outcome of a random process.

#### 2. Application area example

A typical use is to enhance images corrupted by noise, which requires a straightforward extension to a two-dimensional signal. An example is in high resolution electron microscope images [[Marks 1996](#)].

#### 3. Data model/assumptions

The Gaussian signal to be extracted  $s[n]$  is assumed to be embedded in WGN with known variance  $\sigma^2$  so that we observe

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1.$$

The signal is assumed to be the outcome of a zero mean Gaussian random process with a known  $N \times N$  covariance matrix  $\mathbf{C}_s$ .

#### 4. Estimator

The estimator is given as

$$\hat{\mathbf{s}} = \mathbf{C}_s(\mathbf{C}_s + \sigma^2 \mathbf{I})^{-1} \mathbf{x} \quad (9.21)$$

where  $\hat{\mathbf{S}} = [\hat{\mathbf{s}}[0] \ \hat{\mathbf{s}}[1] \ \dots \ \hat{\mathbf{s}}[N - 1]]^T$ . If the signal is from a stationary random process, then the covariance matrix becomes an autocorrelation matrix  $\mathbf{R}_s$  given by

$$\mathbf{C}_s = \mathbf{R}_s = \begin{bmatrix} r_s[0] & r_s[1] & \dots & r_s[N - 1] \\ r_s[1] & r_s[0] & \dots & r_s[N - 2] \\ \vdots & \vdots & \ddots & \vdots \\ r_s[N - 1] & r_s[N - 2] & \dots & r_s[0] \end{bmatrix} \quad (9.22)$$

where  $r_s[k]$  is the autocorrelation sequence for  $s[n]$ . For this case, if the PSD is given by  $P_s(f)$ , then an approximate estimate (which avoids the inversion of the  $N \times N$  autocorrelation matrix) is to filter the data  $x[n]$  with the noncausal filter with frequency response

$$H(f) = \frac{P_s(f)}{P_s(f) + \sigma^2}. \quad (9.23)$$

It is called a *Wiener smoothing filter*. The MATLAB subprogram entitled FSSP3alg9\_12\_sub.m can be used to implement the estimator of (9.21) and (9.22) if the autocorrelation sequence is given. It is contained on the CD.

#### 5. Performance

The estimator given by (9.21) is called a *matrix Wiener smoother*. It is optimal in that it minimizes the Bayesian mean square error of the estimator, assuming the signal and noise are Gaussian (see [Theorem 8.2.2](#) for a scalar parameter). The minimum mean square error for the estimator  $\hat{\mathbf{s}}[n]$  for  $n = 0, 1, \dots, N - 1$  is the  $[n, n]$  diagonal element of the  $N \times N$  mean square error matrix

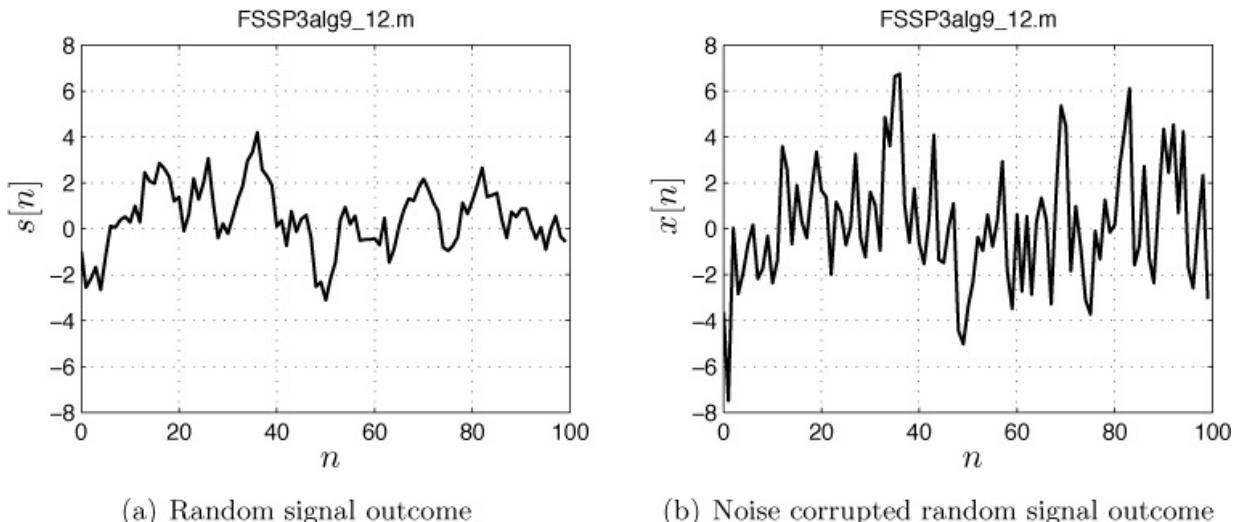
$$\mathbf{M}_s = \mathbf{C}_s - \mathbf{C}_s(\mathbf{C}_s + \sigma^2 \mathbf{I})^{-1} \mathbf{C}_s.$$

## 6. Example

Consider a random signal that is an outcome of a Gaussian AR random process of order one (see [Section 4.4](#)). The parameters of the AR process are  $a[1] = -0.9$  and  $\sigma_u^2 = 1$ . It is embedded in WGN with variance  $\sigma^2 = 5$ .

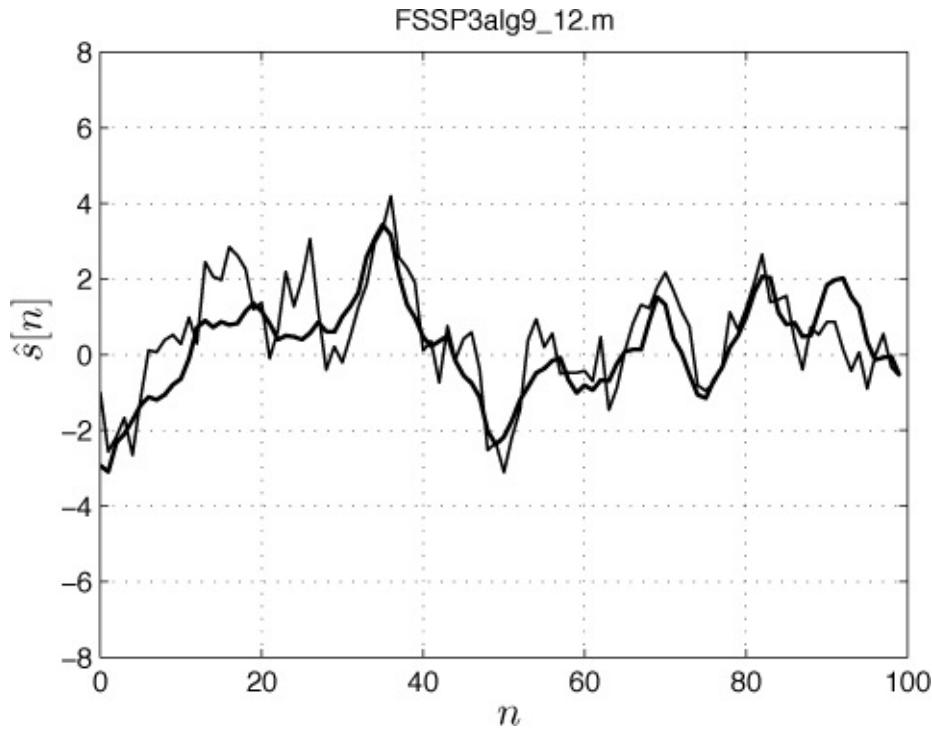
The signal and the noise corrupted signal are shown in [Figure 9.12](#). The signal is an outcome of a stationary random process so that we can estimate it using [\(9.21\)](#) with the covariance matrix replaced by the autocorrelation matrix. The autocorrelation sequence is given by

$$r_s[k] = \frac{\sigma_u^2}{1 - a^2[1]} (-a[1])^{|k|}. \quad (9.24)$$



**Figure 9.12: Outcome of random signal and observed noisy data. The points have been connected by straight lines for easier viewing.**

The Wiener smoothed signal along with the true signal is shown in [Figure 9.13](#). Note that it is closer to the true signal than the noise corrupted signal but exhibits some smoothing.



**Figure 9.13: Estimated signal using the matrix Wiener smoother (heavy line) and the true signal (light line). The points have been connected by straight lines for easier viewing.**

## 7. Explanation

The Wiener smoother cannot separate out the signal from the noise perfectly since the PSDs overlap in frequency. It attempts to reduce the WGN as much as possible in an effort to reduce the overall Bayesian mean square error. This results in reducing the power in some of the high frequency signal bands, resulting in a smoothed estimate.

## 8. Comments/References

The autocorrelation matrix of (9.22) can be efficiently inverted using Levinson algorithm [[Kay 1988](#), pp. 176–177]. See [[Kay 1993](#), pp. 400–406] for further details of the Wiener smoother. Note that the Wiener smoother can be used even if the signal and/or the noise are not Gaussian, in which case it is the optimal *linear estimator* only.



## Exercise 9.12 – A higher SNR

Repeat the example for the data shown in [Figure 9.12a](#) but change the noise variance to  $\sigma^2 = 1$ , which results in a higher SNR. You can generate the random signal outcome and data by using

```
randn('seed',0);
a=-0.9;
sig2u=1;
sig2=1;
N=100;
s=ARgendata(a,sig2u,N);
x=s+sqrt(sig2)*randn(N,1);
```

and then implement [\(9.21\)](#) using `FSSP3alg9_12_sub(x, rs, sig2)`. For the autocorrelation sequence `rs` use [\(9.24\)](#). Compare your results to those in [Figure 9.13](#).

---



---

### **Algorithm 9.13 – Estimation of a signal in noise and interference**

#### **1. Problem**

Reduce the interference of a signal (deterministic or random) by subtracting out an estimate of the interference.

#### **2. Application area example**

An obvious example is to reduce sinusoidal interference [[Glover 1977](#)]. A more interesting one is the reduction of the mother's heart beat in an effort to extract an underlying fetal heart beat [[Widrow \*et al.\* 1975](#)].

#### **3. Data model/assumptions**

For this algorithm it is assumed that there is an additional data source that is correlated with the interference that we wish to mitigate. The data is

$$x[n] = s[n] + w[n] + i[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the signal of interest,  $w[n]$  is observation noise, and  $i[n]$  is the interference. Additionally, we have access to data  $x_R[n]$  for  $n = 0, 1, \dots, N - 1$ , sometimes called *reference data*, that is correlated with  $i[n]$  and hopefully uncorrelated with  $s[n]$ . Ideally, we would like to have  $x_R[n] = i[n]$ , but this is not possible in practice.

#### **4. Estimator**

The interference is estimated as  $\hat{i}[n]$ , which is given as the output of an FIR

filter with the reference data at the input as

$$\hat{i}[n] = \sum_{l=0}^{p-1} h[l]x_R[n-l] \quad n = p-1, p, \dots, N-1.$$

The FIR filter coefficients  $\mathbf{h} = [h[0] \ h[1] \dots \ h[p-1]]^T$  are found as

$$\mathbf{h} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$$

where  $\mathbf{x} = [x[p-1] \ x[p] \ \dots \ x[N-1]]^T$  and  $\mathbf{H}$  is the  $(N-p+1) \times p$  matrix

$$\mathbf{H} = \begin{bmatrix} x_R[p-1] & x_R[p-2] & \dots & x_R[0] \\ x_R[p] & x_R[p-1] & \dots & x_R[1] \\ \vdots & \vdots & \ddots & \vdots \\ x_R[N-1] & x_R[N-2] & \dots & x_R[N-p] \end{bmatrix}.$$

The estimate of the signal becomes  $\hat{\mathbf{s}}[n] = \mathbf{x}[n] - \hat{\mathbf{i}}[n]$  for  $n = p-1, p, \dots, N-1$ . The estimated interference sequence is found from

$$\begin{bmatrix} \hat{i}[p-1] \\ \vdots \\ \hat{i}[N-1] \end{bmatrix} = \mathbf{H}\mathbf{h}.$$

The MATLAB subprogram `FSSP3alg9_13_sub.m`, which is contained on the CD, can be used to implement the estimator.

## 5. Performance

There are no optimality properties.

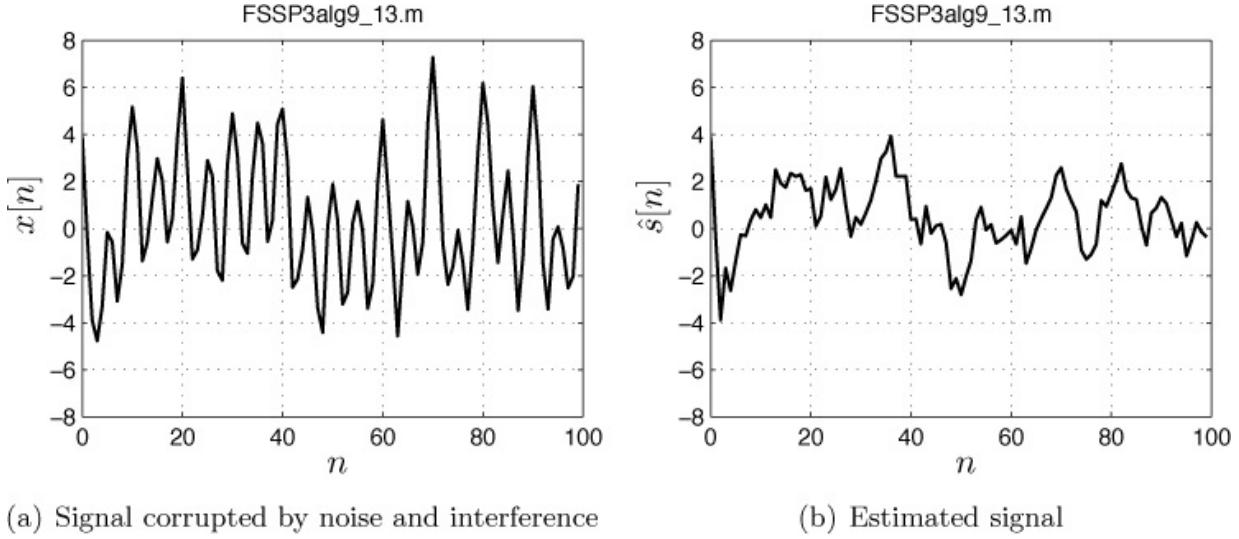
## 6. Example

We consider the signal shown in [Figure 9.12a](#) but add WGN with variance  $\sigma^2 = 0.01$  and also the sinusoidal interference given by

$$i[n] = 2 \cos(2\pi(0.1)n) + 3 \cos(2\pi(0.2)n) \quad n = 0, 1, \dots, N-1.$$

The noise and interference corrupted signal is shown in [Figure 9.14a](#). We have access to the reference data

$$x_R[n] = \cos(2\pi(0.1)n + \pi/4) + \cos(2\pi(0.2)n + \pi/8) \quad n = 0, 1, \dots, N-1 \quad (9.25)$$



**Figure 9.14: Random signal outcome of [Figure 9.12a](#) corrupted by noise and interference and its estimate. The points have been connected by straight lines for easier viewing.**

which is seen to match the interference frequencies but not their amplitudes and phases. The purpose of the FIR filter is to modify  $x_R[n]$  so that the sinusoidal amplitudes and phases match those of the interference. Then, this estimate  $\hat{i}[n]$  can be subtracted from  $x[n]$  to “cancel” the interference. Since the interference consists of two sinusoids, the FIR filter will need to have the frequency response of  $|H(0.1)| = 2$ ,  $\angle H(0.1) = -\pi/4$  and  $|H(0.2)| = 3$ ,  $\angle H(0.2) = -\pi/8$ . This requires the frequency response to be altered at two frequencies so that  $p = 4$  should be used. The results are shown in [Figure 9.14b](#). They are nearly identical to the original signal (see [Figure 9.12a](#)) except for the first 3 points (see Comments/References for the reasons).

## 7. Explanation

The FIR filter coefficients are found by minimizing the least squares error

$$J = \sum_{n=p-1}^{N-1} \left( x[n] - \sum_{l=0}^{p-1} h[l] x_R[n-l] \right)^2.$$

As such it attempts to “cancel out”  $x[n]$ . It is able to accomplish this by having a frequency response that is close to zero whenever there are large frequency components in  $x[n]$ . These frequency components are assumed to be due to interference. Thus, for proper operation the signal should have a broad PSD with no narrowband components. If not, the filter, which

cannot distinguish between what is signal and what is interference, will attempt to “cancel out” the signal as well. Hence, the requirement that the signal and interference are “uncorrelated”.

## 8. Comments/References

This signal extraction approach is usually referred to as an *adaptive noise canceler*. Note that the initial  $p$  samples of the interference are not estimated because of the lack of data samples prior to  $n = 0$  needed for the FIR filter. The interference canceler is usually implemented sequentially for real-time operation and for nonstationary environments. See also [[Haykin 1991](#)] and [[Kay 1993](#), pp. 268–273]. A similar problem was discussed in [Exercises 6.1, 6.2](#).



---

### Exercise 9.13 – Frequency response of adaptive filter

In this exercise we determine the frequency response of the adaptive filter used to produce the estimate shown in [Figure 9.14b](#). To obtain the  $x[n]$  data use load `FSSP3exer9_13`. Next, run `FSSP3alg9_13_sub.m` by using the call `[shat hhat]=FSSP3alg9_13_sub(x, xR, p)`, where  $x_R[n]$  is given by [\(9.25\)](#), and  $p = 4$ . The output `hhat` will be the FIR filter coefficients  $[h[0] \ h[1] \ h[2] \ h[3]]^T$  in vector form. Find the frequency response of this filter and explain what happens at the interference frequencies of  $f = 0.1$  and  $f = 0.2$ . To do so compute  $H(f) = \sum_{k=0}^3 h[k] \exp(-j2\pi f k)$  and plot  $|H(f)|$  and  $\angle H(f)$  versus  $f$  for  $0 \leq f \leq 1/2$ .



---

## References

- Adams, M.D., “Lidar Design, Use, and Calibration Concepts for Correct Environmental Detection”, *IEEE Trans. on Robotics and Automation*, pp. 753–761, Dec. 2000.
- Burdic W.W., *Underwater Acoustic System Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Glover, J.R., Jr, “Adaptive Noise Canceling Applied to Sinusoidal Interferences”, *IEEE Trans. Acoustics, Speech and Signal Processing*, pp. 484–491, Dec. 1977.

- Haykin, S., *Communication Systems*, J. Wiley, NY, 1994.
- Haykin, *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- Johnson, D.H., D.E. Dudgeon, *Array Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory*, Vol. I, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory*, Vol. II, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- Kay, S., S. Saha, “Mean Likelihood Frequency Estimation”, *IEEE Trans. on Signal Processing*, July 2000.
- Lai, X., H. Torp, “Interpolation Methods for Time-delay Estimation Using Crosscorrelation Method for Blood Velocity Measurement”, *IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control*, pp. 277–289, March 1999.
- Levin, M.J., “Power Spectrum Parameter Estimation”, *IEEE Trans. on Information Theory*, pp. 100–107, Jan. 1965.
- Marks, L.D., “Wiener-filter Enhancement of Noisy HREM Images”, *Ultramicroscopy*, pp. 43–52, January 1996.
- McAulay, R., T. Quartieri, “Speech Analysis/Synthesis Based on a Sinusoidal Representation”, *IEEE Trans. Acoustics, Speech and Signal Processing*, pp. 744–754, Aug. 1986.
- Narayanan, R.M., M. Dawood, “Doppler Estimation Using a Coherent Ultrawideband Random Noise Radar”, *IEEE Trans. Antennas and Propagation*, pp. 868–878, June 2000.
- Santamarida, I., C. Pantaleod, J.S. Ibanez, “A Comparative Study of High-Accuracy Frequency Estimation Methods”, *Mechanical Systems and Signal Processing*, pp. 1–17, 2000.
- Skolnik, M., *Introduction to Radar Systems*, McGraw-Hill, NY, 1980.
- Tufts, D., R. Kumaresan, “Estimation of Frequencies of Multiple Sinusoids: Making Linear Prediction Perform Like Maximum Likelihood”, *Proc. IEEE*, pp. 975–982, 1982.

- Urick, R.J., *Principles of Underwater Sound*, McGraw-Hill, NY, 1975.
- Yingyong Q., R.E. Hillman, C. Milstein, “The Estimation of Signal-to-Noise Ratio in Continuous Speech for Disordered Voices”, *Journal of Acoustical Society of America*, pp. 2532–2535, 1999.
- Wen, J., X. Liu, M.S. Scordilis, L. Han, “Speech Enhancement Using Harmonic Emphasis and Adaptive Comb Filtering”, *IEEE Trans. Audio, Speech, and Language Processing*, pp. 356–368, Feb. 2010.
- Widrow, B., J.R. Glover, Jr., J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, E. Dong, Jr., R.C. Goodlin, “Adaptive Noise Cancelling: Principles and Applications”, *IEEE Proceedings*, pp. 1692–1716, Dec. 1975.
- Zrnic, D.S., “Estimation of Spectral Moments for Weather Echos”, *IEEE Trans. on Geoscience and Remote Sensing*, pp. 113–128, 1979.

## Appendix 9A. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

- 9.1** For  $r = 0.6$  the estimate is  $\hat{A} = 0.7895$  and hence the error is  $-0.2105$ . For  $r = 0.9$  we had  $\hat{A} = 0.9893$  for an error of only  $-0.0107$  so that the smaller the value of  $r$ , the larger the error in general. This is due to the decreased signal energy as  $r$  decreases, and hence, the degradation in SNR. You can run the program `FSSP3exer9_1.m`, which is contained on the CD, to obtain the results.
- 9.2** For the increased noise variance the estimates become  $\hat{A} = 0.8195$  and  $\hat{\phi} = 1.1643$ . These are much poorer than in the example and probably not usable. You can run the program `FSSP3exer9_2.m`, which is contained on the CD, to obtain the results.
- 9.3** When the noise variance is zero, the parameter estimates will be perfect. You can run the program `FSSP3exer9_3.m`, which is contained on the CD, to obtain the results. Note that this may not be the case if the true frequency is not at one of the frequencies used to evaluate  $J(f_0)$ .
- 9.4** If the pulse is changed to a square one, then the estimated time delay is again  $\hat{n}_0 = 50$ , with no error. You can run the program `FSSP3exer9_4.m`, which is contained on the CD, to obtain the results.
- 9.5** For  $\beta = \pi/8$  the width of the spatial periodogram becomes larger. For  $\beta =$

$\pi/16$ , the widening leads to a shift in the peak to  $\beta = 0$ . This result is analogous to that observed in [Figure 1A.1](#), in which the positive and negative spectral peaks merge at  $f = 0$ . The CRLB actually goes to infinity at  $\beta = 0$  (see [\(9.11\)](#)). You can run the program FSSP3exer9\_5.m, which is contained on the CD, to obtain the results.

**9.6** When  $N = 2000$  the variance of the estimator is 0.001 and therefore one standard deviation is  $\sqrt{0.001} = 0.0316$ . The estimate for this data record length is  $\hat{\sigma}^2 = 0.9778$ , which is within one standard deviation of the true value of  $\sigma^2 = 1$ . You can run the program FSSP3exer9\_6.m, which is contained on the CD, to obtain the results.

**9.7** The estimate of  $P_0$  using the information of the PSD, i.e., [\(9.14\)](#), is  $\hat{P}_0 = 0.9827$ , while the estimate using [\(9.13\)](#) is  $\hat{P}_0 = 0.8248$ , clearly much poorer. You can run the program FSSP3exer9\_7.m, which is contained on the CD, to obtain the results.

**9.8** The estimate of the center frequency is now  $\hat{f}_0 = 0.1499$ , which is very close to the true value of 0.15. This is because the peak of the true PSD as seen from [Figure 9.6b](#) is about 30 dB. The noise PSD is  $P_w(f) = \sigma^2 = 100$  or 20 dB, which is well below this peak level. Thus, the effect of noise is very small. You can run the program FSSP3exer9\_8.m, which is contained on the CD, to obtain the results.

**9.9** For this lower SNR, which is  $10 \log_{10} A^2/(2\sigma^2) = 3$  dB, the frequency estimates are  $f_1 = 0.1056$  and  $f_2 = 0.1276$ . Below this SNR the frequency estimates are very poor. You can run the program FSSP3exer9\_9.m, which is contained on the CD, to obtain the results.

**9.10** For this low SNR, the frequency estimates are  $f_1 = 0.1023$  and  $f_2 = 0.1287$ , which are comparable to the MLE performance. You can run the program FSSP3exer9\_10.m, which is contained on the CD, to obtain the results.

**9.11** You should obtain an estimate of the periodic signal which is very close to the true one. You can run the program FSSP3exer9\_11.m, which is contained on the CD, to obtain the results.

**9.12** You should obtain an estimate of the AR signal that is less smoothed than that seen in [Figure 9.13](#). Increasing the observation noise variance  $\sigma^2$  has the effect of attenuating all the frequency components that are passed by the Wiener smoother. You can run the program

FSSP3exer9\_12.m, which is contained on the CD, to obtain the results.

**9.13** The filter coefficients are  $h[0] = -1.2772$ ,  $h[1] = 6.1242$ ,  $h[2] = -5.1523$ ,  $h[3] = 2.6033$ . The filter frequency response is

$$|H(0.1)| = 1.7385 \quad |H(0.2)| = 2.9616$$

$$\angle H(0.1) = -0.2364\pi \quad \angle H(0.2) = -0.1406\pi$$

while the theoretical values are 2, 3,  $-0.25\pi$ , and  $-0.125\pi$ , respectively. You can find the estimated values from your on-screen plot by using the MATLAB *data cursor* button.

# Chapter 10. Algorithms for Detection

## 10.1. Introduction

[Chapter 9](#) described commonly used algorithms for parameter estimation. We now turn our attention to the problem of signal detection. These detection algorithms are used in many applications, including radar, sonar, communications, biomedical signal processing of EEGs, ECGs, etc., and image processing, to name just a few. In this chapter we follow the same format in describing the algorithms as in [Chapter 9](#). As before, explicit MATLAB implementations in the form of MATLAB subprograms are provided for the user. A listing of the detection algorithms is given in [Table 10.1](#).

**Table 10.1: Summary of detection algorithms.**

Algorithm	Description	Signal type	Noise type
10.1	Replica-correlator (matched filter)	Known signal	WGN
10.2	Limiter-replica-correlator	Known signal	IID noise
10.3	Generalized matched filter	Known signal	Correlated Gaussian noise
10.4	Estimator-correlator	Random signal	WGN
10.5	Energy detector	Unknown signal	WGN
10.6	GLRT - unknown amplitude	Partially known signal	WGN
10.7	GLRT - unknown $A, \phi$	Partially known sinusoid	WGN
10.8	GLRT - unknown $A, \phi, f_0$	Partially known sinusoid	WGN
10.9	GLRT - unknown arrival time	Partially known signal	WGN

To make the detection algorithms more easily accessible to the practitioner we will categorize them according to the characteristics of the signal we wish to detect. These characteristics take the form of prior knowledge about the signal, which is *critical* to the implementation and successful performance of the algorithm. We will therefore, delineate the various types of signals to be detected

in the following way.

**Model 1.** Signal with known form. For example, we might wish to detect a sinusoidal signal given by

$$s[n] = A \cos(2\pi f_0 n + \phi) \quad n = 0, 1, \dots, N - 1 \quad (10.1)$$

where all the values of  $s[n]$  are *completely known*. This leads to a *replica-correlator* (also known as a *matched filter*), the limiter-replica-correlator, and the generalized matched filter.

**Model 2.** Signal with unknown form. If the form of the signal is not known, then we usually model it as the outcome of a random process (see [Chapter 4](#), which describes “noise” models that can also be used for the unknown form signal). This modeling leads to the *estimator-correlator* and *energy detectors*.

**Model 3.** Signal with known form but with some unknown parameters. A signal that is known except for some parameters might be that given by [\(10.1\)](#) but with any of the parameters  $\{A, f_0, \phi\}$  unknown. This leads to a *generalized likelihood ratio test* (GLRT) detector.

Since the a priori knowledge about the signal is critical to the choice of an appropriate detector, the reader should review [Chapter 3](#), which describes models 1 and 3, and also [Chapter 4](#), which describes model 2. For almost all the detection algorithms to be described, the assumption will be that the signal is embedded in *Gaussian noise*, either *white Gaussian noise* (WGN) or else *colored Gaussian noise* (with *correlated Gaussian noise* as a generalization for nonstationary noise). These are the noise models listed in [Table 4.1](#). The one exception is the detection of a known signal in *independent and identically distributed (IID) nonGaussian noise*, which admits a fairly straightforward algorithm. The reader should be cautioned that all *the algorithms assume that the noise PDF is completely known*. In practice, this is seldom the case, and so the practitioner must estimate the noise characteristics using the methods outlined in [Chapters 6 and 11](#). When the noise PDF cannot be determined (possibly because it is changing), then the best we can do is to attempt to estimate it *on-line*. This means utilizing a detection algorithm that is adaptive (see [Information Acquisition](#) description for [Figure 2.1](#)). Some more advanced methods for accomplishing this goal are described in [Chapter 9](#) of [[Kay 1998](#)] and we refer the reader there. Suffice it to say the algorithms that adapt to noise characteristics can be quite complicated.

As in the previous chapter we assume that the data is real-valued. For sonar and radar applications that require the processing of complex data, a few

extensions are given in [Chapter 12](#).

## 10.2. Signal with Known Form (Known Signal)

---

### Algorithm 10.1 – Replica-correlator (matched filter)

#### 1. Problem

Detect a known signal in WGN.

#### 2. Application area example

Matched filtering is typically used to automatically detect known objects in images [[Pratt 1978](#)]. A two-dimensional version of the matched filter is required.

#### 3. Data model/assumptions

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the known signal, and  $w[n]$  is WGN with known variance  $\sigma^2$ .

#### 4. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \sum_{n=0}^{N-1} x[n]s[n] > \gamma = \sqrt{\sigma^2 \mathcal{E}} Q^{-1}(P_{FA}) \quad (10.2)$$

where  $\mathcal{E} = \sum_{n=0}^{N-1} s^2[n]$  is the signal energy,  $P_{FA}$  is the given probability of false alarm, and  $Q^{-1}(\cdot)$  is the inverse  $Q$  function (see [Section 4.3](#)). This detector is called a *replica-correlator* since it correlates a replica of the signal  $s[n]$  with the data  $x[n]$ . The MATLAB subprogram `FSSP3alg10_1_sub.m` can be used to implement the detector and is contained on the CD.

#### 5. Performance

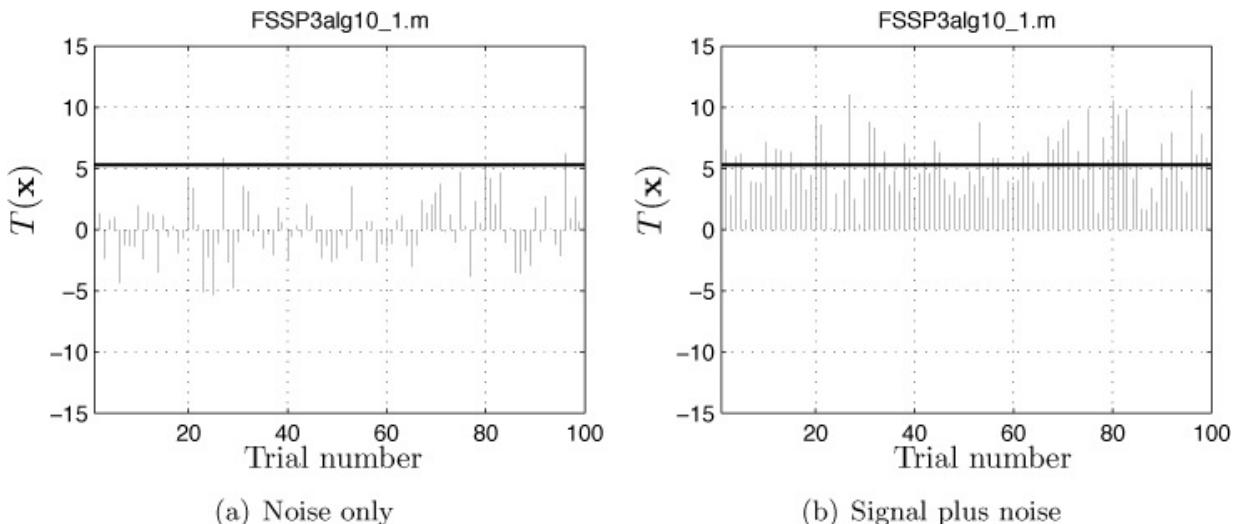
Detector is optimal in that it maximizes the probability of detection  $P_D$  for a given  $P_{FA}$  (Neyman-Pearson optimality – see [Section 8.2.2](#)). The performance is given as

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{\mathcal{E}}{\sigma^2}} \right) \quad (10.3)$$

where  $Q(\cdot)$  is the  $Q$  function (see [Section 4.3](#)).

## 6. Example

Consider a damped exponential signal given by  $s[n] = r^n$ , with  $r = 0.9$ , and  $N = 20$ . The WGN variance is  $\sigma^2 = 1$ . The true signal has been shown in [Figure 9.1a](#). If we set  $P_{FA} = 0.01$ , then the threshold  $\gamma$  is found from [\(10.2\)](#) to be  $\gamma = 5.2974$ . For 100 trials the detection statistic  $T(x)$  given in [\(10.2\)](#) is plotted when there is noise only in [Figure 10.1a](#) and when there is a signal in noise in [Figure 10.1b](#). The threshold is also shown.



**Figure 10.1: Detection statistic outcomes for the replica-correlator detection of a damped exponential signal in WGN.**

## 7. Explanation

The value of the detection statistic due to the signal only will be  $T(s) = \varepsilon$ . The effect of noise will be to introduce the additional term  $\sum_{n=0}^{N-1} w[n]s[n]$ , which hopefully will be small relative to the signal energy.

## 8. Comments/References

The use of the term “matched filter” refers to the historical implementation of  $T(x)$  as an analog *filter* whose impulse response is *matched* to the signal. See [Example 8.3](#) for a special case for which  $s[n] = A$  (a DC level). Also, see [[Kay 1998, Section 4.3](#)].



---

---

### Exercise 10.1 – Lower $P_{FA}$

For the example determine the threshold  $\gamma$  from (10.2) so that  $P_{FA} = 0.0001$ . For this threshold and by observing the outcomes shown in [Figure 10.1](#), does the detection statistic exceed the threshold when there is noise only present, when there is also a signal present? Use the subprogram `[T, gamma] = FSSP3alg10_1_sub(s, x, sig2, Pfa)` to determine the threshold. Since only  $\gamma$  is desired, you can use any arbitrary 20 point data set for  $x$ , say `x=ones(20, 1)`. Also, compute the true  $P_D$  using (10.3) and compare its value to that obtained by counting the number of threshold exceedances in [Figure 10.1b](#) (just use a rough guess from a visual observation of the plot) for the new threshold and dividing by 100.

---



---

## **Algorithm 10.2 – Limiter-replica-correlator**

### **1. Problem**

Detect a known signal in IID nonGaussian noise.

### **2. Application area example**

It is important to be able to measure the thickness of sea ice in order to study possible global warming effects. To do so an upward-looking sonar is used [[Wadhams 1988](#)], but must contend with ice breakups, which produce nonGaussian noise.

### **3. Data model/assumptions**

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the known signal with  $s^2[n] \ll \text{var}(w[n])$  (a weak signal assumption), and  $w[n]$  is IID nonGaussian noise with known probability density function (PDF)  $p(w)$ .

### **4. Detector**

Decide a signal is present if

$$T(x) = \sum_{n=0}^{N-1} g(x[n])s[n] > \gamma = \sqrt{i(A)\mathcal{E}} Q^{-1}(P_{FA}) \quad (10.4)$$

where the function  $g(x)$  is a limiter given by

$$g(w) = -\frac{d \ln p(w)}{dw}. \quad (10.5)$$

The threshold is dependent on  $\mathcal{E} = \sum_{n=0}^{N-1} s^2[n]$ , which is the signal energy, the  $P_{FA}$ , and the term  $i(A)$ , which characterizes the nonGaussian nature of the noise. The latter is defined as

$$i(A) = \int_{-\infty}^{\infty} g^2(w)p(w)dw. \quad (10.6)$$

The MATLAB subprogram `FSSP3alg10_2_sub.m` can be used to implement the detector for  $s[n] = A$ , where  $A > 0$ , and for noise that is IID Laplacian (see [Section 4.6](#)). It is contained on the CD.

## 5. Performance

Detector is optimal for “weak signals” in that it maximizes the probability of detection  $P_D$  for a given  $P_{FA}$  (Neyman-Pearson optimality). The weak signal assumption holds when the data record length is large and  $s[n]$  is small. The performance is given as

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{i(A)\mathcal{E}} \right) \quad (10.7)$$

where  $Q(\cdot)$  is the  $Q$  function and  $Q^{-1}(\cdot)$  is the inverse  $Q$  function.

## 6. Example

Consider the detection of a DC level signal  $s[n] = A$  with  $A > 0$  in IID Laplacian noise with variance  $\sigma^2$  (see [Section 4.6](#)). The weak signal NP detector from [\(10.4\)](#) decides a signal is present if

$$T(\mathbf{x}) = \sum_{n=0}^{N-1} g(x[n])A > \gamma = \sqrt{i(A)\mathcal{E}} Q^{-1}(P_{FA}). \quad (10.8)$$

It can be shown that

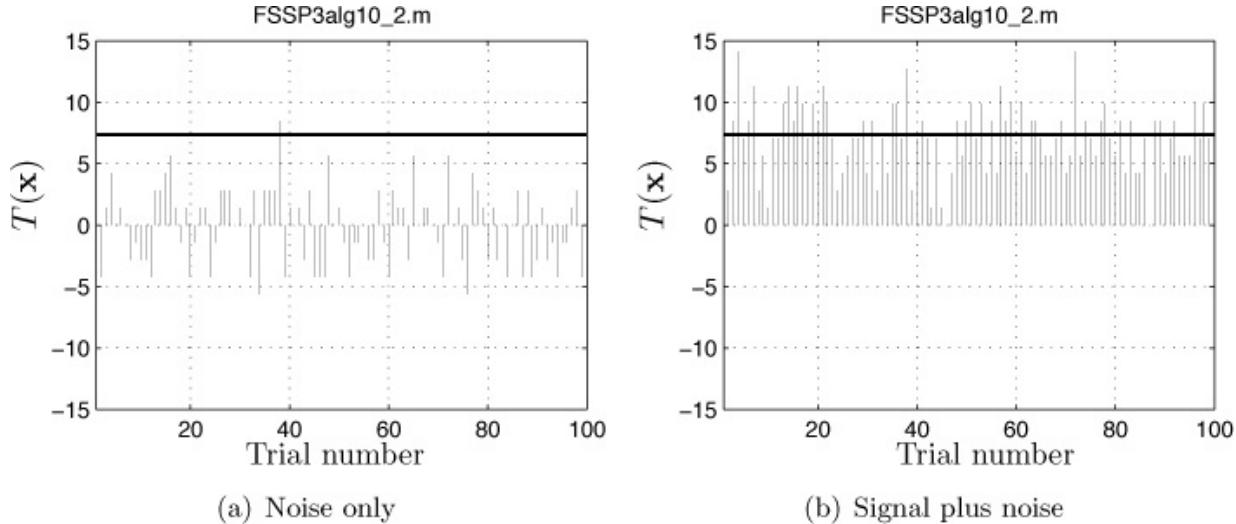
$$g(w) = \sqrt{\frac{2}{\sigma^2}} \operatorname{sgn}(w)$$

where  $\operatorname{sgn}(w) = +1$  if  $w > 0$  and  $\operatorname{sgn}(w) = -1$  if  $w < 0$ . This is the *sign* function. Also,  $i(A) = 2/\sigma^2$ . Therefore, the detector decides a signal is present if

$$T(\mathbf{x}) = A \sqrt{\frac{2}{\sigma^2}} \sum_{n=0}^{N-1} \operatorname{sgn}(x[n]) > \gamma = \sqrt{\frac{2NA^2}{\sigma^2}} Q^{-1}(P_{FA}). \quad (10.9)$$

Now for  $N = 20$ ,  $A = 0.5$ ,  $\sigma^2 = 1$ , and  $P_{FA} = 0.01$  we have that  $\gamma = 7.3566$ .

For 100 trials the detection statistic given in (10.9) is plotted when there is noise only in [Figure 10.2a](#) and when there is a signal in noise in [Figure 10.2b](#). The threshold is also shown.



**Figure 10.2: Detection statistic outcomes for the limiter-replica-correlator detection of a DC level signal in IID Laplacian noise.**

## 7. Explanation

The detector works in a similar manner to the replica-correlator except that it is preceded by a limiter, i.e., the sign nonlinearity. The limiter serves to cut off the spikes typically seen in Laplacian noise (see [Figure 4.15b](#)). If the limiter is not included, the large level spikes will tend to cause many false alarms.

## 8. Comments/References

For IID Laplacian noise the detector of (10.9) is called a *sign detector*. It is often used for robustness, i.e., when the PDF of the noise is known to be IID nonGaussian with heavy “tails” but the exact form of the PDF may not be known (see [Exercise 10.2](#)). See [[Kay 1998](#), pp. 391–392] for further details.



### Exercise 10.2 – Need for limiter

In this exercise we illustrate the effect of a sign detector limiter. We assume that we are under  $H_0$  so that  $x[n] = w[n]$ . Generate 10,000 trials of

a  $N = 50$  sample data record of Gaussian mixture noise with parameters  $\sigma_1^2 = 1/2$ ,  $\sigma_2^2 = 50$  and  $\epsilon = 0.01$ , an example of which has been shown in [Figure 4.17a](#). You can do this by using

`w=Gaussmix_gendata(sig21,sig22,epsilon,N)` for each trial. Then plot the detection statistics

$$T_1(\mathbf{x}) = \sum_{n=0}^{N-1} w[n] / \sqrt{N((1-\epsilon)\sigma_1^2 + \epsilon\sigma_2^2)} \text{ and}$$

$$T_2(\mathbf{x}) = \sum_{n=0}^{N-1} \text{sgn}(w[n]) / \sqrt{N} \text{ and compare the number of spikes.}$$

Note that we have normalized the detection statistics by dividing by the standard deviation so that  $\text{var}(T_1) = \text{var}(T_2) = 1$ . This allows a valid visual comparison of the outcomes. Finally, be aware that the limiter used here is not the optimal one. A better detector would use the limiter given by [\(10.5\)](#), for which the PDF  $p(w)$  of the Gaussian mixture noise needs to be known.



### Algorithm 10.3 – Generalized matched filter

#### 1. Problem

Detect a known signal in correlated Gaussian noise.

#### 2. Application area example

Generalized matched filters are used to improve detection in sonar for reverberation limited environments [[Kay and Salisbury 1990](#)] and in radar for clutter limited environments [[Klemm 2002](#)].

#### 3. Data model/assumptions

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is the known signal, and  $w[n]$  is correlated Gaussian noise with zero mean and known covariance matrix  $\mathbf{C}$ .

#### 4. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \mathbf{x}^T \mathbf{C}^{-1} \mathbf{s} > \gamma = \sqrt{\mathbf{s}^T \mathbf{C}^{-1} \mathbf{s}} Q^{-1}(P_{FA}) \quad (10.10)$$

where  $\mathbf{s} = [s[0] \ s[1] \ \dots \ s[N - 1]]^T$  and  $\mathbf{C}$  is the  $N \times N$  covariance matrix of the noise samples  $\mathbf{w} = [w[0] \ w[1] \ \dots \ w[N - 1]]^T$ . The MATLAB subprogram `FSSP3alg10_3_sub.m` can be used to implement the

detector and is contained on the CD.

## 5. Performance

Detector is optimal in that it maximizes the probability of detection  $P_D$  for a given  $P_{FA}$  (Neyman-Pearson optimality). The performance is given as

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\mathbf{s}^T \mathbf{C}^{-1} \mathbf{s}} \right) \quad (10.11)$$

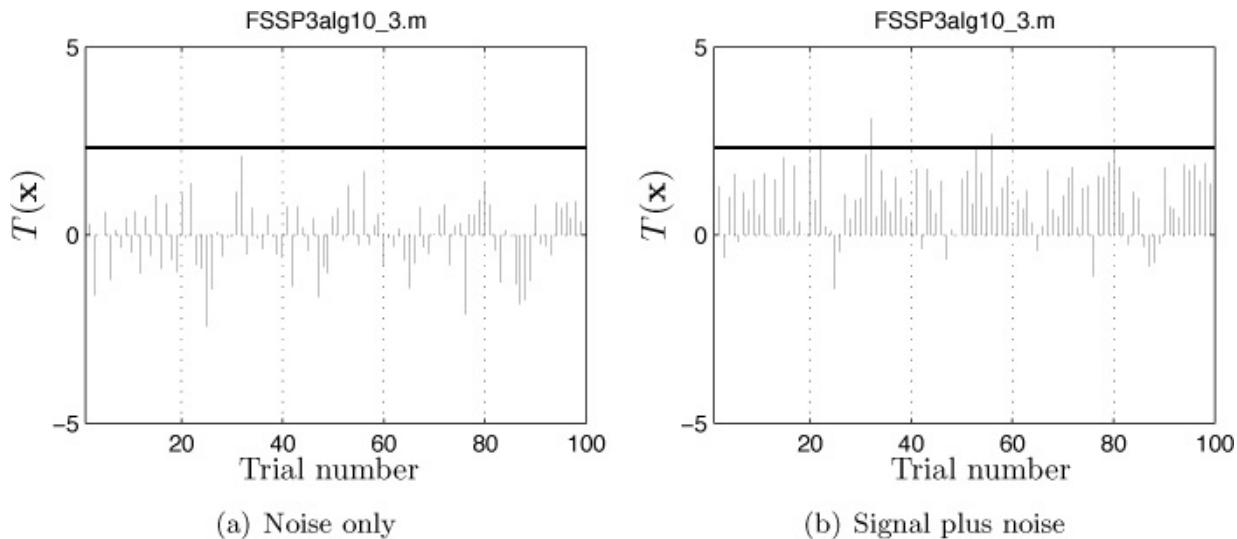
where  $Q(\cdot)$  is the  $Q$  function and  $Q^{-1}(\cdot)$  is the inverse  $Q$  function.

## 6. Example

Consider a damped exponential signal given by  $s[n] = r^n$ , with  $r = 0.9$ , and  $N = 20$ , the same signal as used in the Example of [Algorithm 10.1](#). In contrast to the WGN process used in that example, we now assume the noise is a realization of an autoregressive (AR) random process of order one with PSD shown in [Figure 4.4b](#). Its covariance matrix is given as  $[\mathbf{C}]_{ij} = rw[i - j]$  for  $i = 1, 2, \dots, N; j = 1, 2, \dots, N$ , where  $r_w[k]$  is the ACS

$$r_w[k] = \frac{\sigma_u^2}{1 - a^2[1]} (-a[1])^{|k|}.$$

The parameters of the AR(1) process are  $a[1] = -0.9$  and  $\sigma_u^2 = 0.19$ , which yields a noise variance of unity. For 100 trials the detection statistic given in [\(10.10\)](#) is plotted when there is noise only in [Figure 10.3a](#) and when there is a signal in noise in [Figure 10.3b](#). The threshold  $y = 2.3263$  for  $P_{FA} = 0.01$  is also shown.



**Figure 10.3: Detection statistic outcomes for generalized matched filter for a damped exponential in AR(1) noise.**

## 7. Explanation

The use of  $\mathbf{C}^{-1}$  in the detection statistic, which distinguishes this detector from a replica-correlator, is needed to *prewhiten the noise*. By doing so the data samples are “equalized”, in that noisier samples are downweighted and samples that are heavily correlated are not averaged twice. This enhances the overall SNR of the detection statistic.

## 8. Comments/References

If the Gaussian noise is stationary with PSD  $P_w(f)$ , then an approximate Neyman-Pearson detector decides a signal is present if

$$T(\mathbf{x}) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{X^*(f)S(f)}{P_w(f)} df > \gamma = \sqrt{\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{|S(f)|^2}{P_w(f)} df} Q^{-1}(P_{FA})$$

where  $X(f)$  and  $S(f)$  are the Fourier transforms of  $x[n]$  and  $s[n]$ , respectively. The performance is given by

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{|S(f)|^2}{P_w(f)} df} \right).$$

See [[Kay 1998](#), pp. 105–109] for further details. ■

---

---

### Exercise 10.3 – What is $P_D$ for the example?

For the signal and noise parameters given in the example compute  $\mathbf{s}^T \mathbf{C}^{-1} \mathbf{s}$  using  $[\mathbf{C}]_{ij} = 0.9^{|i-j|}$ . Then, for  $P_{FA} = 0.01$  find the theoretical  $P_D$  using ([10.11](#)). Does it appear to predict the number of threshold crossings as seen in [Figure 10.3b](#)? •

---

## 10.3. Signal with Unknown Form (Random Signals)

### Algorithm 10.4 – Estimator-correlator

#### 1. Problem

Detect a Gaussian random signal in WGN.

## **2. Application area example**

Random signals are used as models for psychophysical experiments to determine the ability of humans to visually detect objects [[Park et al. 2005](#)].

## **3. Data model/assumptions**

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is a Gaussian random signal with zero mean and known  $N \times N$  covariance matrix  $\mathbf{C}_s$ , and  $w[n]$  is WGN with known variance  $\sigma^2$ .

## **4. Detector**

Decide a signal is present if

$$T(\mathbf{x}) = \mathbf{x}^T \mathbf{C}_s (\mathbf{C}_s + \sigma^2 \mathbf{I})^{-1} \mathbf{x} > \gamma \quad (10.12)$$

where  $\mathbf{x} = [x[0] \ x[1] \ \dots \ x[N - 1]]^T$ . This can also be written as

$$T(\mathbf{x}) = \mathbf{x}^T \hat{\mathbf{s}} > \gamma$$

where

$$\hat{\mathbf{s}} = \mathbf{C}_s (\mathbf{C}_s + \sigma^2 \mathbf{I})^{-1} \mathbf{x}$$

is the MMSE estimator of the signal (see [Algorithm 9.12](#)) and hence the name *estimator-correlator*. The MATLAB subprogram `FSSP3alg10_4_sub.m` can be used to implement the detector for a stationary signal with a given autocorrelation matrix (replacing the more general covariance matrix), and is contained on the CD.

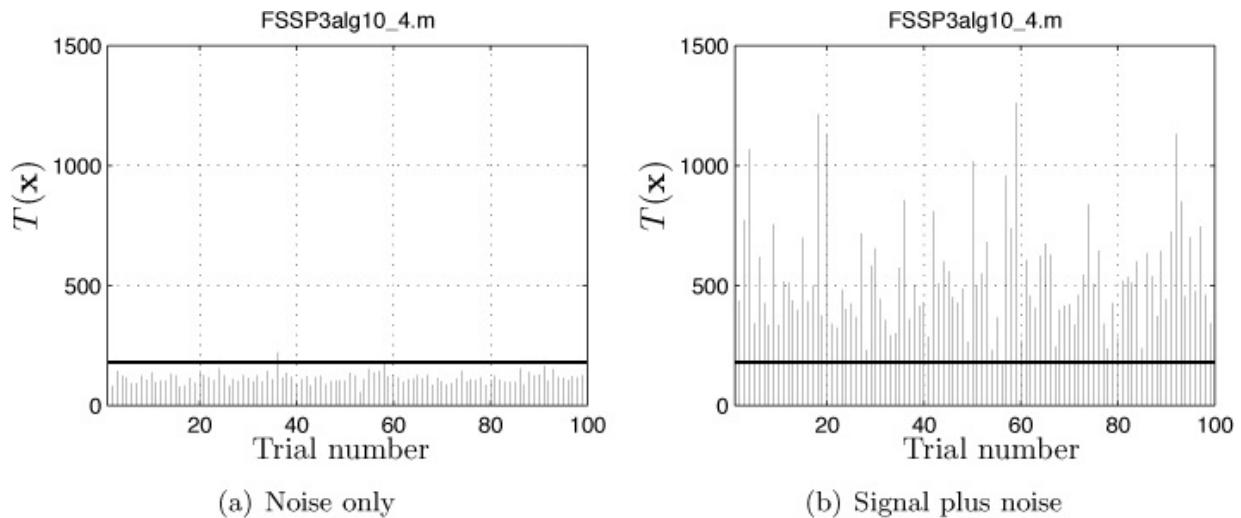
## **5. Performance**

Detector is optimal in that it maximizes the probability of detection  $P_D$  for a given  $P_{FA}$  (Neyman-Pearson optimality). The threshold as well as the detection performance is difficult to express as simply as for the previous algorithms. Methods to do so can be found in [[Kay 1998](#), pp. 149–153] for some special cases.

## **6. Example**

Consider a random signal that is an outcome of an AR random process of order one (see [Section 4.4](#)). The parameters of the AR process are  $a[1] = -0.9$  and  $\sigma_u^2 = 1$ . It is embedded in WGN with variance  $\sigma^2 = 5$ . A typical outcome of the signal and the noise corrupted signal have been shown in [Figure 9.12](#) for  $N = 100$ . Since the signal is an outcome of a stationary

random process, we can replace the covariance matrix by its autocorrelation matrix. This is given by  $[C]_{ij} = rs[i - j]$ . The autocorrelation sequence is given by  $r_s[k] = (\sigma_u^2 / (1 - a^2[1]))(-a[1])^{|k|}$  (see also [Algorithm 9.12](#) for further details). For 100 trials the detection statistic of (10.12) is plotted when there is noise only in [Figure 10.4a](#) and when there is a signal in noise in [Figure 10.4b](#). The threshold of  $\gamma = 179.3756$  for  $P_{FA} = 0.01$  was obtained using a Monte Carlo computer simulation and is also shown (see [Section 7.3.2](#) for more details on how this is done).



**Figure 10.4: Estimator-correlator detection statistic outcomes for a random AR(1) signal in WGN.**

## 7. Explanation

The method attempts to estimate the outcome of the random signal by using a matrix Wiener smoother and then uses this signal estimate to correlate with the data.

## 8. Comments/References

If the Gaussian signal is stationary with PSD  $P_s(f)$ , then an approximate Neyman-Pearson detector decides a signal is present if

$$T(\mathbf{x}) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \left( \frac{P_s(f)}{P_s(f) + \sigma^2} X(f) \right)^* X(f) df > \gamma$$

where the term in parentheses is just the Wiener smoother estimate of the signal in the frequency domain (see [Algorithm 9.12](#)). See [[Kay 1998](#), pp. 144–146] for further details.

---

---

### Exercise 10.4 – Determining the threshold

Implement the detection statistic of (10.12) for the same signal and noise parameters as in the example but for the data use WGN in simulating the noise only condition. Then, determine what the threshold should be so that  $P_{FA} = 0.01$ . You can do this by using a Monte Carlo simulation with 1000 trials as explained in [Section 7.3.2](#). The MATLAB subprogram FSSP3alg10\_4\_sub.m can be used to compute the detection statistic when noise only is present. Does this seem to agree with [Figure 10.4a](#) for which  $\gamma = 179.3756$ ?

---

---

### Algorithm 10.5 – Energy detector

#### 1. Problem

Detect a completely unknown signal in WGN.

#### 2. Application area example

In order to communicate via a cell phone in adverse environments such as in a moving automobile it is necessary to detect the presence of someone speaking. Such *voice activity detection* attempts to automatically sense this condition [[Chang et al. 2006](#)].

#### 3. Data model/assumptions

$$x[n] = s[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is a completely unknown signal and  $w[n]$  is WGN with known variance  $\sigma^2$ .

#### 4. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \sum_{n=0}^{N-1} x^2[n] > \gamma = \sigma^2 Q_{\chi_N^2}^{-1}(P_{FA}) \quad (10.13)$$

where  $Q_{\chi_N^2}^{-1}(\cdot)$  is the inverse chi-squared cumulative distribution function

(see [[Kay 1998](#), pp. 24–26]). The MATLAB subprogram `FSSP3alg10_5_sub.m` can be used to implement the detector, and is contained on the CD.

## 5. Performance

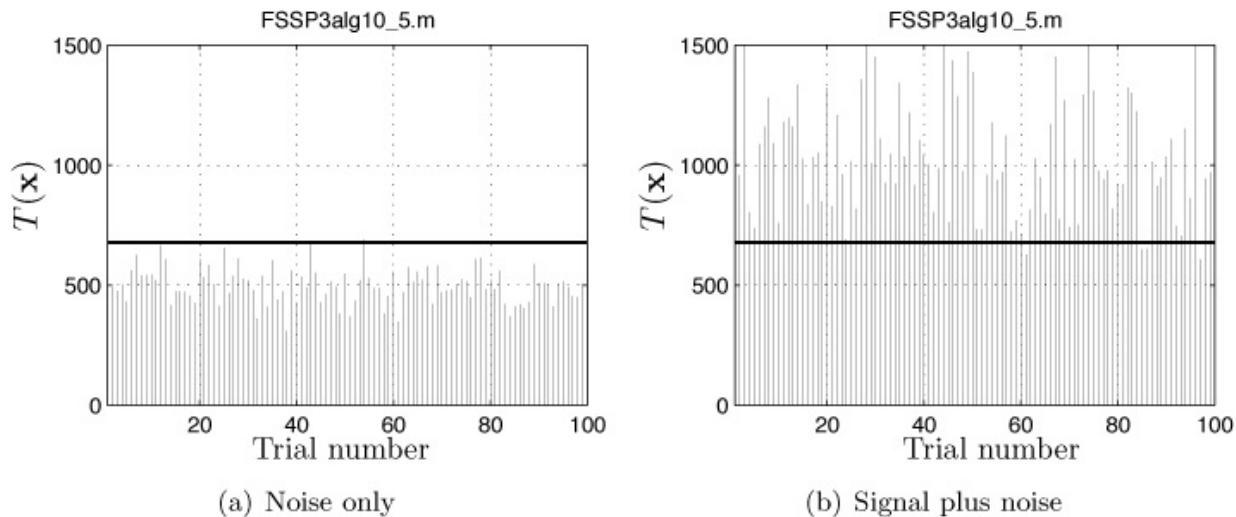
No optimality properties. The probability of false alarm is given by

$$P_{FA} = Q_{\chi_N^2} \left( \frac{\gamma}{\sigma^2} \right)$$

where  $Q_{\chi_N^2}(\cdot)$  is the right-tail probability of a chi-squared random variable with  $N$  degrees of freedom.  $P_D$  cannot be determined without making assumptions on the signal to be detected. The analytical determination of  $P_D$  can be difficult to determine.

## 6. Example

For the same example as in [Algorithm 10.4](#) we now do not assume the knowledge that the signal is a Gaussian AR(1) random process and so must implement an energy detector. For 100 trials the detection statistic given in [\(10.13\)](#) is plotted when there is noise only in [Figure 10.5a](#) and when there is a signal in noise in [Figure 10.5b](#). The threshold is also shown and is given as  $\gamma = 678.71$ . (The MATLAB subprogram `ED_threshold.m` can be used to obtain the threshold.)



**Figure 10.5: Energy detector statistic outcomes for a random AR(1) signal in WGN.**

## 7. Explanation

The energy detector decides that a signal is present if the overall energy of the data increases relative to that expected for noise only. No other

information about the signal is assumed available so that the energy is the only way to discriminate when there is a signal present. This is based on the increased power, i.e., variance. Note that for noise only  $E[x^2[n]] = \sigma^2$  while for a signal in noise  $E[x^2[n]] = s^2[n] + \sigma^2$ , assuming the signal is deterministic.

## 8. Comments/References

If the completely unknown signal is modeled as a white Gaussian signal with known variance  $\sigma_s^2$ , then the detector as given by (10.13) is optimal. In this case the  $P_{FA}$  and  $P_D$  can be found as given in [Kay 1998, pp. 142–144]. This is because in this instance, we have a special case of the Gaussian random signal in WGN (see [Algorithm 10.4](#)). Note that the performance of the energy detector for a Gaussian signal with *known autocorrelation sequence* as in the estimator-correlator of [Algorithm 10.4](#) will be superior to the energy detector, as illustrated in [Exercise 10.5](#). ■

---



---

### **Exercise 10.5 – Poorer performance of energy detector**

From [Figures 10.4b](#) and [10.5b](#) estimate the  $P_D$  by counting the number of threshold crossings in each. Which detector is better? ●

---

## **10.4. Signal with Unknown Parameters (Partially Known Signal)**

---

### **Algorithm 10.6 – Unknown amplitude signal**

#### **1. Problem**

Detect a signal with unknown amplitude in WGN.

#### **2. Application area example**

Square law detectors are typically used to detect signals in radio astronomy, as for example in the search for extraterrestrial intelligence (SETI) [[www.seti.org](http://www.seti.org)].

#### **3. Data model/assumptions**

$$x[n] = As[n] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $As[n]$  is the signal with  $s[n]$  known and  $A$  unknown ( $-\infty < A < \infty$ ), and  $w[n]$  is WGN with known variance  $\sigma^2$ .

#### 4. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \frac{\left( \sum_{n=0}^{N-1} x[n]s[n] \right)^2}{\sigma^2 \sum_{n=0}^{N-1} s^2[n]} > \gamma = \left( Q^{-1}(P_{FA}/2) \right)^2 \quad (10.14)$$

where  $\sigma^2$  is the variance of the WGN. The MATLAB subprogram entitled FSSP3alg10\_6\_sub.m can be used to implement the detector and is contained on the CD.

#### 5. Performance

The detector is a GLRT and so has some optimality properties (see [Section 8.5.2](#)), termed *uniformly most powerful invariant*. The latter means that within a restricted class of detectors it has the highest  $P_D$  [[Scharf 1991](#)].

The exact performance is given as

$$P_D = Q \left( Q^{-1}(P_{FA}/2) - \sqrt{\frac{\mathcal{E}}{\sigma^2}} \right) + Q \left( Q^{-1}(P_{FA}/2) + \sqrt{\frac{\mathcal{E}}{\sigma^2}} \right) \quad (10.15)$$

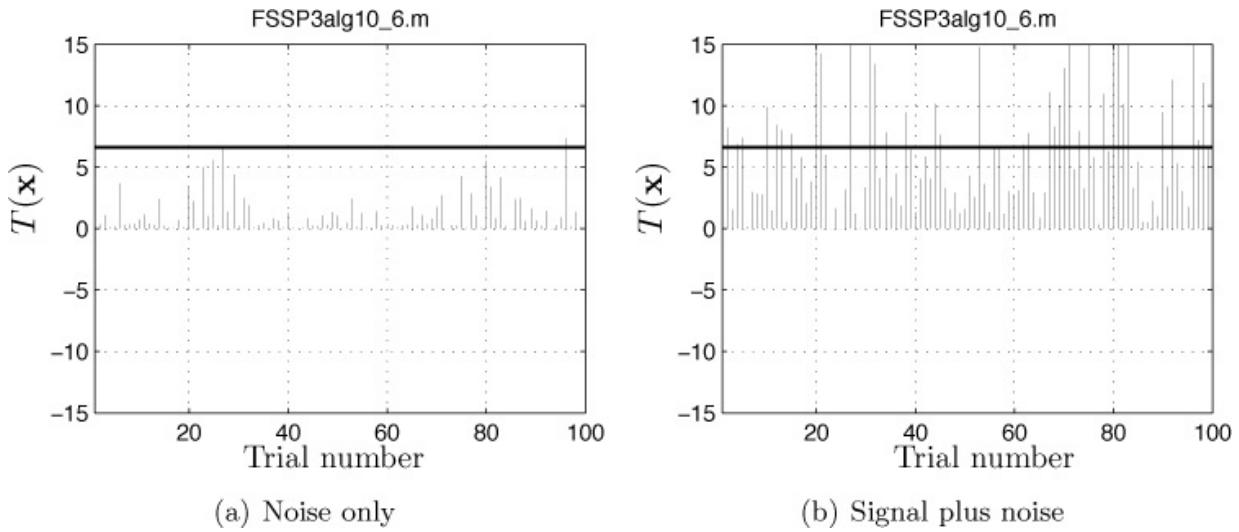
where  $\mathcal{E} = \sum_{n=0}^{N-1} (As[n])^2$  is the signal energy,  $Q(\cdot)$  and  $Q^{-1}(\cdot)$  are the  $Q$  and inverse  $Q$  functions, respectively (see [Section 4.3](#)).

#### 6. Example

Consider a damped exponential, i.e.,  $s[n] = rn$  for  $n = 0, 1, \dots, N - 1$  with  $r = 0.9$ , in WGN with variance  $\sigma^2 = 1$  and  $N = 20$ . We have from (10.14) that a signal is decided to be present if

$$T(\mathbf{x}) = \frac{\left( \sum_{n=0}^{N-1} x[n]r^n \right)^2}{\sigma^2 \sum_{n=0}^{N-1} r^{2n}} > \gamma = \left( Q^{-1}(P_{FA}/2) \right)^2.$$

For 100 trials this detection statistic is plotted when there is noise only in [Figure 10.6a](#) and when there is a signal in noise in [Figure 10.6b](#). The threshold is also shown for  $P_{FA} = 0.01$ , which is  $\gamma = 6.6349$ .



**Figure 10.6: Detection statistic outcomes for the detection of a damped exponential signal with unknown amplitude in WGN.**

## 7. Explanation

The replica-correlator in the presence of a high amplitude signal will produce a very large positive value if  $A > 0$  but a very large negative if  $A < 0$  value. Thus, a single threshold cannot be chosen without knowledge of the sign of  $A$ . The detector therefore squares the replica-correlator detector statistic to make sure both these cases are covered. The denominator normalization produces a simple PDF for the detector statistic when noise only is present and hence, a simple expression for the threshold.

## 8. Comments/References

The  $P_D$  for this detector is only slightly poorer than that of the optimal NP detector, which assumes perfect knowledge of the value of  $A$ . Note that if the amplitude is known to be positive, then the replica-correlator can be used and is optimum (similarly, if it is known that the amplitude is always negative) [Kay 1998, pp. 253–254]. This detector is a special case of the linear model with  $\theta = A$  [Kay 1998, pg. 273]. If desired, once a signal is detected, the value of  $A$  can be estimated using [Algorithm 9.1](#).



### Exercise 10.6 – Effect of squaring

Consider the detection problem of an unknown DC level in WGN. It is defined as  $x[n] = A + w[n]$  when a signal is present and  $x[n] = w[n]$  when

noise only is present for  $n = 0, 1, \dots, N - 1$ . The value of  $A$  is assumed to be unknown. In evaluating the detection performance assume  $A = 1$ ,  $\sigma^2 = 1$ ,  $N = 20$ , and  $P_{FA} = 0.01$ . The effect of squaring can be found by evaluating  $P_D$  ([10.3](#)) for the replica-correlator and ([10.15](#)) for the GLRT. How much does  $P_D$  decrease due to lack of knowledge of the amplitude? Hint: You will need to evaluate  $P_D$  using a computer.

---

### **Algorithm 10.7 – Unknown amplitude and phase of sinusoidal signal**

#### **1. Problem**

Detect a sinusoidal signal that has an unknown amplitude and phase in WGN.

#### **2. Application area example**

In electronic surveillance systems it is the goal to detect narrowband emitters with a known frequency of radiation [[Tsui 1995](#)].

#### **3. Data model/assumptions**

$$x[n] = A \cos(2\pi f_0 n + \varphi) + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $A$  ( $A > 0$ ) and  $\varphi$  ( $-\pi \leq \varphi < \pi$ ) are unknown,  $f_0$  is known, and  $w[n]$  is WGN with known variance  $\sigma^2$ .

#### **4. Detector**

Decide a signal is present if

$$T(\mathbf{x}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n) \right|^2 > \gamma = \sigma^2 \ln \left( \frac{1}{P_{FA}} \right) \quad (10.16)$$

where it has been assumed that the frequency is not near 0 or 1/2 (see [Algorithm 9.3](#) and [Section 1.3](#) for a further discussion). The MATLAB subprogram FSSP3alg10\_7\_sub.m can be used to implement the detector and is contained on the CD.

#### **5. Performance**

The detector is a GLRT and so has some optimality properties, termed *uniformly most powerful invariant* or within a restricted class of detectors

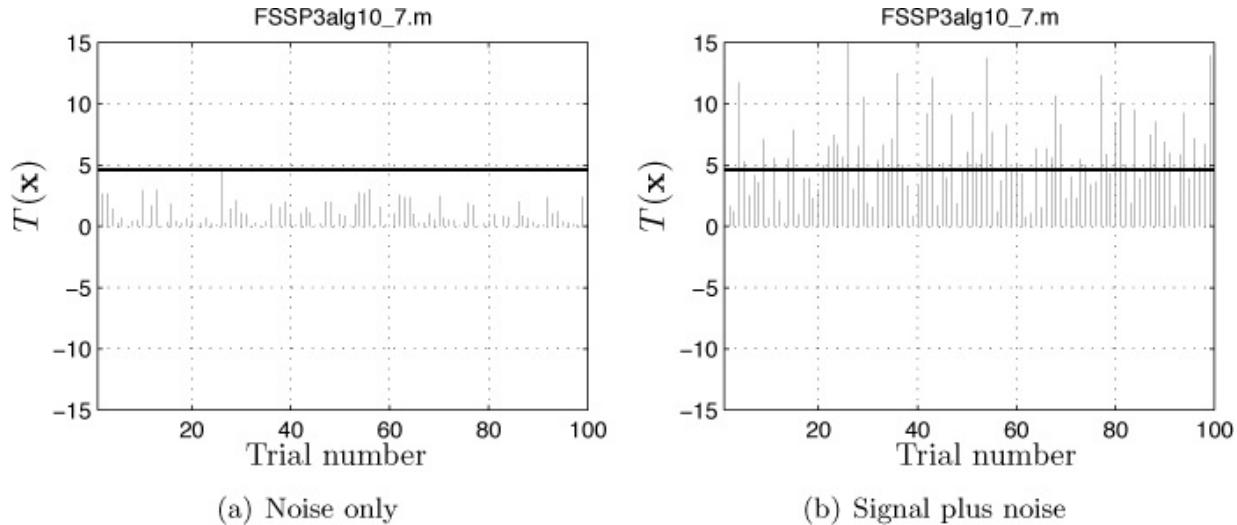
it has the highest  $P_D$  [[Scharf 1991](#)]. The exact performance (to within the approximation that  $f_0$  is not near 0 or 1/2) is

$$P_D = Q_{\chi'^2_2(\lambda)} \left( 2 \ln \frac{1}{P_{FA}} \right)$$

where  $Q_{\chi'^2_2(\lambda)}$  is the right-tail probability of a noncentral chi-squared PDF with 2 degrees of freedom and noncentrality parameter  $\lambda$  [[Kay 1998](#), pp. 26–28]. The value of  $\lambda$  is  $\lambda = N A^2 / (2\sigma^2)$ . The MATLAB subprogram `Qchipr2.m` can be used to evaluate this function and is contained on the CD.

## 6. Example

Consider a sinusoidal signal with  $A = 1$ ,  $f_0 = 0.1$ ,  $\varphi = \pi/2$  and  $N = 20$ . Also, let the WGN variance be  $\sigma^2 = 1$ . Then, for 100 trials the detection statistic given in (10.16) is plotted when there is noise only in [Figure 10.7a](#) and when there is a signal in noise in [Figure 10.7b](#). The threshold  $\gamma = 4.6052$  for  $P_{FA} = 0.01$  is also shown.



**Figure 10.7: Detection statistic outcomes for the detection of a sinusoidal signal with unknown amplitude and phase in WGN.**

## 7. Explanation

The detection statistic is obtained by evaluating the Fourier transform at the sinusoidal frequency, computing the power, and dividing the power by  $1/N$ . Clearly, if the data record length  $N$  is very large, the value of the statistic will also be large since in the limit the Fourier transform is a Dirac delta function at the sinusoidal frequency. The noise PSD, however, is

constant with height  $\sigma^2$ .

## 8. Comments/References

See [[Kay 1998](#), pp. 262–268]. Also, refer back to [Section 1.4](#) to understand the restriction that the frequency not be close to 0 or 1/2. If the amplitude and phase of a detected signal is desired, one can use [Algorithm 9.2](#) to obtain an estimate.



---

## Exercise 10.7 – Effect of data record length

Using FSSP3alg10\_7\_sub.m, input a pure sinusoid  $s[n] = \cos(2\pi(0.1)n)$  for  $n = 0, 1, \dots, N - 1$  and plot the value of the detection statistic versus  $N$  for  $10 \leq N \leq 100$ . You can let  $\sigma^2 = 1$  and  $P_{FA} = 0.01$  in the MATLAB subprogram since these will not affect the detection statistic value. The detection statistic can be shown to be an estimate of the PSD of the signal at the frequency  $f_0$ . How do these values compare to the value of the PSD of the noise, which is  $\sigma^2 = 1$ ?



---

## Algorithm 10.8 – Unknown amplitude, phase, and frequency of sinusoidal signal

### 1. Problem

Detect a sinusoidal signal that has an unknown amplitude, phase, and frequency in WGN.

### 2. Application area example

Typical applications are to radar [[Skolnik 1980](#)] and sonar [[Burdic 1984](#)].

### 3. Data model/assumptions

$$x[n] = A \cos(2\pi f_0 n + \varphi) + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $A$  ( $A > 0$ ),  $\varphi$  ( $-\pi \leq \varphi < \pi$ ), and  $f_0$  ( $0 < f_0 < 1/2$ ) are all unknown, and  $w[n]$  is WGN with known variance  $\sigma^2$ .

### 4. Detector

Decide a signal is present if

Decide a signal is present if

$$T(\mathbf{x}) =$$

$$\max_{0 < f_0 < 1/2} \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f_0 n) \right|^2 > \gamma = \sigma^2 \ln \left( \frac{N/2 - 1}{P_{FA}} \right) \quad (10.17)$$

where it has been assumed that the true frequency is not near 0 or 1/2 (see [Section 1.3](#) for a further discussion). Also, the threshold is approximate, having used some reasonable approximations. The MATLAB subprogram `FSSP3alg10_8_sub.m` can be used to implement the detector and is contained on the CD.

## 5. Performance

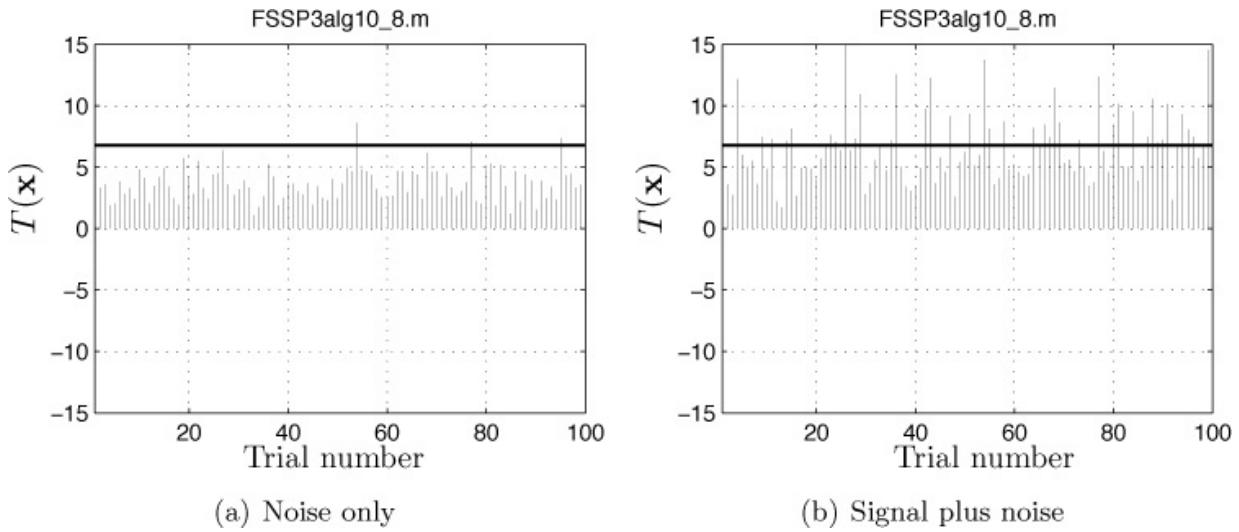
The detector is a GLRT but has no optimality properties. However, in practice the detector performs well. An approximate expression for the performance is given by

$$P_D = Q_{\chi'^2_2(\lambda)} \left( 2 \ln \frac{N/2 - 1}{P_{FA}} \right)$$

where  $Q_{\chi'^2_2(\lambda)}$  is the right-tail probability of a noncentral chi-squared PDF with 2 degrees of freedom and noncentrality parameter  $\lambda$  [[Kay 1998](#), pp. 26–28]. The value of  $\lambda$  is  $\lambda = N A^2 / (2\sigma^2)$ . The MATLAB subprogram `Qchicpr2.m` can be used to evaluate this function and is contained on the CD. Note that the approximation requires  $P_{FA}$  to be small.

## 6. Example

Consider a sinusoidal signal with  $A = 1$ ,  $f_0 = 0.1$ ,  $\varphi = \pi/2$  and  $N = 20$ . Also, let the WGN variance be  $\sigma^2 = 1$ . For 100 trials the detection statistic given in (10.17) is plotted when there is noise only in [Figure 10.8a](#) and when there is a signal in noise in [Figure 10.8b](#). The threshold  $\gamma = 6.8024$  for a  $P_{FA} = 0.01$  is also shown. Note that the threshold must be increased (compare [Figure 10.8a](#) to [Figure 10.7a](#)) due to the increased possibility of false alarms as we search over multiple frequencies for the maximum value.



**Figure 10.8: Detection statistic outcomes for the detection of a sinusoidal signal with unknown amplitude, phase, and frequency in WGN.**

## 7. Explanation

The detection statistic is the periodogram of the data and is also a PSD estimator (see [Chapter 11](#) and [Section 1.3](#)). As in the previous case when the frequency was known, we compute the Fourier transform but not at all the possible frequencies. Since the sinusoid should produce a Dirac delta function for large  $N$ , the detector chooses the value of the magnitude squared Fourier transform divided by  $N$  as the detection statistic.

## 8. Comments/References

The detector statistic is usually computed using an FFT (see the MATLAB subprogram FSSP3alg10\_8\_sub.m) to reduce the computation required to search over frequency. If the amplitude, phase, and frequency of a detected signal is desired, one can use [Algorithm 9.3](#) to obtain an estimate. See [[Kay 1998](#), pp. 268–269] for further details.



### Exercise 10.8 – Increase in $P_{FA}$

For the signal parameters as in the example,  $A = 1$ ,  $\varphi = \pi/2$ ,  $f_0 = 0.1$ ,  $N = 20$  and with  $\sigma^2 = 1$  run the two subprograms FSSP3alg10\_7\_sub.m for the frequency known case and FSSP3alg10\_8\_sub.m for the frequency unknown case. For 10,000 trials determine the number of false

alarms in each case if the threshold is set to yield  $P_{FA} = 0.01$  for the *known frequency case*. Hence, we are using the known frequency case threshold for the unknown frequency case as well to ascertain the increase in false alarms due to having to search for the unknown frequency. Does it increase by the factor  $N/2 - 1$ ? To do so run

`[T, gamma] = FSSP3alg10_7_sub(f0, x, sig2, Pfa)` and  
`[T, gamma] = FSSP3alg10_8_sub(x, sig2, Pfa, Nfft)` 10,000 times. You can use `x=randn(20, 1)` to generate the noise only data. Also, use `Nfft=N` to compute the unknown frequency detection statistic at frequencies between 0 and  $1/2$  (see solution for why we use this value for `Nfft`). ●

---



---

### **Algorithm 10.9 – Unknown arrival time**

#### **1. Problem**

Detect a known signal that arrives at an unknown time in WGN.

#### **2. Application area example**

To determine obstacles in a robot's path a transducer sends an ultrasonic signal and then detects any objects in the robot's path [[Urena et al. 1999](#)].

#### **3. Data model/assumptions**

$$x[n] = s[n - n_0] + w[n] \quad n = 0, 1, \dots, N - 1$$

where  $s[n]$  is a known signal, having samples over the interval  $0 \leq n \leq M - 1$ , and  $w[n]$  is WGN with known variance  $\sigma^2$ . The unknown arrival time  $n_0$  is assumed to take on the possible values  $0 \leq n_0 \leq N - M$ .

#### **4. Detector**

Decide a signal is present if

$$T(\mathbf{x}) = \max_{0 \leq n_0 \leq N-M} \sum_{n=n_0}^{n_0+M-1} x[n]s[n - n_0] > \gamma. \quad (10.18)$$

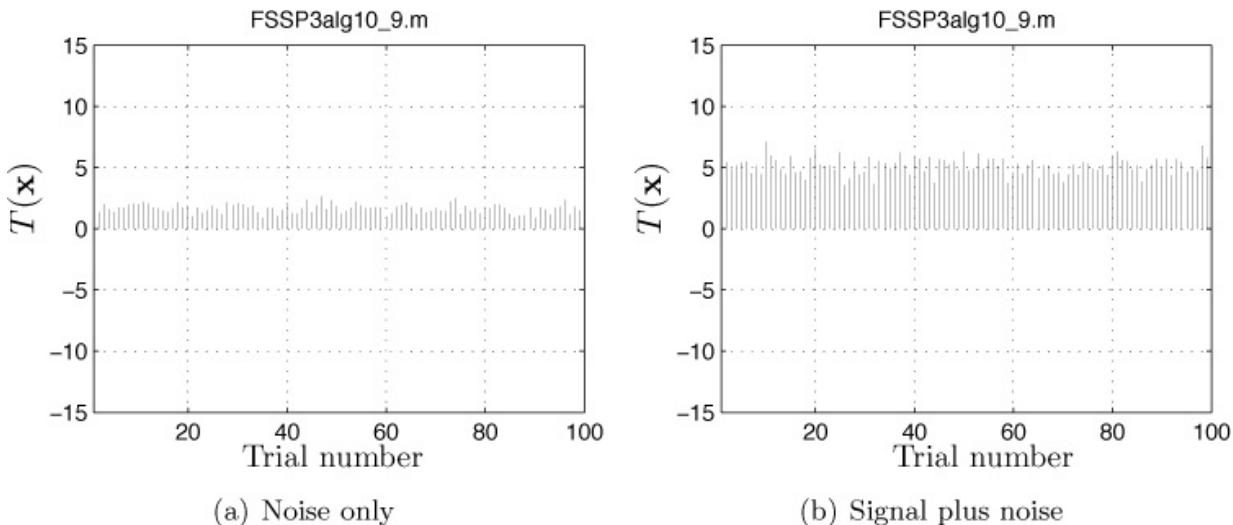
The threshold cannot be obtained in closed form. The MATLAB subprogram `FSSP3alg10_9_sub.m` can be used to compute the detector statistic and is contained on the CD.

## 5. Performance

The detector is a GLRT but has no optimality properties. However, in practice the detector performs well. An analytical evaluation of its performance is difficult to obtain.

## 6. Example

Consider a damped exponential signal given by  $s[n] = r^n$ , with  $r = 0.9$ ,  $M = 20$ , and  $n_0 = 100$ . The data record length, which is the total number of data samples used for computing the correlation statistic, is  $N = 200$ . The WGN variance is  $\sigma^2 = 0.1$ . The true signal has been shown in [Figure 9.1a](#). For 100 trials the detection statistic given in [\(10.18\)](#) is plotted when there is noise only in [Figure 10.9a](#) and when there is a signal in noise in [Figure 10.9b](#).



**Figure 10.9: Detection statistic outcomes for the detection of a damped exponential signal with unknown arrival time in WGN.**

## 7. Explanation

The operation of this detector is nearly identical to that of the replica-correlator. Since the exact position in time of the signal is unknown, the detector attempts to match all possible signals, i.e., it correlates with all possible replicas. This is done by computing  $\sum_{n=n_0}^{n_0+M-1} x[n]s[n-n_0]$  for all possible arrival times  $n_0$ .

## 8. Comments/References

If a signal is detected, its arrival time is also estimated as the time of the maximum obtained from [\(10.18\)](#). When this detector is extended to

accommodate a sinusoid of unknown amplitude, phase, and frequency, the standard radar and active sonar receivers are obtained [[Kay 1998](#), pp. 269–272].

---

---



### Exercise 10.9 – Value of detector statistic when signal is present

For the signal given in the example, determine the value of the signal energy. Then, compare this value to the outcomes of the test statistic shown in [Figure 10.9b](#). Should they be similar? Hint: You will need a computer to do this.

---

---



## References

- Burdic W.W., *Underwater Acoustic System Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Chang, J., N.S. Kim, S.K. Mitra, “Voice Activity Detection Based on Multiple Statistical Models”, *IEEE Trans. on Signal Processing*, pp. 1965–1976, June 2006.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay, S., J. Salisbury, “Improved Active Sonar Detection Using Autoregressive Prewhitening”, *Journal of Acoustical Society of America*, April 1990.
- Klemm, R., *Principles of Space-Time Adaptive Processing*, Institution of Electrical Engineers, London, 2002.
- Park, S., E. Clarkson, M.A. Kupinski, H.H. Barrett, “Efficiency of the Human Observer Detecting Random Signals in Random Backgrounds”, *Journal of Optical Society of America*, pp. 3–16, 2005.
- Pratt, W.K., *Digital Image Processing*, J. Wiley, NY, 1978.
- Scharf, L., *Statistical Signal Processing*, Addison-Wesley, NY, 1991.
- Skolnik, M., *Introduction to Radar Systems*, McGraw-Hill, NY, 1980.
- Tsui, J., *Digital Techniques for Wideband Receivers*, Artech House, Boston, 1995.
- Urena, J., M. Mazo, J.J. Garcia, A. Hernandez, E. Bueno, “Correlation Detector

Based on a FPGA for Ultrasonic Sensors”, *Microprocessors and Microsystems*, Vol. 23, pp. 25–33, June 1999.

Wadhams, P., “The Underside of Arctic Sea Ice Imaged by Sidescan Sonar”, *Nature*, pp. 161–164, May 1988.

## Appendix 10A. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

- 10.1** The threshold is  $\gamma = 8.4687$ . For noise only there are no threshold crossings, and for a signal plus noise there are about 10 threshold crossings, yielding an estimated  $P_D$  of 0.1. The theoretical  $P_D$  for this lower  $P_{FA}$  is  $P_D = 0.0747$ . You can run the program `FSSP3exer10_1.m`, which is contained on the CD, to obtain the results.

- 10.2** You should observe that the “sum” detector given by  $T_1(\mathbf{x})$  will have values of the detection statistic that exceed 4 in magnitude, while the inclusion of the limiter to form  $T_2(\mathbf{x})$  inhibits this from happening most of the time. You can run the program `FSSP3exer10_2.m`, which is contained on the CD, to obtain the results.

- 10.3** The theoretical  $P_D$  is  $P_D = 0.0924$ , which would indicate that about 9 threshold crossings would occur for 100 trials. From [Figure 10.3b](#) there appears to be about 5. You can run the program `FSSP3exer10_3.m`, which is contained on the CD, to obtain the results.

- 10.4** You should get a threshold of  $\gamma = 178.5104$ , which is close to the one used in [Figure 10.4](#) of  $\gamma = 179.3756$ . For a more accurate estimate, increase the number of trials. You can run the program `FSSP3exer10_4.m`, which is contained on the CD, to obtain the results.

- 10.5** From [Figure 10.4b](#) all the detection statistics cross the threshold, while in [Figure 10.5b](#) only 96 out of 100 do so. The estimator-correlator is better.

- 10.6** By evaluating [\(10.3\)](#) for the replica-correlator and [\(10.15\)](#) for the GLRT, we have that  $P_D = 0.9841$  for the replica-correlator and  $P_D = 0.9710$  for the GLRT. Thus, the degradation in performance is very

slight. You can run the program FSSP3exer10\_6.m, which is contained on the CD, to obtain the results.

- 10.7** The detection statistic is approximately given by  $N/4$  and will exceed the noise PSD of  $\sigma^2 = 1$  with high probability if  $N > 40$ . See also [Exercise 1.2](#). You can run the program FSSP3exer10\_7.m, which is contained on the CD, to obtain the results.
- 10.8** You should obtain 116 false alarms if the known frequency case detection statistic is used, and 1098 false alarms if the unknown frequency case detection statistic is used. Thus, the increase in  $P_{FA}$  is about  $1098/116 = 9.46$  and  $N/2 - 1 = 9$ . The reason why we use  $Nfft=1024$  is that it allows us to easily find the  $P_{FA}$ . It is more customary to compute the periodogram at a large number of frequency points, say 1024. However, this will invalidate the use of the simple increase of  $N/2 - 1$  for  $P_{FA}$  since when this is done the periodogram points become heavily correlated, which is at odds with the assumption that they are independent. You can run the program FSSP3exer10\_8.m, which is contained on the CD, to obtain the results.
- 10.9** The signal energy of the damped exponential is  $\varepsilon = 5.1854$ . It is very close to the value of the detection statistic shown in [Figure 10.9b](#). You can run the program FSSP3exer10\_9.m, which is contained on the CD, to obtain the results.

# Chapter 11. Spectral Estimation

## 11.1. Introduction

The power spectral density (PSD) of a random process is a function that quantifies the distribution of power with frequency. The estimation of this function is commonly referred to as *spectral estimation*. How this is accomplished is the subject of this chapter. Unlike estimation or detection problems for which some general theory exists, leading to optimal approaches, spectral estimation has no such theory to help guide us in developing practical methods. This is because it is inherently an ill-posed problem, requiring the estimation of a general curve based on only a finite number of data samples. Its principal and very important utility, however, is as a preliminary data analysis tool. Depending upon the distribution of power with frequency that a spectral analysis provides, more specific approaches to data analysis will often be suggested.

Some of the material in this chapter has been previously described. The statistical modeling and generation of noise as described in [Sections 4.2–4.4](#) includes consideration of the PSD. Also, in [Section 6.4.7](#) we discussed one of the ways in which the PSD is estimated based on its Fourier definition. In particular, the *averaged periodogram* was defined and examples given. The reader may wish to reread these sections before proceeding. In this chapter we will add more detail, and also provide specific algorithms and their MATLAB implementations for the more commonly used approaches. The topic of spectral estimation has been extensively studied. Some useful references are [[Jenkins and Watts 1968](#)], [[Priestley 1981](#)], and [[Kay 1988](#)].

The problem of spectral estimation is to estimate the power spectral density function defined as

$$P_x(f) = \sum_{k=-\infty}^{\infty} r_x[k] \exp(-j2\pi fk) \quad -1/2 \leq f \leq 1/2 \quad (11.1)$$

based on the observed data set  $x[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $r_x[k]$  is the autocorrelation sequence (ACS). This definition says that the PSD is the discrete-time Fourier transform of the ACS. It can equivalently be written as

$$P_x(f) = \lim_{M \rightarrow \infty} \frac{1}{2M + 1} E \left[ \left| \sum_{n=-M}^{M} x[n] \exp(-j2\pi fn) \right|^2 \right] \quad -1/2 \leq f \leq 1/2. \quad (11.2)$$

Recall that  $x[n]$  may represent either a noise process or a random signal process (see [Section 3.6](#) for the signal model and [Chapter 4](#) for the noise model). The observed data set is assumed to be a segment of a random process realization that is zero mean and wide sense stationary (WSS) (see also [Appendix 4A](#) and [\[Kay 2006\]](#)). If the data is nonzero mean, then it is customary to estimate the mean using the sample mean  $\bar{x} = (1/N) \sum_{n=0}^{N-1} x[n]$ , and then subtract  $\bar{x}$  from each data sample. Similarly, if the data contains sinusoidal components, they too should be estimated and removed. Methods to do so have already been described as [Algorithms 9.9](#) and [9.10](#). Note that the very important problem of determining the parameters of these sinusoidal components is *not* a problem in spectral estimation but one in *parameter estimation*, for which the methods of [Chapter 9](#) apply.

We confine our discussion to the estimation of the PSD based on a real data set. The extension to the case of complex data, multichannel data, and multidimensional data can be found in the aforementioned references. The other closely related problems of spectral estimation for a slowly time-varying spectrum (see [Section 4.4](#)) is also briefly discussed.

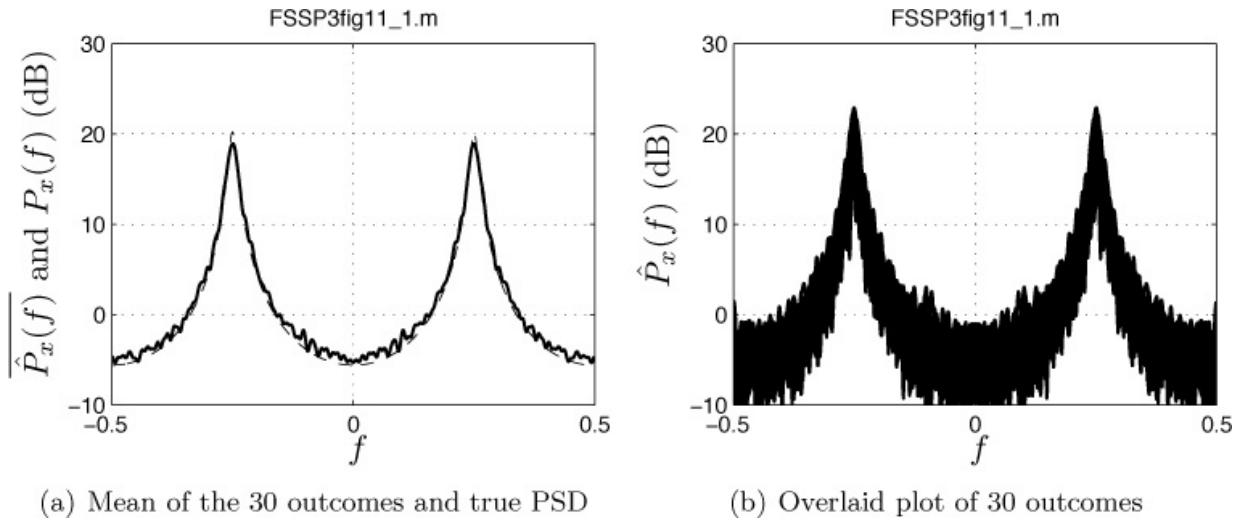
There are two main approaches to spectral estimation: *nonparametric* or Fourier methods, and *parametric* or model-based methods. An example of a nonparametric spectral estimator is the averaged periodogram, described in [Section 6.4.7](#). A commonly used parametric spectral estimator is one based on the autoregressive (AR) model, described in [Section 4.4](#). Nonparametric approaches are more robust, applying to any stationary random process while parametric methods impose the constraint of a particular class of PSD models. The motivation is that the model-based spectral estimator is more accurate *if the model assumed to obtain the estimator is accurate*, hence, the importance of choosing a good model for the data to be analyzed. We will delineate the two approaches to spectral estimation by dividing the algorithms into these two categories.

Finally, note that spectral estimation does not require the data to be Gaussian. It is an analysis of the data with respect to its second moment, i.e., its correlation properties. In assessing the performance of the estimator, however, the assumption of Gaussianity can be useful.

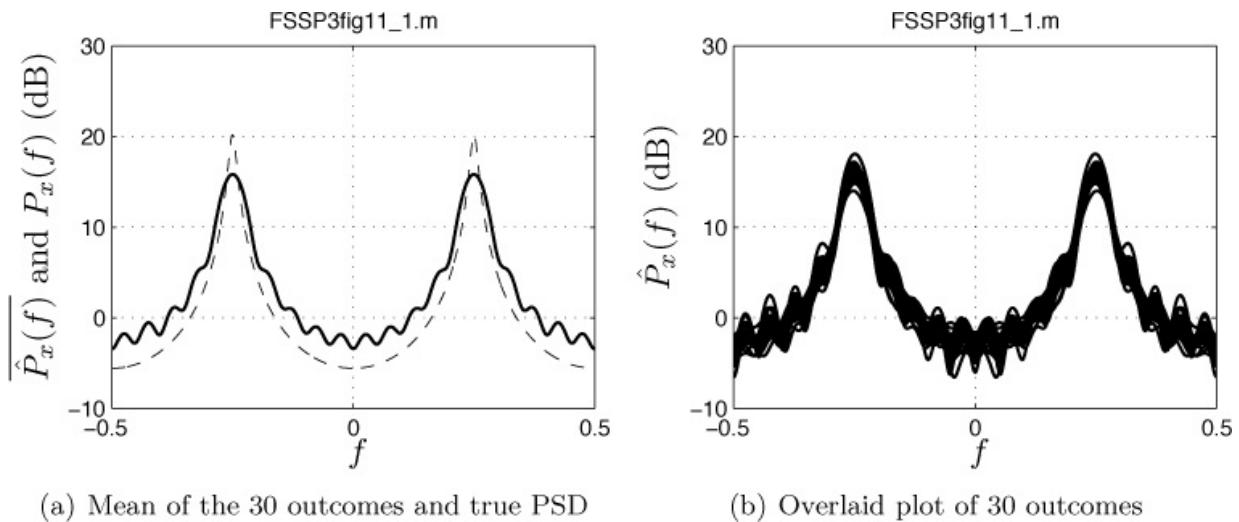
## 11.2. Nonparametric (Fourier) Methods

Before describing the nonparametric algorithms some general comments about their performance need to be made. Good spectral estimates are characterized by

*high resolution* and *low variability*. The former means that the estimate displays fine detail, e.g., narrow peaks, when they are present, and the latter means that the variance is low. Unfortunately, with a fixed number of data samples this goal is generally not possible. An example is shown in [Figures 11.1](#) and [11.2](#), in which the averaged periodogram of [Section 6.4.7](#) is used to obtain the spectral estimates. The data was generated using an AR random process of order  $p = 2$  with  $a[1] = 0$ ,  $a[2] = 0.9025$ , and  $\sigma_u^2 = 1$ . The data record length is  $N = 500$ , and in [Figure 11.1](#)  $I = 5$  nonoverlapping blocks of length  $L = 100$  data samples each were used to compute the estimate. In the figures the results based on 30 different outcomes are shown. In [Figure 11.1a](#) the true PSD is shown as the dashed curve, and the mean of the 30 estimates (denoted by  $\hat{P}_x(f)$ ) is shown as the solid curve. This shows that the *mean* of this spectral estimator is close to the true PSD, i.e., it exhibits high resolution. However, if the 30 estimates are plotted in an overlaid fashion as in [Figure 11.1b](#), it is seen to exhibit a large variability with the range of values at a single frequency of 10 dB or more. Recall that the use of a logarithmic scale (a dB quantity) for the vertical axis is to allow low level details to be observed in the spectral estimate. To reduce the variability it is necessary to use more averages in the averaged periodogram estimator. We could, for example, increase the number of blocks to  $I = 25$  to yield the overlaid plot shown in [Figure 11.2b](#). This has definitely reduced the variance at each frequency. However, in doing so, the length of the data record for each block becomes  $L = 500/25 = 20$  samples, producing the mean of the 30 estimates shown in [Figure 11.2a](#). Now there is a substantial error in that the details of the PSD are distorted, resulting in a low resolution spectral estimate. It is therefore *not possible with a fixed length data record of  $N$  points to simultaneously reduce the variance and increase the resolution*. The best we can do is trade one off for the other. This is the so-called *bias-variance tradeoff*. (Recall that bias is the difference between the expected value of the estimator and the true value.) See [Exercise 11.2](#) for one strategy proposed to make this tradeoff. In [Figure 11.2a](#) it is seen that the peak is underestimated and the other portions of the PSD are overestimated. It appears as if some of the power in the peak *has leaked* into the other frequency bands. This is sometimes referred to as the *leakage problem*. There also appear to be small resonances in the nonpeak bands. This is an artifact of the estimator, which can be ameliorated somewhat as illustrated in the next exercise.



**Figure 11.1: Averaged periodogram spectral estimate using 5 nonoverlapped blocks of length  $L = 100$ . The dashed curve in [Figure 11.1a](#) is the true PSD.**



**Figure 11.2: Averaged periodogram spectral estimate using 25 nonoverlapped blocks of length  $L = 20$ . The dashed curve in [Figure 11.2a](#) is the true PSD.**

### Exercise 11.1 – Reducing windowing ripples

The ripples that are seen in [Figure 11.2a](#) are the so-called windowing effects caused by taking the Fourier transform of a *segment* of a random process realization. The practical restriction to the use of a segment is, of course, due to the availability of only a finite, and sometimes quite short,

data record. The use of a data segment results in taking the Fourier transform of the entire realization of the random process after multiplying by a rectangular window in time. This causes the frequency content (on the average) to be that of the desired Fourier transform convolved with the Fourier transform of the rectangular window, i.e., a sinc type function. To reduce this effect you can multiply each block of data by a Hamming window, as an example, before Fourier transforming. To see the effect of *data windowing* run the code

[Click here to view code image](#)

```
N=500;
L=20; % length of data for each block
a=[0 0.9025]'; % AR filter parameters
sig2u=1; % AR excitation noise variance
for i=1:30
x=ARgendata(a,sig2u,N); % generate N points of data for each outcome
% In the PSD_est_avper.m subprogram replace
% Pper=(1/L)*abs(fftshift(fft(y,1024))) .^2; by
% Pper=(1/(hamming(L) *hamming(L)))...
% *abs(fftshift(fft(y.*hamming(L),Nfft))) .^2;
[freq,Pavper(:,i)]=PSD_est_avper(x,L,1024,0,0);
end
Pm=mean(Pavper'); % find mean of the outcomes
Pavper_mean=Pm'; % convert to column vector for plotting
[powsd_true,freq]=arpsd(a,sig2u,1024); % compute the true PSD
plot(freq,10*log10(Pavper_mean),freq,10*log10(powsd_true)) % plot the
% true PSD and the mean of the averaged periodogram
grid
```

Note that when data windowing is used, the divisor  $L$  in  $P_{\text{per}}$  needs be replaced by  $\sum_{n=0}^{L-1} w^2[n]$ , where  $w[n]$  is the data window, to obtain the correct overall power. What happens to the ripples? How does data windowing affect the resolution of the spectral estimator?

---

Finally, the resolution of a Fourier-based spectral estimator is limited by the length of the data record. Maximum resolution is obtained using a single block, but this unfortunately, also yields maximum variance. The ability of the periodogram to display closely spaced details in the PSD is limited by the block length  $L$ . A rule of thumb is that two narrowband components can be seen if their frequency spacing is at least  $1/L$  cycles/sample in frequency. Hence, if possible, one should choose  $L$  to ensure adequate resolution and then the number of blocks  $I$  to ensure a reasonable variability. The total data record length required then becomes  $N = IL$ .

---

## Algorithm 11.1 – Averaged periodogram

### 1. Problem

Estimate the PSD of a stationary random process.

### 2. Application area example

Nonparametric spectral analysis is used extensively in vibration analysis [[Bendat and Piersol 1971](#)].

### 3. Data model/assumptions

The data  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

### 4. Estimator

Divide the data record into  $I$  successive nonoverlapping blocks of length  $L$  samples each, where it is assumed that  $N = IL$ . The data blocks are given by

$$y_i[n] = x[n + iL] \quad n = 0, 1, \dots, L - 1; i = 0, 1, \dots, I - 1.$$

Then, for each block compute the periodogram as

$$\hat{P}_x^{(i)}(f) = \frac{1}{L} \left| \sum_{n=0}^{L-1} y_i[n] \exp(-j2\pi f n) \right|^2 \quad (11.3)$$

and then average all the periodograms together to yield the spectral estimator as

$$\hat{P}_x(f) = \frac{1}{I} \sum_{i=0}^{I-1} \hat{P}_x^{(i)}(f). \quad (11.4)$$

The MATLAB subprogram `PSD_est_avper.m` can be used to implement the averaged periodogram and is contained on the CD.

### 5. Performance

There are no optimality properties. A 95% confidence interval is

$$10 \log_{10} \hat{P}_x(f) \begin{cases} +10 \log_{10} \frac{2I}{F_{\chi^2_{2I}}^{(-1)}(0.025)} \\ -10 \log_{10} \frac{F_{\chi^2_{2I}}^{(-1)}(0.975)}{2I} \end{cases} \text{ dB} \quad (11.5)$$

where  $F_{\chi^2_v}^{(-1)}(x)$  is the inverse cumulative distribution function (CDF) for a  $\chi^2_v$  random variable, i.e., a chi-squared distribution with  $v$  degrees of freedom. The MATLAB subprogram `chipr2.m` can be used to evaluate the CDF and from this the inverse CDF values can be found. See [Section 6.4.7](#) for an example.

## 6. Example

See [Figures 11.1](#) and [11.2](#).

## 7. Explanation

The averaged periodogram spectral estimator passes the data through a bank of narrowband filters to yield an output sequence that is due to frequencies near the center frequency of each filter. Then, the power at the output of each filter is computed as an average sum of squares (see also [Appendix 11A](#) for more details).

## 8. Comments/References

The averaged periodogram utilizes the fast computation provided by the fast Fourier transform (FFT) algorithm. The data record should always be “zero-padded” before taking the FFT. This means that the data record should be augmented with enough zeros so that the FFT length is much greater than  $N$ . This can be done automatically in MATLAB using the FFT subprogram. Since this process and its relationship to spectral resolution is a frequent source of confusion, some further details are included in [Appendix 11B](#).

It has been suggested that additional variance reduction can be obtained by using overlapped blocks [[Welch 1967](#)]. For other possible data windows and the tradeoffs that ensue see [[Harris 1978](#)]. Further details can be found in [[Kay 1988, Chapter 4](#)].



---

---

### Exercise 11.2 – Window closing approach to choosing block length

It has been suggested that one should initially choose  $L$  small, so that the variability of the spectral estimator is small, and hence what one sees is indeed reliable, but of course smoothed. Then, by increasing  $L$  we start to observe better resolved details, possibly more peaks or sharper peaks appearing in the spectral estimate. The trick is to be able to see the detail

necessary before the value of  $L$  increases so much that the variability of the spectral estimate starts to introduce “noise peaks”, i.e., ones that are not actually there but are artifacts of the particular random process outcome. Try this strategy using the data example of [Figures 11.1](#) and [11.2](#) for  $L = 10$ ,  $L = 20$ ,  $L = 50$ , and  $L = 100$ . You can use the code

[Click here to view code image](#)

```
randn('state',0) % initialize Gaussian random number generator
N=500;
a=[0 0.9025]'; % set AR filter parameters
sig2u=1; % set AR excitation noise variance
[powsd_true,freq]=ARpsd(a,sig2u,1024); % compute the true PSD
x=ARgendata(a,sig2u,N); % generate N samples of data
% put a loop here and plot each spectral estimate as L is increased
%-----
L=??; % length of data block
[freq,Pavper]=PSD_est_avper(x,L,1024,0,0); % call to subprogram
%-----
```

and plot the spectral estimate in dB for each value of  $L$ . Note that the use of the term “window closing” refers to a window in *frequency*, hence as the frequency window width decreases,  $L$  increases [[Jenkins and Watts 1968](#)].

---

A second popular and very useful method for nonparametric spectral estimation is the minimum variance spectral estimator (MVSE). It is easily generalized to multichannel and multidimensional spectral estimation, and therefore is used extensively in array processing [[Johnson and Dudgeon 1993](#)]. There is only a single parameter that determines the bias-variance tradeoff, and that is the size of the covariance matrix that must be estimated. It has been found via empirical studies that for spectra with sharp peaks the MVSE is better able to render the PSD with less smearing than the averaged periodogram. It is summarized next.

---

### Algorithm 11.2 – Minimum variance spectral estimator

#### 1. Problem

Estimate the PSD of a stationary random process.

#### 2. Application area example

The MVSE has been applied to a wide variety of problems. A recent

application utilizes its straightforward extension to multiple dimensions [[Carbone and Kay 2012](#)] for sonar data normalization.

### **3. Data model/assumptions**

The data  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

### **4. Estimator**

$$\hat{P}_x(f) = \frac{p}{\mathbf{e}^H \hat{\mathbf{R}}_x^{-1} \mathbf{e}} \quad (11.6)$$

where  $\mathbf{e} = [1 \exp(j2\pi f) \dots \exp[j2\pi f(p-1)]]^T$ ,  $H$  denotes the conjugate transpose, and

$$[\hat{\mathbf{R}}_x]_{ij} = \frac{1}{2(N-p)} \left[ \sum_{n=p}^{N-1} x[n-i]x[n-j] + \sum_{n=0}^{N-1-p} x[n+i]x[n+j] \right]$$

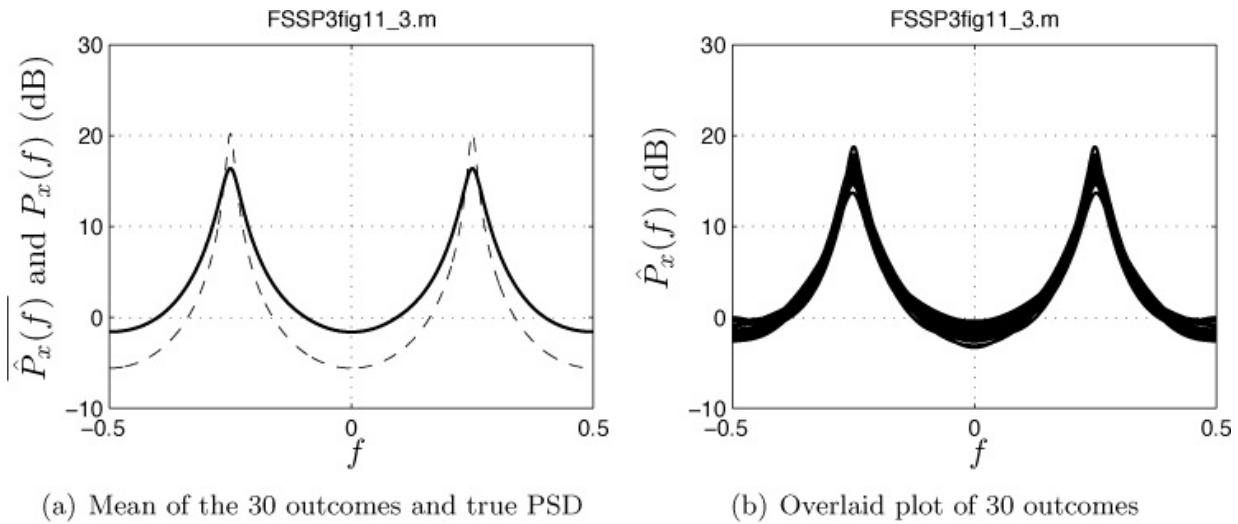
for  $i = 1, 2, \dots, p; j = 1, 2, \dots, p$ . The  $p \times p$  matrix  $\hat{\mathbf{R}}_x$  is an estimate of the autocorrelation matrix (see [Algorithm 9.12](#) for a definition). Although the term  $\mathbf{e}^H \hat{\mathbf{R}}_x^{-1} \mathbf{e}$  is theoretically real, numerical errors will cause it to be complex. Thus, the real part should be taken. The MATLAB subprogram `PSD_est_mvse.m` implements the spectral estimator, and is contained on the CD.

### **5. Performance**

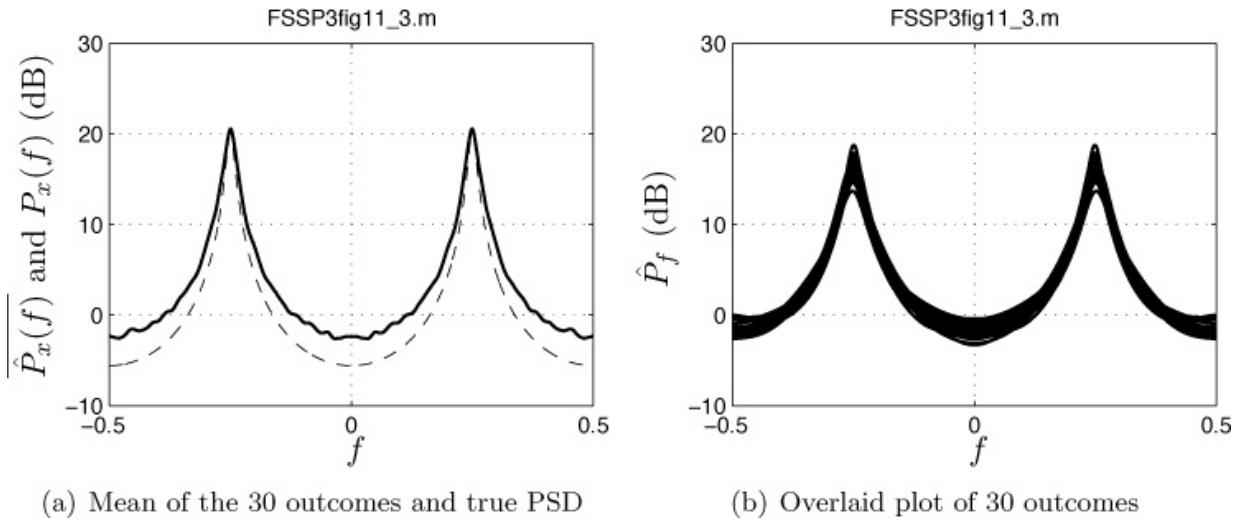
There are no optimality properties. It has been empirically observed that the resolution of the MVSE is better than the averaged periodogram (for the same variance). Confidence intervals are not available for this spectral estimator due to its very nonlinear nature.

### **6. Example**

We consider the same example as for the averaged periodogram, the results of which have been shown in [Figures 11.1](#) and [11.2](#). For  $p = 10$  the MVSE results are shown in [Figure 11.3](#) and for  $p = 50$  they are shown in [Figure 11.4](#). Note that increasing  $p$  improves the resolution while only slightly increasing the variance. A comparison of [Figure 11.4](#) to [Figure 11.2](#) indicates that the MVSE yields a higher resolution spectral estimator with less variance, for overall better performance than the averaged periodogram.



**Figure 11.3: Minimum variance spectral estimate using  $p = 10$ . The dashed curve in [Figure 11.3a](#) is the true PSD.**



**Figure 11.4: Minimum variance spectral estimate using  $p = 50$ . The dashed curve in [Figure 11.4a](#) is the true PSD.**

## 7. Explanation

It can be shown that the MVSE is equivalent to using a bank of bandpass filters to separate the data into its frequency components, estimating the power out of each filter, and finally dividing by the bandwidth. This is similar to the averaged periodogram. The contrasting feature of the MVSE is that the filter frequency responses are a function of the frequency at which the PSD is being estimated. If there is a large frequency component near the frequency band of interest, but outside of the band, then the filter attempts to null it out so that it does not “leak” into the band of interest. In

this way, the resolution appears to be better, i.e., there is less leakage from neighboring frequency bands.

## 8. Comments/References

The bias-variance tradeoff is controlled by the choice of  $p$ . A larger value of  $p$  will improve the resolution but also increase the variance. It can be shown that the true PSD will be obtained if  $\hat{\mathbf{R}}_x$  is the true autocorrelation matrix and  $p \rightarrow \infty$  [Kay and Pakula 2010], as illustrated by the next exercise. Further details can be found in [Kay 1988, Chapter 11].



### Exercise 11.3 – Effect of the choice of $p$

To obtain some guidance as to the appropriate choice of  $p$  for a given type of PSD, one can play the following game. Assume that the ACS  $r_x[k]$  for  $k = 0, 1, \dots, p - 1$  is known. Then,  $[\hat{\mathbf{R}}_x]_{ij} = r_x[i - j]$ . For the AR random process used in the examples for this chapter, in which  $a[1] = 0$ ,  $a[2] = r^2 = 0.9025$ , and  $\sigma_u^2 = 1$ , the true ACS is given by

$$r_x[k] = \frac{\sigma_u^2}{1 - r^4} r^{|k|} \cos[(\pi/2)k].$$

Using this for the “estimated” autocorrelation matrix, plot the MVSE for  $p = 10$  and  $p = 50$ . Do the results appear to coincide with those shown in Figures 11.3a and 11.4a? Could you use this approach to get some idea of an appropriate choice of  $p$ ?



## 11.3. Parametric (Model-Based) Spectral Analysis

In Section 4.4 we discussed the use of the AR model for modeling of a random noise process (could also be used for random signal modeling). The reader may wish to review this material before proceeding. It was found that the AR model resulted in a PSD model given by

$$P_x(f) = \frac{\sigma_u^2}{|1 + a[1] \exp(-j2\pi f) + \dots + a[p] \exp(-j2\pi fp)|^2} \quad (11.7)$$

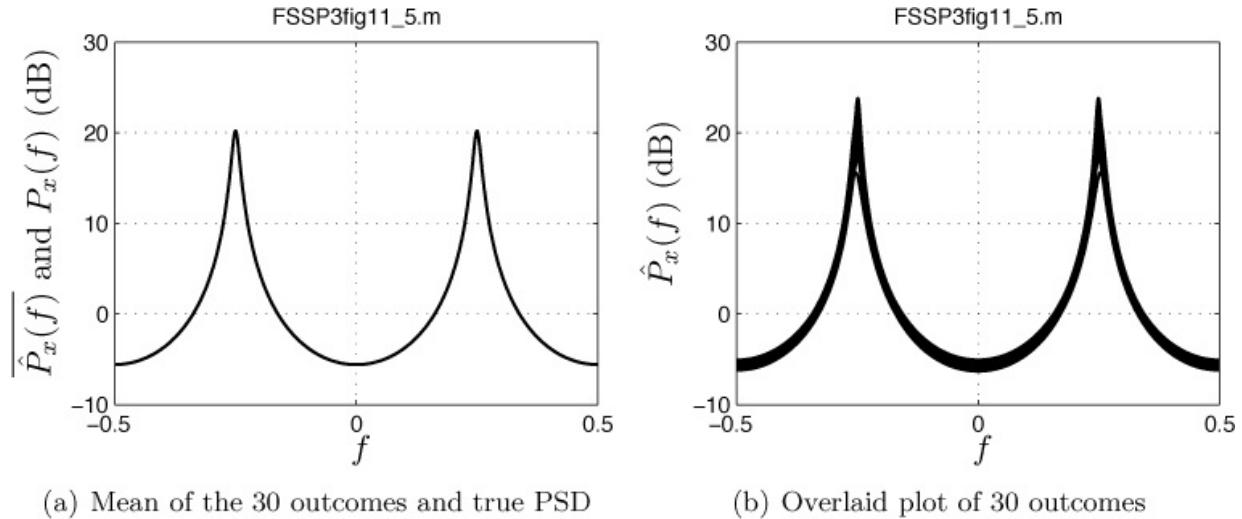
where  $\sigma_u^2 > 0$  is an *excitation noise variance parameter* and  $\{a[1],$

$a[2], \dots, a[p]\}$  are the AR *filter parameters*. The *order* of the model is given by the number of filter parameters  $p$ . It was also mentioned that it can be used to model any PSD, as long as the model order  $p$  is chosen large enough. In that discussion it was mentioned that the parameters could easily be estimated *from data*, i.e.,  $x[n]$  for  $n = 0, 1, \dots, N - 1$ , as opposed to knowledge of the ACS, which was the case in [Section 4.4](#). We now turn to estimation of the AR parameters based on data samples. Once the AR parameters have been estimated, the PSD estimate is obtained by substituting these estimates into [\(11.7\)](#). An obvious approach to estimate the AR parameters is to make use of the Yule-Walker equations in [Section 4.4](#). These equations relate the ACS to the AR parameters. However, it has been observed empirically that the use of the Yule-Walker equations given in [Section 4.4](#), whereby the unknown ACS is estimated directly (using the estimator of [Section 6.4.4](#)), and then substituted into the equations and solved, leads to low resolution spectral estimates. The algorithms to be described next estimate the AR parameters using approximate maximum likelihood methods, which produce the highest resolution and least variance spectral estimates.

Before describing the algorithms one should note (described more fully in [Section 4.4](#)) that for a good model-based spectral estimate, accuracy of the model is paramount to success. If the data is truly an AR process of order  $p$ , then estimating the parameters and substituting into [\(11.7\)](#) will produce good estimates. If, however, the model is incorrect, then very poor results, in terms of low resolution, can be expected.

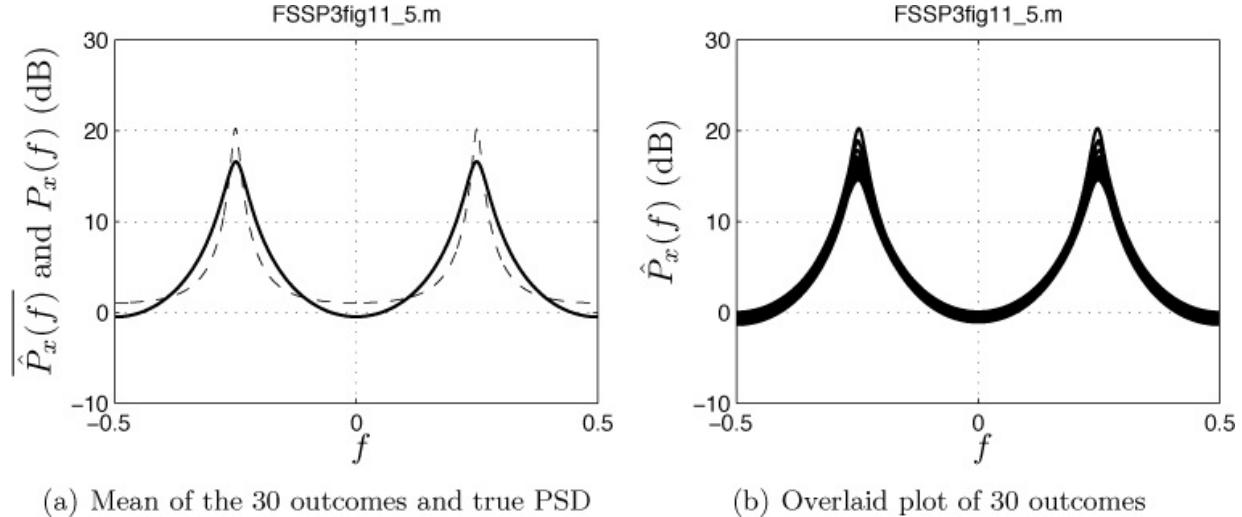
As an example, we consider the data used for the examples in this chapter. It is truly an AR process of order  $p = 2$ . Using the approximate maximum likelihood estimator (MLE), termed the *covariance method*, to estimate the AR parameters  $\{a[1], a[2], \sigma_u^2\}$  we obtain the results shown in [Figure 11.5](#). Note that the mean estimate is nearly identical to the true one and also that the variability is quite low. However, if the original data, which is a pure AR random process of order two, is added to data obtained as the outcome of a white Gaussian noise (WGN) process with variance  $\sigma^2 = 1$ , then the results are poor, as shown in [Figure 11.6](#). Now there is a substantial bias and, as expected, a slight increase in variance due to the WGN. The key result, however, is that *the good resolution of the AR spectral estimator is only attained when the model is accurate*. It can be shown that a random process consisting of an AR process of order  $p$  plus a WGN process is no longer an AR process of order  $p$ , hence, the importance of an accurate model. To improve the resolution one must increase the model order used in [\(11.7\)](#) in an attempt to regain the good resolution. An example of the use

of  $p = 50$  is shown in [Figure 11.7](#), and indeed, improves the resolution. However, it is also noted that increasing the model order to a large value relative to the number of data points  $N$  available (here  $p = 50$  and  $N = 500$ ), leads to large variability. This is because more parameters now need to be estimated. The choice of  $p$  is at best a “balancing act”, with a large  $p$  necessary for good resolution and a small  $p$  necessary for low variance, once again, a bias-variance tradeoff. Good model order estimation approaches are therefore required for the AR spectral estimator, one of which is discussed in [Section 11.3.1](#).



(a) Mean of the 30 outcomes and true PSD      (b) Overlaid plot of 30 outcomes

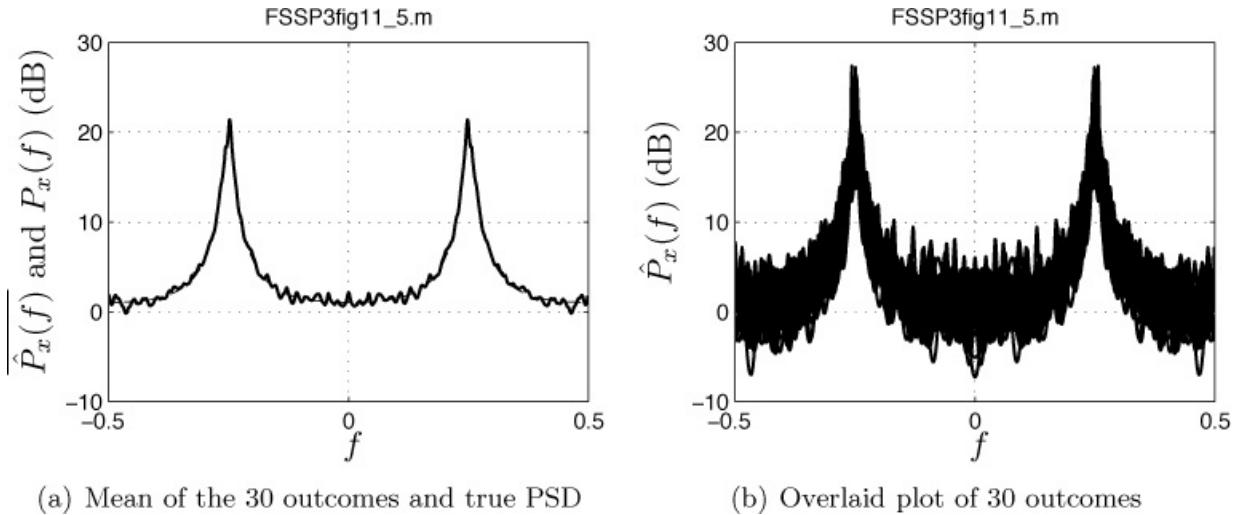
**Figure 11.5: AR PSD estimate for a pure AR random process using the true model order  $p = 2$  for the spectral estimate. The dashed curve in [Figure 11.5a](#) is the true PSD but cannot be seen since it overlaps the mean.**



(a) Mean of the 30 outcomes and true PSD      (b) Overlaid plot of 30 outcomes

**Figure 11.6: AR PSD estimate for a noise-corrupted AR random process using the true AR model order  $p = 2$  for the spectral estimate. The dashed**

curve in [Figure 11.6a](#) is the true PSD of the AR process plus white noise.



**Figure 11.7: AR PSD estimate for a noise-corrupted AR random process using the AR model order  $p = 50$ . The dashed curve in [Figure 11.7a](#) is the true PSD of the AR process plus white noise but cannot be seen since it overlaps the mean.**

There are many algorithms used to estimate the AR parameters, with only slight performance differences between them. We next describe two of the more commonly used ones.

### Algorithm 11.3 – Autoregressive spectral estimator - Covariance method

#### 1. Problem

Estimate the PSD of a stationary random process.

#### 2. Application area example

AR spectral estimation is used extensively for speech processing [[Rabiner and Schafer 1978](#)]. For historical reasons it is referred to as *linear predictive coding*.

#### 3. Data model/assumptions

The data  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

#### 4. Estimator

The AR filter parameters are estimated by solving a set of linear equations

$$\begin{bmatrix} c_x[1,1] & c_x[1,2] & \dots & c_x[1,p] \\ c_x[2,1] & c_x[2,2] & \dots & c_x[2,p] \\ \vdots & \vdots & \ddots & \vdots \\ c_x[p,1] & c_x[p,2] & \dots & c_x[p,p] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} c_x[1,0] \\ c_x[2,0] \\ \vdots \\ c_x[p,0] \end{bmatrix} \quad (11.8)$$

where

$$c_x[i,j] = \frac{1}{N-p} \sum_{n=p}^{N-1} x[n-i]x[n-j]$$

and

$$\widehat{\sigma}_u^2 = c_x[0,0] + \sum_{i=1}^p \hat{a}[i]c_x[0,i].$$

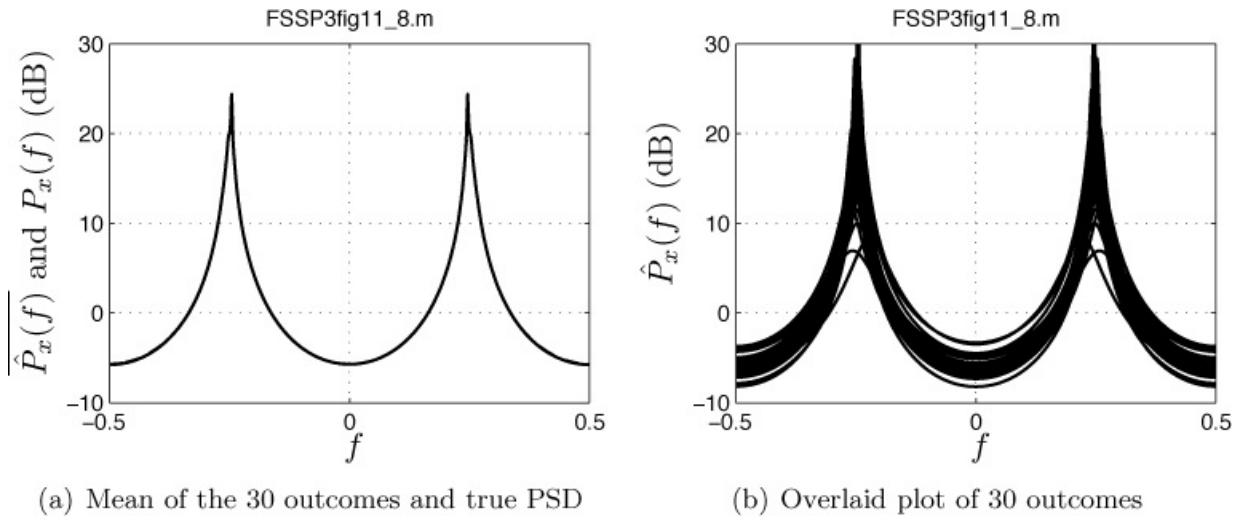
The MATLAB subprogram `FSSP3PSD_est_AR_cov.m` can be used to implement the spectral estimator, and is contained on the CD.

## 5. Performance

It has been empirically observed that the resolution of the AR spectral estimator is better than the averaged periodogram and also the MVSE (for the same variance). Confidence intervals are not available for this spectral estimator due to its very nonlinear nature. The estimator for the AR parameters of a Gaussian AR( $p$ ) process is approximately an MLE and hence is asymptotically optimal. However, there are no optimality properties of the spectral estimator unless  $x[n]$  is truly an AR process of order  $p$ . The approximate Cramer-Rao lower bound for the AR parameters of an AR( $p$ ) process, which is sometimes useful, can be found in [Kay 1988, pg. 191].

## 6. Example

When an AR model is an accurate representation of the data and  $p \ll N$ , then the AR PSD estimator exhibits superior performance. In [Figure 11.5](#) we have an AR process of order  $p = 2$ , and have used the true model order in addition to a relatively long data record length of  $N = 500$ . That example is repeated in [Figure 11.8](#) using only  $N = 50$  data samples. Comparing this to [Figure 11.5](#) it is seen that only the value of the peak is slightly in error and the variance is somewhat increased, attesting to the potential ability of the AR spectral estimator to produce good estimates for short data records.



**Figure 11.8: AR PSD estimate using the Covariance method for a pure AR random process using the true model order  $p = 2$  but only  $N = 50$  data points. The dashed curve in Figure 11.8a is the true PSD but cannot be seen since it overlaps the mean.**

## 7. Explanation

The estimate of the AR filter parameters is obtained using a minimization of the least squares error

$$J(\mathbf{a}) = \sum_{n=p}^{N-1} \left( x[n] + \sum_{i=1}^p a[i]x[n-i] \right)^2.$$

The estimate of the excitation white noise variance is obtained as

$$\widehat{\sigma_u^2} = \frac{1}{N-p} J(\hat{\mathbf{a}})$$

once the AR filter parameters have been estimated. The solution has been implemented in `AR_par_est_cov.m`, and is contained on the CD.

## 8. Comments/References

The estimates of the AR filter parameters are not guaranteed to produce a stable filter, i.e., the estimated poles may fall outside the unit circle. This is usually of no concern for spectral estimation, but sometimes one wishes to generate additional data using the AR model with the estimated parameters. In this case one can either “reflect” the poles outside the unit circle of the z-plane to inside the unit circle [Oppenheim and Schafer, pg. 349] or use the next algorithm to be described, the Burg algorithm. See also [[Kay 1988](#), pp. 185–188] for further details.

---

---

## Exercise 11.4 – Enhanced resolution of AR spectral estimator

Assume that the data is given by  $x[n] = \cos(2\pi(0.1)n) + 0.5 \cos(2\pi(0.13)n)$  for  $n = 0, 1, \dots, N - 1 = 15$ . The use of this data set is meant only to illustrate the potential enhanced resolution. If we indeed had knowledge that  $x[n]$  consisted of sinusoids only or more practically, sinusoids in WGN, we should use an MLE, for example, [Algorithm 9.9](#). For this data set run `PSD_est_AR_cov(x, p, plotit, Nf, plotdB)` with  $p=4$ ,  $plotit=1$ ,  $Nf=1024$ , and  $plotdB=1$  to generate the AR PSD estimate in dB and have it plotted. Compare your results against the magnitude of the Fourier transform shown in [Figure 11B.2b](#), which is also based on the same  $N = 16$  point data record. You can ignore any scaling differences since the intent of the exercise is to ascertain if the two sinusoidal peaks can be resolved. Next add some WGN with variance  $\sigma^2 = 0.0001$  to the sinusoids and repeat. Are the sinusoids resolved?

---

---

## Algorithm 11.4 – Autoregressive spectral estimator - Burg method

### 1. Problem

Estimate the PSD of a stationary random process.

### 2. Application area example

AR spectral estimation can be used for prewhitening data. Once the filter parameters have been estimated, a prewhitener is given by an FIR filter with system function  $A(z) = 1 + a[1] \exp(-j 2\pi f) + \dots + a[p] \exp(-j 2\pi f p)$  [[Birch 1988](#)]. This is called the *inverse filter* since it undoes the coloration of the original filter  $1/A(z)$ , which was used to color the white excitation noise of the AR model (see [Section 4.4](#)).

### 3. Data model/assumptions

The data  $x[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

### 4. Estimator

The Burg method is quite complicated, relying on knowledge of the

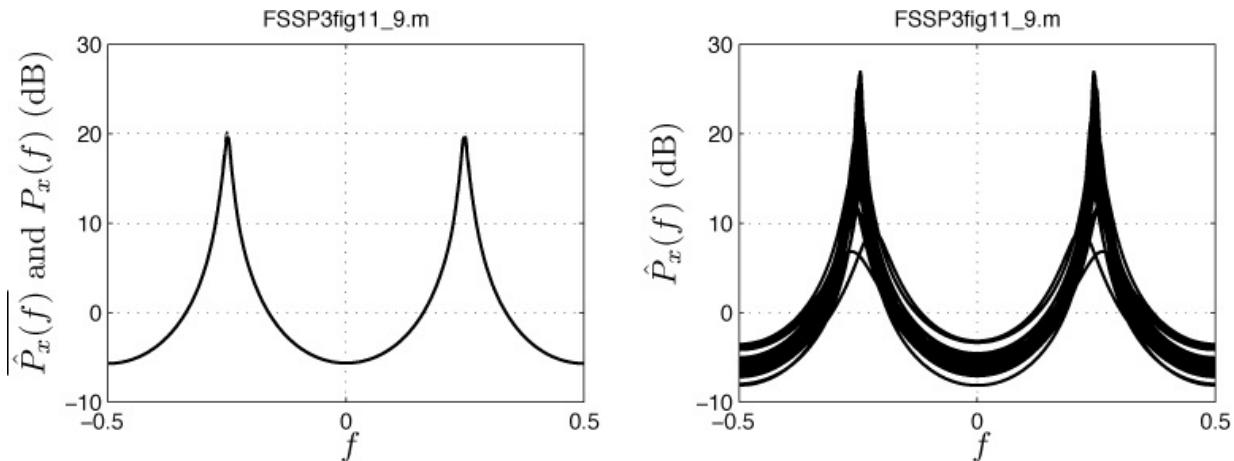
Levinson recursion for solving the Yule-Walker equations, as well as some less familiar linear prediction concepts. See the references for further details. The MATLAB subprogram `PSD_est_AR_Burg.m` can be used to implement the spectral estimator, and is contained on the CD.

## 5. Performance

It has been empirically observed that the resolution of the AR spectral estimator is better than the averaged periodogram and also the MVSE (for the same variance). Confidence intervals are not available for this spectral estimator due to its very nonlinear nature. The estimator for the AR parameters of a Gaussian AR( $p$ ) process is approximately an MLE and hence is asymptotically optimal. However, there are no optimality properties of the spectral estimator unless  $x[n]$  is truly an AR process of order  $p$ . The approximate Cramer-Rao lower bound for the AR parameters of an AR( $p$ ) process, which is sometimes useful, can be found in [Kay 1988, pg. 191].

## 6. Example

For the same example as shown in [Figure 11.8](#) we use the Burg method to produce the spectral estimates shown in [Figure 11.9](#). The results are very similar to the use of the covariance method. For this example, the Burg method outperforms the covariance method slightly (note the peak values in [Figures 11.9a](#) and [11.8a](#)).



(a) Mean of the 30 outcomes and true PSD

(b) Overlaid plot of 30 outcomes

**Figure 11.9: AR PSD estimate using the Burg method for a pure AR random process using the true model order  $p = 2$  but only  $N = 50$  data points. The dashed curve in [Figure 11.3](#) is the true PSD but cannot be seen since it overlaps the mean.**

## **7. Explanation**

The method relies on knowledge of the *reflection coefficients*, also called the *partial correlation coefficients*, which is beyond the scope of our discussions. See the references for further details.

## **8. Comments/References**

For moderate length data records the results are nearly identical to the covariance method. The advantage is that the estimated poles are always within the unit circle, leading to a stable filter. This property also explains the tendency of the Covariance method to display “too sharp” peaks, as illustrated by comparing [Figure 11.8b](#) to [Figure 11.9b](#). Since the Covariance method is not constrained to produce stable poles, i.e., poles within the unit circle of the z-plane, the estimated poles have more freedom to “wander” near the unit circle, producing a sharp peak in the estimated PSD. See [Exercise 15.3](#) for another example and also [[Kay 1988](#), pp. 228–231] for further details.



---

---

### **Exercise 11.5 – Effect of a larger model order than the true one**

For the AR process used in the examples of [Algorithms 11.3](#) and [11.4](#) with  $a[1] = 0$ ,  $a[2] = 0.9025$ , and  $\sigma_u^2 = 1$  run the Burg algorithm on data generated using `ARgendata.m` with only  $N = 30$  data points. To do so run the MATLAB subprogram

`PSD_est_AR_Burg(x,p,plotit,Nf,plotdB)` with  
`plotit=1`, `Nf=1024`, and `plotdB=1`. Use  $p = 2$ ,  $p = 6$ , and  $p = 10$ .  
Use the MATLAB command `axis([-0.5 0.5 -10 30])` after the call to  
`PSD_est_AR_Burg.m` to get the same vertical axis scaling as in [Figure 11.9](#). Describe the spectral estimates.



---

#### **11.3.1. Estimating the AR Model Order**

From [Exercise 11.5](#) you should have observed that too large a model order can result in spurious spectral peaks. In conjunction with model-based spectral analysis it is therefore necessary to have a good model order estimator. In particular, for AR spectral analysis the exponentially embedded family (EEF) method of model order estimation (see [Section 5.5](#)) can be used. It chooses the

AR model order  $p$  as the value of  $k$  that maximizes the criterion

$$\text{EEF}(k) = \begin{cases} \xi_k - k \left[ \ln \left( \frac{\xi_k}{k} \right) + 1 \right] & \text{if } \frac{\xi_k}{k} \geq 1 \\ 0 & \text{if } \frac{\xi_k}{k} < 1 \end{cases} \quad k = 1, 2, \dots, p_{\max} \quad (11.9)$$

where

$$\xi_k = (N - k) \ln \left( \frac{\mathbf{x}_k^T \mathbf{x}_k}{\mathbf{x}_k^T \mathbf{x}_k - \mathbf{x}_k^T \mathbf{H}_k (\mathbf{H}_k^T \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{x}_k} \right),$$

$\mathbf{x}_k = [x[k] \ x[k+1] \ \dots \ x[N-1]]^T$ , and  $\mathbf{H}_k$  is the  $(N - k) \times k$  matrix

$$\mathbf{H}_k = \begin{bmatrix} x[k-1] & x[k-2] & \dots & x[0] \\ x[k] & x[k-1] & \dots & x[1] \\ \vdots & \vdots & \vdots & \vdots \\ x[N-2] & x[N-3] & \dots & x[N-1-k] \end{bmatrix}.$$

As an example, for the AR random process of order  $p = 2$  used for all the figures, if we use  $N = 50$  and  $p_{\max} = 10$ , the EEF chooses the correct order. The MATLAB subprogram `AR_est_order.m` implements the model order estimation algorithm, and is contained on the CD.

---

### Exercise 11.6 – Model order estimation for short data records

Run the model order estimation algorithm for the same data set as in [Exercise 11.5](#) using  $N = 30$  and  $p_{\max} = 10$ . Does it yield the correct order?

Note that it is *never advisable to set the maximum model order more than about  $N/3$* .




---

## 11.4. Time-Varying Power Spectral Densities

In [Section 4.5](#) we discussed PSDs that vary in time. Recall that such a PSD does not theoretically exist since the assumption of stationarity is required to define the PSD. However, in practice, if a random process has a spectral content that is *slowly varying*, it is possible to gain useful information by defining one as  $P_x(f, n)$ . We assume in doing so that the rate of change with  $n$  is slow compared to  $L$ , where  $1/L$  is the desired resolution. For maximum resolution we could use a periodogram for each short block of data, over which we can assume *local stationarity*, and then stack the results in time. This leads to the *spectrogram*, a

nonparametric spectral estimator. Alternatively, a model-based estimator could be used for each block, which is particularly appealing since the increased resolution (if the model is accurate!) will allow shorter blocks, and hence will better be able to follow the change in spectral power. This was illustrated in [Section 4.5](#) using an AR spectral estimator. Various modifications such as overlapping the blocks are used in practice.

## References

- Bendat, J.S., A.G. Piersol, *Random Data: Analysis and Measurement Procedures*, J. Wiley, NY, 1971.
- Birch, G.E., “Application of Prewhitening to AR Spectral Estimation of EEG”, *IEEE Trans. on Biomedical Engineering*, Vol. 35, pp. 640–645, August 1988.
- Carbone, C.P., S. Kay, “A Novel Normalization Algorithm Based on the Three-Dimensional Minimum Variance Spectral Estimation”, *IEEE Trans. on Aerospace and Electronic Systems*, Vol. 48, pp. 430–448, Jan. 2012.
- Harris, F.J., “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform”, *Proceedings of the IEEE*, Vol. 66, pp. 51–83, Jan. 1978.
- Jenkins, G.M., D.G. Watts, *Spectral Analysis and Its Applications*, Holden-Day, San Francisco, 1968.
- Johnson, D.H., D.E. Dudgeon, *Array Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- Kay, S., *Intuitive Probability and Random Processes Using MATLAB*, Springer, NY, 2006.
- Kay, S., L. Pakula, “Convergence of the Multidimensional Minimum Variance Spectral Estimator for Continuous and Mixed Spectra”, *IEEE Signal Processing Letters*, Vol. 17, pp. 28–31, Jan. 2010.
- Oppenheim, A.W., R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.
- Priestley, M.B., *Spectral Analysis and Time Series*, Academic Press, NY, 1981.
- Rabiner, L.R., R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
- Welch, P.D., “The Use of the Fast Fourier Transform for Estimation of Power Spectra: A Method Based on Time Averaging over Short, Modified

Periodograms”, *IEEE Trans. on Audio and Electroacoustics*, Vol. AU15, pp. 70–73, June 1967.

## Appendix 11A. Fourier Spectral Analysis and Filtering

Fourier spectral analysis is based on filtering the data to extract the frequency components in a narrow band, determining the average power of those frequency components, and finally dividing by the bandwidth of the filter to yield a power spectral *density*. For example, consider a filter whose passband is centered about  $f = f_0$  and whose bandwidth is small. The impulse response, which is a complex one, is given by

$$h[n] = \begin{cases} \frac{1}{L} \exp(j2\pi f_0 n) & n = 0, 1, \dots, L - 1 \\ 0 & \text{otherwise.} \end{cases}$$

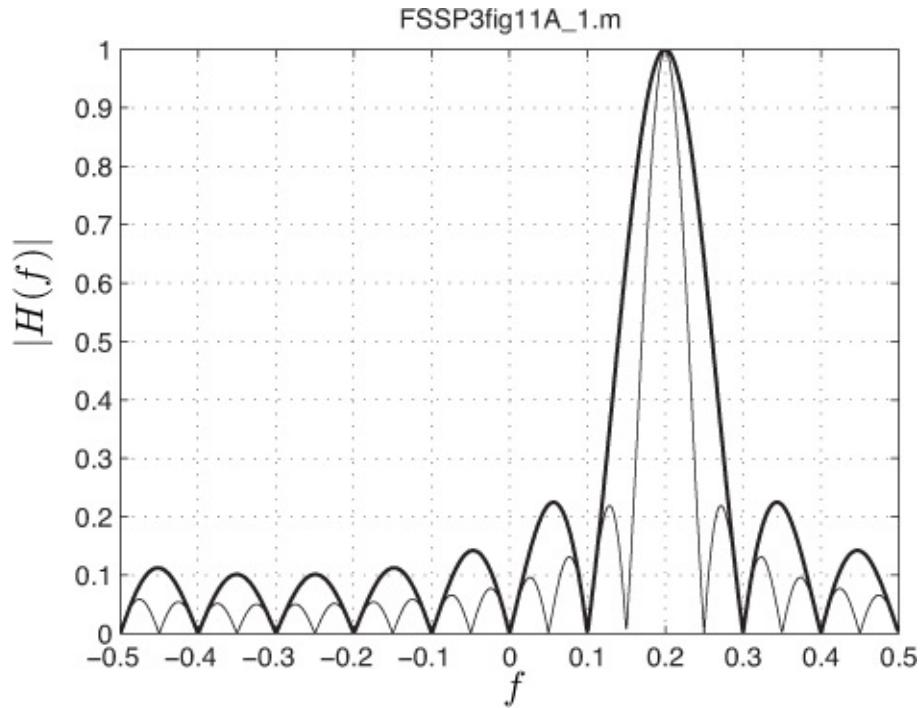
For  $f_0 = 0.25$ ,  $L = 10$ , and  $L = 20$ , the magnitude of the frequency response, where the frequency response is defined as

$H(f) = \sum_{n=0}^{L-1} h[n] \exp(-j2\pi f n)$ , is shown in [Figure 11A.1](#). It is given analytically as

$$|H(f)| = \left| \frac{\sin \pi(f - f_0)L}{L \sin \pi(f - f_0)} \right|.$$

The filter is centered in frequency at  $f = f_0 = 0.2$  and has a 3 dB bandwidth of approximately  $1/L$  (meaning that

$|H(f_0 + 1/(2L))| = |H(f_0 - 1/(2L))| \approx 1/\sqrt{2}$ ). Hence, the filter is able to separate out frequency components that are spaced apart by more than the bandwidth of the filter, i.e., approximately  $1/L$ , which is the so-called *Rayleigh resolution*. In the averaged periodogram we compute the Fourier transform for each block, with the  $i$ th block denoted as  $y_i[n]$  for  $n = 0, 1, \dots, L - 1$ .



**Figure 11A.1: Magnitude of frequency response of a narrowband filter centered at  $f_0 = 0.2$ . The solid curve is for  $L = 10$  and the lighter curve is for  $L = 20$ .**

In particular, consider the PSD estimate for  $f = f_0$ . Then, we have for the Fourier transform of the  $i$ th block of data at  $f = f_0$

$$\begin{aligned}
 \sum_{k=0}^{L-1} y_i[k] \exp(-j2\pi f_0 k) &= \sum_{k=0}^{L-1} y_i[k] \exp[j2\pi f_0(n-k)] \Big|_{n=0} \\
 &= L \sum_{k=0}^{L-1} y_i[k] \frac{1}{L} \exp[j2\pi f_0(n-k)] \Big|_{n=0} \\
 &= L \sum_{k=0}^{L-1} y_i[k] h[n-k] \Big|_{n=0} \\
 &= L z_i(f_0)
 \end{aligned}$$

so that the complex output of the narrowband filter sampled at time  $n = 0$  is

$$z_i(f_0) = \frac{1}{L} \sum_{k=0}^{L-1} y_i[k] \exp(-j2\pi f_0 k). \quad (11A.1)$$

Thus, the output of the narrowband filter centered at  $f = f_0$  is just the Fourier

transform scaled by  $1/L$  and the narrowness of the filter depends upon  $L$ , since its bandwidth is  $1/L$ . To estimate the PSD we need a “bank” of narrowband filters, or equivalently we need to take the Fourier transform at many frequencies and not just  $f = f_0$ . The PSD is estimated as the average power out of the filter, i.e., the value of  $|z_i(f_0)|^2$ , divided by the bandwidth of the narrowband filter, which is  $1/L$ . In fact, it can be shown that for large  $L$ ,  $E[|z_i(f_0)|^2/(1/L)] \approx P_x(f_0)$ . Thus, we have that using (11A.1), the PSD estimate is

$$\begin{aligned}\hat{P}_x(f_0) &= \frac{1}{I} \sum_{i=1}^I \frac{|z_i(f_0)|^2}{1/L} = \frac{1}{I} \sum_{i=1}^I \frac{1}{1/L} \left| \frac{1}{L} \sum_{k=0}^{L-1} y_i[k] \exp(-j2\pi f_0 k) \right|^2 \\ &= \frac{1}{I} \sum_{i=1}^I \frac{1}{L} \left| \sum_{k=0}^{L-1} y_i[k] \exp(-j2\pi f_0 k) \right|^2 \\ &= \frac{1}{I} \sum_{i=1}^I \hat{P}_x^{(i)}(f_0)\end{aligned}$$

which is the averaged periodogram for the frequency  $f_0$ . It is important to note the following. If we do not break up the data record into blocks, then only a single filter output sample is available, i.e.,  $z_1(f_0)$ , to estimate the power.

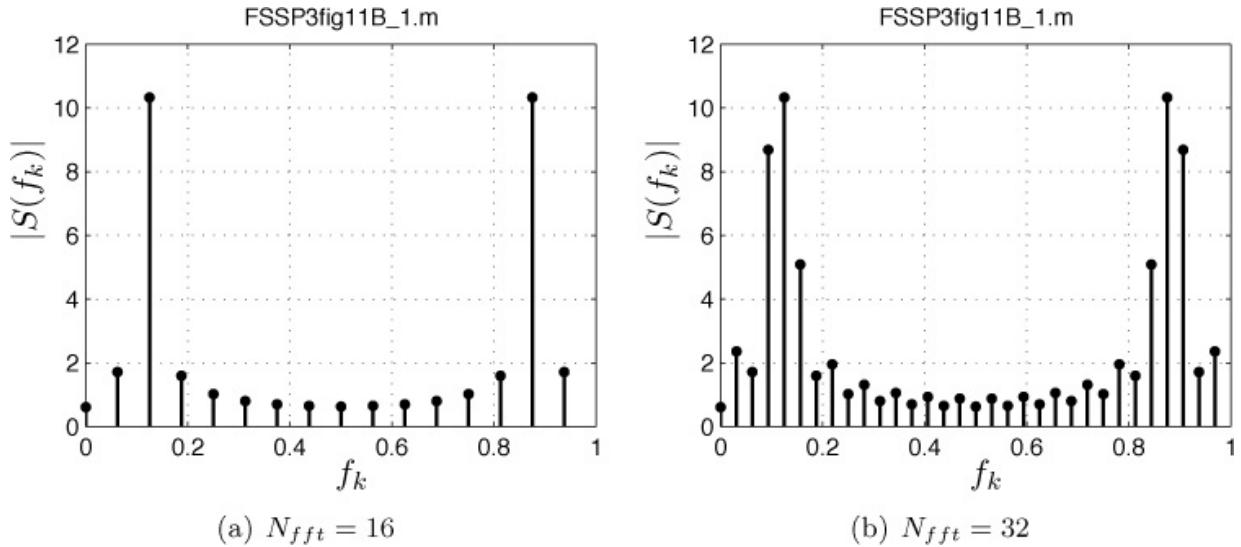
Therefore, no matter how large  $N$  is, if we use all the data for filtering, then there can be no averaging of the periodograms. Thus, the periodogram will never converge to  $P_x(f_0)$ . This was observed in Figure 6.10. By dividing up the data record into blocks, each block is smaller than the original  $N$  point data record and the filter will have a wider passband, as illustrated in Figure 11A.1. Thus, the resolution suffers, but the variance decreases.

## Appendix 11B. The Issue of Zero Padding and Resolution

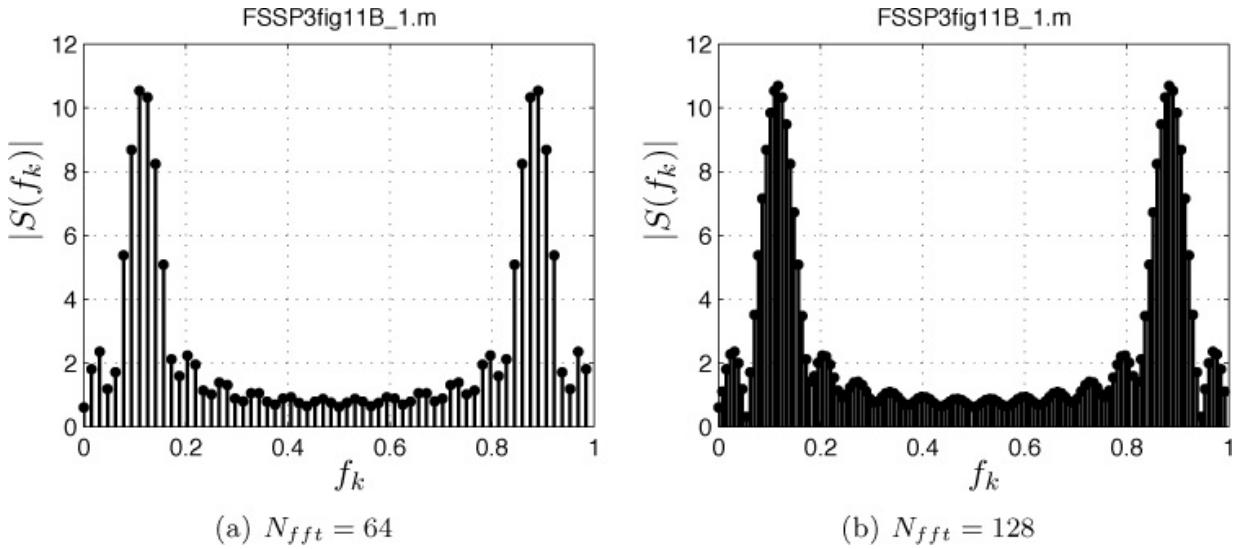
A source of confusion in Fourier spectral analysis when using the FFT to compute the Fourier transform is the use of zero padding. Since a computer evaluation of a function can only provide the function value at a finite number of points, it is important to use a sufficient number of points so that the function is rendered accurately. *It is essential to observe that the discrete-time Fourier transform of a finite-length sequence  $s[n]$  is defined as*

$$S(f) = \sum_{n=0}^{N-1} s[n] \exp(-j2\pi f n) \quad -1/2 \leq f \leq 1/2$$

which is a *continuous function of frequency*. It is only our inability to compute it for *all* frequency points in the interval  $-1/2 \leq f \leq 1/2$  that leads to an evaluation at a finite number of frequency points. For example, in [Figures 11B.1](#) and [11B.2](#) we show the result of computing  $|S(f)|$  for  $s[n] = \cos(2\pi(0.11)n) + 0.5 \cos(2\pi(0.13)n)$  for  $n = 0, 1, \dots, N - 1 = 15$  at the frequency points  $f_k = 0, 1/N_{fft}, \dots, (N_{fft} - 1)/N_{fft}$ , where  $N_{fft} = 16$ ,  $N_{fft} = 32$ ,  $N_{fft} = 64$ , and  $N_{fft} = 128$ . Of course, these frequency points lie in the interval  $[0, 1]$  so that if we desire an evaluation in the interval  $[-1/2, 1/2]$ , as we have always agreed to, we must “flip around” the upper frequency points, which is allowable due to the periodicity of the discrete-time Fourier transform (see the MATLAB command `fftshift`). It is seen that the two sinusoids are not resolvable since their frequency separation is 0.02, which would require at least  $1/0.02 = 50$  data points. Increasing the FFT size only yields a better depiction of the true  $|S(f)|$ , *not any increased spectral resolution*. The method of “zero padding” amounts to nothing more than adding zeros to the data so that when the augmented data set is input to the FFT algorithm, the output of the algorithm yields the desired number of frequency points. Usually, for maximum computational speed we set  $N_{fft} = 2^v$ , for  $v$  an integer.



**Figure 11B.1:** FFT evaluation of discrete-time Fourier transform for  $N_{fft} = 16$  and  $N_{fft} = 32$ .



**Figure 11B.2: FFT evaluation of discrete-time Fourier transform for  $N_{fft} = 64$  and  $N_{fft} = 128$ .**

## Appendix 11C. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand ('state', 0)` and `randn ('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

- 11.1 You should observe that there are no ripples. However, the width of the main peak is broadened somewhat due to the data windowing. You can run the program `FSSP3exer11_1.m`, which is contained on the CD, to obtain the results.
- 11.2 For the smallest value of  $L$  you will observe a smooth spectral estimate with a main resonance that is too wide. As  $L$  increases the width of the main peak will be nearly equal to the true one but the variability (as evidenced by the small wiggles) will increase. For this particular PSD it appears that  $L = 100$  is sufficient to attain good resolution. This indicates a 3 dB bandwidth of the main peak of about  $1/L = 0.01$  cycles/sample. You can run the program `FSSP3exer11_2.m`, which is contained on the CD, to obtain the results.
- 11.3 The MVSE using a known autocorrelation matrix of dimension  $50 \times 50$  produces a PSD very close to the true one. Only the value of the peak appears to be too small. Comparing these results to those in [Figures 11.3a](#) and [11.4a](#), we see that they are overly optimistic. However, it does give

us some general guidance as to the appropriate order to use, if we have some idea of the type of PSDs we may encounter. You can run the program FSSP3exer11\_3.m, which is contained on the CD, to obtain the results.

**11.4** Using the parameters given, the two sinusoids should be well resolved. This is in contrast to the use of a Fourier transform, for which the two spectral peaks are merged together and cannot be seen in the plot (see [Figure 11B.2b](#)). By adding WGN with variance  $\sigma^2 = 0.0001$  (SNR = 10  $\log_{10}((1/2)^2/2)/0.0001 = 31$  dB for the lower level sinusoid), the resolution decreases dramatically and the sinusoids are no longer resolved. This is due to the modeling inaccuracy. You can run the program FSSP3exer11\_4.m, which is contained on the CD, to obtain the results.

**11.5** You should observe that for  $p = 2$ , the true model order, the PSD estimate is very close to the true one. As the order increases, however, “spurious peaks” appear, rendering the spectral estimate useless. Clearly, a model order estimator is needed to avoid this problem. You can run the program FSSP3exer11\_5.m, which is contained on the CD, to obtain the results.

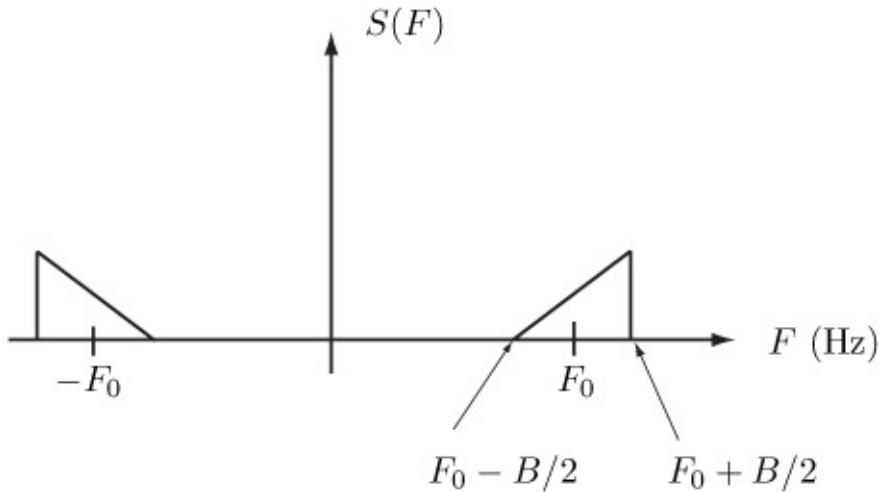
**11.6** The estimated model order should be the true one  $p = 2$ . You can run the program FSSP3exer11\_6.m, which is contained on the CD, to obtain the results.

# **Part III. Real-World Extensions**

# Chapter 12. Complex Data Extensions

## 12.1. Introduction

In all our previous discussions of statistical signal processing we have assumed that the data consists of a single record of  $N$  real data samples. The data is usually presumed to have originated as the result of sampling a real analog temporal waveform at equally spaced instants of time. The intent in this chapter is to address *complex* data, primarily used in the fields of radar, sonar, and communications. Complex data naturally arises when one is interested in processing a signal that is contained within a given frequency band, with the center frequency of the band at a very high frequency, relative to the bandwidth of the signal. These so-called *bandpass signals* have *continuous-time* Fourier transforms that are concentrated about a center frequency. An example of the Fourier transform of such a signal is shown in [Figure 12.1](#), where it should be noted that the frequency is *analog frequency* and is measured in Hz (cycles/sec). We use a capital  $F$  to distinguish this frequency from the discrete-time frequency  $f$ , measured in cycles/sample.

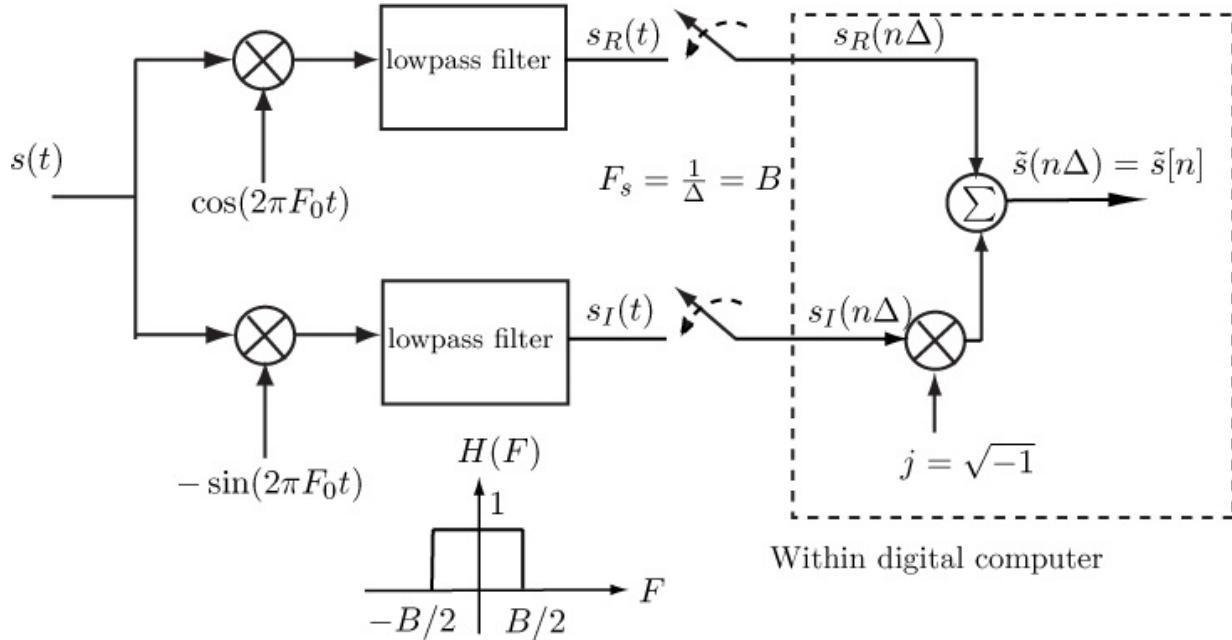


**Figure 12.1: Fourier transform of a real bandpass signal  $s(t)$ .**

From a hardware perspective it is more economical to translate the signal down to *baseband* so that its Fourier transform is centered at  $F = 0$ . Then, Nyquist sampling can be performed on the lowpass signal whose highest frequency is much lower. This demodulation can be accomplished via the operation shown in [Figure 12.2](#). The lowpass analog signals  $s_R(t)$  and  $s_I(t)$  are sampled at the Nyquist rate of  $F_s = 1/B$  samples per second. Note that two

channels of demodulation are necessary, i.e., the signal is demodulated with both a cosine oscillator and a sine oscillator. Otherwise, the lowpass signal will not retain all the information of the bandpass signal. For example, consider the bandpass signal

$$s(t) = \cos(2\pi F_2 t) - \cos(2\pi F_1 t) \quad (12.1)$$

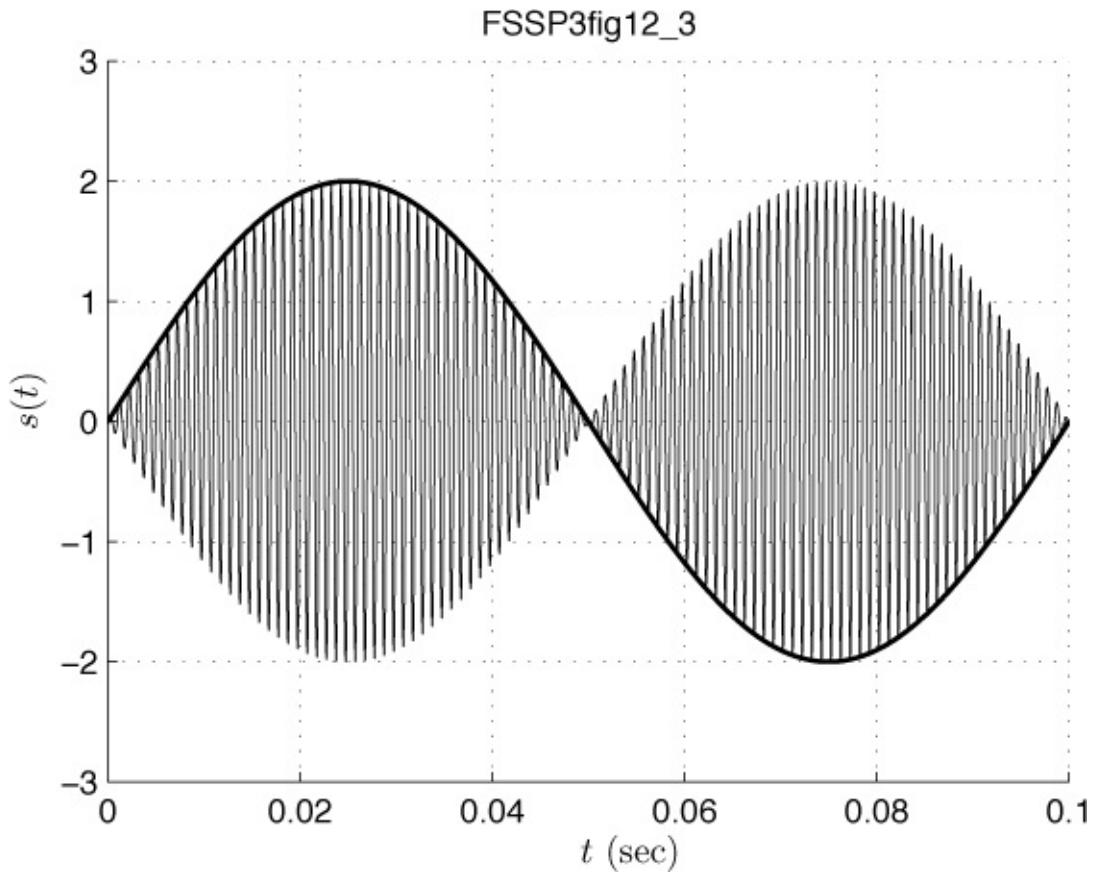


**Figure 12.2: Demodulation of a real bandpass signal to baseband.**

where  $F_1 < F_2$ . The center frequency of the band can be considered as the frequency midpoint  $F_0 = (F_1 + F_2)/2$ . An illustration is given in [Figure 12.3](#). The center frequency of 1000 Hz manifests itself as the high frequency sinusoid having 10 periods in 0.01 seconds for a fundamental period of 0.001 seconds. The “envelope” is sinusoidal with a frequency of  $F_2 - F_0 = 10$  Hz. It can be shown that using  $\exp(j\theta) = \cos \theta + j \sin \theta$ , the signal can be written as

$$s(t) = \cos(2\pi(1010)t) - \cos(2\pi(990)t) = 2\operatorname{Re}[j \sin(2\pi(10)t) \exp(j2\pi(1000)t)]$$

where  $\operatorname{Re}$  denotes the real part. The solid line in [Figure 12.3](#) is  $2\sin(2\pi(10)t)$  and is clearly related to the envelope of the bandpass signal. It is this lowpass signal, i.e., the envelope, that maintains the information encoded in the bandpass signal.



**Figure 12.3: A bandpass signal with  $F_0 = 1000$  Hz as the center frequency and  $F_1 = 990$  Hz and  $F_2 = 1010$  Hz.**

---

### Exercise 12.1 – Finding the complex envelope

The analog *complex envelope*  $\tilde{s}(t)$  of a real bandpass signal  $s(t)$  is defined implicitly as

$$s(t) = 2\operatorname{Re} [\tilde{s}(t) \exp(j2\pi F_0 t)].$$

Find the complex envelope for the bandpass signal given in (12.1).

For the bandpass signal of (12.1) we next find the output of the lowpass filter shown in Figure 12.2. Multiplying by  $\cos(2\pi F_0 t)$  and also by  $-\sin(2\pi F_0 t)$ , and then lowpass filtering, produces upon using  $\cos(A) \cos(B) = (\cos(A+B)+\cos(A-B))/2$

$$\begin{aligned}
s_R(t) = [s(t) \cos(2\pi F_0 t)]_{LPF} &= [(\cos(2\pi F_2 t) - \cos(2\pi F_1 t)) \cos(2\pi F_0 t)]_{LPF} \\
&= \frac{1}{2} [\cos[2\pi(F_2 - F_0)t] - \cos[2\pi(F_1 - F_0)t]] \\
&= 0
\end{aligned}$$

where the sum frequency terms fall outside the lowpass filter bandwidth and so are filtered out. Using  $\sin(A) \cos(B) = (\sin(A + B) + \sin(A - B))/2$  produces

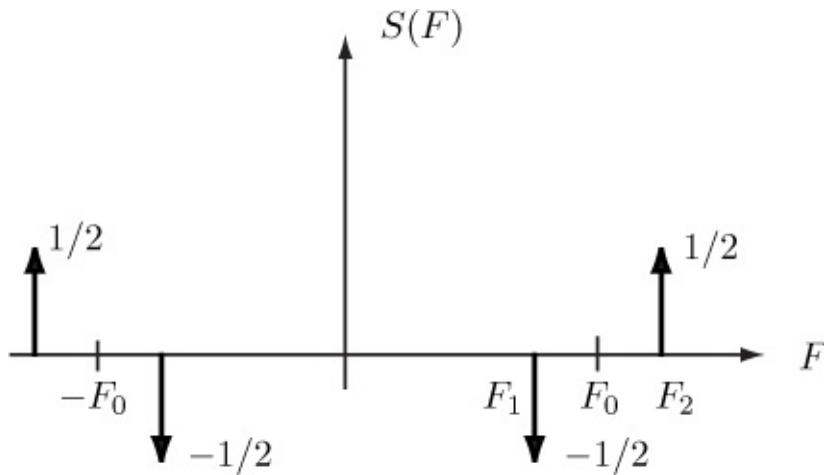
$$\begin{aligned}
s_I(t) = -[s(t) \sin(2\pi F_0 t)]_{LPF} &= -[(\cos(2\pi F_2 t) - \cos(2\pi F_1 t)) \sin(2\pi F_0 t)]_{LPF} \\
&= -\frac{1}{2} [\sin[2\pi(F_0 - F_2)t] - \sin[2\pi(F_0 - F_1)t]] \\
&= \sin[2\pi(F_2 - F_0)t].
\end{aligned}$$

Note that if we had only used cosine demodulation, the output of the lowpass filter  $s_R(t)$  would have been zero. Hence, we need both channels of demodulation.



### Exercise 12.2 – Cosine demodulation

Since  $\cos(2\pi F t) = (\exp(j2\pi F t) + \exp(-j2\pi F t))/2$ , the Fourier transform, also called the spectrum, of  $s(t)$  given by (12.1) can be depicted as shown in [Figure 12.4](#). Each arrow represents a Dirac delta function, and the downward arrow indicates a negative amplitude of the complex sinusoid. Noting that demodulating by a cosine with frequency  $F_0$  shifts the spectrum up in frequency by  $F_0$  (to the right) and down in frequency (to the left), can you explain why cosine demodulation for this signal results in a zero output? Hint: Don't forget that after shifting the spectrum, the signal is lowpass filtered.



**Figure 12.4: Fourier transform of two real sinusoids. Each upward arrow represents a Dirac impulse of strength 1/2. The downward arrow represents a sinusoid with a negative amplitude.**

Referring to [Figure 12.2](#) it is customary to combine the two sequences  $s_R[n] = s_R(n\Delta)$  and  $s_I[n] = s_I(n\Delta)$  together to form a complex valued sequence,  $\tilde{s}[n] = s_R[n] + js_I[n]$ . This is called the discrete-time *complex envelope*. Also,  $s_R[n]$  is called the *in-phase* component and  $s_I[n]$  is called the *quadrature* component, which refers to the demodulating sinusoids being a cosine and a sine function. It is this process of combining the two channels of data into a complex signal that leads to the need to analyze complex data. Otherwise, we would have to consider a vector signal at each time instant consisting of real components as  $[s_R[n] s_I[n]]^T$ . The use of complex data representations proves to be convenient and may be thought of as analogous to the use of phasors in electrical circuits or the use of a complex Fourier series instead of a real one. Ultimately, after processing the complex data the final result is a real number, for example, a real part only or a magnitude or an angle.

In this chapter we consider complex data, which we will denote as  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$ . The tilde will be used whenever it is necessary to call attention to the complex nature in order to avoid confusion. Also, the observed data set will consist of  $N$  *complex samples*, which is equivalent to  $2N$  real data samples comprising the real and imaginary parts.



**Be careful of definitions**

In dealing with complex signals and noise the Fourier transform is often used. Engineering convention defines the Fourier transform of a continuous-time signal as  $S(F) = \int_{-\infty}^{\infty} s(t) \exp(-j2\pi F t) dt$ , using a  $-j$  for the definition in the exponential. Of course, for the inverse Fourier transform we use a  $+j$ . In some fields just the opposite convention is used, with a  $+j$  for the Fourier transform. If not noted, this will cause wrong results and much consternation! Similar comments apply to the definition of the complex envelope.



In the next section we summarize the important differences between real and complex signals. Also, since the complex signals we will be analyzing are obtained via demodulation and subsequent concatenation into a complex quantity, the same processing converts the real noise into complex noise. In [Section 12.3](#) we will summarize the key differences between real noise and complex noise. With this background we will then be able to present the extensions to some of the detection, estimation, and spectral estimation algorithms.

## 12.2. Complex Signals

The complex envelope signal was defined in [Exercise 12.1](#). It is this signal that is sampled in time to produce the complex data. Some salient points about this signal should be noted.

1. Because the real bandpass signal has been shifted down to  $F = 0$ , the magnitude of the Fourier transform of the complex envelope need not be symmetric about  $F = 0$ , nor the phase of the Fourier transform be antisymmetric. This can be easily seen by visualizing the spectrum shown in [Figure 12.1](#) after it has been shifted to the left by  $F_0$  and lowpass filtered. Thus, it is imperative that any plot should be of the spectrum over the entire interval  $-1/2 \leq f \leq 1/2$ , where  $f = F/F_s$  is the discrete-time frequency. A similar comment applies to the noise PSD as will be described in the next section.
2. The average power of a complex signal will be defined as

$$P_{\tilde{s}} = \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{s}[n]|^2. \quad (12.2)$$

For example, if we have a complex sinusoid given by  $\tilde{s}[n] = \tilde{A} \exp(j2\pi f_0 n)$ , then the average power is  $|\tilde{A}|^2$ . This is in contrast to a real sinusoid for which the power is  $A^2/2$ . Such factors of two will plague the poor soul that does not heed this warning!

3. In general the real signal cannot be recovered by setting the imaginary part equal to zero. Both real and imaginary parts of the signal carry important information.

## 12.3. Complex Noise

The analog bandpass signal  $s(t)$  shown in [Figure 12.3](#) is always corrupted by noise  $w(t)$ . As such, the received waveform at the input to the demodulator is more realistically modeled as  $x(t) = s(t) + w(t)$ . This results in the complex data  $\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n]$  within the digital computer. We now examine the modeling and properties of the complex noise samples  $\tilde{w}[n] = u[n] + jv[n]$ , where  $u[n]$  and  $v[n]$  are the real and imaginary parts. To do so we describe the usual approach to handling complex random variables and the extension to complex random vectors and processes. For the most part the extension to complex noise is intuitive and straightforward, but some potential pitfalls can arise if one is not careful.

### 12.3.1. Complex Random Variables

We define a complex random variable as  $\tilde{X} = U + jV$ , where  $U$  and  $V$  are real random variables and are *independent* of each other. We emphasize that these are random variables by using capital letters. Later we will revert back to lowercase letters when there is less possibility for confusion. Its mean or expected value is defined as the complex quantity  $E[\tilde{X}] = E[U] + jE[V]$ . It can be obtained by using the formulas for real random variables, and applying them to the real and imaginary parts of  $\tilde{X}$ . The second moment is defined as the real quantity  $E[|\tilde{X}|^2] = E[U^2] + E[V^2]$ , where  $|\tilde{X}|$  is the complex magnitude of  $\tilde{X}$ . This definition yields the average power of  $\tilde{X}$ , similar to the use of [\(12.2\)](#) for the average power of a complex signal. The variance of  $\tilde{X}$  is defined as

$$\text{var}(\tilde{X}) = E[|\tilde{X} - E[\tilde{X}]|^2] \quad (12.3)$$

and can be shown to be equivalent to

$$\text{var}(\tilde{X}) = E[|\tilde{X}|^2] - |E[\tilde{X}]|^2 \quad (12.4)$$

which is analogous to the real random variable case. This can be verified by

expressing all the expected values using the real and imaginary parts of  $\tilde{X}$ .

---

### Exercise 12.3 – Working with complex random variables

Derive the equivalent expression for the variance of a complex random variable given by (12.4) by expressing the variance given by (12.3) in terms of  $U$  and  $V$ , and finally converting back to expected values of  $|\tilde{X}^2|$  and  $\tilde{X}$ . Expressing the complex random variables in terms of real random variables is the preferred way to work with them algebraically. This minimizes errors and misplaced complex conjugates, which can be especially troublesome when complex random vectors must be manipulated.




---

It is usually assumed that the real and imaginary parts of  $\tilde{X}$  have a Gaussian PDF and that the PDF is the same for both  $U$  and  $V$  (and recall that  $U$  and  $V$  are independent). Hence, the PDF, which is actually a *two-dimensional PDF* of the real random variables  $U$  and  $V$ , is given by

$$p_{U,V}(u, v) = \frac{1}{\sqrt{2\pi(\sigma^2/2)}} \exp \left[ -\frac{1}{2(\sigma^2/2)}(u - \mu_u)^2 \right] \cdot \frac{1}{\sqrt{2\pi(\sigma^2/2)}} \exp \left[ -\frac{1}{2(\sigma^2/2)}(v - \mu_v)^2 \right] \quad (12.5)$$

where it is assumed that both  $U$  and  $V$  have variance of  $\sigma^2/2$ . The reason for this assumption on the variances is that the variance of  $\tilde{X}$  then becomes  $\sigma^2$ , consistent with the notation for a real Gaussian random variable. Note that the joint PDF of (12.5) can also be written using complex notation as

$$p_{\tilde{X}}(\tilde{x}) = \frac{1}{\pi\sigma^2} \exp \left[ -\frac{1}{\sigma^2} |\tilde{x} - \tilde{\mu}|^2 \right] \quad (12.6)$$

where  $\tilde{x} = u + jv$  and  $\tilde{\mu} = \mu_u + j\mu_v$ . This is the complex Gaussian PDF for a complex Gaussian random variable with complex mean  $\tilde{\mu}$  and variance  $\sigma^2$ , and is denoted by the shorthand notation as  $\mathcal{CN}(\tilde{\mu}, \sigma^2)$ . Note that  $p_{\tilde{X}}(\tilde{x})$  is still a two-dimensional PDF of two real random variables and retains all the usual properties. In computer simulations one can generate an outcome of a complex Gaussian random variable  $\tilde{X}$  by using

```
u=sqrt(sig2/2)*randn(1,1)+mu_u;
```

```

v=sqrt(sigz/z)*randn(1,1)+mu_v;
x=u+j*v;

```

where  $j$  is  $\sqrt{-1}$ .

### 12.3.2. Complex Random Vectors

Consider a  $2 \times 1$  complex random vector given as  $[\tilde{X} \ \tilde{Y}]^T$ . The covariance between the complex random variables is defined as

$$\text{cov}(\tilde{X}, \tilde{Y}) = E_{\tilde{X}, \tilde{Y}}[(\tilde{X} - E[\tilde{X}])^*(\tilde{Y} - E[\tilde{Y}])].$$

The subscripts on the left-most expected value is to remind us that the expectation is taken with respect to two complex random variables or four real random variables. *It is critical that one notices the placement of the complex conjugate on the first random variable.* There is no standard convention, and some authors place it on the second random variable, which will lead to errors if one is not consistent. Our choice is consistent with that in [Kay 1993, 1998]. The covariance is a complex quantity but  $\text{cov}(\tilde{X}, \tilde{X}) = \text{var}(\tilde{X})$ , which is real as required. Also, it is seen that  $\text{cov}(\tilde{Y}, \tilde{X}) = \text{cov}^*(\tilde{X}, \tilde{Y})$ . This property leads to the complex covariance matrix  $\mathbf{C}$  being *hermitian*, which means that the *conjugate-transpose* of the matrix gives back the same covariance matrix (the extension of the property that the covariance matrix of a real random vector is symmetric). Denoting the conjugate-transpose by  $H$  we have that  $\mathbf{C}^H = \mathbf{C}$ . The covariance matrix is defined in an analogous manner to the real case (see [Section 6.4.5](#)). By letting  $\tilde{\mathbf{Z}} = [\tilde{X} \ \tilde{Y}]^T$ , we have

$$\begin{aligned} \mathbf{C}_{\tilde{z}} &= E[(\tilde{\mathbf{Z}} - E[\tilde{\mathbf{Z}}])(\tilde{\mathbf{Z}} - E[\tilde{\mathbf{Z}}])^H] \\ &= E \left\{ \begin{bmatrix} \tilde{X} - E[\tilde{X}] \\ \tilde{Y} - E[\tilde{Y}] \end{bmatrix} \begin{bmatrix} (\tilde{X} - E[\tilde{X}])^* & (\tilde{Y} - E[\tilde{Y}])^* \end{bmatrix} \right\} \\ &= \begin{bmatrix} \text{var}(\tilde{X}) & \text{cov}(\tilde{X}, \tilde{Y}) \\ \text{cov}(\tilde{Y}, \tilde{X}) & \text{var}(\tilde{Y}) \end{bmatrix}^*. \end{aligned}$$

Clearly, one can extend this to the case of  $L$  complex random variables by letting  $\mathbf{Z}$  be an  $L \times 1$  random vector. Let the  $L \times 1$  complex random vector be denoted by  $\tilde{\mathbf{X}} = [\tilde{X}_1 \ \tilde{X}_2 \dots \tilde{X}_L]^T$ . Then, analogous to the PDF for a real Gaussian random vector (see [Section 6.4.5](#)), the PDF for a complex Gaussian random vector is defined as

$$p_{\tilde{\mathbf{X}}}(\tilde{\mathbf{x}}) = \frac{1}{\pi^L \det(\mathbf{C}_{\tilde{x}})} \exp \left[ -(\tilde{\mathbf{x}} - \tilde{\boldsymbol{\mu}})^H \mathbf{C}_{\tilde{x}}^{-1} (\tilde{\mathbf{x}} - \tilde{\boldsymbol{\mu}}) \right] \quad (12.7)$$

and can be shown to reduce to (12.6) for  $L = 1$ . The argument of the exponential is guaranteed to be real since a complex matrix  $\tilde{\mathbf{A}}$  that is hermitian results in a real *hermitian form*  $\tilde{\mathbf{x}}^H \tilde{\mathbf{A}} \tilde{\mathbf{x}}$ . Also, since the covariance matrix is assumed to be positive definite and hence its inverse is positive definite, the hermitian form in (12.7) is positive. See [Kay 1993] for some further necessary restrictions on the covariance matrix of a complex Gaussian random vector.

### 12.3.3. Complex Random Processes

For most practical algorithm derivations and subsequent computer simulations, it is usually assumed that the random variables are all Gaussian. We will assume this to be the case. Also, to orient our discussion toward noise modeling, all complex random variables will be assumed to have zero mean. We will denote the random process by  $\tilde{x}[n]$  for  $-\infty < n < \infty$ , reverting back to lowercase notation. In the complex case, where the noise is assumed to be stationary, we can define an autocorrelation sequence (ACS) and a power spectral density (PSD). The ACS is defined as

$$r_{\tilde{x}}[k] = E[\tilde{x}^*[n]\tilde{x}[n+k]]$$

where again the placement of the complex conjugate should be noted. The PSD is once again the Fourier transform of the ACS

$$P_{\tilde{x}}(f) = \sum_{k=-\infty}^{\infty} r_{\tilde{x}}[k] \exp(-j2\pi fk) \quad -1/2 \leq f \leq 1/2.$$

Since the complex ACS is no longer symmetric but can be shown to have the hermitian property  $r_{\tilde{x}}[-k] = r_{\tilde{x}}^*[k]$ , the PSD need not be symmetric about  $f = 0$ . It, of course, must be real.

Two common models for complex Gaussian noise is *complex white Gaussian noise* (CWGN) and *complex colored Gaussian noise*. The former has a flat PSD and the latter can have any shape. Complex white Gaussian noise is defined as

$$\tilde{x}[n] = u[n] + jv[n]$$

where  $u[n]$  and  $v[n]$  are both *real white Gaussian noise* processes, independent of each other, and each one has a variance of  $\sigma^2/2$ . The overall average power  $E[|\tilde{x}[n]|^2]$  is given by  $\sigma^2$  since  $E[u^2[n]] = E[v^2[n]] = \sigma^2/2$ . The MATLAB program `cwgn.m` can be used to generate a segment of a CWGN realization.

### Exercise 12.4 – Estimating the ACS

Since the PSD for CWGN is  $P_{\tilde{x}}(f) = \sigma^2$ , the ACS, obtained by taking the inverse Fourier transform, is  $r_{\tilde{x}}[k] = \sigma^2$  for  $k = 0$  and  $r_{\tilde{x}}[k] = 0$  for  $k \neq 0$ . This is an “impulsive” sequence. Generate a realization of  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $N = 1000$  and  $\sigma^2 = 1$ . Then, estimate the ACS using the estimator (see also [Section 6.4.4](#) for the ACS estimator for a real random process)

$$\hat{r}_{\tilde{x}}[k] = \frac{1}{N} \sum_{n=0}^{N-1-k} \tilde{x}^*[n] \tilde{x}[n+k]$$

for  $k = 0, 1, \dots, 9$ . Is the estimate close to an impulse? Does the estimated value of  $\hat{r}_{\tilde{x}}[0]$  agree with the theoretical value of  $\sigma^2 = 1$ ?

A flexible model for complex colored Gaussian noise  $\tilde{x}[n]$  is the autoregressive (AR) model. The analogous model for real colored Gaussian noise was described in [Section 4.4](#). The simplest case is when  $p = 1$ , for which the definition is

$$\tilde{x}[n] = -a[1]\tilde{x}[n-1] + \tilde{u}[n]$$

where the AR filter parameter  $a[1]$  can now be *complex* and the excitation noise  $\tilde{u}[n]$  is CWGN with variance  $\sigma_{\tilde{u}}^2$ . The ACS and PSD can be shown to be

$$r_{\tilde{x}}[k] = \begin{cases} \frac{\sigma_{\tilde{u}}^2}{1-|a[1]|^2} (-a[1])^k & k \geq 0 \\ r_{\tilde{x}}^*[-k] & k < 0 \end{cases} \quad (12.8)$$

and

$$P_{\tilde{x}}(f) = \frac{\sigma_{\tilde{u}}^2}{|1 + a[1] \exp(-j2\pi f)|^2} \quad (12.9)$$

respectively.



### Exercise 12.5 – Nonsymmetric nature of PSD

For  $a[1] = -r \exp(j2\pi f_0)$  plot the AR PSD for  $f_0 = 0.2$ ,  $r = 0.8$ , and also for  $f_0 = -0.2$ ,  $r = 0.8$ . Assume  $\sigma_{\tilde{u}}^2 = 1$ . Be sure to plot it over the frequency

interval  $-1/2 \leq f \leq 1/2$ . Hint: You will need a computer to do this.

---

The complex AR model with  $p = 1$  can be easily extended to the more general AR model, having any number of complex filter coefficients. It is important to note that whereas a real AR model with  $p = 2$  is required to represent a bandpass resonance, the complex AR model only requires  $p = 1$ . This is because the poles of the AR process need not occur in complex conjugate pairs due to the complex AR filter parameters. A MATLAB subprogram `ARgendata_complex.m`, which is contained on the CD, can be used to generate a segment of an AR realization.

---

### Exercise 12.6 – Calculating the signal-to-noise ratio (SNR)

Consider a noise corrupted complex sinusoid given by

$\tilde{x}[n] = \tilde{A} \exp(j2\pi(0.1)n) + \tilde{w}[n]$ , where  $\tilde{w}[n]$  is a realization of a complex AR process of order  $p = 1$ . The SNR is defined as

$$\text{SNR} = 10 \log_{10} \frac{|\tilde{A}|^2}{r_{\tilde{w}}[0]} \quad \text{dB.}$$

For  $\tilde{A} = 2 \exp(j\pi/8)$ ,  $a[1] = -0.9 \exp(j2\pi(0.3))$ ,  $\sigma_{\tilde{u}}^2 = 1$ , find the SNR.

Then, generate  $\tilde{s} = \tilde{A} \exp(j2\pi(0.1)n)$  and also a realization of  $\tilde{w}[n]$  for  $N = 1000$ , using the MATLAB subprogram `ARgendata_complex.m` for the latter. Finally, calculate the average powers from your generated data by using  $(1/N) \sum_{n=0}^{N-1} |\tilde{s}[n]|^2$  for the signal and  $(1/N) \sum_{n=0}^{N-1} |\tilde{w}[n]|^2$  for the noise. What is the estimated SNR? How does it compare to the theoretical SNR?

---

A summary of the difference between modeling of real data and complex data is given in [Table 12.1](#).

**Table 12.1: Summary of Real and Complex Random Variable/Vector/Process Definitions.**

Concept	Real - Notation and definition	Complex - Notation and definition
Random variable	$X$	$\tilde{X} = U + jV$ ( $U$ and $V$ independent)
Mean	$E[X] = \int x p_X(x) dx$	$E[\tilde{X}] = \int u p_U(u) du + j \int v p_V(v) dv$
Variance	$\text{var}(X) = E[(X - E[X])^2] = \int (x - E[X])^2 p_X(x) dx$	$\text{var}(\tilde{X}) = E[ \tilde{X} - E[\tilde{X}] ^2] = \int  \tilde{x} - E[\tilde{X}] ^2 p_{U,V}(u, v) dudv$
Scalar Gaussian PDF	$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x - \mu)^2\right]$	$p_{\tilde{X}}(\tilde{x}) = \frac{1}{\pi\sigma^2} \exp\left[-\frac{1}{\sigma^2} \tilde{x} - \tilde{\mu} ^2\right]$
Covariance	$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$	$\text{cov}(X, Y) = E[(\tilde{X} - E[\tilde{X}])(\tilde{Y} - E[\tilde{Y}])]$
Random vector	$\mathbf{X} = [X_1 \ X_2 \ \dots \ X_L]^T$	$\tilde{\mathbf{X}} = [\tilde{X}_1 \ \tilde{X}_2 \ \dots \ \tilde{X}_L]^T$
Mean vector	$E[\mathbf{X}] = [E[X_1] \ E[X_2] \ \dots \ E[X_L]]^T$	$E[\tilde{\mathbf{X}}] = [E[\tilde{X}_1] \ E[\tilde{X}_2] \ \dots \ E[\tilde{X}_L]]^T$
Covariance matrix	$\mathbf{C}_x = E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^T]$ $(\mathbf{C}_x^T = \mathbf{C}_x)$	$\mathbf{C}_{\tilde{x}} = E[(\mathbf{X} - E[\mathbf{X}])(\mathbf{X} - E[\mathbf{X}])^H]$ $(\mathbf{C}_{\tilde{x}}^H = \mathbf{C}_{\tilde{x}})$
Multivariate Gaussian PDF	$p_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{L/2} \det^{1/2}(\mathbf{C}_x)} \cdot \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C}_x^{-1}(\mathbf{x} - \boldsymbol{\mu})\right]$	$p_{\tilde{\mathbf{X}}}(\tilde{\mathbf{x}}) = \frac{1}{\pi^L \det(\mathbf{C}_{\tilde{x}})} \cdot \exp\left[-(\tilde{\mathbf{x}} - \tilde{\boldsymbol{\mu}})^H \mathbf{C}_{\tilde{x}}^{-1}(\tilde{\mathbf{x}} - \tilde{\boldsymbol{\mu}})\right]$
Autocorrelation	$r_x[k] = E[x[n]x[n+k]]$	$r_{\tilde{x}}[k] = E[x^*[n]x[n+k]]$
Power spectral density	$P_x(f) = \sum_{k=-\infty}^{\infty} r_x[k] \exp(-j2\pi fk)$ $(P_x(-f) = P_x(f))$	$P_{\tilde{x}}(f) = \sum_{k=-\infty}^{\infty} r_{\tilde{x}}[k] \exp(-j2\pi fk)$
White Gaussian noise	$w[n]$	$\tilde{w}[n] = u[n] + jv[n]$ ( $u[n]$ , $v[n]$ each real WGN process and independent of each other)
Autoregressive process	$x[n] = -\sum_{k=1}^p a[k]x[n-k] + u[n]$ ( $u[n]$ is WGN)	$\tilde{x}[n] = -\sum_{k=1}^p a[k]\tilde{x}[n-k] + \tilde{u}[n]$ ( $a[k]$ 's complex and $\tilde{u}[n]$ is CWGN)

## 12.4. Complex Least Squares and the Linear Model

In [Section 3.5](#) the modeling of a real signal was discussed in detail. There the concepts of least squares and the linear model were defined and illustrated. Recall that for a real signal expressed succinctly by the  $N \times 1$  vector  $\mathbf{s}$ , the linear signal model is expressed as  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ , where  $\mathbf{H}$  is a real  $N \times p$  matrix and  $\boldsymbol{\theta}$  is a real  $p \times 1$  parameter vector. Typically, the parameters are unknown and we wish to estimate them. The least squares estimator is  $\hat{\boldsymbol{\theta}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}$ , and is obtained by minimizing the least squares error criterion

$$J(\boldsymbol{\theta}) = (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})^T (\mathbf{x} - \mathbf{H}\boldsymbol{\theta})$$

over  $\boldsymbol{\theta}$ . The minimum value of  $J(\boldsymbol{\theta})$  is

$$J(\hat{\boldsymbol{\theta}}) = \mathbf{x}^T \mathbf{x} - \mathbf{x}^T \mathbf{H} (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{x}.$$

When the real data  $\mathbf{x}$  is assumed to consist of the linear signal model  $\mathbf{s}$  embedded in WGN  $\mathbf{w}$ , then we have the real linear data model given by

$$\mathbf{x} = \mathbf{H}\boldsymbol{\theta} + \mathbf{w}.$$

The least squares estimator of  $\boldsymbol{\theta}$  then becomes an optimal estimator. Finally, we also saw that if the matrix  $\mathbf{H}$  contained unknown parameters  $\boldsymbol{\beta}$  as well, then the maximum likelihood estimator (MLE) of these extra parameters is obtained by maximizing (see [Section 5.4](#))

$$L(\boldsymbol{\beta}) = \mathbf{x}^T \mathbf{H}(\boldsymbol{\beta})(\mathbf{H}(\boldsymbol{\beta})^T \mathbf{H}(\boldsymbol{\beta}))^{-1} \mathbf{H}^T(\boldsymbol{\beta})\mathbf{x}.$$

Analogous results are available for complex data. Assume that the linear signal model is  $\tilde{\mathbf{s}} = \mathbf{H}\boldsymbol{\theta}$ , where  $\mathbf{H}$  and  $\boldsymbol{\theta}$  are now complex quantities. The analogous least squares estimator of  $\boldsymbol{\theta}$  is given by minimizing

$$J(\boldsymbol{\theta}) = (\tilde{\mathbf{x}} - \mathbf{H}\boldsymbol{\theta})^H (\tilde{\mathbf{x}} - \mathbf{H}\boldsymbol{\theta})$$

which can be shown to yield

$$\hat{\boldsymbol{\theta}} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \tilde{\mathbf{x}}.$$

When substituted in  $J(\boldsymbol{\theta})$  we have the minimum least squares error

$$J(\hat{\boldsymbol{\theta}}) = \tilde{\mathbf{x}}^H \tilde{\mathbf{x}} - \tilde{\mathbf{x}}^H \mathbf{H} (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \tilde{\mathbf{x}}.$$

It is seen that the only difference in the formulas is the replacement of the matrix transpose ("T") by the matrix complex-conjugate and transpose ("H"). For the case of unknown real or complex parameters in  $\mathbf{H}$  we estimate them by maximizing

$$L(\boldsymbol{\beta}) = \tilde{\mathbf{x}}^H \mathbf{H}(\boldsymbol{\beta})(\mathbf{H}(\boldsymbol{\beta})^H \mathbf{H}(\boldsymbol{\beta}))^{-1} \mathbf{H}^H(\boldsymbol{\beta})\tilde{\mathbf{x}}.$$

When the complex linear signal is corrupted by additive CWGN, we have the *complex linear model* given by  $\tilde{\mathbf{x}} = \mathbf{H}\boldsymbol{\theta} + \tilde{\mathbf{w}}$ . In this case the estimator can be shown to be optimal. Extensions to the case when the additive noise is correlated with a complex multivariate Gaussian PDF as listed in [Table 12.1](#) are easily obtained [[Kay 1993](#), pp. 529–531].

---

### Exercise 12.7 – Some practice with complex signal estimation

Consider the least squares estimation of the complex amplitude of a sinusoid embedded in complex noise. The signal is given by  $\tilde{s}[n] = \tilde{A} \exp(j2\pi f_0 n)$  for  $n = 0, 1, \dots, N - 1$ . First express  $\tilde{\mathbf{s}}$  as  $\mathbf{H}\boldsymbol{\theta}$  in order to identify  $\mathbf{H}$ . Then, find the least squares estimator of  $\tilde{A}$ . Does it look familiar?

---

## 12.5. Algorithm Extensions for Complex Data

We now summarize the extensions of algorithms described in [Chapters 9–11](#) for real data to the case of complex data. In order not to burden the reader with an overly detailed account, only an essential description, which includes the data assumptions and the algorithm, will be given. Much of what is known about performance is summarized in [[Kay 1988](#), [1993](#), [1998](#)], [[Van Trees 2002](#)] as well as in numerous research papers with [[Picinbono 1995](#)], [[Adali and Roy 2009](#)], as examples. In practice, because of the complexity of an analytical performance evaluation, Monte Carlo computer simulation is frequently used. The reader will have an opportunity to do this by completing the exercises.

The listing is delineated into estimation, detection, and spectral estimation and follows the same sequence as in [Chapters 9–11](#). The corresponding real data algorithm is noted in parentheses within the title of each algorithm, which can be consulted for more details. Important differences from the real algorithms are noted in the comments. Most of the algorithms are easily coded in MATLAB as simple extensions of the code already provided for real data.

### 12.5.1. Estimation with Complex Data

---

#### **Algorithm 12.1 – Deterministic signal amplitude estimation ([Algorithm 9.1](#))**

##### **1. Data model/assumptions**

$$\tilde{x}[n] = \tilde{A}\tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{A}$  is the complex amplitude to be estimated,  $\tilde{s}[n]$  is the known signal, and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ .

##### **2. Estimator**

$$\hat{\tilde{A}} = \frac{\sum_{n=0}^{N-1} \tilde{x}[n]\tilde{s}^*[n]}{\sum_{n=0}^{N-1} |\tilde{s}[n]|^2}. \quad (12.10)$$

##### **3. Comments**

This is a special case of the complex linear model.




---



---

#### **Exercise 12.8 – How to get the conjugate in the right place**

Verify that (12.10) is probably correct in its placement of the complex

conjugate by letting  $\tilde{x}[n] = \tilde{A}\tilde{s}[n]$ . Do you obtain  $\tilde{\mathbf{A}}$ ? What would happen if you had conjugated the  $\tilde{x}[n]$  term in (12.10) instead of the  $\tilde{s}[n]$  term?

---



---

### **Algorithm 12.2 – Sinusoidal amplitude and phase estimation (Algorithm 9.2)**

#### **1. Data model/assumptions**

$$\tilde{x}[n] = \tilde{A} \exp(j2\pi f_0 n) + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{\mathbf{A}}$  is the complex amplitude to be estimated. The complex magnitude of the complex amplitude estimate yields the amplitude estimate, and its phase yields the phase estimate. The frequency is assumed known with  $-1/2 < f_0 < 1/2$  and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ .

#### **2. Estimator**

$$\hat{\tilde{A}} = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f_0 n) \quad (12.11)$$

#### **3. Comments**

This is just the same problem and solution as in [Algorithm 12.1](#) for  $\tilde{s}[n] = \exp(j2\pi f_0 n)$ . Note that there are no restrictions on the frequency as in the real case (recall that in the real case we required  $2/N < f_0 < 1/2 - 2/N$ ).

---



---

### **Algorithm 12.3 – Sinusoidal amplitude, phase, and frequency estimation (Algorithm 9.3)**

#### **1. Data model/assumptions**

$$\tilde{x}[n] = \tilde{A} \exp(j2\pi f_0 n) + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{\mathbf{A}}$  and  $f_0$  are to be estimated. The complex magnitude of  $\hat{\tilde{A}}$  yields the amplitude estimate, and its phase yields the phase estimate. The frequency is assumed unknown with  $-1/2 < f_0 < 1/2$ , and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ .

## 2. Estimator

The frequency is estimated as the value that maximizes the periodogram

$$I(f_0) = \frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f_0 n) \right|^2 \quad (12.12)$$

over the frequency range  $-1/2 < f_0 < 1/2$ . Once the frequency estimate is found as  $\hat{f}_0$ , the complex amplitude estimate is given as

$$\hat{\tilde{A}} = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi \hat{f}_0 n). \quad (12.13)$$

## 3. Comments

The matrix inversion required for the real case to implement the exact MLE (see (9.6)) is not needed here.



---

### Exercise 12.9 – Sinusoidal estimation

Consider a complex sinusoid  $\tilde{s}[n] = \tilde{A} \exp(j2\pi f_0 n)$  for  $n = 0, 1, \dots, N - 1$  embedded in CWGN with variance  $\sigma^2$ . Then, if the sinusoidal parameters are  $\tilde{A} = -1 + j$ ,  $f_0 = -0.2$  and the noise variance is  $\sigma^2 = 2$ , generate  $N = 100$  data points. Finally, estimate the sinusoid's amplitude, phase, and frequency. Hint: Use the `atan2` subprogram in MATLAB to find the four-quadrant phase.



---

### Algorithm 12.4 – Estimation of time delay ([Algorithm 9.4](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n - n_0] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is the known signal,  $n_0$  is the delay to be estimated, and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ . It is assumed that  $\tilde{s}[n]$  is  $M$  samples in length, where  $n = 0, 1, \dots, M - 1$  and that for all possible values of  $n_0$ , the delayed signal  $\tilde{s}[n - n_0]$  is contained in the observation interval  $0 \leq n \leq N - 1$ .

## 2. Estimator

The estimate is chosen as the value of  $n_0$  that maximizes

$$J(n_0) = \operatorname{Re} \left( \sum_{n=n_0}^{n_0+M-1} \tilde{x}[n] \tilde{s}^*[n - n_0] \right) \quad 0 \leq n_0 \leq N - M. \quad (12.14)$$

## 3. Comments

When  $n_0$  is the true value in (12.14), the value due to the signal should be the energy  $\mathcal{E} = \sum_{n=0}^{M-1} |\tilde{s}[n]|^2$ . The imaginary part of  $J(n_0)$  is therefore due to noise only and so is omitted.

■

## **Algorithm 12.5 – Bearing estimation ([Algorithm 9.5](#))**

### 1. Data model/assumptions

As in the real case, a uniform line array is assumed (refer back to [Algorithm 9.5](#) for the problem description and symbol definitions). The only difference now is that the arrival angle, i.e., bearing,  $\beta$  to be estimated can be in the interval  $0 < \beta < \pi$ . The data model is

$$\tilde{x}[n] = \tilde{A} \exp [j2\pi(F_0(d/c) \cos \beta)] + \tilde{w}[n] \quad n = 0, 1, \dots, M - 1$$

where the complex amplitude  $\tilde{A}$  is unknown, and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ . The parameters  $F_0$ ,  $d$ , and  $c$  are assumed known.

## 2. Estimator

The estimate of the arrival angle is the value of  $\beta$  over the interval  $0 < \beta < \pi$  that maximizes

$$I_s(\beta) = \frac{1}{M} \left| \sum_{n=0}^{M-1} \tilde{x}[n] \exp \left[ -j2\pi \left( F_0 \frac{d}{c} \cos \beta \right) n \right] \right|^2. \quad (12.15)$$

## 3. Comments

In the real case we had to assume that  $\beta$  was not close to 0 or  $\pi/2$ . For the complex case there are no restrictions on  $\beta$  being close to its interval endpoints.

■

## Algorithm 12.6 – Estimation of power of CWGN ([Algorithm 9.6](#))

### 1. Data model/assumptions

The data model is  $\tilde{x}[n] = \tilde{w}[n]$  for  $n = 0, 1, \dots, N - 1$ , where  $\tilde{w}[n]$  is CWGN with variance, i.e., power, of  $\sigma^2$ .

### 2. Estimator

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2. \quad (12.16)$$

### 3. Comments

In the estimator for the complex case we are averaging  $2N$  real samples, as opposed to  $N$  for the real case. Thus, comparison of performance of a real parameter estimator to a complex parameter estimator must be done very carefully.



---

---

## Exercise 12.10 – Expected value of estimator

For the estimator given by (12.16) find the expected value. Is the estimator unbiased? Hint: Recall the definition of the variance for a complex random variable.



---

---

## Algorithm 12.7 – Random signal power estimation ([Algorithm 9.7](#))

### 1. Data model/assumptions

It is assumed that the signal  $\tilde{s}[n]$  is a zero mean Gaussian random process, and has the PSD

$$P_{\tilde{s}}(f) = P_0 Q(f) \quad -1/2 \leq f \leq 1/2$$

where  $P_0$  is real and  $P_0 > 0$ ,  $Q(f)$  is a real positive function (not necessarily symmetric about  $f = 0$ ) with  $\int_{-\frac{1}{2}}^{\frac{1}{2}} Q(f) df = 1$  so that  $P_0$  is the total power in the signal process.

### 2. Estimator

The estimator is

$$\hat{P}_0 = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{I(f)}{Q(f)} df \quad (12.17)$$

where

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{s}[n] \exp(-j2\pi f n) \right|^2$$

is the periodogram of the signal process.

### 3. Comments

None



## **Algorithm 12.8 – Random signal center frequency estimation (Algorithm 9.8)**

### 1. Data model/assumptions

It is assumed that the zero mean Gaussian random process  $\tilde{x}[n]$  has the PSD

$$P_{\tilde{x}}(f) = P_{\tilde{s}}(f - f_0) + \sigma^2 \quad -1/2 \leq f \leq 1/2$$

where  $P_{\tilde{s}}(f)$  is a lowpass signal PSD not necessarily symmetric about  $f = 0$ . The lowpass PSD is assumed to be known, as is the CWGN variance  $\sigma^2$ . The center frequency to be estimated is such that the entire content of  $P_{\tilde{s}}(f - f_0)$  lies within the  $[-1/2, 1/2]$  frequency band.

### 2. Estimator

The estimator is the value of  $f_0$  that minimizes

$$J(f_0) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{I(f)}{P_{\tilde{s}}(f - f_0) + \sigma^2} df \quad (12.18)$$

where  $I(f)$  is the periodogram given by

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f n) \right|^2.$$

### 3. Comments

None



### **Algorithm 12.9 – Multiple sinusoids in CWGN - optimal solution (Algorithm 9.9)**

#### **1. Data model/assumptions**

$$\tilde{x}[n] = \sum_{i=1}^p \tilde{A}_i \exp(j2\pi f_i n) + \tilde{w}[n] \quad n = 0, 1, \dots, N-1$$

where the complex amplitudes  $\tilde{A}_i$ , and frequencies  $f_i$  are to be estimated, and  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ .

#### **2. Estimator**

The frequencies are first estimated by maximizing the function

$$J(\mathbf{f}) = \tilde{\mathbf{x}}^H \mathbf{H}(\mathbf{f}) (\mathbf{H}^H(\mathbf{f}) \mathbf{H}(\mathbf{f}))^{-1} \mathbf{H}^H(\mathbf{f}) \tilde{\mathbf{x}} \quad (12.19)$$

where  $\mathbf{f} = [f_1 \ f_2 \ \dots \ f_p]^T$ , over the region  $-1/2 < f_1 < f_2 < \dots < f_p < 1/2$ . The  $N \times p$  matrix  $\mathbf{H}(\mathbf{f})$  is defined as

$$\mathbf{H}(\mathbf{f}) = [\mathbf{e}_1 \ \mathbf{e}_2 \ \dots \ \mathbf{e}_p]$$

where

$$\mathbf{e}_i = \begin{bmatrix} 1 \\ \exp(j2\pi f_i) \\ \vdots \\ \exp[j2\pi f_i(N-1)] \end{bmatrix}$$

for  $i = 1, 2, \dots, p$ . Once the estimates of the frequencies, denoted by  $\hat{\mathbf{f}}$ , have been found, the complex amplitudes are estimated by

$$\hat{\mathbf{A}} = \left( \mathbf{H}^H(\hat{\mathbf{f}}) \mathbf{H}(\hat{\mathbf{f}}) \right)^{-1} \mathbf{H}^H(\hat{\mathbf{f}}) \tilde{\mathbf{x}}.$$

The amplitude estimates are obtained by taking the complex magnitudes of  $\hat{\mathbf{A}}$  and similarly the phase estimates are obtained by taking the phases of  $\hat{\mathbf{A}}$ .

#### **3. Comments**

For  $p = 1$  we have the same problem and estimator as in [Algorithm 12.3](#). The objective function given by (12.19) is real and nonnegative since the matrix  $\mathbf{H}(\mathbf{f})(\mathbf{H}^H(\mathbf{f})\mathbf{H}(\mathbf{f}))^{-1}\mathbf{H}^H(\mathbf{f})$  is hermitian and positive semidefinite (actually it is a projection matrix).

---



### **Algorithm 12.10 – Multiple sinusoids in WGN - suboptimal solution ([Algorithm 9.10](#))**

#### **1. Data model/assumptions**

Same as [Algorithm 12.9](#).

#### **2. Estimator**

Compute the forward  $\mathbf{H}_f$  and backward  $\mathbf{H}_b$  observation matrices, both of dimension  $(N - L) \times L$ , where  $L$  is the largest integer less than or equal to  $(3/4)N$ , and

$$\mathbf{H}_f = \begin{bmatrix} \tilde{x}[L-1] & \tilde{x}[L-2] & \dots & \tilde{x}[0] \\ \tilde{x}[L] & \tilde{x}[L-1] & \dots & \tilde{x}[1] \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}[N-2] & \tilde{x}[N-3] & \dots & \tilde{x}[N-L-1] \end{bmatrix}$$

$$\mathbf{H}_b = \begin{bmatrix} \tilde{x}[1] & \tilde{x}[2] & \dots & \tilde{x}[L] \\ \tilde{x}[2] & \tilde{x}[3] & \dots & \tilde{x}[L+1] \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}[N-L] & \tilde{x}[N-L+1] & \dots & \tilde{x}[N-1] \end{bmatrix}^*$$

and the forward  $\mathbf{h}_f$  and backward  $\mathbf{h}_b$  data vectors, both of dimension  $(N - L) \times 1$  are

$$\mathbf{h}_f = \begin{bmatrix} \tilde{x}[L] \\ \tilde{x}[L+1] \\ \vdots \\ \tilde{x}[N-1] \end{bmatrix} \quad \mathbf{h}_b = \begin{bmatrix} \tilde{x}[0] \\ \tilde{x}[1] \\ \vdots \\ \tilde{x}[N-L-1] \end{bmatrix}^*.$$

Next form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_f \\ \mathbf{H}_b \end{bmatrix}$$

which has dimension  $2(N - L) \times L$  and find the eigenvectors and eigenvalues of the  $\mathbf{H}^H \mathbf{H}$ . Arrange the eigenvalues (which are real and positive) in descending order as  $\lambda_1 > \lambda_2 > \dots > \lambda_p > \lambda_{p+1} > \dots > \lambda_L$ . Retain the  $p$  eigenvectors with eigenvalues  $\lambda_1, \dots, \lambda_p$ , which is equivalent to finding the pseudoinverse of the matrix  $\mathbf{H}$ , assumed to have a rank of  $p$ . Call this  $\mathbf{H}^\#$ . Then, the principal component AR filter parameter solution of length  $L \times 1$  is

$$\hat{\mathbf{a}} = -\mathbf{H}^\# \mathbf{h}$$

where  $\hat{\mathbf{a}} = [\hat{a}[1] \hat{a}[2] \dots \hat{a}[L]]^T$ . Finally, solve for the roots of the polynomial

$$\hat{\mathcal{A}}(z) = 1 + \hat{a}[1]z^{-1} + \hat{a}[2]z^{-2} + \dots + \hat{a}[L]z^{-L}$$

and choose the  $p$  roots that are in the z-plane and are closest to the unit circle, i.e., whose magnitudes are closest to one. Calling these  $z_1, z_2, \dots, z_p$ , the final frequency estimates are

$$\hat{f}_i = \frac{1}{2\pi} \angle z_i$$

for  $i = 1, 2, \dots, p$ . The amplitude and phase estimates are obtained as in [Algorithm 12.9](#) once the frequency estimates are found.

### 3. Comments

The MATLAB subprogram `FSSP3alg9_10_sub.m` can be used to implement this algorithm once the complex conjugates are added and the matrix transposes are replaced with matrix hermitians. Also, only the  $p$  principal eigenvectors are retained, as opposed to  $2p$  principal eigenvectors in the real data case.



### **Algorithm 12.11 – Estimation of random periodic signal in noise ([Algorithm 9.11](#))**

This algorithm has not been extended to the complex data case. Conceivably, however, the estimator will be an average over all the periods as for the real case. However, the period estimate may need

refinement.



### Algorithm 12.12 – Estimation of random signal in noise ([Algorithm 9.12](#))

#### 1. Data model/assumptions

The random signal  $\tilde{s}[n]$  to be extracted is embedded in CWGN with known variance  $\sigma^2$  so that we observe

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1.$$

The signal is assumed to be the outcome of a zero mean complex random process with a known  $N \times N$  covariance matrix  $\mathbf{C}_{\tilde{s}}$ .

#### 2. Estimator

The estimator is given as

$$\hat{\tilde{s}} = \mathbf{C}_{\tilde{s}}(\mathbf{C}_{\tilde{s}} + \sigma^2 \mathbf{I})^{-1} \tilde{x} \quad (12.20)$$

where  $\hat{\tilde{s}} = [\hat{\tilde{s}}[0] \ \hat{\tilde{s}}[1] \ \dots \ \hat{\tilde{s}}[N - 1]]^T$ . If the signal is from a stationary random process, then the covariance matrix becomes an autocorrelation matrix  $\mathbf{R}_{\tilde{s}}$  given by

$$\mathbf{C}_{\tilde{s}} = \mathbf{R}_{\tilde{s}} = \begin{bmatrix} r_{\tilde{s}}[0] & r_{\tilde{s}}^*[1] & \dots & r_{\tilde{s}}^*[N - 1] \\ r_{\tilde{s}}[1] & r_{\tilde{s}}[0] & \dots & r_{\tilde{s}}^*[N - 2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{\tilde{s}}[N - 1] & r_{\tilde{s}}[N - 2] & \dots & r_{\tilde{s}}[0] \end{bmatrix} \quad (12.21)$$

where  $r_{\tilde{s}}[k]$  is the autocorrelation sequence for  $\tilde{s}[n]$ . If  $\tilde{s}[n]$  is stationary with PSD given by  $P_{\tilde{s}}(f)$ , then an approximate estimate (which avoids the inversion of the  $N \times N$  autocorrelation matrix) is to filter the data  $\tilde{x}[n]$  with the noncausal filter with frequency response

$$H(f) = \frac{P_{\tilde{s}}(f)}{P_{\tilde{s}}(f) + \sigma^2}. \quad (12.22)$$

#### 3. Comments

The autocorrelation matrix given in (12.21) utilizes the property that  $r_{\tilde{s}}[-k] = r_{\tilde{s}}^*[k]$  to yield the hermitian equivalent of the real autocorrelation matrix for which  $r_s[-k] = r_s[k]$ . The MATLAB

subprogram FSSP3alg9\_12\_sub.m can be used to implement this algorithm by replacing the real autocorrelation matrix by its complex version.

---

### **Algorithm 12.13 – Estimation of a signal in noise and interference (Algorithm 9.13)**

#### **1. Data model/assumptions**

For this algorithm it is assumed that there is an additional data source that is correlated with the interference that we wish to mitigate. The data is

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] + \tilde{i}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is the signal of interest,  $\tilde{w}[n]$  is observation noise, and  $\tilde{i}[n]$  is the interference. Additionally, we have access to data  $\tilde{x}_R[n]$  for  $n = 0, 1, \dots, N - 1$ , sometimes called *reference data*, which is correlated with  $\tilde{i}[n]$ , and hopefully, uncorrelated with  $\tilde{s}[n]$ .

#### **2. Estimator**

The interference is estimated by  $\hat{\tilde{i}}[n]$ , which is given by the output of a complex FIR filter with the reference data at the input as

$$\hat{\tilde{i}}[n] = \sum_{l=0}^{p-1} h[l] \tilde{x}_R[n-l] \quad n = p - 1, p, \dots, N - 1.$$

The complex FIR filter coefficients  $\mathbf{h} = [h[0] \ h[1] \ \dots \ h[p - 1]]^T$  are found as

$$\mathbf{h} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \tilde{\mathbf{x}}$$

where  $\tilde{\mathbf{x}} = [\tilde{x}[p - 1] \ \tilde{x}[p] \ \dots \ \tilde{x}[N - 1]]^T$  and  $\mathbf{H}$  is the  $(N - p + 1) \times p$  matrix

$$\mathbf{H} = \begin{bmatrix} \tilde{x}_R[p - 1] & \tilde{x}_R[p - 2] & \dots & \tilde{x}_R[0] \\ \tilde{x}_R[p] & \tilde{x}_R[p - 1] & \dots & \tilde{x}_R[1] \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{x}_R[N - 1] & \tilde{x}_R[N - 2] & \dots & \tilde{x}_R[N - p] \end{bmatrix}.$$

The estimate of the signal becomes  $\hat{\tilde{s}}[n] = \tilde{x}[n] - \hat{\tilde{i}}[n]$  for  $n = p - 1,$

$p, \dots, N - 1$ .

### 3. Comments

The MATLAB subprogram `FSSP3alg9_13_sub.m` can be used to implement this algorithm once the complex conjugates are added and the matrix transposes are replaced with matrix hermitians.



---

## 12.5.2. Detection with Complex Data

---

### Algorithm 12.14 – Replica-correlator (matched filter) ([Algorithm 10.1](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is the known signal, and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

#### 2. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \operatorname{Re} \left( \sum_{n=0}^{N-1} \tilde{x}[n] \tilde{s}^*[n] \right) > \gamma = \sqrt{(\sigma^2/2)\mathcal{E}} Q^{-1}(P_{FA}) \quad (12.23)$$

where  $\mathcal{E} = \sum_{n=0}^{N-1} |\tilde{s}[n]|^2$  is the signal energy,  $P_{FA}$  is the given probability of false alarm, and  $Q^{-1}(\cdot)$  is the inverse Q function (see [Section 4.3](#)).

#### 3. Comments

The MATLAB subprogram `FSSP3alg10_1_sub.m` can be used to implement this algorithm once the appropriate modifications are made. Note that the threshold is slightly different as well.



---

### Algorithm 12.15 – Limiter-replica-correlator ([Algorithm 10.2](#))

This algorithm has not been extended to the complex data case. Conceivably it might just require limiters for the real and imaginary parts

of  $\tilde{x}[n]$ .

---

---



### Algorithm 12.16 – Generalized matched filter ([Algorithm 10.3](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is the known signal, and  $\tilde{w}[n]$  is correlated Gaussian noise with zero mean and known covariance matrix  $\mathbf{C}$ .

#### 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \operatorname{Re} (\tilde{\mathbf{s}}^H \mathbf{C}^{-1} \tilde{\mathbf{x}}) > \gamma = \sqrt{\tilde{\mathbf{s}}^H \mathbf{C}^{-1} \tilde{\mathbf{s}} / 2} Q^{-1}(P_{FA}) \quad (12.24)$$

where  $\mathbf{s} = [\tilde{s}[0] \ \tilde{s}[1] \ \dots \ \tilde{s}[N - 1]]^T$  and  $\mathbf{C}$  is the  $N \times N$  covariance matrix of the noise samples  $\tilde{\mathbf{w}} = [\tilde{w}[0] \ \tilde{w}[1] \ \dots \ \tilde{w}[N - 1]]^T$ .

#### 3. Comments

For  $\mathbf{C} = \sigma^2 \mathbf{I}$ , this reduces to [Algorithm 12.14](#). The MATLAB subprogram FSSP3alg10\_3\_sub.m can be used to implement this algorithm once the appropriate modifications are made. Note that the threshold is slightly different as well.

---

---



### Algorithm 12.17 – Estimator-correlator ([Algorithm 10.4](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is a complex Gaussian random signal with zero mean and known  $N \times N$  covariance matrix  $\mathbf{C}_{\tilde{s}}$  and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

#### 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^H \mathbf{C}_{\tilde{s}} (\mathbf{C}_{\tilde{s}} + \sigma^2 \mathbf{I})^{-1} \tilde{\mathbf{x}} > \gamma \quad (12.25)$$

where  $\tilde{\mathbf{x}} = [\tilde{x}[0] \ \tilde{x}[1] \dots \tilde{x}[N-1]]^T$ .

### 3. Comments

Note that the matrix  $\mathbf{C}_{\tilde{s}}(\mathbf{C}_{\tilde{s}} + \sigma^2 \mathbf{I})^{-1}$  is hermitian and positive definite. Hence, the hermitian form  $T(\tilde{\mathbf{x}})$  is theoretically a real and positive quantity. The MATLAB subprogram FSSP3alg10\_4\_sub.m can be used to implement this algorithm by replacing the real autocorrelation matrix by its complex version.

---



---

## **Algorithm 12.18 – Energy detector ([Algorithm 10.5](#))**

### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N-1$$

where  $\tilde{s}[n]$  is a completely unknown complex signal and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

### 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \sum_{n=0}^{N-1} |\tilde{x}[n]|^2 > \gamma = (\sigma^2/2) Q_{\chi_{2N}^2}^{-1}(P_{FA}) \quad (12.26)$$

where  $Q_{\chi_{2N}^2}^{-1}(\cdot)$  is the inverse chi-squared cumulative distribution function with  $2N$  degrees of freedom.

### 3. Comments

Note that the degrees of freedom  $2N$  is different from the real case since we are summing the squares of  $2N$  real random variables. Each real random variable has a variance of  $\sigma^2/2$ .

---



---

## **Algorithm 12.19 – Unknown complex amplitude ([Algorithm 10.6](#))**

### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{A}\tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N-1$$

where  $\tilde{A}\tilde{s}[n]$  is the signal with  $\tilde{s}[n]$  known,  $\tilde{A}$  is an unknown complex amplitude, and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

## 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \frac{|\sum_{n=0}^{N-1} \tilde{x}[n] \tilde{s}^*[n]|^2}{(\sigma^2/2) \sum_{n=0}^{N-1} |\tilde{s}[n]|^2} > \gamma = 2 \ln \left( \frac{1}{P_{FA}} \right). \quad (12.27)$$

## 3. Comments

The MATLAB subprogram FSSP3alg10\_6\_sub.m can be used to implement this algorithm once the appropriate modifications are made. Note that the threshold is slightly different as well.

---



---

### **Algorithm 12.20 – Unknown complex amplitude of sinusoidal signal (Algorithm 10.7)**

## 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{A} \exp(j2\pi f_0 n) + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{A}$  is an unknown complex amplitude, the frequency  $f_0$  is known, and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

## 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \frac{\frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f_0 n) \right|^2}{(\sigma^2/2)} > \gamma = 2 \ln \left( \frac{1}{P_{FA}} \right). \quad (12.28)$$

## 3. Comments

This algorithm is a special case of [Algorithm 12.19](#) with  $\tilde{s}[n] = \exp(j2\pi f_0 n)$ . The MATLAB subprogram

FSSP3alg10\_7\_sub.m can be used to implement this algorithm once the appropriate modifications are made. Note that the threshold is slightly different as well.

---



---

### Exercise 12.11 – Necessary amplitude for reliable detection

Consider the requirements that  $P_{FA} = 0.01$  and  $P_D \geq 0.95$  for a complex data record of length  $N = 50$ . If  $f_0 = 0.1$  and  $\sigma^2 = 1$ , determine via computer simulation, what the value of  $\tilde{A}$  should be to meet these specs. Hint: You can use the MATLAB subprogram `cwgn.m` to generate the noise. Also, the performance should not depend on the phase of  $\tilde{A}$ .

---

---

### Algorithm 12.21 – Unknown complex amplitude and frequency ([Algorithm 10.8](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{A} \exp(j2\pi f_0 n) + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{A}$  is an unknown complex amplitude,  $f_0$  is an unknown frequency, and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ .

#### 2. Detector

Decide a signal is present if

$$T(\mathbf{x}) = \frac{\max_{-1/2 < f_0 < 1/2} \frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f_0 n) \right|^2}{(\sigma^2/2)} > \gamma \quad (12.29)$$
$$= \ln \left( \frac{N - 1}{P_{FA}} \right)$$

#### 3. Comments

As in the real case, the threshold is approximate.

---

---

### Algorithm 12.22 – Unknown arrival time ([Algorithm 10.9](#))

#### 1. Data model/assumptions

$$\tilde{x}[n] = \tilde{s}[n - n_0] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is a known signal, having samples over the interval  $0 \leq n \leq M - 1$ , and  $\tilde{w}[n]$  is CWGN with known variance  $\sigma^2$ . The unknown delay time  $n_0$

is assumed to take on the possible values  $0 \leq n_0 \leq N - M$ .

## 2. Detector

Decide a signal is present if

$$T(\tilde{\mathbf{x}}) = \max_{0 \leq n_0 \leq N-M} \operatorname{Re} \left( \sum_{n=n_0}^{n_0+M-1} \tilde{x}[n] \tilde{s}^*[n-n_0] \right) > \gamma. \quad (12.30)$$

The threshold cannot be obtained in closed form.

## 3. Comments

The MATLAB subprogram `FSSP3alg10_9_sub.m` can be used to implement this algorithm once the appropriate modifications are made.



### 12.5.3. Spectral Estimation with Complex Data

#### Algorithm 12.23 – Averaged periodogram ([Algorithm 11.1](#))

##### 1. Data model/assumptions

The data  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually wide sense stationary (WSS)) and need not be Gaussian.

##### 2. Spectral estimator

Divide the data record into  $I$  successive nonoverlapping blocks of length  $L$  samples each, where it is assumed that  $N = IL$ . The data blocks are given by

$$\tilde{y}_i[n] = \tilde{x}[n + iL] \quad n = 0, 1, \dots, L - 1; i = 0, 1, \dots, I - 1.$$

Then, for each block compute the periodogram as

$$\hat{P}_{\tilde{x}}^{(i)}(f) = \frac{1}{L} \left| \sum_{n=0}^{L-1} \tilde{y}_i[n] \exp(-j2\pi fn) \right|^2 \quad (12.31)$$

and then average all the periodograms together to yield the spectral estimator as

$$\hat{P}_{\tilde{x}}(f) = \frac{1}{I} \sum_{i=0}^{I-1} \hat{P}_{\tilde{x}}^{(i)}(f). \quad (12.32)$$

##### 3. Comments

The MATLAB subprogram `PSD_est_avper.m` can be used to implement this algorithm once the appropriate modifications are made.

---



---

### Exercise 12.12 – Estimating the PSD for CWGN

Make the necessary modifications to `PSD_est_avper.m` to be able to implement the averaged periodogram spectral estimator for complex data. Then, generate  $N = 1000$  data points of CWGN for  $\sigma^2 = 1$  using `cwgn.m`. Use your program to estimate and plot the spectral estimate over the frequency interval  $-1/2 \leq f \leq 1/2$  in dB quantities. Choose  $I = 100$  blocks of data. What is the true PSD? Is the estimate reasonably accurate? Hint: Use the axis scaling command `axis([-0.5 0.5 -10 10])` so that the vertical axis runs from  $-10$  dB to  $+10$  dB.

---



---

### Algorithm 12.24 – Minimum variance spectral estimator ([Algorithm 11.2](#))

#### 1. Data model/assumptions

The data  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

#### 2. Spectral estimator

$$\hat{P}_{\tilde{x}}(f) = \frac{p}{\mathbf{e}^H \hat{\mathbf{R}}_{\tilde{x}}^{-1} \mathbf{e}} \quad (12.33)$$

where  $\mathbf{e} = [1 \exp(j2\pi f) \dots \exp[j2\pi f(p-1)]]^T$ , and

$$[\hat{\mathbf{R}}_{\tilde{x}}]_{ij} = \frac{1}{2(N-p)} \left[ \sum_{n=p}^{N-1} \tilde{x}^*[n-i]\tilde{x}[n-j] + \sum_{n=0}^{N-1-p} \tilde{x}[n+i]\tilde{x}^*[n+j] \right]$$

for  $i = 1, 2, \dots, p; j = 1, 2, \dots, p$ .

#### 3. Comments

Although the term  $\mathbf{e}^H \hat{\mathbf{R}}_{\tilde{x}}^{-1} \mathbf{e}$  is theoretically real and positive, numerical errors will cause it to be complex. Thus, the real part should be taken. The MATLAB subprogram `FSSP3alg11_2_sub.m` can be used to

implement this algorithm once the appropriate modifications are made.




---



---

### **Algorithm 12.25 – Autoregressive spectral estimator - Covariance method ([Algorithm 11.3](#))**

#### **1. Data model/assumptions**

The data  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

#### **2. Spectral estimator**

The complex AR filter parameters are estimated by solving a set of linear equations

$$\begin{bmatrix} c_{\tilde{x}}[1, 1] & c_{\tilde{x}}[1, 2] & \dots & c_{\tilde{x}}[1, p] \\ c_{\tilde{x}}[2, 1] & c_{\tilde{x}}[2, 2] & \dots & c_{\tilde{x}}[2, p] \\ \vdots & \vdots & \ddots & \vdots \\ c_{\tilde{x}}[p, 1] & c_{\tilde{x}}[p, 2] & \dots & c_{\tilde{x}}[p, p] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} c_{\tilde{x}}[1, 0] \\ c_{\tilde{x}}[2, 0] \\ \vdots \\ c_{\tilde{x}}[p, 0] \end{bmatrix} \quad (12.34)$$

where

$$c_{\tilde{x}}[i, j] = \frac{1}{N-p} \sum_{n=p}^{N-1} \tilde{x}^*[n-i] \tilde{x}[n-j]$$

and

$$\widehat{\sigma}_{\tilde{u}}^2 = c_{\tilde{x}}[0, 0] + \sum_{i=1}^p \hat{a}[i] c_{\tilde{x}}[0, i].$$

The spectral estimator is given as

$$\hat{P}_{\tilde{x}}(f) = \frac{\widehat{\sigma}_{\tilde{u}}^2}{|1 + \hat{a}[1] \exp(-j2\pi f) + \dots + \hat{a}[p] \exp(-j2\pi fp)|^2}.$$

#### **3. Comments**

The MATLAB subprogram FSSP3alg11\_3\_sub.m can be used to implement this algorithm once the appropriate modifications are made.



## **Algorithm 12.26 – Autoregressive spectral estimator - Burg method**

[\*\*\(Algorithm 11.4\)\*\*](#)

### **1. Data model/assumptions**

The data  $\tilde{x}[n]$  for  $n = 0, 1, \dots, N - 1$  is assumed to be zero mean and stationary (actually WSS) and need not be Gaussian.

### **2. Spectral estimator**

See [[Kay 1988](#), pp. 228–231] for the details.

### **3. Comments**

None



---

## **12.6. Other Extensions**

In practice, there are numerous situations in which data is obtained from other types of information sources. For example, in radar it is customary to sample the output of  $M$  multiple sensors in time. This produces multiple temporal waveforms, with one data set for each sensor that is sampled. If we view the output at time  $t$  of all the sensors, then the “data point” at time  $t$  is  $\mathbf{x}(t) = [x_1(t) \ x_2(t) \ \dots \ x_M(t)]^T$ , when written as a vector. This data point is sometimes called a *snapshot*, likening it to a camera snapshot in time. In this case, it is more appropriate to consider the data as a string of vectors, one for each time sample. This type of data is termed *multichannel* and its processing requires extensions of the concepts and algorithms used for a single temporal data set. Fortunately, the extensions are not overly complicated but do require some care in their description and implementation. Familiarity with some advanced matrix algebra is often essential. A second type of data that is of practical importance is encountered in image processing problems, as an example. The data consists of the two-dimensional spatial samples of an analog image, producing a set of pixel values. These pixel values are usually arranged in an array. To effectively utilize the information provided by the image it is essential that the ordering in the plane of the pixel values be maintained. Thus, the data will consist of samples of a *two-dimensional* analog signal. This type of data is termed *multidimensional*, in general, with the predominant application to two-dimensions. For either multichannel (also called *multivariate* in the statistics community) or multidimensional data many, but not all, of the algorithms described in [Chapters 9–11](#) have been extended. However, their descriptions are involved and are

beyond the scope of this text. The interested reader should consult [[Kay 1998, Chapter 13](#)] for multichannel detection, [[Kay 1988, Chapter 14](#)] for multichannel spectral estimation, [[Kay 1988, Chapter 15](#)] for two-dimensional spectral estimation. For an overview of multichannel signal processing, sometimes called *array processing*, the excellent texts [[Johnson and Dudgeon 1993](#)] and [[Van Trees 2002](#)] should be consulted, and for multidimensional signal processing the excellent book [[Dudgeon and Mersereau 1984](#)] is a valuable resource.

## 12.7. Lessons Learned

- To retain all the information of a bandpass signal it is necessary to employ demodulation using both cosine and sine demodulators.
- The average signal power can be defined in many ways, but being consistent will avoid misinterpretation of SNR measures.
- To avoid errors in working with complex random variables, it is advisable to first express them in their real and imaginary parts so that the algebra for real random variables can be employed.
- The definition of joint moments of a complex random variable is not consistent throughout the literature. One should pay particular attention to the placement of the complex conjugates.
- The PSD of a complex random process need not be symmetric. As a result, a complex AR PSD model can produce a bandpass PSD using an order of  $p = 1$ .

## References

- Adali, T., A. Roy, “Circularity and Gaussianity Detection Using the Complex Generalized Gaussian Distribution”, *IEEE Signal Processing Letters*, pp. 993–996, Nov. 2009.
- Dudgeon, D.E., R.M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Johnson, D.H., D.E. Dudgeon, *Array Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory, Vol. I*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.

Kay, S., *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Picinbono, B., "Widely Linear Estimation with Complex Data", *IEEE Trans. on Signal Processing*, pp. 2030–2033, Aug. 1995.

Van Trees, H.L., *Optimum Array Processing*, J. Wiley, NY, 2002.

## Appendix 12A. Solutions to Exercises

To obtain the results described below, initialize the random number generators to `rand ('state', 0)` and `randn ('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

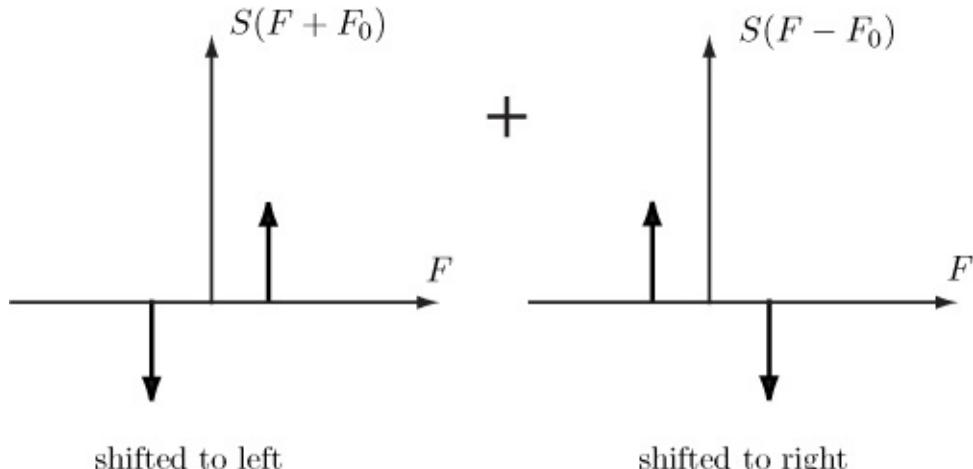
### 12.1

$$\begin{aligned} s(t) &= \cos(2\pi F_2 t) - \cos(2\pi F_1 t) = \operatorname{Re} [\exp(j2\pi F_2 t) - \exp(j2\pi F_1 t)] \\ &= 2\operatorname{Re} \left[ \underbrace{\frac{1}{2} (\exp[j2\pi(F_2 - F_0)t] - \exp[j2\pi(F_1 - F_0)t])}_{\tilde{s}(t)} \exp(j2\pi F_0 t) \right]. \end{aligned}$$

Using  $F_2 - F_0 = -(F_1 - F_0)$  we have finally

$$\tilde{s}(t) = \frac{1}{2} (\exp[j2\pi(F_2 - F_0)t] - \exp[-j2\pi(F_2 - F_0)t]) = j \sin[2\pi(F_2 - F_0)t].$$

**12.2** If we shift to the left and lowpass filter, we obtain the left most spectrum shown in [Figure 12A.1](#), while a shift to the right and lowpass filtering produces the other spectrum. Summing the two spectra together produces zero.



**Figure 12A.1: Example of cosine demodulation**

### 12.3 To derive (12.4)

$$\begin{aligned}
E[|\tilde{X} - E[\tilde{X}]|^2] &= E[|U + jV - E[U + jV]|^2] \\
&= E[|(U - E[U]) + j(V - E[V])|^2] \\
&= E[(U - E[U])^2 + (V - E[V])^2] \\
&= E[(U - E[U])^2] + E[(V - E[V])^2] \\
&= \text{var}(U) + \text{var}(V) = E[U^2] - E^2[U] + E[V^2] - E^2[V] \\
&= E[U^2 + V^2] - (E^2[U] + E^2[V]) \\
&= E[|U + jV|^2] - |E[U] + jE[V]|^2 = E[|\tilde{X}|^2] - |E[\tilde{X}]|^2.
\end{aligned}$$

Note that we have converted the complex random variables to real random variables, used the variance properties of real random variables, then reexpressed the result in terms of the complex random variables.

### 12.4 The theoretical ACS is given as the inverse Fourier transform of

$P_{\tilde{x}}(f) = 1$ , which is  $r_{\tilde{x}}[k] = 1$  for  $k = 0$  and  $r_{\tilde{x}}[k] = 0$  for  $k \geq 1$ . You should observe that the ACS estimate is very close to being zero for  $k = 1, 2, \dots, 9$ , and for  $k = 0$  it is close to the theoretical value of one. You can run the program FSSP3exer12\_4.m, which is contained on the CD, to obtain the results.

### 12.5 You should observe that the PSD for $f_0 = 0.2$ has a peak at this frequency but is not symmetric about $f = 0$ . For $f_0 = -0.2$ the peak is at this negative frequency. Hence, the AR model for a bandpass process PSD only requires a single complex AR filter coefficient, as opposed to the real case in which an order of $p = 2$ is required. You can run the program FSSP3exer12\_5.m, which is contained on the CD, to obtain the results.

### 12.6 The estimated SNR is found to be $\text{SNR} = -1.0119$ dB while the true value is $-1.1919$ . To find the latter use the expression for $r_{\tilde{w}}[0]$ given in (12.8). You can run the program FSSP3exer12\_6.m, which is contained on the CD, to obtain the results.

### 12.7 The signal model can be expressed as

$$\tilde{\mathbf{s}} = \underbrace{\begin{bmatrix} 1 \\ \exp(j2\pi f_0) \\ \vdots \\ \exp[j2\pi f_0(N-1)] \end{bmatrix}}_{\mathbf{H}} \underbrace{\tilde{\mathbf{A}}}_{\boldsymbol{\theta}}.$$

Then,  $\hat{\tilde{A}} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \tilde{\mathbf{x}}$  and since  $\mathbf{H}^H \mathbf{H} = N$ , we have

$$\hat{\tilde{A}} = \frac{1}{N} \mathbf{H}^H \tilde{\mathbf{x}} = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f_0 n).$$

This is just the normalized Fourier transform at  $f = f_0$ .

**12.8** Replacing  $\tilde{x}[n]$  by  $\tilde{A}\tilde{s}[n]$  we obtain

$$\hat{\tilde{A}} = \frac{\sum_{n=0}^{N-1} \tilde{A}\tilde{s}[n]\tilde{s}^*[n]}{\sum_{n=0}^{N-1} |\tilde{s}[n]|^2} = \frac{\tilde{A} \sum_{n=0}^{N-1} \tilde{s}[n]\tilde{s}^*[n]}{\sum_{n=0}^{N-1} |\tilde{s}[n]|^2} = \tilde{A}.$$

If instead, we had conjugated the  $\tilde{x}[n]$  term, then

$$\hat{\tilde{A}} = \frac{\sum_{n=0}^{N-1} \tilde{x}^*[n]\tilde{s}[n]}{\sum_{n=0}^{N-1} |\tilde{s}[n]|^2} = \frac{\sum_{n=0}^{N-1} (\tilde{A}\tilde{s}[n])^*\tilde{s}[n]}{\sum_{n=0}^{N-1} |\tilde{s}[n]|^2} = \tilde{A}^*$$

clearly the wrong result!

**12.9** For the frequency estimate you should obtain  $\hat{f}_0 = -0.2000$ . The complex amplitude estimate is  $\hat{\tilde{A}} = -0.9692 + j0.9898$ , which produces the amplitude estimate (the magnitude of  $\hat{\tilde{A}}$ ) of 1.3853, and the phase estimate (the angle of  $\hat{\tilde{A}}$ ) of 2.3457 radians. The true values of the amplitude and phase are  $\sqrt{2} = 1.4142$  and  $\arctan(\frac{-1}{1}) = 2.3562$ . You can run the program FSSP3exer12\_9.m, which is contained on the CD, to obtain the results.

**12.10** Since  $\tilde{x}[n]$  is CWGN, we have that  $\tilde{x}[n] = u[n] + jv[n]$ , where  $u[n]$  and  $v[n]$  are each zero mean and each have variance  $\sigma^2/2$ . Therefore,

$$\begin{aligned} E[\hat{\sigma}^2] &= E\left[\frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2\right] = E\left[\frac{1}{N} \sum_{n=0}^{N-1} (u^2[n] + v^2[n])\right] \\ &= \frac{1}{N} \sum_{n=0}^{N-1} (E[u^2[n]] + E[v^2[n]]) \\ &= \frac{1}{N} \sum_{n=0}^{N-1} (\text{var}(u[n]) + \text{var}(v[n])) = \sigma^2 \end{aligned}$$

and the estimator is unbiased.

**12.11** The value of  $|\tilde{A}|$  required is about 0.46 and can be found from the

computer simulation by trial and error. For this value we have that  $P_D = 0.9575$ . The exact value can be found using the formula [[Kay 1998](#), pg. 485]

$$P_D = Q_{\chi'_2(\lambda)} \left( 2 \ln \frac{1}{P_{FA}} \right)$$

where the  $Q$  function here is the right-tail probability for a noncentral  $\chi^2_2$  PDF with 2 degrees of freedom and noncentrality parameter  $\lambda$ . For the problem at hand the noncentrality parameter is found as

$$\lambda = \frac{2N|\tilde{A}|^2}{\sigma^2}.$$

The exact value of  $P_D$  for  $|\tilde{A}| = 0.46$  is  $P_D = 0.9491$ . See also the MATLAB subprogram `chipr2.m` to evaluate this right-tail probability. You can run the program `FSSP3exer12_11.m`, which is contained on the CD, to obtain the results.

- 12.12** The true PSD is  $P_{\tilde{x}}(f) = \sigma^2 = 1$ . In dB quantities it is 0 dB for all  $-1/2 \leq f \leq 1/2$ . The estimate should be fairly close to the true PSD but note that it is not symmetric. Finally, there is no modification necessary to `PSD_est_avper.m`. As written, it can handle either real or complex data. You can run the program `FSSP3exer12_12.m`, which is contained on the CD, to obtain the results.

# **Part IV. Real-World Applications**

# Chapter 13. Case Studies - Estimation Problem

## 13.1. Introduction

We now illustrate the concepts and techniques presented in the previous chapters by applying them to real-world problems. In so doing, the general methodology for algorithm design summarized in [Figure 2.1](#) (“Roadmap for Algorithm Development”) will be used as a guide. Every problem seems to exhibit its own peculiarities, so it is impossible to give a “one size fits all” methodology. Thus, the roadmap and its application should be regarded as only a rough sequence of steps, each step to be either included or omitted depending upon the problem at hand. In an ideal world all the steps would be followed but in practice there are sometimes insufficient resources, time constraints, or even just a lack of physical knowledge that preclude implementing all the steps. For instance, in some problems “hard” specifications cannot be given since it is unclear what performance is desired. As an example, for an image processing problem, some viewers may object to the quality of an image. When asked what their objections are, the viewer may only say that it doesn’t look right. Quantifying what *does* look right is a difficult psychometric problem [[Winkler 1999](#)]. In addition to the methodology, a key aspect of algorithm design is the choice of the signal and noise models. Roadmaps to follow for model selection were given in [Figures 5.1](#) and [6.1](#) for the signal and noise, respectively. We will also use these as a guide.

Three case studies will be presented, encompassing an estimation problem in this chapter, a detection problem in [Chapter 14](#), and a spectral estimation problem in [Chapter 15](#). All are actual problems that the author has encountered. The estimation problem is to estimate the Doppler center frequency of a radar signal return. The detection problem is to detect a magnetic signal obscured by ambient and geomagnetic noise. Finally, the spectral estimation problem is one of determining the power spectral density of exercise induced muscle noise which complicates the signal analysis of an electrocardiogram (ECG) waveform.

## 13.2. Estimation Problem - Radar Doppler Center Frequency

Referring to [Figure 2.1](#) we proceed to discuss the various steps.

### 1. Problem Statement

The problem as defined by discussions with the client was to determine and implement the best radar Doppler frequency estimator. Also, it should be implemented “in real time and at a reasonable cost”

implemented in real time and at a reasonable cost.

## 2. Goals

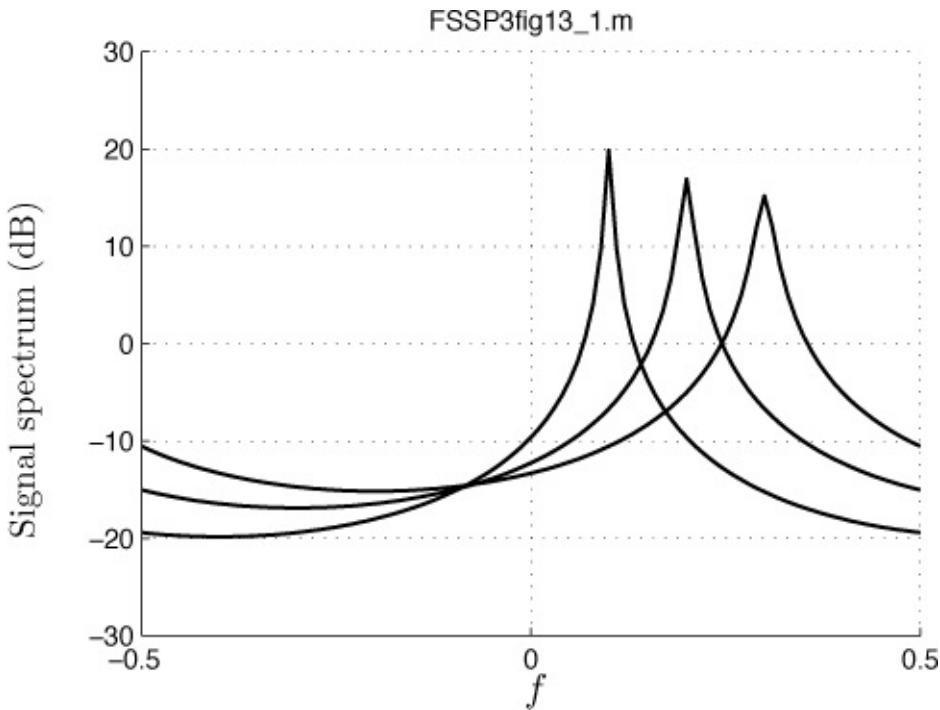
The overall goal was to exceed the performance of the currently used estimator, which was based on a fast Fourier transform (FFT). The frequency estimate was the location of the frequency at which the magnitude-squared of the FFT had a maximum, essentially a periodogram frequency estimator. And of course, the computational complexity had to be comparable to the currently used algorithm.

## 3. Specifications

No specifications were given other than the desire to outperform the FFT.

## 4. Information Acquisition

As we have already seen in [Algorithm 12.3](#) (“Sinusoidal amplitude, frequency, and phase estimation” for complex data), if the signal is a deterministic sinusoid embedded in complex white Gaussian noise (CWGN), then the FFT is optimal. This is because the FFT is an implementation of the periodogram, which is a maximum likelihood estimator (MLE) for complex data. Hence, no improvements are possible (at least for a signal-to-noise ratio (SNR) at which the MLE attains the Cramer-Rao lower bound (CRLB)). However, upon further discussions it was learned that the observed signal spectrum changed in width as the center frequency increased. This type of behavior is not consistent with modeling the signal as a deterministic sinusoid, whose bandwidth is fixed as the frequency changes. As an illustration, the signal power spectral density (PSD) dependence on center frequency would be as shown in [Figure 13.1](#). (Recall that because the data is complex, the PSD need not be symmetric about  $f = 0$ .) For the problem at hand it was stated that one could assume that the ratio of the center frequency to bandwidth was a fixed and known quantity. Specifically, letting  $f_0$  be the center frequency and  $B$  be the bandwidth, the ratio  $f_0/B = 15$  could be assumed. With this added information, it is possible that a better estimator could be obtained by leveraging the a priori knowledge that the bandwidth was not constant but depended upon the center frequency.



**Figure 13.1: Illustration of signal spectrum variation with center frequency. The bandwidth widens as the center frequency increases. Each power spectrum has the same total power of one.**

## 5. Mathematical Model Selection

### a. Signal model selection

Recall from [Figure 12.1](#) that for a radar problem the analog signal is originally a bandpass signal but becomes complex when demodulated to a lower frequency. Hence, we will need to choose complex signal and noise models as described in [Chapter 12](#). Referring to [Figure 5.1](#) for the roadmap for signal model selection, the physical origin of the signal was not specified so that an accepted model was not available. Furthermore, field data was also not provided. Hence, a decision about the signal model rested solely on the information provided in the previous “[Information Acquisition](#)” step. With the possible signal models of a random process, a transient signal, or a periodic signal, the obvious choice would be a random process model. This is because a stationary random process can be narrowband or broadband, with the bandwidth adjusted as needed. (There are also physical scattering theories that can be employed to justify the random process model.) Such was the case for the autoregressive (AR) model, with an example of its PSD shown in [Figure 4.4](#) for two different bandwidths. Recall from [Section 12.3.3](#) that a complex AR random process with a model order of  $p = 1$  is capable of modeling a bandpass process. The pole of the AR model is

obtained by finding the zero of the denominator of the AR system function, i.e.,  $A(z) = 1 + a[1]z^{-1}$ . Setting  $A(z) = 0$  and solving produces the pole  $z_p = -a[1]$ , and assuming it is complex to represent a bandpass PSD, the AR filter parameter is given by

$$a[1] = -r \exp(j2\pi f_0). \quad (13.1)$$

The PSD will then have a peak at  $f = f_0$  and the bandwidth of the peak will be determined by the pole radius  $r$  (see also [Appendix 4C](#) for a geometric interpretation of the effect of pole placement on the PSD). As  $r$  becomes nearer to one, the bandwidth of the PSD will decrease. Finally, for mathematical tractability we will assume a *Gaussian* AR signal process with zero mean.

---

### Exercise 13.1 – Effect of pole placement

Recall that if we model the complex signal  $\tilde{s}[n]$  by a complex AR random process, its PSD for  $p = 1$  is given by

$$P_{\tilde{s}}(f) = \frac{\sigma_{\tilde{u}}^2}{|1 + a[1] \exp(-j2\pi f)|^2}. \quad (13.2)$$

Plot this PSD for the parameter values of  $a[1]$  obtained by letting  $r = 0.8$ ,  $f_0 = 0.1$  and also for  $r = 0.6$ ,  $f_0 = 0.2$ . Let  $\sigma_{\tilde{u}}^2 = 1$ . Comment upon the observed center frequency and bandwidth.




---

Next recall that the center frequency and bandwidth are related to each other. This means that  $f_0$  and  $r$  are not independent parameters but must satisfy the constraint that  $Q = f_0/B = 15$ . Equivalently, the AR filter parameter cannot take on any arbitrary value. To utilize this constraint it is necessary to relate the bandwidth  $B$  to the pole radius. In [Appendix 13A](#) it is shown that

$$r = \frac{1}{1 + \pi B} = \frac{1}{1 + \pi f_0/Q} \quad (13.3)$$

under the assumption that  $B \ll 1$  or the signal bandwidth is small relative to the entire bandwidth (normalized to discrete-frequency) of 1. As expected, with an increase in center frequency, the pole radius decreases and hence the bandwidth increases, as was illustrated in [Figure 13.1](#). Also, note that the model breaks down if  $f_0 = 0$  since then  $r = 1$ , producing a pole on the unit circle and an unstable filter. Hence, we will assume that  $f_0 > 0$ .

### b. Noise model selection

Referring to [Figure 6.1](#) there was again no indication of the physical origin of the noise or even field data provided. The current FFT processor is only optimal for WGN. Hence, the use of an FFT processor indicates that the assumed noise is thought to be WGN or that the noise is colored, but not significantly so.

Additionally, no mention was made of any radar clutter, usually modeled as colored noise and possibly nonGaussian as well. The noise background therefore was probably due to ambient and electronic noise, which typically has a flat PSD. So as to not overcomplicate the problem, a WGN assumption was therefore made. The client had no reservations about this assumption and so it was adopted.

---



### Assuming WGN

Since it is inherently more difficult to design signal processing algorithms in colored and/or nonGaussian noise, it behooves us to assume white Gaussian noise whenever possible. Some instances when a flat noise PSD *cannot be assumed*, is when the signal is in a frequency band that exhibits clutter in radar or reverberation in sonar. In either of these cases, the noise is heavily correlated and in fact, its PSD resembles the signal PSD.

However, it is important to realize that the assumption of a flat PSD for the noise can be made even if the *overall* noise PSD is colored. Since any good signal processing algorithm will only process the frequency band over which the signal resides, the issue of whether the noise PSD is flat or colored *only applies to the signal band*. Thus, if the signal PSD is concentrated over a portion of the frequency spectrum where the noise PSD is relatively flat, a flat PSD assumption is valid for the noise. In a similar vein the assumption of nonGaussian noise will greatly complicate the algorithm design, and so is to be avoided whenever possible. An example of the difficulty encountered is given in [Chapter 14](#) for a signal detection problem.



---

### c. Overall data model

We will use the following model for the data

$$\tilde{x}[n] = \tilde{s}[n] + \tilde{w}[n] \quad n = 0, 1, \dots, N - 1$$

where  $\tilde{s}[n]$  is the complex signal modeled as a complex Gaussian AR process

with  $p = 1$  and whose parameters are  $a[1] = -r \exp(j2\pi f_0)$ , with  $r$  being a function of  $f_0$  according to (13.3), and the excitation noise being CWGN variance with variance  $\sigma_{\tilde{u}}^2$ . The additive noise  $\tilde{w}[n]$  is CWGN with variance  $\sigma^2$ . Note that the signal has the time domain representation

$$\tilde{s}[n] = -a[1]\tilde{s}[n-1] + \tilde{u}[n]$$

which can be used for computer simulation purposes.

## 6. Feasibility Study

Since this is an estimation problem, it is good practice to compute the CRLB on the variance of any unbiased estimator (see [Section 8.2.1](#)). Two problems arise when we attempt to do so. One is that the data is complex and we have not described how to obtain bounds for this case. Secondly, it is difficult, but not impossible, to derive the bounds when the AR signal process is added to a CWGN process. This is because the resulting process as discussed in [Section 11.3](#) is no longer an AR process, complicating the derivation of the bound. As an approximation, we will ignore the CWGN when computing the CRLB. Of course, if the resulting bound exhibits any peculiarities, then we must be aware that they are probably due to this omission.

It can be shown that the CRLB for  $\hat{f}_0$  based on the complex data  $\tilde{s}[n]$  for  $n = 0, 1, \dots, N-1$  is approximately

$$\text{var}(\hat{f}_0) \geq \frac{1 - r^2}{2N \left( \frac{\pi^2 r^4}{Q^2} + 4\pi^2 r^2 \right)} \quad (13.4)$$

where  $r$  is given by (13.3). It should be noted that according to the bound, if  $r \rightarrow 1$ , then the bound approaches zero! This is due to our unrealistic assumption that there is no additive CWGN. In the absence of additive noise the signal spectrum would approach an impulse as  $r \rightarrow 1$ , and so a perfect frequency estimate would be obtained.

## 7. Specification Modifications (if necessary)

Not applicable.

## 8. Proposed Solution

Again to simplify the subsequent development of the estimator, we assume no additive noise. Then, we have the familiar problem of estimating the center frequency of a Gaussian random process. This has been described in [Algorithm 9.8](#) for real data and in [Algorithm 12.8](#) for complex data, as “Random signal

center frequency estimation”. For the problem under consideration, whereby we assume that there is no noise, we let  $\sigma^2 = 0$  in (12.18). To find the approximate MLE of the center frequency we then need to minimize

$$J(f_0) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{I(f)}{P_{\tilde{s}}(f - f_0)} df \quad (13.5)$$

over  $0 < f_{0_{\min}} < f_0 < f_{0_{\max}} < 0.5$ . The function  $I(f)$  is the periodogram given by

$$I(f) = \frac{1}{N} \left| \sum_{n=0}^{N-1} \tilde{x}[n] \exp(-j2\pi f n) \right|^2.$$

Upon using (13.1) and (13.2) this reduces to

$$J(f_0) = \frac{1}{\sigma_{\tilde{u}}^2} \int_{-\frac{1}{2}}^{\frac{1}{2}} I(f) |1 - r \exp(-j2\pi(f - f_0))|^2 df$$

or equivalently we can minimize

$$J'(f_0) = \int_{-\frac{1}{2}}^{\frac{1}{2}} I(f) |1 - r \exp(-j2\pi(f - f_0))|^2 df$$

since  $\sigma_{\tilde{u}}^2 > 0$ . After some manipulation and a few small approximations, we arrive at the final objective function

$$J''(f_0) = 1 + r^2 - 2r \operatorname{Re} [\hat{\rho}_s \exp(-j2\pi f_0)] \quad (13.6)$$

where  $r$  is given by (13.3) and  $\hat{\rho}_s$  is an estimate of the normalized autocorrelation sequence at lag  $k = 1$ . The normalized autocorrelation for  $k = 1$  is defined as

$$\rho_{\tilde{s}} = \frac{r_{\tilde{s}}[1]}{r_{\tilde{s}}[0]}. \quad (13.7)$$

Its estimator is

$$\hat{\rho}_{\tilde{s}} = \frac{(1/N) \sum_{n=0}^{N-2} x^*[n]x[n+1]}{(1/N) \sum_{n=0}^{N-1} |\tilde{x}[n]|^2}. \quad (13.8)$$

See also [Section 6.4.4](#) for estimation of the autocorrelation sequence of a real random process. The objective function has a nice intuitive interpretation.

---

### Exercise 13.2 – Interpreting the resultant estimator

It is always useful and serves as a good reality check to make sure the final estimator is a sensible one. Show that the theoretical normalized autocorrelation coefficient given by (13.7) is equal to

$\rho_{\tilde{s}} = r \exp(j2\pi f_0)$  for a complex AR process of order  $p = 1$ . Next replace  $\hat{\rho}_{\tilde{s}}$  in the objective function of (13.6) by  $\rho_{\tilde{s}}$ . Finally, noting that we should choose  $f_0$  to minimize the objective function, interpret the operation of the estimator. Hint: Use (12.8).

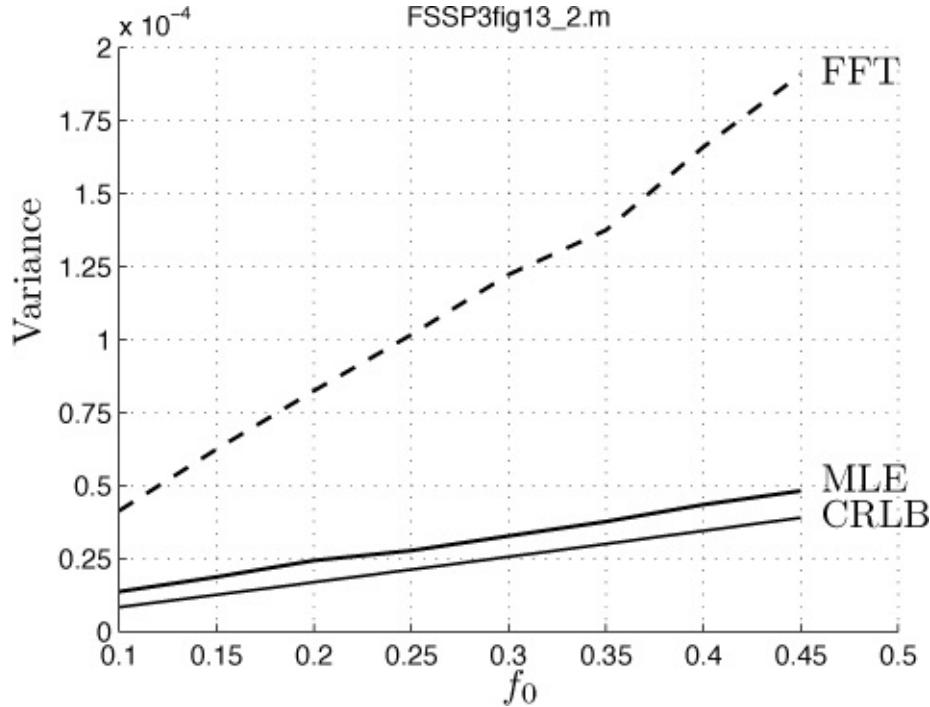
---

## 9, 10. Performance Analysis

To ascertain the performance of the proposed MLE, we use a computer simulation. As a benchmark and also as a means of ensuring that the results are reasonable, we also plot the CRLB. If the performance of the MLE is below the CRLB, then most likely the estimator is biased since the CRLB assumes an *unbiased estimator*. Hence, *before employing the CRLB*, it behooves us to also estimate the mean of the estimator to ensure that the estimator is unbiased. Computer simulations indicate that the MLE as well as the FFT estimator are both unbiased estimators. Since the original signal is a bandpass process that has been demodulated, we will assume that the demodulated discrete-time lowpass process has a center frequency in the range  $0.1 \leq f_0 \leq 0.45$ . This is consistent with the modeling, since otherwise, the radius of the AR pole for  $f_0 = 0$  would be  $r = 1$ , from (13.3), producing a pole on the unit circle and an impulsive PSD. This is clearly unrealistic. *As is always the case in practice, we need to temper the ease of use of the model with the physical accuracy of that model.*

We use a data record length of  $N = 64$  in order to determine the variances of the MLE and FFT center frequency estimators. An exact analytical evaluation of the performance is, unfortunately, not possible, and so we resort to a Monte Carlo evaluation using 10,000 trials. To determine the length of the FFT, we use a trial and error procedure, in which we keep increasing the size until the results “stabilize”. A length of 2048 appears to be adequate. Recall that we need to search for the maximum of the periodogram (see also [Algorithm 9.9](#) for a further discussion). To minimize the objective function given in (13.6) over the interval  $0.1 \leq f_0 \leq 0.45$ , we use the same trial and error procedure to find that a grid size of 0.001 is adequate. In either case any error due to quantization in evaluating the objective function will be at most 1/2 of the grid size. In the case of the FFT this will produce a maximum squared frequency error of  $[1/(2(2048))]^2 = 6 \times 10^{-8}$ , which is negligible compared to the variance. Similarly, for the MLE the

maximum squared error due to grid search size would be  $(1/0.001)^2 = 10^{-6}$ , again well below the variance. The results are shown in [Figure 13.2](#). It is seen that the CRLB does indeed yield a lower bound on performance. The performance improvement of the MLE as compared to the FFT is largest for higher center frequencies. The variance of the MLE at  $f_0 = 0.45$  is  $0.4821 \times 10^{-4}$  while that for the FFT is  $1.908 \times 10^{-4}$  for a variance reduction of about 4. The reason for the improvement is that for higher center frequencies the spectrum widens, and so the optimality property of the periodogram for a sinusoid no longer applies, with the approximation becoming progressively worse as the bandwidth increases. The MATLAB program `FSSP3fig13_2_exer.m`, which is included on the CD, has been used to produce this figure. It can be consulted for the details of the computer simulation, the algorithm implementation, and the performance results.



**Figure 13.2: Variances for the MLE and the FFT center frequency estimators with the Cramer-Rao lower bound as a benchmark.**

As discussed previously, the MLE only attains the CRLB for large data record lengths. Here the data record length of  $N = 64$  appears to be too small to attain the CRLB. The increase in variance of the MLE as compared to the CRLB, however, is small. Hence, it is probably not worth attempting to find a better estimator than the MLE, *even if it exists, which it may not!*

### **Exercise 13.3 – Attaining the CRLB**

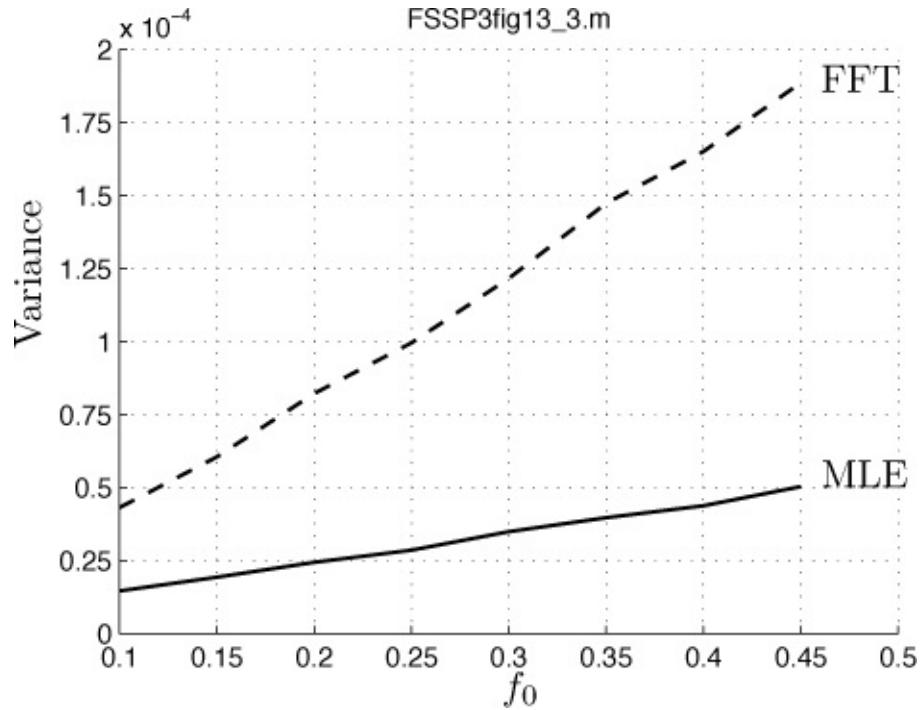
Run the MATLAB program `FSSP3fig13_2_exer.m`, which was used to generate [Figure 13.2](#), but increase the data record length to  $N = 300$ .

How does the MLE now compare to the CRLB? Hint: It may take several minutes to run.

---

## **11, 12. Sensitivity Analysis**

To ensure that the computer simulation is a realistic prediction of how the algorithm will work in practice, we need to reassess the assumptions under which the algorithm was developed. The obvious one, which has already been highlighted, is the assumption of no additive noise. This was done in an attempt to make the problem easier. Adding noise complicates the calculation of the CRLB, and will necessitate a much more complicated MLE algorithm. It is even possible that the algorithm may not be implementable in real-time, as was our goal. Yet, we cannot totally ignore the additive noise. A sensitivity analysis is required to determine the effect that additive noise will have upon the estimator's performance. To do so we repeat the computer simulation but now add CWGN with variance  $\sigma^2 = 0.01$ . The power of the AR signal process is set at  $r_{\tilde{s}}[0] = 1$  so that the SNR is 20 dB. The results are shown in [Figure 13.3](#). It is seen by comparing [Figure 13.3](#) to [Figure 13.2](#) that the estimator is not very sensitive to the SNR, as long as it is not too low. Hence, our initial approach of ignoring the additive noise to simplify the algorithm development is justified.



**Figure 13.3: Variances for the MLE and the FFT estimator when additive CWGN is included.**

### 13, 14, 15. Field Testing

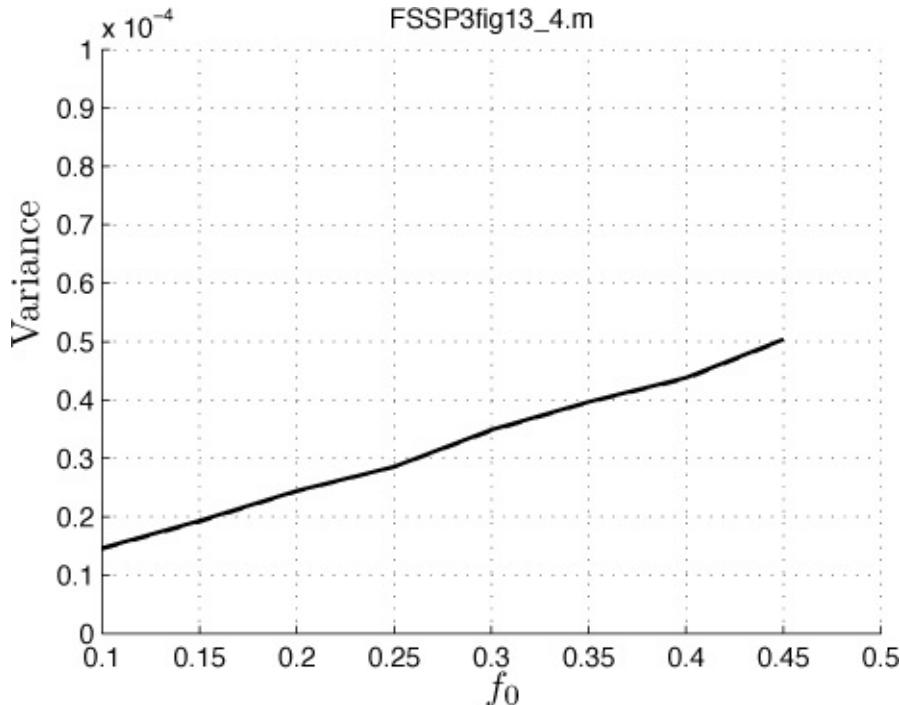
The project upon which this case study was based did not progress to the field testing stage (at least not that the author is aware of). If it had, it is possible that the algorithm proposed may have been too computationally complex to be implemented. It is usually advisable to have an alternative approach, which is computationally less demanding. This approach is best obtained by “paring down” the optimal or near optimal algorithm, in the hope of obtaining a more easily implementable algorithm, but with only slightly diminished performance. For the problem at hand the MLE suggests how to do this. Recall from [Exercise 13.2](#) that in minimizing the objective function, we expect the minimum to occur when  $f_0$  is chosen so that the estimated  $\rho_{\tilde{s}}$  is near its theoretical value. That value is  $\rho_{\tilde{s}} = r_{\tilde{s}}[1]/r_{\tilde{s}}[0] = \exp(j2\pi f_0)$ . Thus, we might propose to use

$$\hat{f}_0 = \frac{1}{2\pi} \angle \left( \frac{\hat{r}_{\tilde{s}}[1]}{\hat{r}_{\tilde{s}}[0]} \right)$$

where  $\angle$  denotes the angle of the complex quantity, as our estimator. The estimate is then given by the angle of the complex quantity in [\(13.8\)](#) after dividing by  $2\pi$ . This yields as our alternative estimator

$$\hat{f}_0 = \frac{1}{2\pi} \angle \left( \frac{(1/N) \sum_{n=0}^{N-2} \tilde{x}^*[n] \tilde{x}[n+1]}{(1/N) \sum_{n=0}^{N-1} |\tilde{x}[n]|^2} \right). \quad (13.9)$$

The computational complexity will be much reduced since it is on the order of only  $2N$  complex multiply/adds. (It can be reduced further by omitting the  $N$  divisors and also the denominator term since these positive quantities will not affect the phase.) To assess the loss in performance of this *autocorrelation estimator* we can rerun the computer simulation used to produce [Figure 13.3](#) (includes CWGN) but now compare the MLE to the autocorrelation estimator. The results are shown in [Figure 13.4](#). It is seen that there is no discernable difference in performance between the two estimators. In fact, the two curves lie on top of each other. Although somewhat surprising at first blush, there is some justification for this seemingly “too good” performance [[Kay 1989](#)]. Note that by first appealing to the MLE, we were able to identify another estimator with less computation and about the same performance. This observation seems to hold for many practical problems! It would be expected, however, that the MLE would be more robust in practice, and in particular, at lower SNR and/or shorter data record lengths. The MATLAB program `FSSP3fig13_4_exer.m`, which is included on the CD, has been used to produce this figure. It can be consulted for the details of the computer simulation, the algorithm implementation, and the performance results.



**Figure 13.4: Variances for the MLE and the autocorrelation estimators**

when additive CWGN is included. The two curves lie on top of each other.

---

### Exercise 13.4 – MLE and autocorrelation estimators for a more difficult set of conditions

Rerun the program FSSP3fig13\_4\_exer.m. The program used a noise variance of  $\sigma^2 = 0.01$ , but increase it to  $\sigma^2 = 0.1$ . Also, decrease the data record length from  $N = 64$  to  $N = 32$ . Be sure to change the axes scaling using `axis ([0.1 0.45 0 1e-3])`. What do you observe? Can you explain this? Hint: Assuming the PDF of the autocorrelation estimator is Gaussian ( $N(\mu, \sigma^2)$ ), what might happen if the mean  $\mu$  were the true value of  $f_0 = 0.45$  and the standard deviation  $\sigma$  were

$\sqrt{0.0002} = 0.014$ ? What is  $\mu + 3\sigma$ ? Hint: It may take several minutes to run.

•

---

### 13.3. Lessons Learned

- For parameters that need to be estimated, such as the `a[1]` filter parameter, any constraints on the possible set of values will improve the accuracy of the estimate. This is because the estimator is not allowed to produce values that are far away from the true parameter value.
- Whenever possible, assume white Gaussian noise as opposed to colored and/or nonGaussian noise. White Gaussian noise leads to an “easy” problem while the latter results in a much harder one.
- The assumption of an AR random process model for a signal leverages many useful theoretical and practical properties of such a model. More general models such as an autoregressive moving average (ARMA) lead to very difficult optimization and therefore implementation problems in practice.
- As a reality check, it is always good to understand the operation of the estimator from an intuitive standpoint. As in [Exercise 13.2](#), replacing the data by the quantity it is attempting to estimate and then minimizing the objective function, should produce the true parameter values.
- As is always the case in practice, we need to temper the ease of use of the model with the physical accuracy of that model.
- It is advisable that for algorithms designed to work in real time, a

“backup” algorithm based on the proposed algorithm should also be available. This is useful if the computational requirements of the proposed algorithm are too demanding for implementation.

## References

Kay, S. “A Fast and Accurate Single Frequency Estimator”, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Dec. 1989.

Winkler, S., “Issues in Vision Modeling for Perceptual Video Quality Assessment”, *Signal Processing*, Vol. 78, pp. 231–252, Oct. 1999.

## Appendix 13A. 3 dB Bandwidth of AR PSD

In this appendix we derive a relationship between the pole radius  $r$  of the AR filter and the center frequency  $f_0$  of the process. Substituting the complex filter parameter, using  $a[1] = -r \exp(j2\pi f_0)$ , into the PSD, we have

$$P_{\tilde{s}}(f) = \frac{\sigma_{\tilde{u}}^2}{|1 - r \exp(-j2\pi(f - f_0))|^2}.$$

The 3 dB bandwidth  $B$  is defined by

$$\frac{P_{\tilde{s}}(f_0 + B/2)}{P_{\tilde{s}}(f_0)} = \frac{1}{2}$$

or

$$\frac{\sigma_{\tilde{u}}^2}{|1 - r \exp(-j\pi B)|^2} = \frac{1}{2} \frac{\sigma_{\tilde{u}}^2}{|1 - r|^2},$$

then

$$(1 - r)^2 = \frac{1}{2} |1 - r \exp(-j\pi B)|^2.$$

Using  $\exp(-j\pi B) \approx 1 - j\pi B$ , which is valid for  $B \ll 1$ , this simplifies to

$$B = \frac{1 - r}{\pi r}.$$

Thus, we have that

$$r = \frac{1}{1 + \pi B} = \frac{1}{1 + \pi f_0/Q}.$$

It is seen that the pole radius and the pole frequency are related to each other.

## Appendix 13B. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators, respectively.

**13.1** For the first radius and angle pair the PSD is centered at  $f_0 = 0.1$  with a given bandwidth. For the second radius and angle pair, which has a smaller pole radius, the PSD has a wider bandwidth, and the PSD is centered at  $f_0 = 0.2$ . You can run the program `FSSP3exer13_1.m`, which is contained on the CD, to obtain the results.

**13.2** To evaluate the theoretical normalized autocorrelation for  $k = 1$  we need the ACS for a complex AR random process of order one. This was given by (12.8) as

$$r_{\tilde{s}}[k] = \begin{cases} \frac{\sigma_{\tilde{u}}^2}{1-|a[1]|^2}(-a[1])^k & k \geq 0 \\ r_{\tilde{s}}^*[-k] & k < 0. \end{cases}$$

Hence,  $r_{\tilde{s}}[1]/r_{\tilde{s}}[0] = -a[1]$  and since  $a[1] = -r \exp(j2\pi f_0)$ , we have that

$$\rho_{\tilde{s}} = r_{true} \exp(j2\pi f_{0true})$$

where we have subscripted  $r$  and  $f_0$  to indicate that these are the true values. Substituting this into the objective function produces

$$\begin{aligned} J''(f_0) &= 1 + r^2 - 2r \operatorname{Re}[r_{true} \exp(j2\pi f_{0true}) \exp(-j2\pi f_0)] \\ &= 1 + r^2 - 2rr_{true} \cos[2\pi(f_{0true} - f_0)]. \end{aligned}$$

Clearly, this is minimized and equals zero when  $r = r_{true}$  and  $f_0 = f_{0true}$ .

**13.3** By increasing the data record length to  $N = 300$  you will observe that all the variances decrease. Now, however, the MLE attains the CRLB since the two curves should coalesce. You can run the program `FSSP3exer13_3.m`, which is contained on the CD, to obtain the results.

**13.4** The autocorrelation estimator uses the phase angle, which is assumed to be in the interval  $[-\pi, \pi]$ , corresponding to a center frequency range of  $[-0.5, 0.5]$ . However, at the decreased SNR and decreased data record length, the width of the Gaussian PDF increases and some of the frequency estimates fall outside this interval, in fact, they “wrap around”. These estimates then become closer to  $-0.5$  than  $0.5$ , leading to large

errors and also a large variance. Note that the  $\sigma$  given in the exercise as  $\sqrt{0.0002}$  is the approximate standard deviation of the MLE at  $f_0 = 0.45$ . This should be observed from your plot. You can run the program FSSP3exer13\_4.m, which is contained on the CD, to obtain the results.

# Chapter 14. Case Studies - Detection Problem

## 14.1. Introduction

In this chapter we address a real-world signal detection problem. The project sought to improve upon the detection capability of a current system by improving the sensitivity of the sensor as well as by leveraging the improvements possible through digital signal processing technology. Development and testing of the new system is ongoing.

## 14.2. Detection Problem - Magnetic Signal Detection

Referring to [Figure 2.1](#) we proceed to discuss the various steps.

### 1. Problem Statement

The problem was to detect a target that was emitting a periodic electromagnetic signal on an intermittent basis and was embedded in ambient noise and very spiky *geomagnetic noise*. The sensor that acquires the signal is a magnetometer, similar to the device that is used in airport screening to detect metal objects.

### 2. Goals

Because the signal is generally unknown, it was not possible to employ a matched filter. The current system utilized an energy detector. The goal was to improve detection performance, i.e., to yield a higher probability of detection for a given false alarm probability. For the preliminary design the foremost goal was performance, and so a real-time implementation was not required.

### 3. Specifications

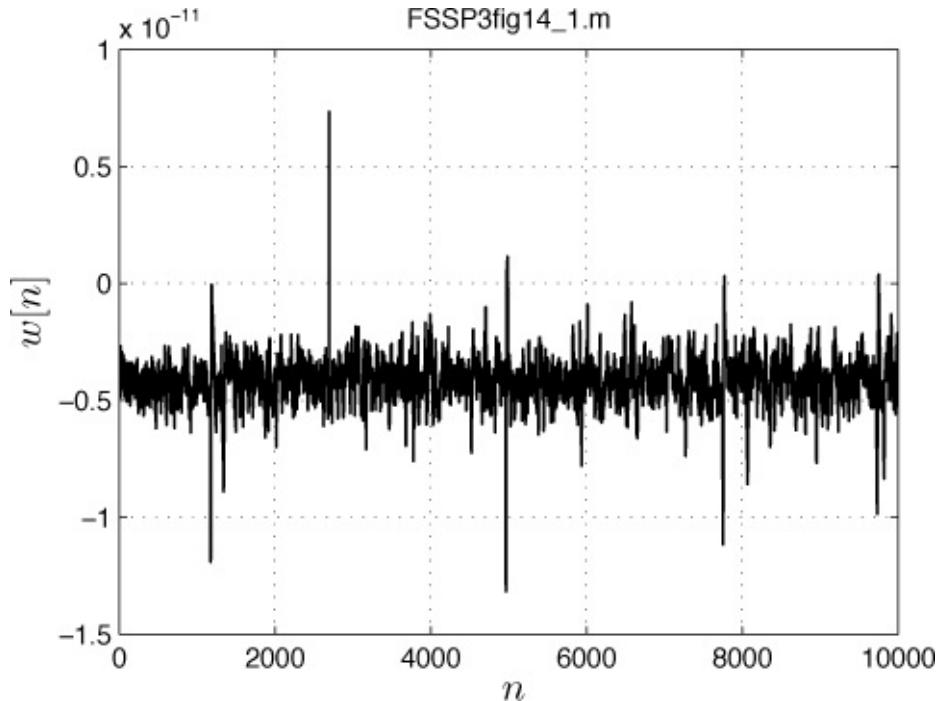
No other specifications were given except the desire to outperform an energy detector.

### 4. Information Acquisition

A list of questions was prepared to narrow down the possible assumptions that could be made in developing the detector. As a result of subsequent discussions with the client, a number of modeling assumptions were agreed upon as being reasonable, in light of what was known about the physics of the problem. These assumptions were, of course, made so as to *allow the subsequent development of the detector to leverage known optimal or asymptotically optimal results, or at least to require only simple extensions of those known results*. Otherwise, it is

likely the project would not be completed on time and within budgetary constraints. As a result, the following assumptions were made. The signal is a very lowpass signal (sometimes called an *extremely low frequency* (ELF) signal) with the received data being a real data set. Additionally, the signal could be assumed to be periodic but of short duration, the so-called “on” time of the signal. The period of the signal or equivalently its fundamental frequency was not known but could be assumed to be within a narrow range. Also, because of the propagation losses of the radiated target signal to the magnetometer, it was assumed that only the first few harmonics had significant amplitudes.

The modeling of the noise benefited by having representative noise data available. As an example, 10,000 samples of typical noise data appears as shown in [Figure 14.1](#). It is seen that the noise consists of a background noise of lower level plus large level spikes on an infrequent basis. The background noise is most likely ambient noise, while the spikes are due to atmospheric disturbances such as lightning strikes, which can propagate long distances at low frequencies. The latter is referred to as *geomagnetic noise*. Because of the way the data was collected, the mean level is not zero. In any practical system the mean level of the data is always subtracted out before processing.



**Figure 14.1: Typical noise data consisting of real data samples. The data samples have been connected with straight lines for easier viewing.**

The very small value of the amplitude is due to the units used to represent

magnetic flux density, with the amplitudes usually on the order of  $10^{-10}$  Teslas. Henceforth, to avoid these small values, we will normalize the data by dividing it by the maximum absolute value in the data record, yielding values in the interval  $-1 \leq w[n] \leq 1$ .

---

### Exercise 14.1 – Discerning nonGaussian noise

It is clearly of interest to discern if the noise shown in [Figure 14.1](#) is nonGaussian. A simple method is to estimate the background noise variance  $\sigma^2$  to determine if the spikes exceed  $\mu \pm 3\sigma$ . (See also [Section 6.3](#) for the kurtosis method.) Using the data shown in [Figure 14.1](#), first subtract out the mean, and then estimate the background noise variance using the part of the data for  $3000 \leq n \leq 4000$ , which appears to consist of ambient noise only. Then find the  $\pm 3\sigma$  limits. Comment on the results. You can load in the entire data record using load `FSSP3exer14_1`.

•

---

## 5. Mathematical Model Selection

### a. Signal model selection

We refer to Table 5.1, which describes the signal model selection process. The physical origin of the signal radiation is known. However, there is no single model that will suffice, since the signal characteristics will change with the target motion and the sensing environment. From the physics of the radiating process, the periodic signal model is a reasonable one, albeit with unknown parameters to accommodate changes in the target and the environment. Hence, the client accepted the periodic signal with unknown parameters model. The periodic signal was discussed in [Section 3.4.6](#). It was mentioned there, but not described in detail, that a periodic signal could also be represented as a Fourier series. For the problem at hand, this is a useful representation. This is because for a known fundamental frequency and a known number of harmonics, the Fourier series will allow us to apply the linear model. When the fundamental frequency is unknown, the signal model becomes a *partially* linear model as discussed in [Section 3.5.4](#), for which the maximum likelihood estimator (MLE) can be easily found. Coupled with a generalized likelihood ratio test (GLRT), which requires maximum likelihood estimation of the unknown signal parameters, we have a viable approach to designing a detection algorithm. Since the signal is known except for some parameters, it is anticipated that any derived

detector will outperform the energy detector, for which no signal information is used.

The Fourier series model is given as [Kay 1993, pp. 88–90]

$$s[n] = \sum_{k=1}^M a_k \cos(2\pi k f_0 n) + \sum_{k=1}^M b_k \sin(2\pi k f_0 n) \quad n = 0, 1, \dots, N-1 \quad (14.1)$$

where  $f_0$  is the *fundamental frequency* and the remaining frequency components are generally referred to as the *harmonics*. According to the problem information acquired, it is reasonable to assume that  $M = 3$  so that the signal is composed of the fundamental and the first two harmonics (the higher harmonics are generally attenuated to the point that they are comparable in level to the noise and so provide very little additional detectability). In this model the unknown parameters are  $\{a_1, a_2, a_3, b_1, b_2, b_3, f_0\}$ , and thus we have the partially linear model in matrix/vector format expressed as

$$\begin{bmatrix} s[0] \\ s[1] \\ \vdots \\ s[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots \\ \cos(2\pi f_0) & \cos(4\pi f_0) & \cos(6\pi f_0) & \dots \\ \vdots & \vdots & \vdots & \vdots \\ \cos[2\pi f_0(N-1)] & \cos[4\pi f_0(N-1)] & \cos[6\pi f_0(N-1)] & \dots \\ 0 & 0 & 0 & \dots \\ \sin(2\pi f_0) & \sin(4\pi f_0) & \sin(6\pi f_0) & \dots \\ \vdots & \vdots & \vdots & \vdots \\ \sin[2\pi f_0(N-1)] & \sin[4\pi f_0(N-1)] & \sin[6\pi f_0(N-1)] & \dots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (14.2)$$

or succinctly as  $\mathbf{s} = \mathbf{H}\boldsymbol{\theta}$ , where  $\mathbf{s}$  is  $N \times 1$ ,  $\mathbf{H}$  is  $N \times 6$ , and  $\boldsymbol{\theta}$  is  $6 \times 1$ . Note that the matrix  $\mathbf{H}$  is known *except for*  $f_0$ , hence, the name *partially* linear model. It can be shown that if  $f_0 >> 1/N$ , then the columns of  $\mathbf{H}$  are approximately orthogonal (see [Exercise 14.2](#)). This will be useful in our algorithm development and is the same “trick” we used in [Algorithm 9.3](#) to reduce the estimator to

finding the peak location of a periodogram.

Finally, it should be observed that because the parameters of the signal are unknown a priori, and can change with the target motion and the sensing environment, an adaptive detector will be necessary. In [Figure 5.1](#) this is designated as a *data-adaptive model*.

---

### Exercise 14.2 – Orthogonality of H columns

Referring to the **H** matrix given in [\(14.2\)](#), let  $f_0 = 0.1$  and  $N = 100$ . Then compute  $\mathbf{H}^T \mathbf{H}/(N/2)$  to show that it is approximately a  $6 \times 6$  identity matrix.

•

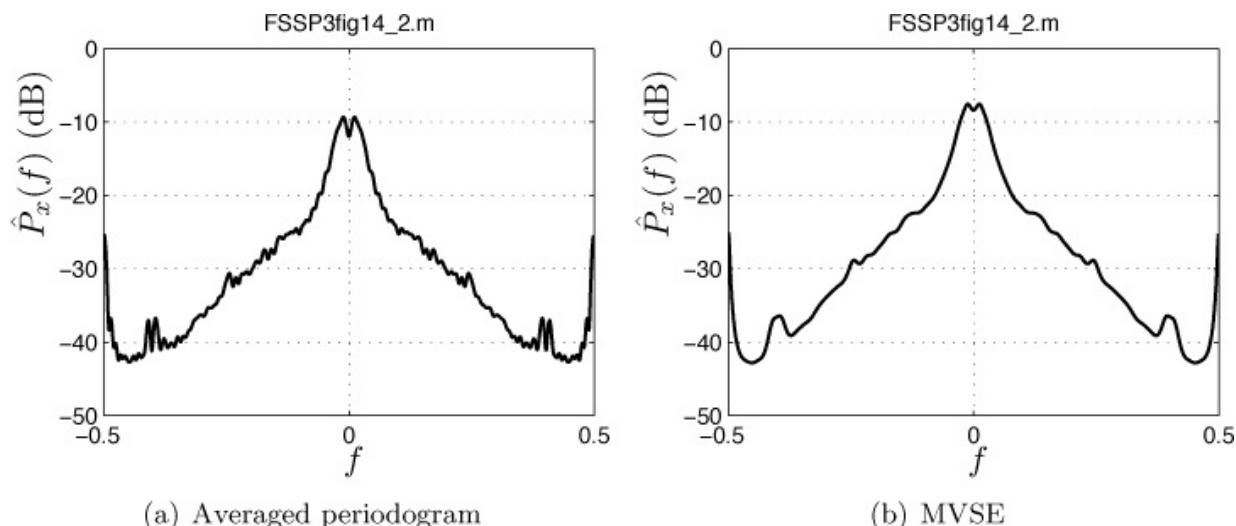
---

#### b. Noise model selection

Referring to [Figure 6.1](#), it is first concluded that although the physical origin of the noise is known, there is no generally accepted fixed model for geomagnetic noise. This is because the statistical characteristics change with the sensing location, time of day, weather conditions, *etc*. As a result, any successful algorithm will have to be data-adaptive, in that it will need to estimate the noise characteristics on-line. The use of the field data, as was shown in [Figure 14.1](#), will allow us to formulate a class of models whose parameters can be estimated at regular intervals. As to the question of noise stationarity, since the signal is intermittent and only on for a very short time, it can be assumed that *over the signal duration interval*, the noise is stationary. This assumption leads us in [Figure 6.1](#) to the decision of whether or not the noise is Gaussian. As is evident from [Figure 14.1](#), the noise appears to be mostly Gaussian but exhibits high level spikes at times. Therefore, overall the noise needs to be modeled as nonGaussian.

With a nonGaussian noise assumption it is important to be able to assume a flat PSD. If this is reasonable, then one can conclude that the noise samples are uncorrelated, and therefore, we could justify the independent and identically distributed (IID) noise model. Of course, this is a theoretical “stretch”, since for nonGaussian noise uncorrelated does not imply independence. But otherwise, if we have to assume dependent nonGaussian noise samples, then we face a very difficult problem. Can we assume a flat PSD? Using the data shown in [Figure 14.1](#) we plot the averaged periodogram and the minimum variance spectral estimator (MVSE) of the data in [Figure 14.2](#) after the mean has been subtracted

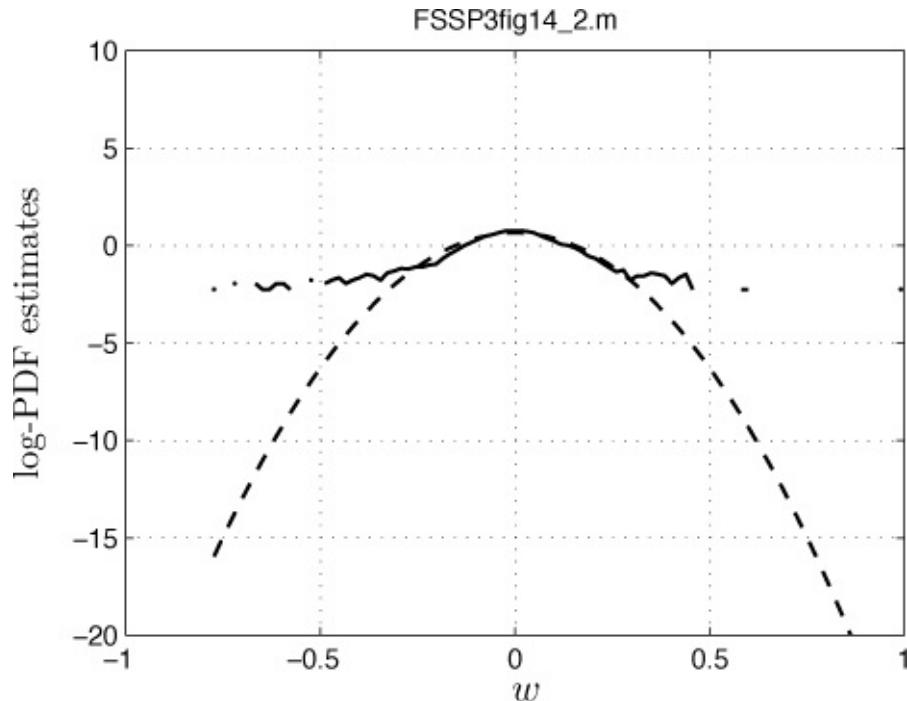
(see [Algorithms 11.1](#) and [11.2](#)). The MATLAB programs `PSD_est_avper.m` and `PSD_est_mvse.m`, which are contained on the CD, have been used. The data record is  $N = 10,000$  samples, and for the averaged periodogram spectral estimator the block length is  $L = 100$  for a resolution of about 0.01. For the MVSE we have used  $p = 50$ . It is seen that both spectral estimates produce very similar spectral estimates, lending credibility to the results. It is seen that even over a small band the noise PSD does not appear to be flat. It might be supposed then that a prewhitener would be required. But if we use a prewhitener, which is a linear filtering operation, the nonGaussian noise samples will become more Gaussian, due to the averaging effect of the prewhitening filter (recall the central limit theorem). Hence, large level spikes may “bleed through”. On the other hand, if we do not use the prewhitener, there may be some detection loss. As long as the bandwidth of the signal components, which is the reciprocal of the signal duration, is small relative to the variation of the noise PSD over that bandwidth, we can argue that the loss due to the absence of a prewhitener will be small. Therefore, we will make the assumption of IID noise samples since it is believed that it is more important to accommodate the nonGaussian nature of the noise than its coloration (see also the “[Field Testing](#)” section for a further discussion).



**Figure 14.2: Averaged periodogram and minimum variance spectral estimate (MVSE) using  $p = 50$  for data shown in [Figure 14.1](#).**

To increase our confidence that the nonGaussian noise assumption is required, we estimate the probability density function (PDF) as explained in [Section 6.4.6](#). Using the MATLAB subprogram `pdf_hist_est.m`, which is contained on the CD, we estimate the PDF using a histogram, and then compare the estimated

PDF to a Gaussian PDF of the same variance. The sample mean has been subtracted from the data so that the mean of both PDFs is zero. Using the data shown in [Figure 14.1](#), but scaling it so that the maximum absolute value is unity, we have the results shown in [Figure 14.3](#). The histogram estimate is “missing” in many of the bins since no samples fell within those bins. The usual range of a Gaussian random variable, which is  $\pm 3\sigma = \pm 0.2$  for this data, is exceeded a large fraction of the time as can be observed from the tails of the histogram PDF estimate shown as the solid line. It is clear from this plot that the noise is very nonGaussian.



**Figure 14.3: Histogram estimated PDF of typical noise data (solid curve) as compared to a Gaussian PDF (dashed curve) with the same variance. The plot is on a logarithmic scale (the common logarithm of both estimates have been taken before displaying the results). The  $3\sigma$  point of the Gaussian PDF is at 0.2.**

### c. Overall data model

We will use the following model for the real data

$$\begin{aligned} x[n] &= s[n] + w[n] \\ &= \sum_{k=1}^3 a_k \cos(2\pi k f_0 n) + \sum_{k=1}^3 b_k \sin(2\pi k f_0 n) + w[n] \quad (14.3) \end{aligned}$$

for  $n = 0, 1, \dots, N - 1$ , where the Fourier series coefficients  $a_k$ 's and  $b_k$ 's are

unknown, the fundamental frequency  $f_0$  is unknown, but lies in the range of  $f_{0_{\min}} \leq f_0 \leq f_{0_{\max}}$ , and the noise samples  $w[n]$  are zero mean, IID, with known PDF  $p(w)$ .

## 6. Feasibility Study

As discussed in [Section 7.4](#) it is advantageous to employ a performance bound for the detection problem. Typically, we use the Neyman-Pearson (NP) detector performance which assumes that the PDFs when a signal is present (the  $H_1$  hypothesis) and when noise only is present (the  $H_0$  hypothesis) are known.

Because we assume this additional information, the performance must be better than any other detector that does not have access to this prior knowledge.

Determining this upper bound on performance will allow us to ascertain the *potential improvement* of a detector relative to the energy detector.

For the problem at hand we will need to specify the noise PDF as well as the unknown signal parameters to obtain the PDFs, and hence the NP detector statistic, and finally the bound. The NP detector decides a signal is present if (see [Section 8.2.2](#))

$$L(\mathbf{x}) = \frac{p(\mathbf{x}; \mathcal{H}_1)}{p(\mathbf{x}; \mathcal{H}_0)} > \gamma_{NP}$$

and for the model given in [\(14.3\)](#), this becomes

$$L(\mathbf{x}) = \frac{\prod_{n=0}^{N-1} p(x[n] - s[n])}{\prod_{n=0}^{N-1} p(x[n])}$$

where  $p(w)$  is the PDF of each noise sample  $w[n]$ . We have assumed that the noise samples are IID and the signal is known. Alternatively, we have

$$\ln L(\mathbf{x}) = \sum_{n=0}^{N-1} \ln \frac{p(x[n] - s[n])}{p(x[n])}$$

so that it remains to find  $p(w)$ . We could estimate the PDF based on the data given in [Figure 14.1](#), an example of which is shown in [Figure 14.3](#), but this is only one realization of data, obtained from a particular field test. Instead, we assume that the noise is Laplacian since this exhibits the heavy tails that we are seeing in the data, and is commonly used for *robustness*. In the “[Field Testing](#)” section to be described later, we comment about an alternative adaptive approach.

Hence, assuming the noise is IID Laplacian (see [Section 4.6](#)), the PDF is

$$p(w) = \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\sqrt{\frac{2}{\sigma^2}}|w|\right) \quad -\infty < w < \infty$$

where  $\sigma^2$  is the noise variance. The NP upper bound detection statistic then becomes

$$\ln L(\mathbf{x}) = - \sum_{n=0}^{N-1} \sqrt{\frac{2}{\sigma^2}} (|x[n] - s[n]| - |x[n]|)$$

or equivalently we can use the NP detection statistic

$$T_{NP}(\mathbf{x}) = \sum_{n=0}^{N-1} (|x[n]| - |x[n] - s[n]|). \quad (14.4)$$

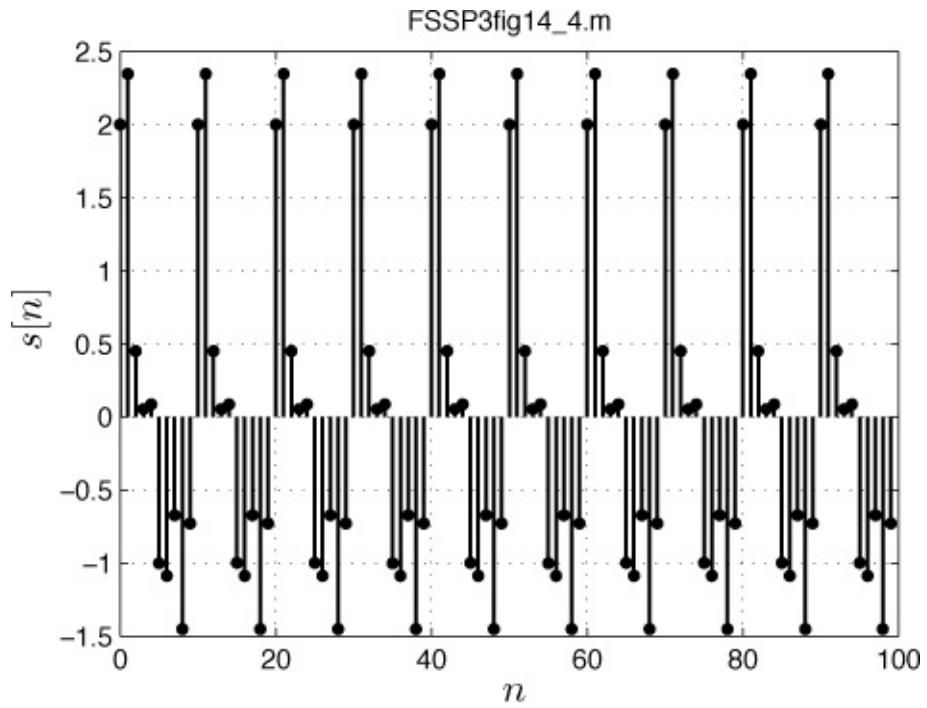
The currently employed detector, an energy detector, uses the detection statistic

$$T_{ED}(\mathbf{x}) = \sum_{n=0}^{N-1} x^2[n]. \quad (14.5)$$

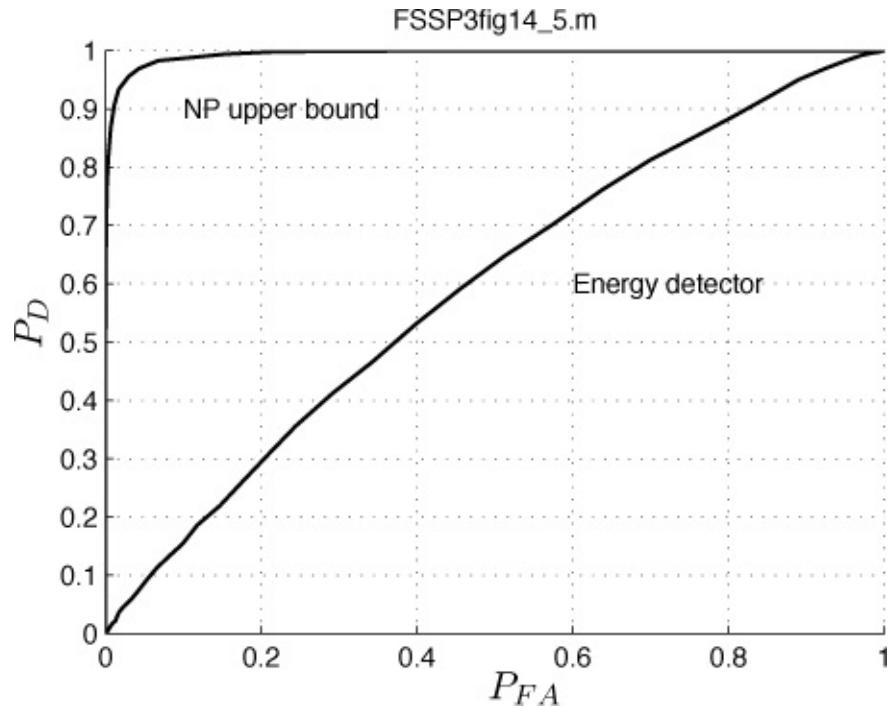
Using the approach described in [Section 4.6](#) to generate IID Laplacian noise, we can compare the detection performance of the upper bound NP detector given by [\(14.4\)](#) to that of the energy detector given by [\(14.5\)](#). To do so we choose the signal to be

$$s[n] = \sum_{k=1}^3 a_k \cos(2\pi k(0.1)n) + \sum_{k=1}^3 b_k \sin(2\pi k(0.1)n) \quad n = 0, 1, \dots, N-1$$

where  $a_1 = b_1 = 1$ ,  $a_2 = b_2 = 0.5$ , and  $a_3 = b_3 = 0.5$ . The signal is shown in [Figure 14.4](#). For a noise variance of  $\sigma^2 = 20$  a computer simulation yields the receiver operating characteristics shown in [Figure 14.5](#). It is seen that there is a potential for greatly improved performance. Of course, we do not expect to attain this upper bound because in practice we will have to estimate all the signal parameters.



**Figure 14.4: Periodic signal used to determine NP upper bound performance.**



**Figure 14.5: Receiver operating characteristics for the Neyman-Pearson upper bound and the energy detector.**

## 7. Specification Modifications (if necessary)

Not applicable.

## 8. Proposed Solution

The limiter-replica correlator for detection in IID nonGaussian noise that was described as [Algorithm 10.2](#) only applies to the case of a known signal (and actually only to the detection of a “weak” known signal). For the problem at hand it is unrealistic to assume a known signal. Our model, however, is a partially linear one for which a detection approach is known. It is shown in [[Kay 1998](#), pg. 400] that a variant of the GLRT (see [Section 8.5.2](#)), called the Rao test, is directly applicable to our problem. It differs from the GLRT in that *we do not have to estimate the signal amplitude parameters*. This property of the Rao detector is especially important when dealing with nonGaussian noise.

Determining the MLE of the signal amplitude parameters for the case of nonGaussian noise can be quite complex, requiring a nonlinear optimization, and so we prefer to avoid this difficulty. The asymptotic (large data record) performance of the Rao detector is comparable to the GLRT.

The Rao test for our problem produces the detection statistic (to within a positive scale factor which will not change its performance)

$$T_{Rao}(\mathbf{x}) = \max_{f_0} \mathbf{y}^T \mathbf{H}(f_0) (\mathbf{H}^T(f_0) \mathbf{H}(f_0))^{-1} \mathbf{H}^T(f_0) \mathbf{y}$$

where  $\mathbf{y} = [y[0] \ y[1] \ \dots \ y[N - 1]]^T$  and  $y[n] = g(x[n])$ . The function  $g(\cdot)$  is the same nonlinearity used in the limiter-replica-correlator (see [Algorithm 10.2](#)) and is given as

$$g(w) = -\frac{d \ln p(w)}{dw} \quad (14.6)$$

where  $p(w)$  is the nonGaussian noise PDF. The matrix  $\mathbf{H}(f_0)$  has dimension  $N \times 6$  and is given in [\(14.2\)](#). For our problem we had assumed that the fundamental frequency was such that  $f_0 >> 1/N$ , which makes the columns of  $\mathbf{H}(f_0)$  approximately orthogonal, as illustrated in [Exercise 14.2](#). It can be shown therefore that  $\mathbf{H}^T(f_0) \mathbf{H}(f_0) \approx (N/2) \mathbf{I}$ , resulting in the detection statistic

$$T_{Rao}(\mathbf{y}) = \frac{2}{N} \max_{f_0} \mathbf{y}^T \mathbf{H}(f_0) \mathbf{H}^T(f_0) \mathbf{y}.$$

Since

$$\mathbf{H}^T(f_0)\mathbf{y} = \begin{bmatrix} \sum_{n=0}^{N-1} y[n] \cos(2\pi f_0 n) \\ \sum_{n=0}^{N-1} y[n] \cos(4\pi f_0 n) \\ \sum_{n=0}^{N-1} y[n] \cos(6\pi f_0 n) \\ \sum_{n=0}^{N-1} y[n] \sin(2\pi f_0 n) \\ \sum_{n=0}^{N-1} y[n] \sin(4\pi f_0 n) \\ \sum_{n=0}^{N-1} y[n] \sin(6\pi f_0 n) \end{bmatrix} \quad (14.7)$$

we have that

$$\mathbf{y}^T \mathbf{H}(f_0) \mathbf{H}^T(f_0) \mathbf{y} = [(\mathbf{H}^T(f_0) \mathbf{y})^T (\mathbf{H}^T(f_0) \mathbf{y})] = \|\mathbf{H}^T(f_0) \mathbf{y}\|^2$$

where  $\|\mathbf{x}\|^2$  denotes the sum of squares of the vector  $\mathbf{x}$  (its squared-length). Thus, the result is the sum of the squares of the elements of  $\mathbf{H}^T(f_0)\mathbf{y}$ . By rearranging the elements in the  $6 \times 1$  vector in (14.7), we have

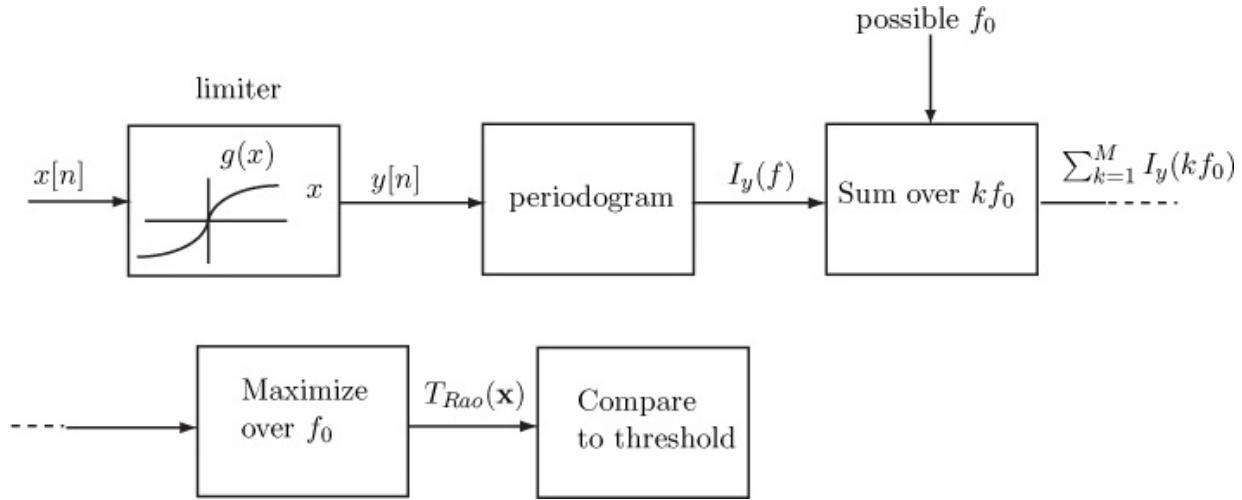
$$\begin{aligned} \mathbf{y}^T \mathbf{H}(f_0) \mathbf{H}^T(f_0) \mathbf{y} &= \sum_{k=1}^3 \left[ \left( \sum_{n=0}^{N-1} y[n] \cos(2\pi k f_0 n) \right)^2 \right. \\ &\quad \left. + \left( \sum_{n=0}^{N-1} y[n] \sin(2\pi k f_0 n) \right)^2 \right] \\ &= \sum_{k=1}^3 \left| \sum_{n=0}^{N-1} y[n] \exp(-j2\pi k f_0 n) \right|^2. \end{aligned}$$

Finally, we have that the detection statistic is (scaling by 1/2)

$$T_{Rao}(\mathbf{y}) = \max_{f_0} \sum_{k=1}^3 \underbrace{\frac{1}{N} \left| \sum_{n=0}^{N-1} y[n] \exp(-j2\pi k f_0 n) \right|^2}_{I_y(k f_0)} \quad (14.8)$$

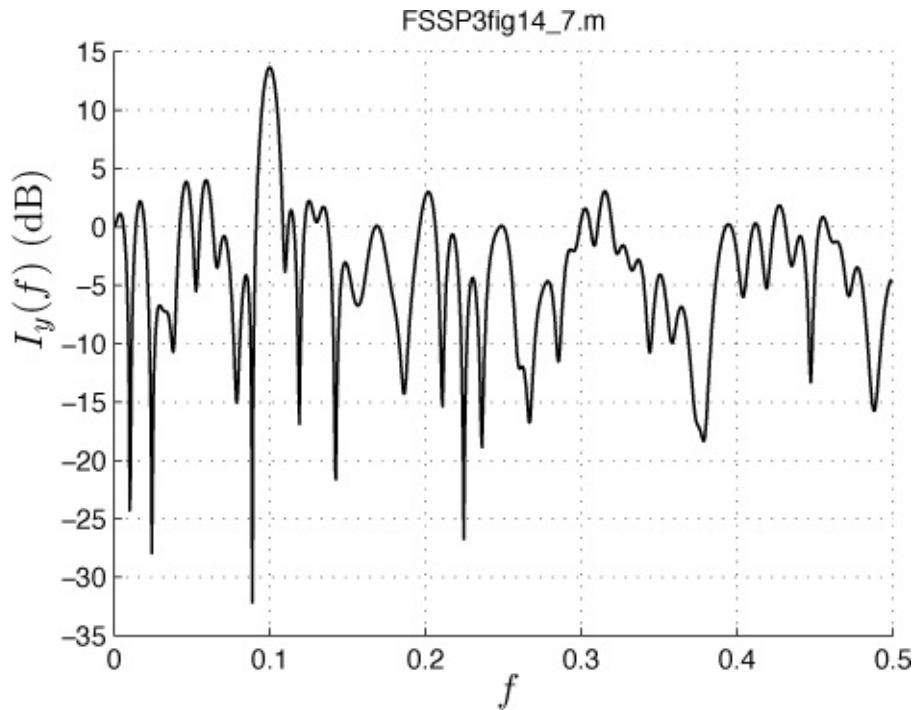
where  $I_y(k f_0)$  is the periodogram of the data at the output of the limiter and evaluated at the frequencies  $f = k f_0$ . The resultant detector is summarized in [Figure 14.6](#). It has a nice intuitive explanation. Because the noise is IID nonGaussian, it employs a limiter to suppress the high level noise spikes. Next, because each sinusoidal component of the signal has an unknown amplitude and phase (equivalent to not knowing  $a_k$  and  $b_k$ ) the periodogram is computed. This

is identical to the operation of the detector that must detect a sinusoid of unknown amplitude and phase (see [Algorithm 10.7](#)). Since the sinusoidal components have frequencies that are “well resolved”, with their spacing exceeding  $1/N$  (there is no “interaction” between the sinusoidal components), we sum up the values of the periodogram. Finally, we do not actually know the fundamental frequency, so we compute the sum of the periodogram values for all possible values of  $f_0$ , and choose the statistic with the maximum over  $f_0$ . This can be shown to be equivalent to a “comb filter” and energy detector [[Kay 1998](#), pp. 321–325].



**Figure 14.6: Rao detector for a periodic signal in IID nonGaussian noise. The Fourier series coefficients as well as the fundamental frequency of the signal are assumed to be unknown.**

As an example, the periodogram of  $y[n]$ , for the periodic signal shown in [Figure 14.4](#) and a typical realization of the Laplacian noise with  $\sigma^2 = 1$ , is shown in [Figure 14.7](#). It can be seen that there is a strong peak at  $f = 0.1$  and a smaller one at  $f = 0.2$ . The peak at  $f = 0.3$  does not appear.



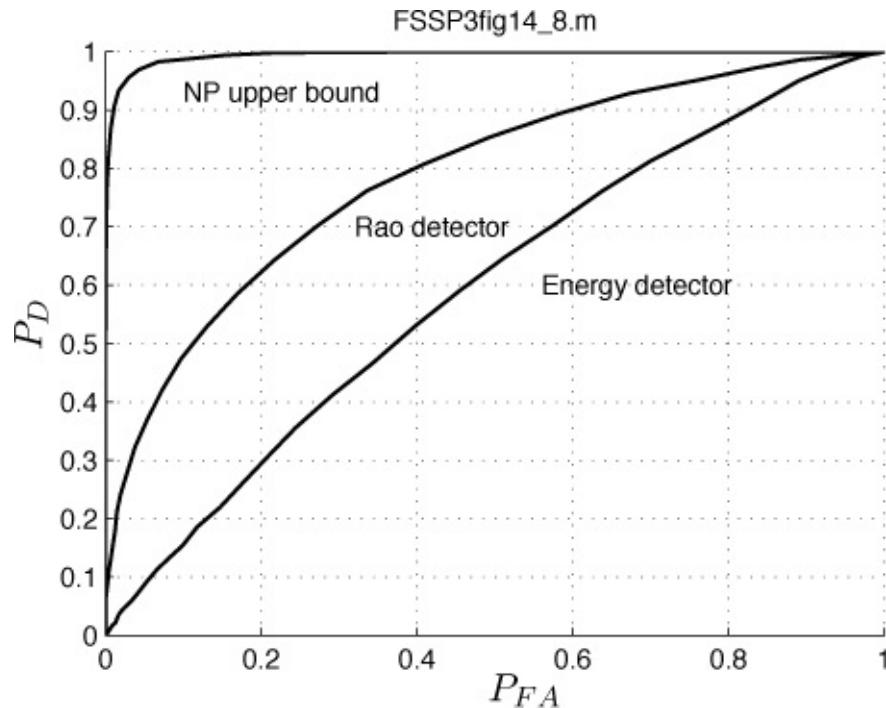
**Figure 14.7: Periodogram of periodic signal shown in [Figure 14.4](#) embedded in Laplacian noise after the limiter.**

## 9, 10. Performance Analysis

We now revisit the detection of the periodic signal shown in [Figure 14.4](#) in IID Laplacian noise, using  $N = 100$  data samples. Repeating the same computer simulation that produced the receiver operating characteristics (ROC) in [Figure 14.5](#), we implement the Rao detector under the same conditions. The limiter can be shown to be [[Kay 1998](#), pg. 391–392]

$$g(x) = \sqrt{\frac{2}{\sigma^2}} \operatorname{sgn}(x)$$

where  $\operatorname{sgn}(x) = 1$  for  $x > 0$  and  $\operatorname{sgn}(x) = -1$  for  $x < 0$ . For the Rao detector it is assumed that the fundamental frequency is known to lie within the interval  $0.09 \leq f_0 \leq 0.11$ . This produces a minimum spacing between the frequencies of 0.09 which is much greater than  $1/N = 0.01$ , justifying our assumption in the Rao detector derivation. The ROC for the Rao and energy detectors are shown in [Figure 14.8](#), along with the NP upper bound. The MATLAB program FSSP3fig14\_8\_exer.m, which is contained on the CD, has been used to produce this figure. It can be consulted for the details of the computer simulation, the algorithm implementation, and the performance results.



**Figure 14.8: Receiver operating characteristics for the Neyman-Pearson upper bound, Rao detector, and the energy detector.**

---

### Exercise 14.3 – Effect of the limiter

Using the MATLAB program FSSP3fig14\_8\_exer.m, which was used to produce [Figure 14.8](#), rerun the performance evaluation but omit the limiter for the Rao detector. Compare your results against those shown in [Figure 14.8](#), especially at a small  $P_{FA}$ . In particular, examine what happens at  $P_{FA} = 0.1$ . Does it make a difference in the performance? Hint: Use the command `axis([0 0.1 0 1])` to plot the ROC for  $0 \leq P_{FA} \leq 0.1$ . (Using actual data, the improvement in detection performance due to the limiter was found to be about 2 dB. This means that the detector using a limiter could detect a signal that had 2 dB less power than that of a detector not using a limiter.)



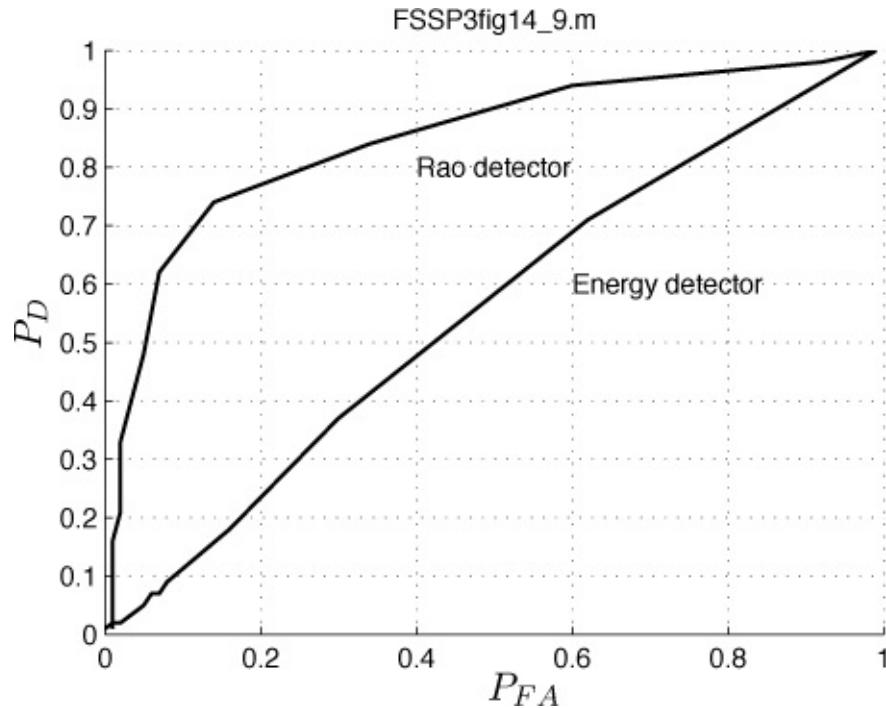

---

## 11, 12. Sensitivity Analysis

The principal assumption that was made for the ease of obtaining a detector was to ignore the coloration of the noise. Hence, it is important to see if the detector performance is sensitive to this assumption. Unfortunately, even generating

nonGaussian noise with a prescribed PDF and a nonflat PSD is a difficult task. One recent way that has been proposed to do this can be found in [Kay 2010]. It was not available during the course of the project described herein. In lieu of having this method available, an alternative approach uses the actual data provided and shown in [Figure 14.1](#) to perform a detection experiment.

We therefore repeat the example that yielded [Figure 14.8](#) except that we replace the computer generated IID Laplacian noise with the actual field data noise. In doing so, we will need more samples than the 10,000 shown in [Figure 14.1](#) since this data will produce only  $10000/N = 100$  blocks of data, and therefore, only 100 outcomes of the detection statistics. There were available 100,000 samples and so for the same data record length of  $N = 100$  samples used previously, we can break up the 100,000 samples into 1000 blocks. We then use each 100 sample block as the noise only realization, scaling the noise samples to have a variance of  $\sigma^2 = 20$  as before, and also add in the computer generated signal for the signal plus noise realization. Then we apply both the energy and Rao detectors and finally compute the ROCs. Since we will have only 1000 detector statistics (each of the 100 blocks produces one detector statistic), we limit the number of thresholds used in the ROC to 20. This accounts for the piece-wise linear appearance of the generated ROC as shown in [Figure 14.9](#). It is seen that the improvement in performance is maintained. In fact, the improvement appears to be even greater than in [Figure 14.8](#). This is attributed to the very spiky nature of the geomagnetic noise seen in [Figure 14.1](#) and its estimated PDF shown in [Figure 14.3](#). Recall that for the geomagnetic noise the standard deviation was estimated to be about  $\sigma = 0.2/3$  (see [Figure 14.3](#)). Thus, as seen in [Figure 14.3](#) there are high level events that are about  $0.6 = 9\sigma$  away from the mean. The Laplacian PDF used for the simulation that resulted in [Figure 14.8](#) would not exhibit such extreme events. Hence, the nonGaussianity of the geomagnetic noise appears to be much greater than for Laplacian noise and hence the improved performance seen in [Figure 14.8](#).



**Figure 14.9: Receiver operating characteristics for the Rao detector and the energy detector using actual field data.**

The results obtained from utilizing the field data lends credibility to the argument that the noise coloration may be ignored. Clearly, however, further testing is needed.

### 13, 14, 15. Field Testing

The results shown in [Figure 14.9](#) illustrate a preliminary approach to field testing. Actual noise data was used, although with a computer generated signal. For further verification of the efficacy of the proposed detector, data that includes a real target signal would need to be analyzed.

One issue that needed to be addressed but was not completed was the question of what noise PDF should be assumed and hence what limiter should be used.

Should we keep the sign limiter or maybe even make the limiter data-adaptive? For the latter the system would have to estimate the PDF in real time and then implement the limiter as per [\(14.6\)](#). If an adaptive limiter is to be chosen, then a PDF estimate that can be differentiated is needed. One possibility was described in [Section 6.4](#) as the autoregressive (AR) PDF estimator. The advantage of using this is that the PDF estimator is a smooth curve with an analytical form that can be easily differentiated. Another advantage that might become a critical one, occurs when the noise is very nonstationary. For example, if the noise PDF

changes over a duration of say 300 samples, then it would need to be updated every few hundred samples. This means that only a few hundred samples would be available with which to estimate the PDF. As we have seen from [Figure 14.3](#) even 10,000 samples may not be adequate if a histogram is used. An AR PDF estimator might be used since it only requires a few parameters, possibly 10 or less, and so would be able to provide good estimates with only a few hundred data samples. Note that in practice the procedure would be to estimate the noise PDF from the previous samples, say a data window of 300 samples and then apply that PDF estimate to detecting the signal in the *next block of 100 samples*. In essence, a sliding window approach would be used (see also [Section 4.5](#) for a more general discussion of nonstationary noise).

For the AR PDF estimator it can be shown that if the data is first normalized so that its values lie in the interval  $-1/2 \leq x \leq 1/2$ , then by estimating the AR PDF and computing the form of the limiter, we have that

$$g(x) = \frac{4\pi}{|A(x)|^2} \operatorname{Re} \left[ jA(x) \sum_{k=1}^p ka[k] \exp(j2\pi kx) \right] \quad (14.9)$$

where

$$A(x) = 1 + \sum_{k=1}^p a[k] \exp(-j2\pi kx).$$

The AR filter parameters are real since the noise PDF is assumed to be even ( $p(-w) = p(w)$ ). The author's involvement in the project ended before this could be tested.

---

### **Exercise 14.4 – Adaptive limiter**

Show that the AR derived limiter for  $p = 1$  is given by

$$g(x) = -\frac{4\pi a[1] \sin(2\pi x)}{|A(x)|^2}.$$

Next for  $a[1] = -0.95$  and also for  $a[1] = -0.8$  plot  $g(x)$ .




---

### **14.3. Lessons Learned**

- Always try to make signal and noise assumptions that we know *beforehand* will allow us to use one of the known solutions to detection

problems.

- In analyzing data it is good practice to use more than one means of analysis. For example, in spectral estimation several estimators can be tried to make sure that any peculiar results obtained are not artifacts of the estimation method.
- The Rao detector is especially useful when we have unknown signal amplitude parameters for a nonGaussian detection problem. It avoids the difficult problem of finding the MLE of the signal amplitude parameters.
- The use of a limiter for nonGaussian noise leads to improved detection performance, the improvement depending upon how heavy the tails of the nonGaussian PDF are. Increases of up to 20 dB or more are possible [[Kay 1998](#), pg. 404].
- When actual noise data is available, it can be advantageous to run detection experiments using this data with computer simulated signals. This allows some realism to be injected into the experiment while at the same time maintaining the flexibility of changing the signal parameters. In practice, it is sometimes difficult to obtain signal data under various operating conditions.
- For nonstationary noise environments parametric estimators, ones that need only estimate a few parameters, are preferable to nonparametric ones *as long as the parametric model is reasonably accurate*. Such is the case for PDF estimation of nonstationary noise samples.

## References

- Kay, S., *Fundamentals of Statistical Signal Processing: Estimation Theory, Vol. I*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Kay, S., *Fundamentals of Statistical Signal Processing: Detection Theory, Vol. II*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
- Kay S., “Representation and Generation of NonGaussian Wide Sense Stationary Random Processes with Arbitrary PSDs and a Class of PDFs”, *IEEE Trans. on Signal Processing*, July 2010.

## Appendix 14A. Solutions to Exercises

To obtain the results described below initialize the random number generators to `rand('state', 0)` and `randn('state', 0)` at the beginning of any code. These commands are for the uniform and Gaussian random number generators,

respectively.

**14.1** The standard deviation is estimated as  $\hat{\sigma} = 9.5501 \times 10^{-13}$ . By plotting the entire data record of 10,000 samples (after subtracting its mean), you will see that many of the samples fall outside the range of  $-3\hat{\sigma} \leq w \leq 3\hat{\sigma}$ . You can run the program `FSSP3exer14_1.m`, which is contained on the CD, to obtain the results.

**14.2** You should obtain a  $6 \times 6$  matrix that is an approximate identity matrix to a few decimal places. You can run the program `FSSP3exer14_2.m`, which is contained on the CD, to obtain the results.

**14.3** Run `FSSP3fig14_8_exer.m` but omit the limiter. You will notice that for the Rao detector at  $P_{FA} = 0.1$ ,  $P_D$  decreases from about 0.5 to 0.4. You can run the program `FSSP3exer14_3.m`, which is contained on the CD, to obtain the results.

**14.4** Using (14.9) with  $p = 1$  produces

$$\begin{aligned} g(x) &= \frac{4\pi}{|A(x)|^2} \operatorname{Re} [jA(x)a[1] \exp(j2\pi x)] \\ &= \frac{4\pi a[1]}{|A(x)|^2} \operatorname{Re} [j(1 + a[1] \exp(-j2\pi x)) \exp(j2\pi x)] \\ &= \frac{4\pi a[1]}{|A(x)|^2} \operatorname{Re} [j(\exp(j2\pi x) + a[1])] \\ &= -\frac{4\pi a[1] \sin(2\pi x)}{|A(x)|^2} \end{aligned}$$

since  $a[1]$  is assumed to be real. The limiters are seen to be odd functions that pass linearly low level amplitudes but attenuate higher level amplitudes. You can run the program `FSSP3exer14_4.m`, which is contained on the CD, to obtain the results.

# Chapter 15. Case Studies - Spectral Estimation Problem

## 15.1. Introduction

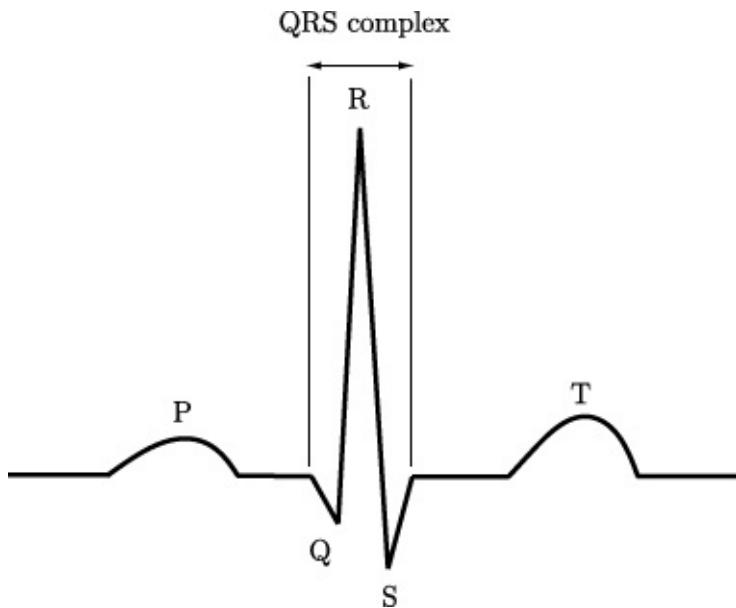
In this chapter we discuss the use of spectral analysis to obtain information about the data of interest. Since spectral analysis is a preliminary data analysis tool, we do not investigate in detail an actual algorithm but instead use our analysis *to formulate a noise power spectral density (PSD) model*. Other necessary information about the noise such as its probability density function (PDF) is not studied. We do describe, however, the use of a matched filter to improve the data quality and why it ultimately proved unsuccessful for the heart rate monitor problem. In contrast to the previous two chapters in which we followed the flow chart of [Figure 2.1](#) to design an actual algorithm, we now relegate our discussion to the PSD modeling of the noise.

The particular problem that we will discuss is that of designing a heart rate monitor for exercise equipment. This is a research project that the author has recently been involved. The customer would be commercial manufacturers of exercise equipment, with other potential applications to accurate heart rate monitoring to improve peak performance of athletes and to assess heart rate functions for cardiac impaired patients. More specifically, it is customary for modern exercise equipment such as stationary bicycles, elliptical cross-trainers, etc. to have built-in heart rate monitors so that a person exercising can instantly obtain a readout of his/her heart rate in beats per minute (BPM). To do so the standard equipment employs metallic hand contacts that allow a person, while exercising, to place their hands on the contacts to obtain the reading.

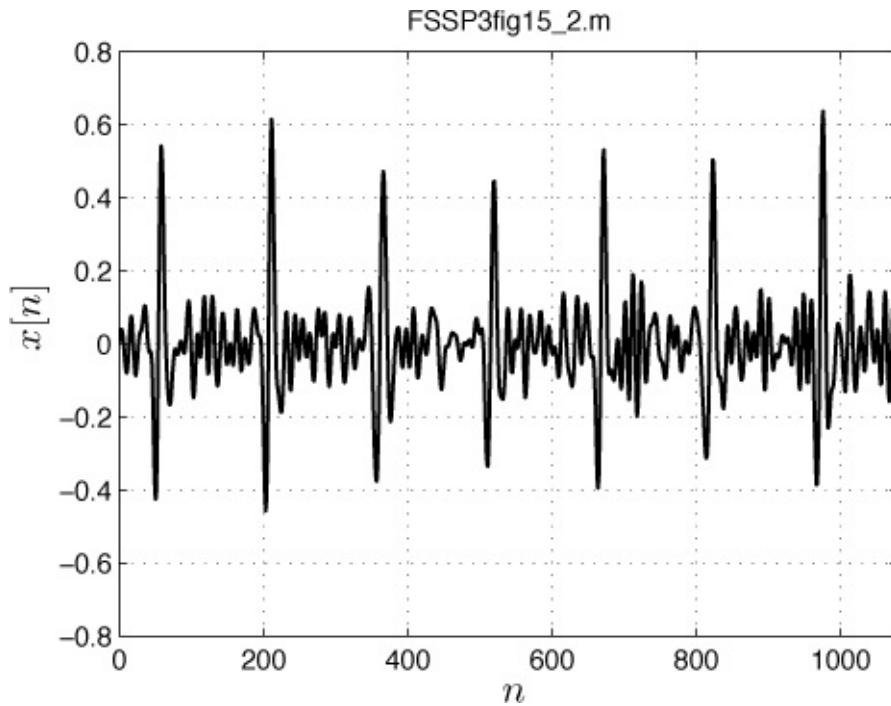
Unfortunately, the hand-hold contacts produce a poor electrical sensing mechanism of the heart beat waveform due to the low voltages of the signal, having been propagated from the heart to the hands, as well as high noise levels due to increased muscle nerve firing rates. This low signal-to-noise ratio (SNR) creates a particularly difficult problem in which detection of the heart beat waveform for translation into heart rate has produced unreliable results in current commercial machines.

The ideal signal waveform of a generic heart beat is shown in [Figure 15.1](#). It exhibits several waves, but for heart rate estimation the most important portion of the waveform is the QRS portion, usually referred to as the *QRS complex*. In a clinical setting, this high-level signal is easily detected using a standard

electrocardiogram (ECG) recording device on a patient at rest. A typical portion of an ECG recording that has been obtained from a subject while exercising is shown in [Figure 15.2](#). It was obtained by attaching chest lead electrodes to the subject, and then acquiring data while the subject exercised on an elliptical cross-trainer. The analog waveform was first filtered using a lowpass filter and then sampled at a sampling rate of  $F_s = 360$  samples/sec to produce the 1080 sample data record shown. Next the digital data was filtered with a digital linear phase finite impulse response (FIR) bandpass filter having a passband of 9 Hz to 39 Hz. This band is a typical one for heart rate data, and the use of the linear phase FIR filtering is to ensure that the QRS waveform was not distorted due to filter phase distortion. It is seen that the QRS waveform is easily discerned, but there is also some lower level noise. The heart rate is found from the figure to correspond to a period of  $519 - 366 = 153$  samples. For the sampling rate of  $F_s = 360$ , this is a period of 0.425 seconds or the beats per minute is  $BPM = 60/0.425 = 141.2$ .

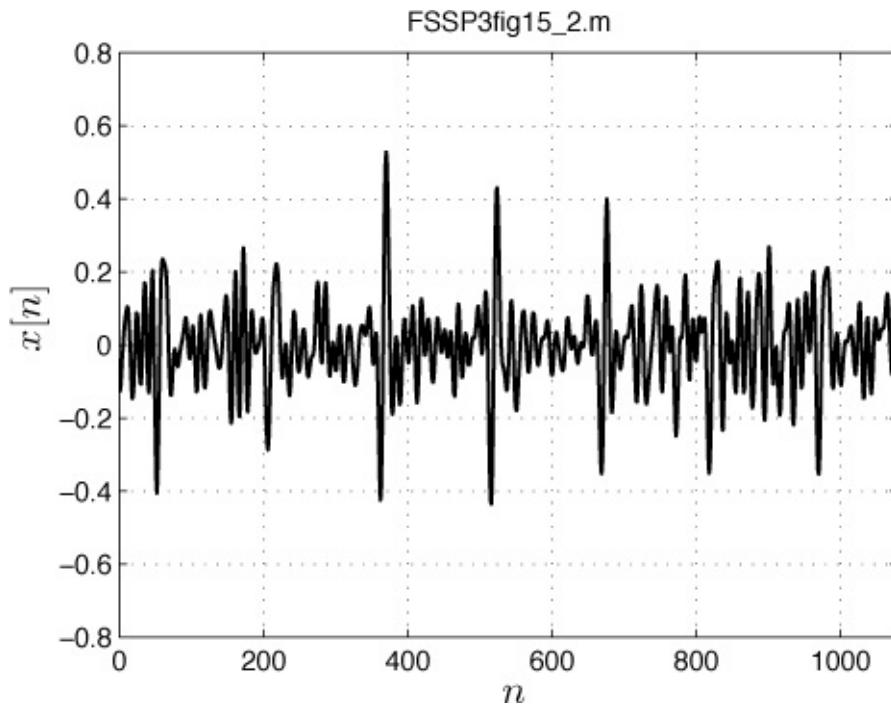


**Figure 15.1: Generic heart beat waveform.**



**Figure 15.2: Heart rate data obtained from chest electrodes. The data samples have been connected with straight lines for easier viewing.**

The heart rate waveform obtained from the hand-hold metal contacts for the same period of time is shown in [Figure 15.3](#). Note that some of the QRS complexes are missing and there is increased noise. This is mainly due to muscle noise. The question arises if it is possible to mitigate the effects of the noise. Some of the methods that come to mind are to implement a generalized matched filter to detect each QRS complex if a signal replica could be obtained (see [Algorithm 10.3](#)) or to extract the signal from the noise using a Wiener filter, assuming the signal is a periodic random process with known PSD (see [Algorithm 9.12](#)).



**Figure 15.3: Heart rate data obtained from hand-hold contacts. The data samples have been connected with straight lines for easier viewing.**

---

### Exercise 15.1 – Why can't we use a periodogram?

In [Chapter 14](#) the proposed detector for a periodic signal in nonGaussian noise with a flat PSD utilized a periodogram. Indeed if the signal waveform is purely periodic, as it might be supposed for a regular heartbeat, then we would expect to see peaks in the spectrum at the fundamental and harmonic frequencies. Here the period is  $1/F_0 = 0.425$  seconds so that the fundamental frequency component is at  $1/0.425 = 2.35$  Hz. Within the passband of 9 to 39 Hz we would expect to see peaks every 2.35 Hz or at  $4F_0 = 9.40$ ,  $5F_0 = 11.75$ ,  $6F_0 = 14.10$ ,  $7F_0 = 16.45$ , etc. within the passband. Execute the MATLAB program FSSP3exer15\_1.m, which computes and plots the periodogram in dB of the data shown in [Figure 15.2](#). This program uses the high SNR data obtained using the chest electrodes. Do you see peaks at the appropriate frequency locations? What else do you see? Do you think this would work for the lower SNR data shown in [Figure 15.3](#)? Try replacing in the program the chest data with the hand-hold data of [Figure 15.3](#) (see code for how to do this).

---

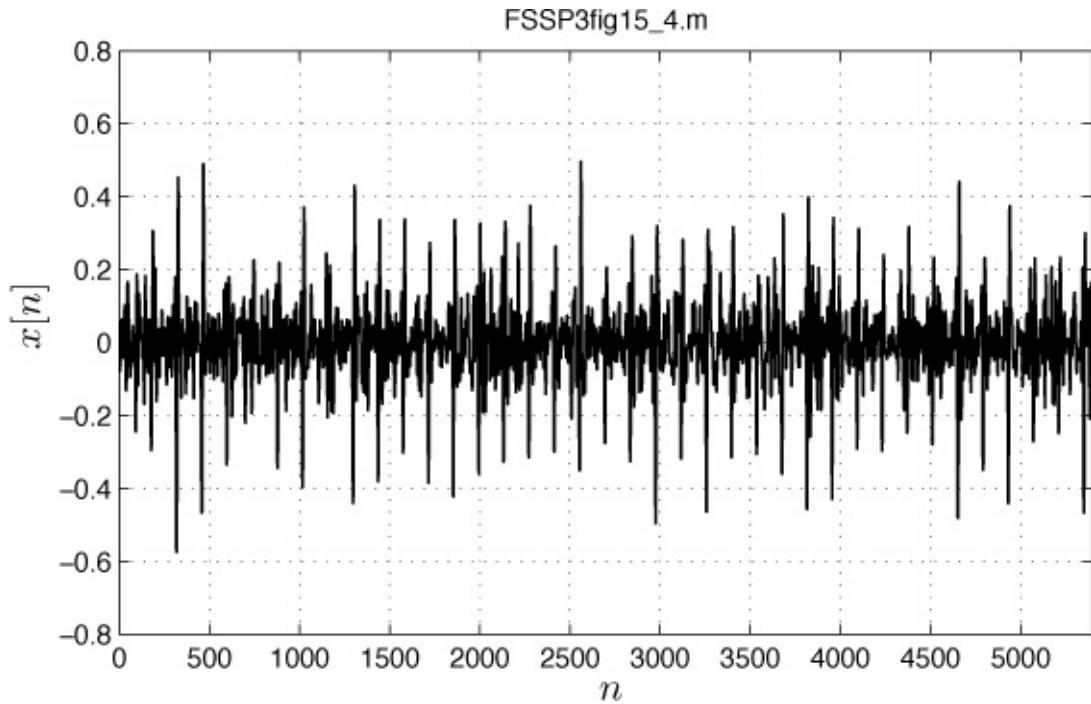
You should have observed from [Exercise 15.1](#) that due to the excessive noise it is not possible to reliably extract the peaks of the periodogram. In order to mitigate the noise, we first need to know the noise PSD over the signal bandwidth. This is because both a generalized matched filter and a Wiener filter require knowledge of the noise PSD. A spectral analysis of the noise is therefore necessary. In the next section we describe how this is done.

## 15.2. Extracting the Muscle Noise

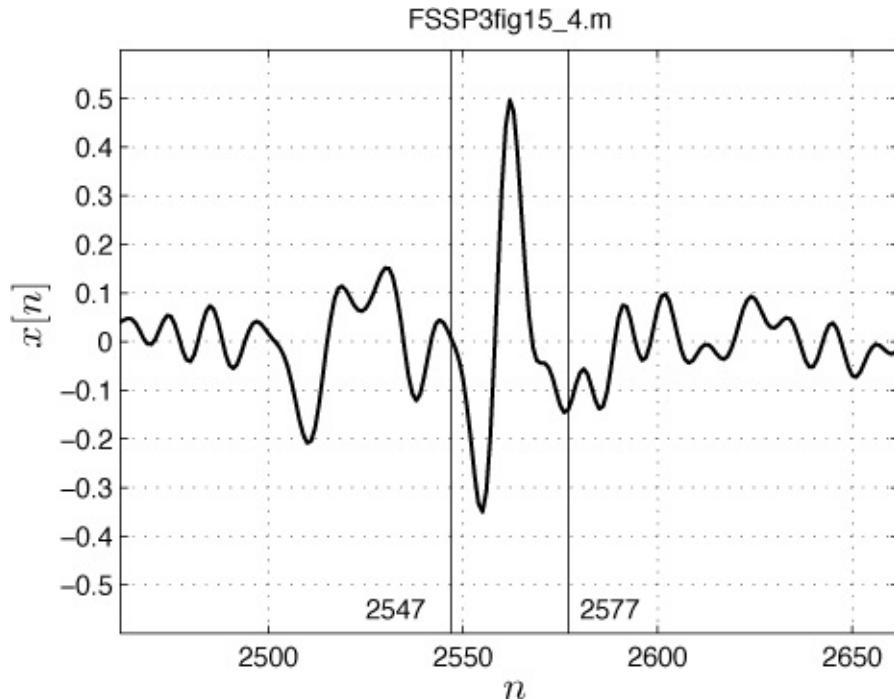
The 3 seconds of data shown in [Figure 15.2](#) is about the length of the data record that an actual algorithm might use to produce a timely heart rate estimate. In order to estimate the PSD of the noise, however, we will need more data.

Consider a 15 second data record obtained from the hand-hold contacts as shown in [Figure 15.4](#). With a heart rate period of about  $P = 143$  samples (as measured from the data by visual observation), this provides about  $5400/143 \approx 38$  segments of noise data. Thus, if we can extract the QRS signal from the data, we could then use the remaining data to carry out a spectral analysis of the muscle noise. To do so the first thing we need to determine is the length in samples of the QRS complex. Typically, this signal duration is about 0.1 seconds, but can vary from person to person. We thus expect the duration to be about  $0.1(360) = 36$  samples. From [Figure 15.4](#) we see that there is a particularly strong QRS complex with a peak at sample location  $n = 2562$ . In [Figure 15.5](#) we plot this QRS waveform using 100 samples on either side of the peak. It is seen that the QRS complex, which is contained between the vertical lines shown, is about 30 samples in duration. The positioning of the vertical lines shown in [Figure 15.5](#) is somewhat arbitrary, and therefore we choose a duration of 35 samples. This choice ensures that the QRS complex will be extracted. Note that the duration is about what we expect. Next, to actually extract the QRS complexes, leaving the noise segments to use for spectral analysis, it is necessary to manually delete those samples in each interval  $[n_{peak} - 17, n_{peak} + 17]$ , where  $n_{peak}$  is the peak of each QRS complex seen in [Figure 15.4](#). This is done by visually placing a cursor over each peak and determining its location, using the MATLAB “data cursor” function. It is found that there are 38 QRS complexes, and thus these are edited out of the data of [Figure 15.4](#) by replacing the QRS complex samples by zeros. These stretches of zeros serve to delineate the different sections of noise-only data, making it convenient for spectral analysis of each section. The data shown in [Figure 15.6](#) consists of the original data of [Figure 15.4](#) but with the QRS complex samples replaced by zeros. We now have 38 segments of noise-only

data to use in our spectral analysis.

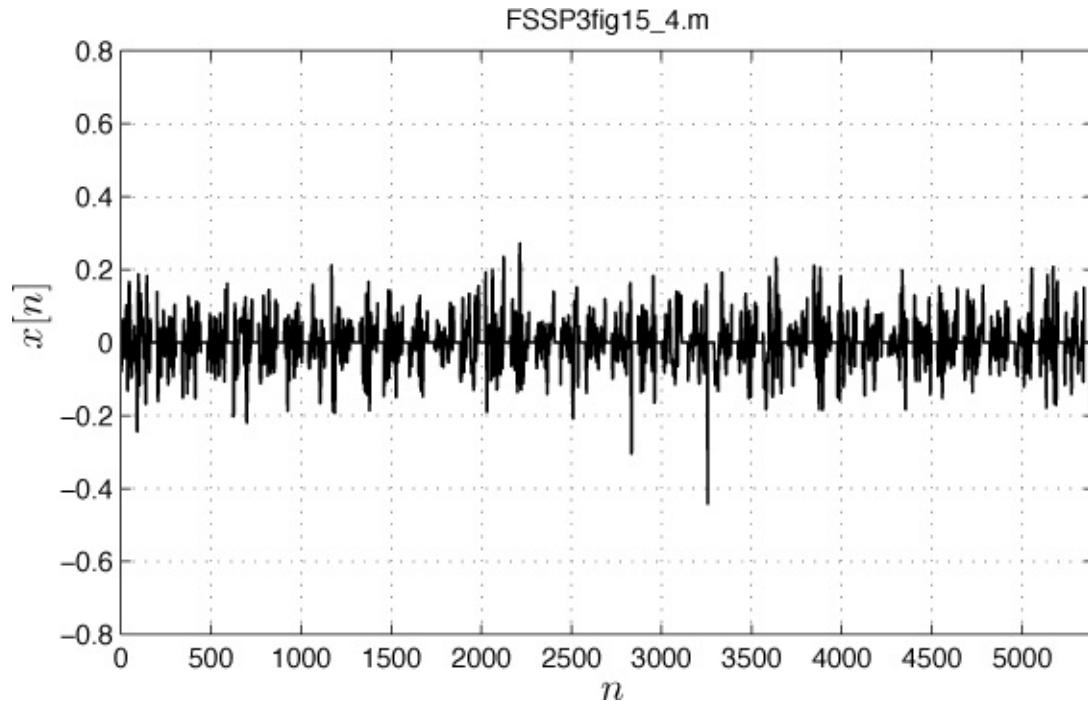


**Figure 15.4:** Heart rate data obtained from hand-hold contacts. The data samples have been connected with straight lines for easier viewing.



**Figure 15.5:** Heart rate data obtained from hand-hold contacts—zoomed in version showing the QRS complex, which peaks at  $n = 2562$  of [Figure 15.4](#).

**The data samples have been connected with straight lines for easier viewing.**



**Figure 15.6: Heart rate data obtained from hand-hold contacts with QRS complexes edited out and replaced by zeros. The data samples have been connected with straight lines for easier viewing.**

---

### Exercise 15.2 – Spectrum of QRS complex

The extracted QRS complex shown in [Figure 15.5](#) is seen to have about the correct duration of 0.1 sec. How about its spectrum? We had originally bandpass filtered the data to obtain the signal band of 9 to 39 Hz. Should it be even narrower in bandwidth, which would help reduce the noise? Load in the QRS complex data shown in [Figure 15.5](#) by using `load FSSP3exer15_2`. You will obtain an array `QRS_complex` of size  $35 \times 1$ . Then, find the periodogram of the data over the analog frequency interval  $0 \leq F < 90$  Hz. To do so this use the code

[Click here to view code image](#)

```
Fs=360; % sampling rate in samples/sec  
Nfft=4096; % length of FFT  
freq=[0:Nfft-1]'/Nfft; % frequencies at which FFT is computed,  
% 0 <= freq <1  
N=length(QRS_complex); % length of QRS complex
```

```

per=(1/N)*abs(fft(QRS_complex,Nfft)).^2; % compute periodogram
figure
plot(Fs*freq(1:Nfft/4),10*log10(per(1:Nfft/4))) % plot periodogram
% over discrete frequency range 0<= freq < 0.25 or analog frequency
% range 0 <= F < 90 Hz.
xlabel('F (Hz)')
ylabel('Periodogram (dB)')
grid

```

Is the entire band from 9 to 39 Hz needed to preserve the QRS complex?

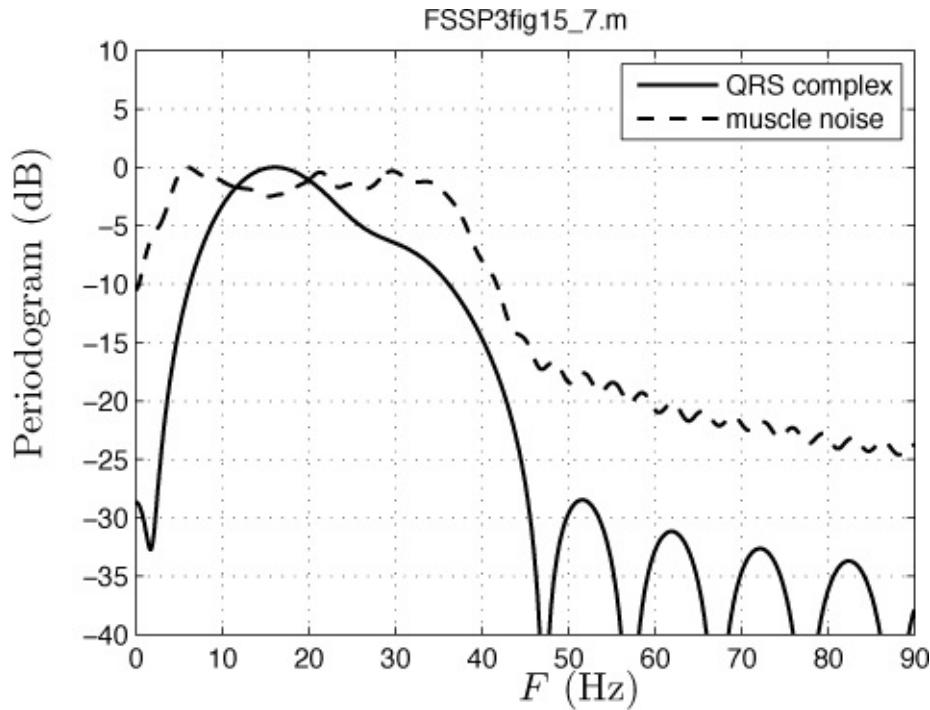
---

### 15.3. Spectral Analysis of Muscle Noise

We next use an averaged periodogram spectral estimator (see [Algorithm 11.1](#)) applied to the muscle noise data shown in [Figure 15.6](#). There are 38 segments of noise which we treat as blocks of data to be used in the spectral estimator. Observe that the resolution of the spectral estimator can be found by noting that the period between heart beats is about 143 samples. Of this data record the QRS complex comprises about 35 samples in duration, leaving  $L = 108$  samples, which consist of the muscle noise only. Thus, the resolution is about  $1/L = 0.00926$  cycles/sample or  $Fs/L = 3.33$  Hz (see [Chapter 11](#)). Thus, details in the PSD of the noise can be seen to within this resolution limit. Also, because we can average 38 periodograms together, it is possible to reduce the variability of the spectral estimator. Confidence limits can be placed on the spectral estimate as explained in the description of [Algorithm 11.1](#) (see (11.5)). A slight difference in this problem is that the segments need not all be of the same length. Recall that we edited out the QRS complex by deleting 35 points about each QRS peak. The peaks, however, will not in general be uniformly spaced since the heart rate does change over a period of a few seconds. This necessitates slight changes to the MATLAB program `PSD_est_avper.m`, which we have used previously.

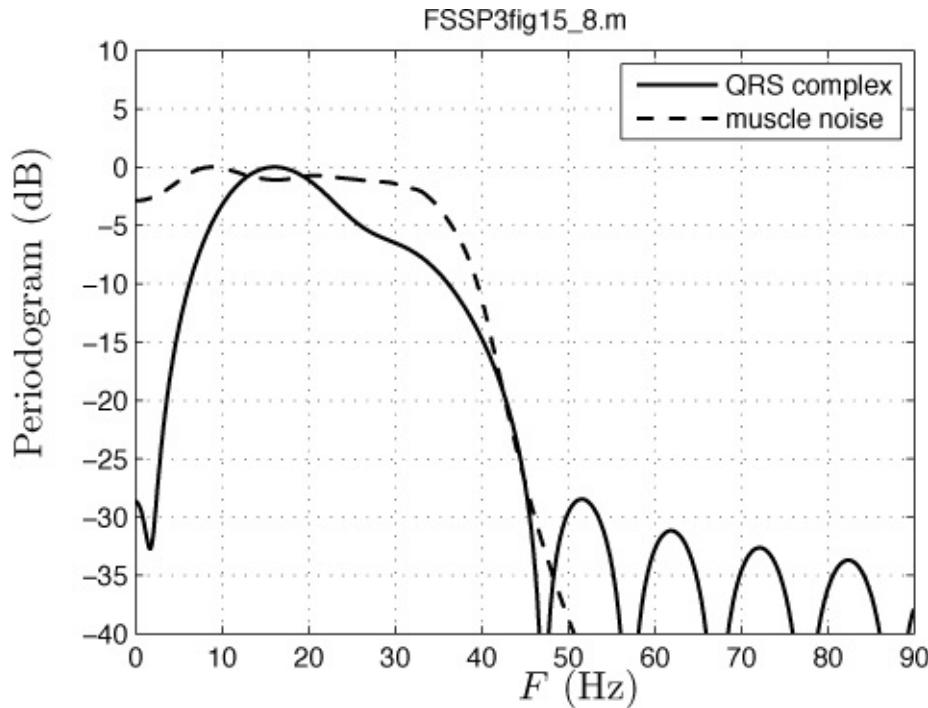
In [Figure 15.7](#) we show the averaged periodogram of the muscle noise (shown as the dashed curve) as well as the periodogram of the QRS complex (shown as the solid curve). To make the visual comparison easier we have normalized both PSDs to their maximum values, hence, the peak values for both are at 0 dB. It appears that the PSD of the muscle noise is fairly flat over the signal bandwidth. Hence, it is doubtful that the use of a generalized matched filter versus a standard matched filter, which assumes no noise coloration, would provide any improved performance. Similarly, the use of a Wiener filter would probably not be successful in extracting the signal waveform since both signal and noise

spectra are nearly the same.



**Figure 15.7: The periodogram of the QRS complex (solid curve) and averaged periodogram PSD estimate of the muscle noise (dashed curve).**

The question might be asked as to whether the resolution provided by the averaged periodogram is sufficient. Since the minimum variance spectral estimator (MVSE) has slightly better resolution, it might be worthwhile to employ it. Using an autocorrelation matrix of dimension  $20 \times 20$  produces the spectral estimate shown in [Figure 15.8](#). Again the PSD of the muscle noise over the signal band appears to be quite flat.



**Figure 15.8: The periodogram of the QRS complex (solid curve) and the MVSE PSD estimate of the muscle noise (dashed curve).**

The only conclusion that can be drawn is that over the signal band, the muscle noise should be modeled as white.

---

### Exercise 15.3 – Further confirmation using an AR spectral estimate

For the same 38 segments of muscle noise data shown in [Figure 15.6](#) we next try an AR spectral estimator. Run the program

FSSP3exer15\_3.m, which implements the AR spectral estimator using the covariance method with a model order of  $p = 10$ . You will see the result of the AR spectral estimate for each of the 38 segments plotted in an overlaid fashion as well as the average of the spectral estimates. What do you notice? Then, modify the program to replace the covariance method by the Burg method. How this is done is explained in the code. Recall that the Burg method constrains the estimated poles to be within the unit circle of the z-plane so that a pole cannot be on the unit circle. Now comment upon the possibility of peaks in the noise PSD. Are they actually there or are they due to the particular noise realization for each segment of muscle noise data?



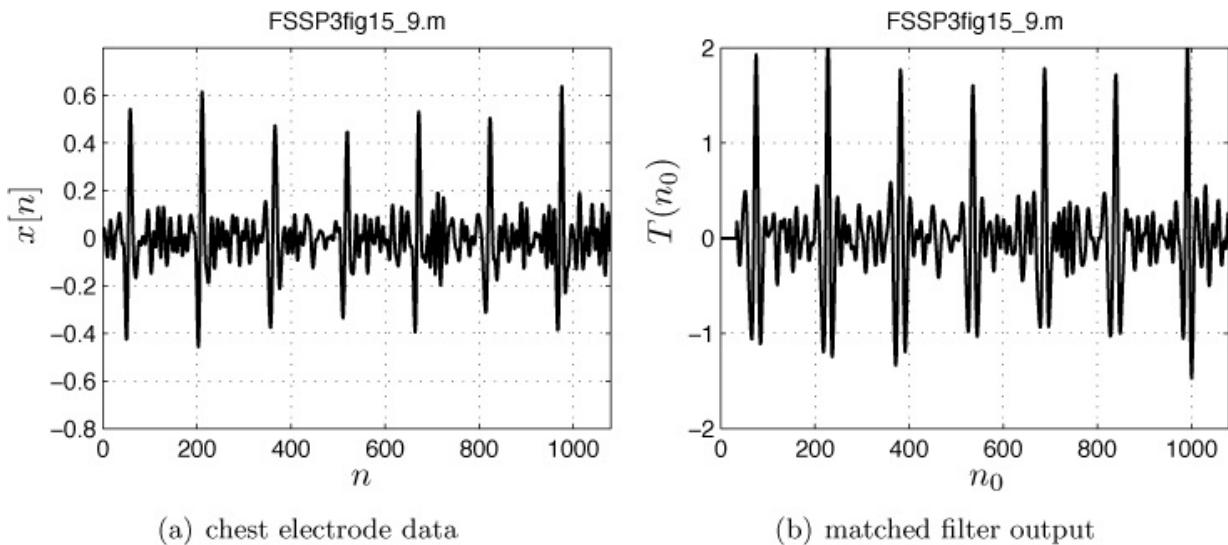
## 15.4. Enhancing the ECG Waveform

Although we are not designing an actual algorithm to estimate heart rate, it is of interest to see whether or not a matched filter could enhance the QRS complexes. If so, then it would simplify the task of designing a reliable heart rate estimation algorithm. Using a matched filter to enhance an ECG has been extensively explored with typical studies described in [[Sormmo et al. 1985](#)], [[Xue et al. 1992](#)], [[Ruha et al. 1997](#)].

As observed in [Figure 15.7](#) there is about a 15 dB dynamic range in the QRS spectrum. (Note that the maximum is at 0 dB and at  $F = 39$  Hz the spectrum is about -15 dB). Therefore, there are some differences between the signal and noise spectra within the signal band, although not large. In [Figure 15.9a](#) we show a 3 second segment of the chest electrode data (same data as shown in [Figure 15.2](#)). The QRS complexes are very evident. A matched filter of length  $M = 35$  samples using the previously measured QRS complex shown in [Figure 15.5](#), which consists of the peak with 17 samples taken on either side, is used as the replica. The matched filter, also called the replica-correlator, produces the output (see [Algorithm 10.9](#))

$$T(n_0) = \sum_{n=n_0}^{n_0+M-1} x[n]s[n - n_0]$$

for  $n_0 = 0, 1, \dots, N - M$ . The results of matched filtering are shown in [Figure 15.9b](#). There is some improvement, although the chest electrode data is already quite good.

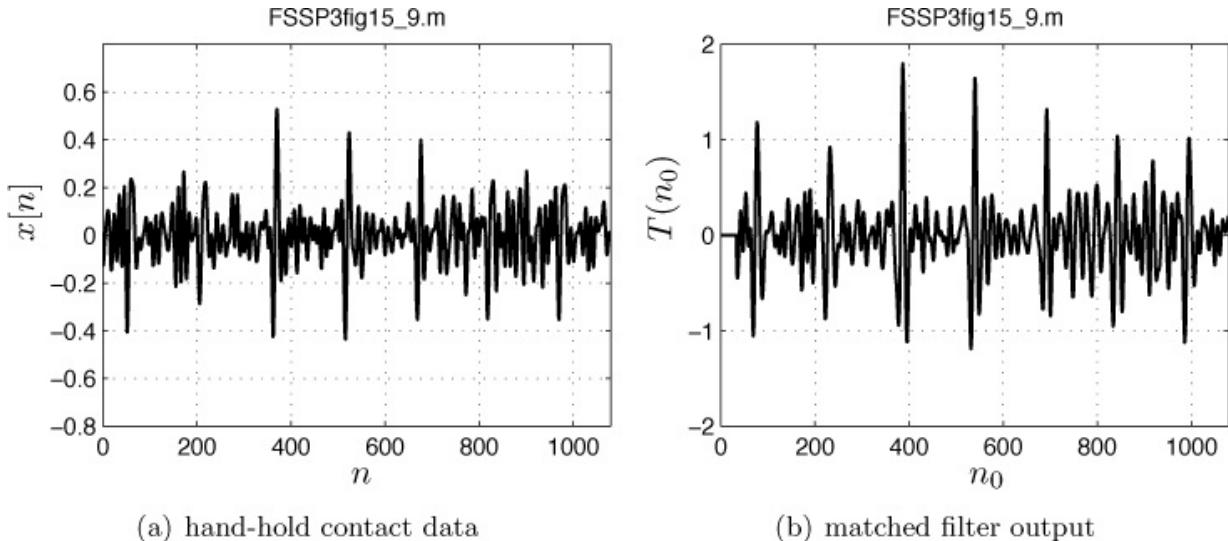


**Figure 15.9: Chest electrode data and matched filtered output. The data samples have been connected with straight lines for easier viewing.**

**Samples have been connected with straight lines for easier viewing.**

Next, applying the same matched filtering to the hand-hold data shown in [Figure 15.10a](#) (same data as shown in [Figure 15.3](#)) produces the output depicted in [Figure 15.10b](#). It is seen that the matched filtering has improved the quality of the QRS complexes for the hand-hold contact data. This is especially true near the beginning and ends of the data record, which were highly noisy. Thus, it might be supposed that simple thresholding and determining some average time between QRS complexes would produce a good heart rate estimate.

Unfortunately, subsequent experimentation proved that the heart rate estimates obtained this way were not reliable enough. Some of the difficulties encountered were: obtaining an accurate enough replica of the QRS complex while a person is exercising, choosing thresholds that could be used over the duration of the exercise session, among others. In this case, the obvious approach was not successful. Such is not an uncommon situation, and not unexpected as we have skipped many of the important steps first outlined in [Figure 2.1](#). A more precise mathematical model for the noise, including its PDF, and of the signal, including its variability over time and from person to person, as examples, would be the next logical step. Such an effort is ongoing.



**Figure 15.10: Hand-hold contact data and matched filtered output. The data samples have been connected with straight lines for easier viewing.**

In summary, difficult problems such as the one described in this chapter require a full analysis to formulate good signal and noise models. The approach outlined in [Figure 2.1](#) reflects this philosophy. Guessing at a solution sometimes produces an algorithm that “works well”. However, *to design algorithms with the best performance, we require reasonably accurate models coupled with*

*sound statistical approaches.*

## 15.5. Lessons Learned

- Always filter the data to reduce the analysis bandwidth to the signal band of interest. This improves the SNR by deleting the noise-only bands.
- The use of the periodogram for determining the fundamental frequency of a periodic waveform works well for long enough data records. For shorter data records, however, the peaks may not be easily discernible from the background noise. This is exacerbated if the waveform is not purely periodic but may exhibit a change in its period, such as is encountered with heart rate estimation during exercising.
- It is always a good approach to use several different spectral estimators to check for consistency of results. This avoids misinterpreting details in the spectral estimate that may be artifacts of the particular estimator used. A prime example is the possibility of spurious peaks in AR spectral estimation.
- Guessing at an algorithm to solve a problem always leaves the open question of whether there might exist an algorithm with better performance.

## References

- Ruha, A., S. Sallinen, S. Nissila, “A Real-Time Microprocessor QRS Detector System with a 1-ms Timing Accuracy for the Measurement of Ambulatory HRV”, *IEEE Trans. on Biomedical Engineering*, pp. 159–167, March 1997.
- Sornmo, L., O. Pahlm, M. Nygards, “Adaptive QRS Detection: A Study of Performance”, *IEEE Trans. on Biomedical Engineering*, pp. 392–401, June 1985.
- Xue, Q., Y.H. Hu, W.J. Tompkins, “Neural-Network Based Adaptive Matched Filtering for QRS Detection”, *IEEE Trans. on Biomedical Engineering*, pp. 317–329, April 1992.

## Appendix 15A. Solutions to Exercises

- 15.1** If you run the program `FSSP3exer15_1.m` you will see the periodogram plotted over the analog frequency range  $0 \leq F \leq 40$  Hz. You should see that, as expected, the first few peaks are at 9.45, 11.80, 14.14, and 16.52 Hz. If you then replace the chest electrode data with the hand-

hold contact data by enabling `x=hecg`; in the program, you will observe additional peaks due to the noise. If you were to threshold the periodogram to obtain the frequencies, it is possible that you may include noise peaks or even miss some signal peaks, depending upon which portion of the band you examined. Note that you can use the “data-cursor” function provided on the MATLAB figure window toolbar to determine the frequencies at which the peaks occur.

- 15.2** You should observe that the spectrum extends over the entire band from 9 to 39 Hz. Note that it rolls off below 9 Hz and above 39 Hz due to the bandpass filtering. To preserve the QRS complex the entire band must be maintained. You can run the program `FSSP3exer15_2.m`, which is contained on the CD, to obtain the results.
- 15.3** The AR spectral estimate using the covariance method will exhibit many different peaks due to statistical error. Because the method does not constrain the estimated poles to be within the unit circle, it is likely that some may stray too close to the unit circle. When they do, the estimated PSD exhibits a sharp peak, which is due to statistical error. This is why the average of the estimates is very “peaky”. The Burg method, on the other hand, constrains the estimated poles to be within the unit circle. Therefore, these extraneous peaks are less likely to occur. You can run the program `FSSP3exer15_3.m`, which is contained on the CD, to obtain the results.

# **Appendix A. Glossary of Symbols and Abbreviations**

## **A.1. Symbols**

Boldface characters denote vectors or matrices. All others are scalars. All vectors are column vectors. Random variables are denoted by capital letters such as  $U, V, W, X, Y, Z$  and random vectors by  $\mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$  and their values by corresponding lowercase letters.

$*$	complex conjugate
$\hat{\cdot}$	denotes estimator
$\sim$	denotes complex quantity
$\sim$	denotes <i>is distributed according to</i>
$A$	amplitude of signal (DC level)
$a[i]$	AR filter parameter
$[\mathbf{A}]_{ij}$	$ij$ th element of $\mathbf{A}$
$[\mathbf{b}]_i$	$i$ th element of $\mathbf{b}$
$\chi_n^2$	chi-squared distribution with $n$ degrees of freedom
$\chi_n^2(\lambda)$	noncentral chi-squared distribution with $n$ degrees of freedom and noncentrality parameter $\lambda$
$\text{cov}(x, y)$	covariance of $x$ and $y$
$\mathbf{C}$ or $\mathbf{C}_x$	covariance matrix of $\mathbf{x}$
$\mathcal{CN}(\tilde{\mu}, \sigma^2)$	complex normal distribution with mean $\tilde{\mu}$ and variance $\sigma^2$
$d^2$	deflection coefficient
$\delta(t)$	Dirac delta function
$\delta[n]$	discrete-time impulse sequence
$\Delta$	time sampling interval
$\det(\mathbf{A})$	determinant of matrix $\mathbf{A}$
$E$	expected value
$E_x$	expected value with respect to PDF of $\mathbf{x}$
$\mathcal{E}$	signal energy
$\eta$	signal-to-noise ratio
$f$	discrete-time frequency in cycles/sample
$F$	continuous-time frequency in Hz
$F_s$	sampling rate in samples/sec
$F_X(x)$	cumulative distribution function of random variable $X$
$F_X^{-1}(x)$	inverse cumulative distribution function of random variable $X$
$\gamma$ ( $\gamma'$ , $\gamma''$ , etc.)	threshold
$H$	conjugate transpose of vector or matrix
$\mathbf{H}$	observation matrix
$\mathcal{H}_0$	null hypothesis (noise only)
$\mathcal{H}_1$	alternative hypothesis (signal in noise)
$\mathcal{H}_i$	$i$ th hypothesis
$h[n]$	impulse response of linear shift invariant system

$H(f)$	frequency response of linear shift invariant system
$\mathcal{H}(z)$	system function of linear shift invariant system
$I(\theta)$	Fisher information for $\theta$
$\mathbf{I}$ or $\mathbf{I}_n$	identity matrix or identity matrix of dimension $n \times n$
$\mathbf{I}(\boldsymbol{\theta})$	Fisher information matrix for vector $\boldsymbol{\theta}$
$I(f)$	periodogram
$I_x(f)$	periodogram based on $x[n]$
$\text{Im}(\ )$	imaginary part of
$j$	$\sqrt{-1}$
$L(\mathbf{x})$	likelihood ratio
$L_G(\mathbf{x})$	generalized likelihood ratio
$\mu$	mean
$\boldsymbol{\mu}$	mean vector
$n$	sequence index
$N$	length of observed data set
$N_0/2$	level of PSD for continuous-time white Gaussian noise
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean $\mu$ and variance $\sigma^2$
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$	multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\mathbf{C}$
$\ \mathbf{x}\ $	norm of $\mathbf{x}$
$p(x)$ or $p_X(x)$	probability density function of $X$
$p(\mathbf{x}; \theta)$	PDF of $\mathbf{x}$ with $\theta$ as a parameter
$p(\mathbf{x} \mathcal{H}_i)$	conditional PDF of $\mathbf{x}$ conditioned on $\mathcal{H}_i$ being true
$P_D$	probability of detection
$P_e$	probability of error
$P_{FA}$	probability of false alarm
$\mathbf{P}$	projection matrix
$\mathbf{P}^\perp$	orthogonal projection matrix
$P_x(f)$	power spectral density of discrete-time process $x[n]$
$P_x(F)$	power spectral density of continuous-time process $x(t)$
$\rho_s$	signal correlation coefficient
$Q(x)$	probability that a $\mathcal{N}(0, 1)$ random variable exceeds $x$
$Q^{-1}(u)$	value of $\mathcal{N}(0, 1)$ random variable which is exceeded with probability of $u$

$Q_{\chi_n^2}(x)$	probability that a $\chi_n^2$ random variable exceeds $x$
$Q_{\chi_n'^2(\lambda)}(x)$	probability that a $\chi_n'^2(\lambda)$ random variable exceeds $x$
$r_x[k]$	autocorrelation sequence of discrete-time process $x[n]$
$r_x(\tau)$	autocorrelation function of continuous-time process $x(t)$
$\mathbf{R}_x$	autocorrelation matrix of $\mathbf{x}$
$\text{Re}(\ )$	real part
$\sigma^2$	variance
$s[n]$	discrete-time signal
$S(f)$	discrete-time Fourier transform of $s[n]$
$\mathbf{s}$	vector of signal samples
$s(t)$	continuous-time signal
$\text{sgn}(x)$	signum function ( $= 1$ for $x > 0$ and $= -1$ for $x < 0$ )
$T(\mathbf{x})$	detection statistic
$t$	continuous time
$\theta(\boldsymbol{\theta})$	unknown parameter (vector)
$\hat{\theta}(\hat{\boldsymbol{\theta}})$	estimator of $\theta(\boldsymbol{\theta})$
$T$	transpose of vector or matrix
$\text{var}(x)$	variance of $x$
$w[n]$	observation noise sequence
$\mathbf{w}$	vector of noise samples
$w(t)$	continuous-time noise
$x[n]$	observed discrete-time data
$\mathbf{x}$	vector of data samples
$x(t)$	observed continuous-time waveform
$\bar{x}$	sample mean of $x$
$z$	complex variable
$\mathbf{0}$	vector or matrix of all zeros

## A.2. Abbreviations

ACS	autocorrelation sequence
AR	autoregressive
AR( $p$ )	autoregressive process of order $p$
CDF	cumulative distribution function
CRLB	Cramer-Rao lower bound
CWGN	complex white Gaussian noise
DC	constant level (direct current)
DFT	discrete Fourier transform
EEF	exponentially embedded family
ENR	energy-to-noise ratio
FFT	fast Fourier transform
FIR	finite impulse response
GLRT	generalized likelihood ratio test
IID	independent and identically distributed
IIR	infinite impulse response
LS	least squares
MAP	maximum a posteriori
ML	maximum likelihood
MLE	maximum likelihood estimator
MMSE	minimum mean square error
MSE	mean square error
MVSE	minimum variance spectral estimator
MVU	minimum variance unbiased
NP	Neyman-Pearson
PDF	probability density function
PSD	power spectral density
ROC	receiver operating characteristics
SNR	signal-to-noise ratio
WGN	white Gaussian noise
WSS	wide sense stationary

## Appendix B. Brief Introduction to MATLAB

A brief introduction to the scientific software package MATLAB is contained in this appendix. Further information is available at the web site [www.mathworks.com](http://www.mathworks.com). MATLAB is a scientific computation and data presentation language.

### B.1. Overview of MATLAB

The chief advantage of MATLAB is its use of high-level instructions for matrix algebra and built-in routines for data processing. In this appendix as well as throughout the text a MATLAB command is indicated with the typewriter font such as `end`. MATLAB treats matrices of any size (which includes vectors and scalars as special cases) as *elements*, and hence matrix multiplication is as simple as  $C=A*B$ , where  $A$  and  $B$  are conformable matrices. In addition to the usual matrix operations of addition  $C=A+B$ , multiplication  $C=A*B$ , and scaling by a constant  $c$  as  $B=c*A$ , certain matrix operators are defined that allow convenient manipulation. For example, assume we first define the column vector  $\mathbf{x} = [1 \ 2 \ 3 \ 4]^T$ , where  $T$  denotes transpose, by using `x=[1:4] . '`. The vector starts with the element 1 and ends with the element 4, and the colon indicates that the intervening elements are found by incrementing the start value by one, which is the default. For other increments, say 0.5, we use `x=[1:0.5:4] . '`. To define the vector  $\mathbf{y} = [1^2 \ 2^2 \ 3^2 \ 4^2]^T$ , we can use the matrix *element by element* exponentiation operator `.^` to form `y=x.^2 if x=[1:4] . '`. Similarly, the operators `.*` and `./` perform element by element multiplication and division of the matrices, respectively. For example, if

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\end{aligned}$$

then the statements `C=A.*B` and `D=A./B` produce the results

$$\begin{aligned}\mathbf{C} &= \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix} \\ \mathbf{D} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\end{aligned}$$

respectively. A listing of some common characters is given in [Table B.1](#).

MATLAB has the usual built-in functions of `cos`, `sin`, etc. for the trigonometric functions, `sqrt` for a square root, `exp` for the exponential function, and `abs` for absolute value, as well as many others. When a function is applied to a matrix, the function is applied to each element of the matrix. Other built-in symbols and functions and their meanings are given in [Table B.2](#).

**Table B.1: Definition of common MATLAB characters.**

Character	Meaning
<code>+</code>	addition (scalars, vectors, matrices)
<code>-</code>	subtraction (scalars, vectors, matrices)
<code>*</code>	multiplication (scalars, vectors, matrices)
<code>/</code>	division (scalars)
<code>^</code>	exponentiation (scalars, square matrices)
<code>.*</code>	element by element multiplication
<code>./</code>	element by element division
<code>.^</code>	element by element exponentiation
<code>;</code>	suppress printed output of operation
<code>:</code>	specify intervening values
<code>,</code>	conjugate transpose (transpose for real vectors, matrices)
<code>...</code>	line continuation (when command must be split)
<code>%</code>	remainder of line interpreted as comment
<code>==</code>	logical equals
<code> </code>	logical or
<code>&amp;</code>	logical and
<code>~ =</code>	logical not

**Table B.2: Definition of useful MATLAB symbols and functions.**

Function	Meaning
<code>pi</code>	$\pi$
<code>i</code>	$\sqrt{-1}$
<code>j</code>	$\sqrt{-1}$
<code>round(x)</code>	rounds every element in $\mathbf{x}$ to the nearest integer
<code>floor(x)</code>	replaces every element in $\mathbf{x}$ by the nearest integer less than or equal to $x$
<code>inv(A)</code>	takes the inverse of the square matrix $\mathbf{A}$
<code>x=zeros(N,1)</code>	assigns an $N \times 1$ vector of all zeros to $\mathbf{x}$
<code>x=ones(N,1)</code>	assigns an $N \times 1$ vector of all ones to $\mathbf{x}$
<code>x=rand(N,1)</code>	generates an $N \times 1$ vector of all uniform random variables
<code>x=randn(N,1)</code>	generates an $N \times 1$ vector of all Gaussian random variables
<code>rand('state',0)</code>	initializes uniform random number generator
<code>randn('state',0)</code>	initializes Gaussian random number generator
<code>M=length(x)</code>	sets $M$ equal to $N$ if $\mathbf{x}$ is $N \times 1$
<code>sum(x)</code>	sums all elements in vector $\mathbf{x}$
<code>mean(x)</code>	computes the sample mean of the elements in $\mathbf{x}$
<code>flipud(x)</code>	flips the vector $\mathbf{x}$ upside down
<code>abs</code>	takes the absolute value (or complex magnitude) of every element of $\mathbf{x}$
<code>fft(x,N)</code>	computes the FFT of length $N$ of $\mathbf{x}$ (zero pads if $N > \text{length}(\mathbf{x})$ )
<code>ifft(x,N)</code>	computes the inverse FFT of length $N$ of $\mathbf{x}$
<code>fftshift(x)</code>	interchanges the two halves of an FFT output
<code>pause</code>	pauses the execution of a program
<code>break</code>	terminates a loop when encountered
<code>whos</code>	lists all variables and their attributes in current workspace
<code>help</code>	provides help on commands, e.g., <code>help sqrt</code>

Matrices and vectors are easily specified. For example, to define the column vector  $\mathbf{c}_1 = [1 \ 2]^T$ , just use `c1=[1 2].'` or equivalently `c1=[1;2]`. To define the  $\mathbf{C}$  matrix given previously, the construction `C=[1 4; 9 16]` is used. Or we could first define  $\mathbf{c}_2 = [4 \ 16]^T$  by `c2=[4 16].'` and then use `C=[c1 c2]`. It is also possible to extract portions of matrices to yield smaller matrices or vectors. For example, to extract the first column from the matrix  $\mathbf{C}$  use `c1=C(:,1)`. The colon indicates that all elements in the first column should be extracted. Many other convenient manipulations of matrices and vectors are possible.

Any vector that is generated whose dimensions are not explicitly specified is assumed to be a *row* vector. For example, if we say `x=ones(10)`, then it will be designated as the  $1 \times 10$  row vector consisting of all ones. To yield a column vector use `x=ones(10, 1)`.

Loops are implemented with the construction

```
for k=1:10
    x(k,1)=1;
end
```

which is equivalent to `x=ones(10, 1)`. Logical flow can be accomplished with the construction

```
if x>0
    y=sqrt(x);
else
    y=0;
end
```

Finally, a good practice is to begin each program or script, which is called an “m” file (due to its syntax, for example, `pdf.m`), with a `clear all` command. This will clear all variables in the workspace, since otherwise, the current program may inadvertently (on the part of the programmer) use previously stored variable data.

## B.2. Plotting in MATLAB

Plotting in MATLAB is illustrated in the next section by example. Some useful functions are summarized in [Table B.3](#).

### An Example Program

A complete MATLAB program is given here, and is contained on the CD, to illustrate how one might compute the samples of several sinusoids of different amplitudes. It also allows the sinusoids to be clipped. The sinusoid is  $s(t) = A \cos(2\pi F_0 t + \pi/3)$ , with  $A = 1$ ,  $A = 2$ , and  $A = 4$ ,  $F_0 = 1$ , and  $t = 0, 0.01, 0.02, \dots, 10$ . The clipping level is set at  $\pm 3$ , i.e., any sample above  $+3$  is clipped to  $+3$ , and any sample less than  $-3$  is clipped to  $-3$ .

**Table B.3: Definition of useful MATLAB plotting functions.**

Function	Meaning
<b>figure</b>	opens up a new figure window
<b>plot(x,y)</b>	plots the elements of <b>x</b> versus the elements of <b>y</b>
<b>plot(x1,y1,x2,y2)</b>	same as above except multiple plots are made
<b>plot(x,y,'.'</b> )	same as <b>plot</b> except the points are not connected
<b>title('my plot')</b>	puts a title on the plot
<b>xlabel('x')</b>	labels the x axis
<b>ylabel('y')</b>	labels the y axis
<b>grid</b>	draws grid on the plot
<b>axis([0 1 2 4])</b>	plots only the points in range $0 \leq x \leq 1$ and $2 \leq y \leq 4$
<b>text(1,1,'curve 1')</b>	places the text "curve 1" at the point (1,1)
<b>hold on</b>	holds current plot
<b>hold off</b>	releases current plot

[Click here to view code image](#)

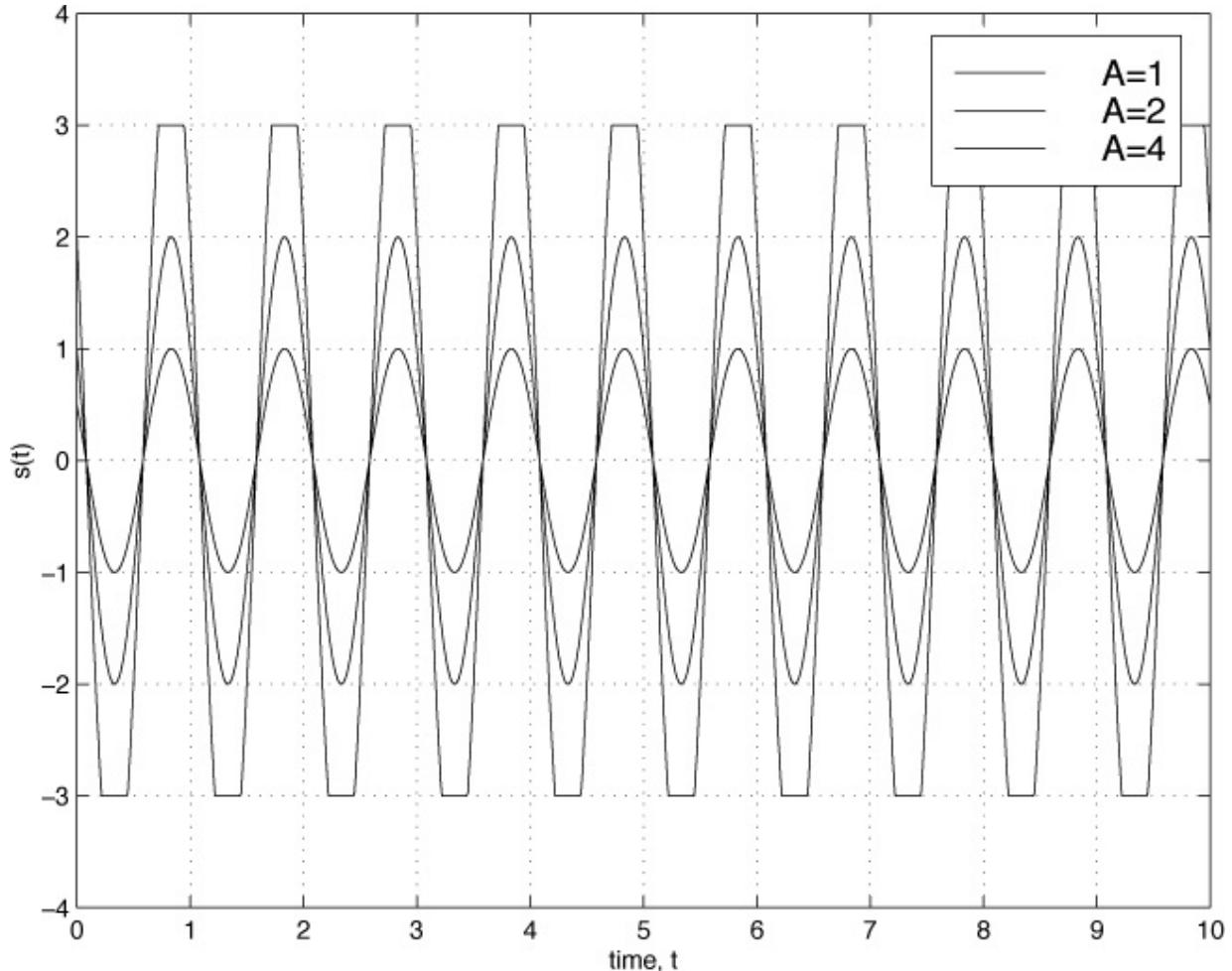
```
% matlabexample.m
%
% This program computes and plots samples of a sinusoid
% with amplitudes 1, 2, and 4. If desired, the sinusoid can be
% clipped to simulate the effect of a limiting device.
% The frequency is 1 Hz and the time duration is 10 seconds.
% The sample interval is 0.1 seconds. The code is not efficient but
% is meant to illustrate MATLAB statements.
%
clear all % clear all variables from workspace
delt=0.01; % set sampling time interval
F0=1; % set frequency
t=[0:delt:10]'; % compute time samples 0,0.01,0.02,...,10
A=[1 2 4]'; % set amplitudes
clip='yes'; % set option to clip
for i=1:length(A) % begin computation of sinusoid samples
    s(:,i)=A(i)*cos(2*pi*F0*t+pi/3); % note that samples for sinusoid
                                         % are computed all at once and
                                         % stored as columns in a matrix
    if clip=='yes' % determine if clipping desired
        for k=1:length(s(:,i)) % note that number of samples given as
                               % dimension of column using length
                               % command
            if s(k,i)>3 % check to see if sinusoid sample exceeds 3
                s(k,i)=3; % if yes, then clip
            elseif s(k,i)<-3 % check to see if sinusoid sample is less
                s(k,i)=-3; % than -3 if yes, then clip
            end
        end
    end
end
figure % open up a new figure window
```

```

plot(t,s(:,1),t,s(:,2),t,s(:,3)) % plot sinusoid samples versus time
                                    % samples for all three sinusoids
grid % add grid to plot
xlabel('time, t') % label x-axis
ylabel('s(t)') % label y-axis
axis([0 10 -4 4]) % set up axes using axis([xmin xmax ymin ymax])
legend('A=1','A=2','A=4') % display a legend to distinguish
                            % different sinusoids

```

The output of the program is shown in [Figure B.1](#). Note that the different graphs will appear as different colors.



**Figure B.1: Output of MATLAB program** matlabexample.m.

# Appendix C. Description of CD Contents

## C.1. CD Folders

The version of MATLAB used to create the files is R2009a (Version 7.8.0.347), configured for operation on a Windows PC machine. The included CD contains the following folders:

1. Algorithm\_implementations - These are MATLAB m files that implement the algorithms described in [Chapters 9 \(Algorithms for Estimation\)](#), [Chapter 10 \(Algorithms for Detection\)](#), and [Chapter 11 \(Spectral Estimation\)](#). Each file is a function subprogram with a typical file being `FSSP3alg_9_1_sub.m`, which implements [Algorithm 9.1](#). In some instances external programs are called for. In these instances the additional programs required are given in the folder “Utility\_programs”.
2. Exercise\_solutions - These are MATLAB m files that can be run to obtain the solutions of the exercises that are included within each chapter. A typical file is `FSSP3exer_9_1.m`, which is the solution for [Exercise 9.1](#). In some cases, external programs are required and these can be found in the folder “Utility\_programs”. Also, there are MATLAB mat files, containing data, that can be loaded in as required by the exercise solution program.
3. Utility\_programs - These are MATLAB programs and function subprograms that are generally useful for common operations that have been described in the book. They are described next.

## C.2. Utility Files Description

1. AR\_est\_order.m This program estimates the AR model order needed to estimate the AR PSD, based on the EEF approach (see [\(11.9\)](#)).
2. AR\_par\_est\_cov.m This program estimates the AR parameters using the covariance method, an approximate MLE (see [\(11.8\)](#)).
3. ARgenda.m This program generates real data samples of an AR random process given the filter coefficients and excitation noise variance (see [Section 4.4](#)).
4. ARgenda\_complex.m This program generates complex data samples of a complex AR random process given the complex filter coefficients and excitation noise variance (see [Section 12.3.3](#)).

5. ARpsd.m This program computes a set of PSD values for the frequency band  $[-1/2, 1/2]$ , given the parameters of an AR model (see [\(4.2\)](#)).
6. ARpsd\_model.m This program determines the AR model PSD to approximate a desired PSD (see [Section 4.4](#)).
7. autocorrelation\_est.m This program estimates the autocorrelation sequence from data (see [\(6.14\)](#)).
8. Burg.m This program estimates the AR parameters using the Burg method (an approximate MLE). Note that the AR filter parameters will produce a stable all-pole filter (see [Algorithm 11.4](#)).
9. chipr2.m This program computes the right-tail probability of a central or noncentral chi-squared PDF (see [Section 6.4](#)).
10. cwgn.m This program generates complex white Gaussian noise (see [Section 12.3](#)).
11. detection\_demo.m This program compares the performance of an autocorrelator detector with that of a matched filter and an energy detector by computing an ROC. The signal is a sinusoid of unknown frequency embedded in white Gaussian noise (see Section 7B12.3).
12. ED\_threshold.m This program computes the threshold required for an energy detector to achieve a given  $P_{FA}$  (see [Algorithm 10.5](#)).
13. Gaussmix\_gendata.m This program generates a realization of IID Gaussian mixture noise given the mixture parameters and the noise variances (see [\(4.19\)](#)).
14. Laplacian\_gendata.m This program generates a realization of IID Laplacian noise given the noise variance (see [\(4.17\)](#)).
15. matlabexample.m This is the program given in [Appendix B](#) to demonstrate plotting in MATLAB.
16. mDmax.m This program finds the maximum and location of the maximum of a D-dimensional function, where D can be 2, 3, or 4. It is more convenient to use than MATLAB's max function (see [Algorithm 9.9](#)).
17. pdf\_AR\_est.m This program estimates and plots the PDF of a set of data using an AR estimator (see [\(6.20\)](#)).
18. pdf\_AR\_est\_order.m This program estimates the AR model order needed to estimate the PDF using an AR model. The value can be input to the program pdf\_AR\_est.m to estimate the PDF (see [\(6.27\)](#)).

- 19.** pdf\_hist\_est.m This program estimates and plots the PDF of a set of data using a histogram estimator (see [\(6.18\)](#)).
- 20.** plotlineroutine.m This program was used to generate the MATLAB plots in the textbook.
- 21.** pole\_plot\_PSD.m This program allows the user to specify poles in the unit circle of the z-plane. These poles are then converted to autoregressive parameters and the corresponding power spectral density  $P(f)$  is plotted, assuming the excitation noise variance is  $\sigma_u^2 = 1$  (see [Appendix 4C](#)).
- 22.** Q.m This program computes the right-tail probability (complementary cumulative distribution function) for a  $N(0, 1)$  random variable (see [Section 4.3](#)).
- 23.** Qinv.m This program computes the inverse Q function or the value that is exceeded by a  $N(0, 1)$  random variable with a probability of  $x$  (see [Section 8.2.2](#)).
- 24.** roccurve.m This program determines the ROCs for a given set of detector outputs under  $H_0$  and  $H_1$  (see [Section 7.3.2](#)).
- 25.** shift.m This program shifts the given sequence by  $N_s$  points. Zeros are shifted in either from the left or right (see [Algorithm 9.8](#)).
- 26.** sinusoid\_gen.m This program generates a sum of damped sinusoids and also computes the magnitude of the Fourier transform (see [\(3.9\)](#)).
- 27.** stepdown.m This program implements the stepdown procedure to find the coefficients and prediction error powers for all the lower order predictors given the coefficients and prediction error power for the  $p$ th order linear predictor (needed for ARgenda.m).
- 28.** stepdown\_complex.m This program implements the stepdown procedure for a complex linear predictor (needed for ARgenda\_complex.m).
- 29.** WGNgenda.m This program generates white Gaussian noise (see [Section 4.3](#)).
- 30.** YWsolve.m This program solves the Yule-Walker equations (see [\(4.10\)](#)).

# Index

Abbreviations, [457](#)

ACS, *see* [Autocorrelation](#)

Adaptive noise canceler, [308](#)

Algorithms, *see* Algorithm implementations folder on CD

detection, [315](#), [423](#)

estimation, [274](#), [405](#)

fixed vs. adaptive, [129](#), [151](#)

software development, [210](#)

spectral estimation, [343](#), [443](#)

All-pole modeling, [56](#), [96](#)

Amplitude estimation/detection, [274](#), [326](#)

AR, *see* [Autoregressive](#)

Array processing, [345](#)

Arrival time estimation/detection, [281](#), [332](#)

Asymptotics

autocorrelation variance, [205](#)

detection, [321](#), [323](#)

Gaussian PDF, [120](#)

performance, [204](#)

variance, [30](#)

Autocorrelation

definition

complex, [375](#)

real, [117](#)

detection, [220](#), *see* `detection_demo.m`

estimation of, [165](#), [375](#), *see* `autocorrelation_est.m`

estimation of frequency, [415](#)

matrix, [302](#)

Autoregressive

definition

complex, [375](#), [408](#)

real, [94](#), [98](#)

examples, [91](#), [95](#), [101](#), *see ARpsd.m*  
frequency response, [121](#)  
generation of data, *see ARgendata.m*, *see ARgendata\_complex.m*  
model order estimation, [355](#), *see AR\_est\_model.m*  
parameter estimation, [100](#), [351](#), *see AR\_par\_est\_cov.m*, *see Burg.m*,  
[354](#)  
PDF model estimation, [171](#), [438](#)  
PSD model estimation, [100](#), *see ARpsd\_model.m*, [348](#)  
time-varying, [104](#)  
Averaged periodogram, [343](#), *see PSD\_est\_avper.m*  
Bandpass signal, [367](#)  
Bandwidth, mean squared, [282](#)  
Beamforming, [285](#)  
Bearing estimation, [284](#), [382](#)  
Bias, [193](#)  
Bias-variance tradeoff, [342](#)  
Burg method, [353](#), *see also PSD\_est\_AR\_Burg.m*  
Cautions  
    Always do a computer simulation first, [44](#)  
    Always use a computer simulation first to determine performance, [190](#)  
    An upper bound is hard (actually impossible) to beat, [44](#)  
    Assuming WGN, [409](#)  
    Bayesian versus classical estimator performance comparisons, [244](#)  
    Be careful of definitions, [371](#)  
    Computing the MLE via a grid search, [259](#)  
    If in doubt, increase the data record length,  $N$ , [162](#)  
    Linear filtering alters the PDF of an IID nonGaussian random process, [111](#)  
    Performance metrics and assumptions for detection and classification, [248](#)  
    The siren call of linearization, [76](#)  
    There is no “free lunch”, [209](#)  
    Using multiple approaches lends credibility, [159](#)  
    What do we mean by mean?, [153](#)  
    You can never be sure!, [160](#)  
Central limit theorem, [78](#)  
Classification, [38](#), [254](#)

Clipper, [317](#)

Clutter in radar, [94](#)

Colored Gaussian noise, [94](#)

Comb filter, [300](#), [434](#)

Communication systems, [199](#), [248](#), [324](#)

Complex demodulation, [35](#), [368](#)

Complex envelope, [35](#), [369](#)

Complex Gaussian random variable, [373](#)

Confidence intervals

    definition, [179](#)

    kurtosis, [181](#)

    mean, [162](#)

    PDF, [170](#)

    PSD, [175](#), [344](#)

    variance, [163](#)

Correlator, running, [281](#), *see also* [Estimator-correlator](#), *see also* [Replica-correlator](#)

Covariance

    definition

        complex, [374](#)

        real, [164](#)

    estimation of, [164](#)

Covariance matrix

    definition

        complex, [374](#)

        real, [119](#)

    estimation of, [166](#)

Covariance method, [350](#), *see also* `PSD_est_ARcov.m`

Cramer-Rao lower bound

    DC level in WGN, [7](#)

    definition, [237](#)

    description, [5](#)

    Inverse cumulative distribution function, [111](#)

    use as sanity check, [202](#)

CRLB, *see* [Cramer-Rao lower bound](#)

Data windowing, [342](#)  
DC level in noise, [5](#)  
Deflection coefficient, [40](#)  
Delay time estimation, [281](#)  
Detection curves, [197](#), *see also* [Receiver operating characteristics](#)  
Doppler effect, [33](#), [49](#), [279](#), [406](#)  
Dynamical signal model, [105](#)  
EEF, *see* [Exponentially embedded family](#)  
Electrocardiogram (ECG), [56](#), [152](#), [305](#), [444](#)  
Electroencephlogram (EEG), [30](#)  
Energy detector, [324](#), *see also* `ED_threshold.m`, [431](#)  
Energy-to-noise ratio, [32](#), [282](#)  
ENR, *see* [Energy-to-noise ratio](#)  
Estimator-correlator, [322](#)  
Estimators, [274](#), *see* [Table 9.1](#)  
Excitation noise (AR), [97](#)  
Exponential signals  
    definition, [63](#)  
    examples, [133](#)  
Exponentially embedded family, [140](#), 358  
False alarm probability, [32](#), [194](#)  
Fast Fourier transform, [4](#), [344](#), [361](#), [412](#)  
FFT, *see* [Fast Fourier transform](#)  
Finite impulse response filter, [155](#), [306](#), [444](#)  
FIR, *see* [Finite impulse response filter](#)  
FM signal, [64](#)  
Frequency response, [118](#)  
Fundamental frequency, [426](#)  
Gaussian random process, [119](#)  
Generalized likelihood ratio test  
    definition, [260](#)  
    examples, *see* [Algorithms](#), detection  
Generalized matched filter, [319](#)  
Geomagnetic noise, [423](#)  
GLRT, *see* [Generalized likelihood ratio test](#)

Grid search, [76](#), [137](#)  
Harmonic frequency, [426](#)  
Hermitian form, [374](#)  
Histogram, [168](#), *see* `pdf_hist_est.m`  
IID, *see* [Independent and identically distributed](#)  
Image signal processing, [315](#), [322](#)  
Impulse response, [118](#)  
In-phase signal, [35](#), [370](#)  
Independent and identically distributed  
    Definition, [92](#)  
    Detection, [317](#)  
    MATLAB, *see* `Gauss_mixture.m`, *see* `Laplacian_gendata.m`  
    Noise, [109](#)  
Interference suppression, [30](#), [113](#), [299](#), [305](#)  
Inverse filter, [354](#)  
Kurtosis  
    asymptotic variance, [181](#), [204](#)  
    confidence interval, [181](#)  
    definition, [157](#)  
Least squares estimator, *see also* [Linear model](#)  
    complex, [378](#)  
    covariance method, [353](#)  
    line signal, [68](#)  
    real, [252](#)  
Levinson algorithm, [101](#), [354](#)  
Light detection and ranging (LIDAR), [281](#)  
Likelihood ratio test, *see* [Neyman-Pearson detection/criterion](#)  
Limiter, [317](#), [432](#)  
Line arrays, [285](#), [382](#)  
Linear FM signal, [65](#)  
Linear model, Bayesian  
    definition  
        complex, [379](#)  
        real, [81](#)  
    examples, [114](#)

parameter estimation, [252](#)  
Linear model, classical  
classification, [254](#)  
definition  
    complex, [378](#)  
    real, [71](#)  
detection, [253](#)  
examples, [10](#), [69](#), [72](#), [426](#)  
parameter estimation, [252](#)  
Linear predictive coding, [27](#)  
Linear shift invariant system, [98](#), [118](#)  
Local stationarity, [103](#), [356](#)  
MAP, *see* [Maximum a posteriori decision rule](#)  
Matched filter, [316](#), [319](#), [451](#)  
MATLAB, [3](#), *see also* `matlabexample.m`, [14](#), [461](#)  
MATLAB subprograms, [467](#)  
Matrix  
    autocorrelation, [119](#)  
    eigenanalysis, [297](#)  
    hermitian, *see* [Covariance matrix](#), definition, complex positive definite, *see* [Covariance matrix](#), definition, real projection, [155](#)  
    pseudoinverse, [297](#)  
    Toeplitz, [119](#)  
Maximum a posteriori decision rule  
    definition, [250](#)  
    upper bound, [203](#)  
Maximum likelihood decision rule, [248](#)  
Maximum likelihood estimator  
    definition, [258](#)  
    insight into, [263](#)  
    practical utility, [240](#)  
Mean  
    Definition, [162](#), [165](#)  
    Estimation of, [162](#), [165](#)  
Mean square error (classical), [193](#)

Minimum distance classifier, [43](#)  
Minimum mean square error estimator, Bayesian  
definition, [242](#)  
insight into, [265](#)  
linear model, [252](#)  
Minimum probability of error, [42](#), [191](#)  
Minimum variance spectral estimator, [346](#), *see PSD\_est\_mvse.m*, [450](#)  
Minimum variance unbiased estimator  
definition, [237](#)  
theorem, [240](#)  
ML, *see Maximum likelihood decision rule*  
MLE, *see Maximum likelihood estimator*  
MMSE estimator, *see Minimum mean square error estimator*  
Model order estimation  
AR PDF, [176](#), *see pdf\_AR\_est\_order.m*  
AR PSD, [355](#), *see also AR\_est\_order.m*  
exponential signal, [140](#)  
polynomial signal, [141](#)  
Models  
identifiability, [62](#), [137](#)  
linear/nonlinear, [61](#), [74](#), [137](#)  
necessity of good, [236](#)  
noise, [90](#), *see Table 4.1*  
signal, [32](#), [57](#), *see Table 3.1*, [74](#), *see Table 3.2*  
Moments, [163](#)  
Monte Carlo method  
need for, [192](#)  
performance evaluation  
probability of detection, [197](#)  
probability of error, [200](#)  
ROC, [195](#)  
MSE, *see Mean square error*  
Multipath, *see Rayleigh fading*, *see Rician fading*  
MVSE, *see Minimum Variance spectral estimator*  
MVU, *see Minimum variance unbiased estimator*

Neyman-Pearson detection/criterion  
definition, [244](#)  
theorem, [246](#)  
upper bound, [203](#), [221](#)

NonGaussian noise  
models, [90](#), [109](#), [112](#)  
test of, [157](#)

Nonparametric estimation, [171](#), [340](#)

Nonstationary random process, [102](#)

Normal PDF, *see* [Probability density functions](#)

Notational conventions, [15](#), [457](#)

NP, *see* [Neyman-Pearson detection/criterion](#)

Orthogonality, [427](#)

Outliers, *see* [Spikes](#)

Partial correlation, [354](#)

Partially linear model, [61](#), [426](#)

Pattern recognition, *see* [Classification](#)

PDF, *see* [Probability density functions](#)

Performance metrics, [191](#)

Period estimation, [300](#)

Periodic signals, [66](#), [72](#), [426](#)

Periodogram  
detector, [330](#)  
frequency estimation, [206](#), [446](#)

PDF, asymptotic PDF, [120](#)

real-data application, [449](#)

resolution, [9](#), [342](#), [361](#)

spatial, [285](#)  
spectral estimator, [174](#), *see* `PSD_est_avper.m`

Poles, [121](#), [297](#), [355](#), [408](#), *see* `pole_plot_PSD.m`

Polynomial signal, [66](#)

Posterior PDF, [242](#)

Power spectral density  
approximation of, [100](#)  
definition, [117](#), [173](#)

estimator, center frequency, [289](#)  
estimator, power, [288](#)  
models, [96](#), [121](#)  
Prewhitenning, [321](#), [354](#)  
Principal components, [296](#)  
Prior PDF, [243](#)  
Prior probability, *see* [Probabilities](#)  
Probabilities  
    detection, [191](#)  
    error, [191](#)  
    false alarm, [191](#), [247](#)  
    Monte Carlo evaluation, [195](#), [197](#), [200](#)  
    prior, [248](#)  
    right-tail probability, [40](#), *see* `Q.m`  
Probability density functions  
    chi-squared (central), [175](#), *see* `chirpr2.m`  
    chi-squared (noncentral), [329](#), *see* `chirpr2.m`  
    complex Gaussian, [373](#)  
    definition, [167](#)  
    estimation of, [79](#), [156](#), [168](#), [171](#), *see* `pdf_AR_est.m`, *see* `pdf_hist_est.m`  
    Gaussian, [80](#), [93](#), [119](#), [166](#), *see* `Q.m`, *see* `Qinv.m`  
    Gaussian mixture, [112](#), *see* `Gaussmix_gendata.m`  
    Laplacian, [109](#), [430](#), *see* `Laplacian_gendata.m`  
    Normal, *see* Gaussian  
    Rayleigh, [80](#)  
    Rician, [81](#)  
    Von Mises, [57](#)  
PSD, *see* [Power spectral density](#)  
Q function  
    definition, [40](#)  
    MATLAB subprogram, *see* `Q.m`  
QRS complex, [444](#)  
Quadrature signal, [35](#), [370](#)  
Radar signal processing, [279](#), [320](#), [406](#)

Random process

- autoregressive, [94](#)
- complex, [374](#)
- Gaussian, [89](#), [102](#), [119](#)
- General concepts, [117](#)
- nonstationary, [91](#), [102](#), [150](#)
- random walk, [106](#)

Random signals

- center frequency, estimation, [289](#)
- definition, [77](#)
- examples, [242](#), [253](#)
- power, estimation, [288](#)
- signal samples, estimation, [302](#)

Random variable complex, *see* [Probability density functions](#), complex Gaussian

Rao test, [432](#)

Rayleigh fading, [80](#), [113](#)

Receiver operating characteristics, *see also* [roccurve.m](#)

- definition, [194](#)
- examples, [41](#), [435](#)
- Monte Carlo evaluation of, [195](#)

Reflection coefficient, [354](#)

Replica-correlator, [315](#), [317](#)

Resolution, [341](#), [359](#)

Reverberation, in sonar, [94](#)

Rician fading, [81](#)

Right-tail probability, *see* [Probabilities](#)

Road maps

- algorithm design, [24](#)
- noise modeling, [150](#)
- signal modeling, [131](#)

Robustness (sensitivity), [8](#), [30](#), [45](#), [59](#), [206](#), [318](#), [413](#), [436](#)

ROC, *see* [Receiver operating characteristics](#)

Sample mean estimator, [6](#)

Sign detector, [318](#), [435](#)

Signal averager, [299](#)

Signal design, [282](#)  
Signal-to-noise ratio, [277](#), [287](#), [376](#)  
Sinusoidal signal, *see also* `sinusoid_gen.m`  
    amplitude/phase, estimation, [75](#), [276](#)  
    detection, [328](#), [330](#)  
    frequency, estimation, [279](#)  
    model, [57](#), [62](#)  
    multiple sinusoids, parameter estimation, [292](#), [296](#)  
    random, [77](#), [80](#)  
    spatial, [285](#)  
    spectrum, [12](#)  
Sliding window power estimator, [103](#)  
Snapshot, [286](#)  
SNR, *see* [Signal-to-noise ratio](#)  
Sonar signal processing, [279](#), [284](#), [320](#)  
Spatial frequency, [285](#)  
Spectrogram, [356](#)  
Speech modeling, [89](#), [276](#)  
Spikes, [112](#), [318](#), [424](#)  
Standardized random variable, [158](#)  
Surveillance systems, [284](#), [328](#)  
System function, [98](#)  
Tapped delay line, [56](#)  
Target echo modeling, [78](#)  
TDL, *see* [Tapped delay line](#)  
Toeplitz, *see* [Matrix](#), [Toeplitz](#)  
Ultrasound, [31](#)  
Uniformly most powerful invariant test, [327](#), [329](#)  
Variance  
    definition, [163](#), [372](#)  
    estimation of, [163](#)  
    nonstationary, [102](#), [105](#)  
Vibrational analysis, [133](#), [343](#)  
WGN, *see* [White Gaussian noise](#)  
White Gaussian noise

definition

complex, [375](#), [408](#), *see* cwgn.m

real, [6](#), [93](#), *see* WGNgenda.m

from continuous-time, [36](#)

power, estimation, [286](#)

Wide sense stationary, [117](#), [119](#), [375](#)

Wiener filter, [303](#), [324](#)

Wiener process, *see* [Random process](#), random walk Wiener-Khintchine theorem, [117](#)

Window closing, [345](#)

Wrap-around effect, [421](#)

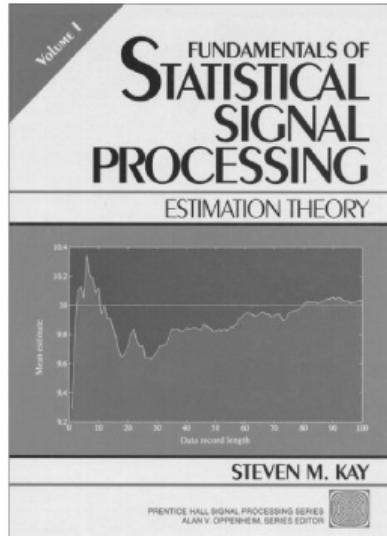
WSS, *see* [Wide sense stationary](#)

Yule-Walker equations, [100](#), [349](#), *see* YWsolve.m

Zero padding, [361](#)

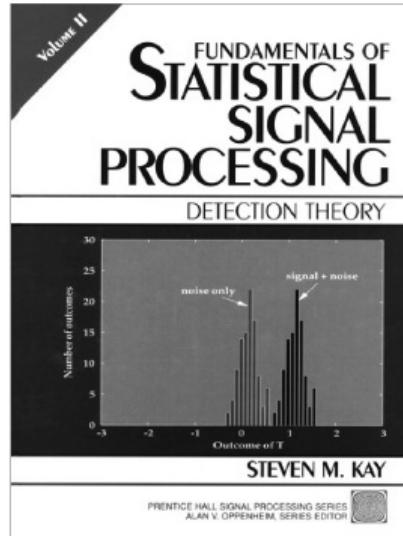


## More from Steven M. Kay



ISBN-13: 978-0-13-345711-7

A unified presentation of parameter estimation for those involved in the design and implementation of statistical signal processing algorithms. Covers important approaches to obtaining an optimal estimator and analyzing its performance; and includes numerous examples as well as applications to real-world problems



ISBN-13: 978-0-13-504135-2

The most comprehensive overview of signal detection available. Focuses extensively on real-world signal processing applications, including state-of-the-art speech and communications technology as well as traditional sonar/radar systems.



For more information and to order visit [informit.com](http://informit.com)

## **CD-ROM Warranty**

Prentice Hall warrants the enclosed CD-ROM to be free of defects in materials and faulty workmanship under normal use for a period of ninety days after purchase (when purchased new). If a defect is discovered in the CD-ROM during this warranty period, a replacement CD-ROM can be obtained at no charge by sending the defective CD-ROM, postage prepaid, with proof of purchase to:

Disc Exchange  
Prentice Hall  
Pearson Technology Group  
75 Arlington Street, Suite 300  
Boston, MA 02116  
Email: [disc.exchange@pearson.com](mailto:disc.exchange@pearson.com)

Prentice Hall makes no warranty or representation, either expressed or implied, with respect to this software, its quality, performance, merchantability, or fitness for a particular purpose. In no event will Prentice Hall, its distributors, or dealers be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the software. The exclusion of implied warranties is not permitted in some states. Therefore, the above exclusion may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you may have that vary from state to state. The contents of this CD-ROM are intended for personal use only.

More information and updates are available at:  
[informit.com/ph](http://informit.com/ph)

## Where Are the Companion Content Files?

Register this digital version of **Fundamentals of Statistical Signal Processing, Volume III**, to access important downloads.

Register this digital version to unlock the companion files that are included on the disc that accompanies the print edition. Follow the steps below.

1. Go to [www.informit.com/register](http://www.informit.com/register) and log in or create a new account.
  2. Enter this ISBN: **9780132808033** NOTE: This is the ISBN for the Print book which must be used to register the eBook edition.
  3. Answer the challenge question as proof of purchase.
  4. Click on the “Access Bonus Content” link in the “Registered Products” section of your account page, which will take you to the page where your downloadable content is available.
- 

The Professional and Personal Technology Brands of Pearson



Cisco Press



informIT

PEARSON IT Certification



que

SAMS

VMware PRESS

```
clear all % clear out all variables
randn('state',0) % initialize random number generator
count=0; % initialize threshold exceedance counter
for i=1:10000
    x=0.1+randn(100,1); % generate data set
    for j=1:100 % take the cube root
        if x(j)>=0
            y(j,1)=x(j)^(1/3);
        else
            y(j,1)=-((-x(j))^(1/3)); % this avoids the complex cube root
        end
    end
    T=mean(y); % compute detection statistic
    if T>=0 % determine if threshold exceeded
        count=count+1;
    end
end
P_Dest=count/10000 % estimate of P_D
```



```
clear all
rand('state',0)
randn('state',0)
epsilon=0.5;
sig21=1;
sig22=1;
N=10000;
u=rand(N,1);
for i=1:N
    if u(i)<1-epsilon
        w(i,1)=sqrt(sig21)*randn(1,1)-5; % now add -5 for
                                         % validation purposes only
    else
        w(i,1)=sqrt(sig22)*randn(1,1)+5; % now add +5 for
                                         % validation purposes only
    end
end
[xx,pdfest]=pdf_hist_est(w,50,1,-10,10,0.5);
```

```

% Gaussmix_gendata.m
%
% This program generates a realization of IID Gaussian mixture noise
% given the mixture parameters and the noise variances.
%
% Input Parameters:
%
%     sig21 - variance of first component of Gaussian mixture
%     sig22 - variance of second component of Gaussian mixture
%     epsilon - mixing probability
%     N       - number of data points desired
%
% Output Parameters:
%
%     x      - array of dimension Nx1 of data samples
%
function x=Gaussmix_gendata(sig21,sig22,epsilon,N)
u=rand(N,1);
for i=1:N
    if u(i)<1-epsilon
        x(i,1)=sqrt(sig21)*randn(1,1);
    else
        x(i,1)=sqrt(sig22)*randn(1,1);
    end
end

```

```
clear all
rand('state',0)
randn('state',0)
epsilon=0.1;
sig21=1;
sig22=100;
N=10;
A=1;
for i=1:10000
x=A+Gaussmix_gendata(sig21,sig22,epsilon,N);
Ahat(i,1)=mean(x);
end
```

```
mse=sum((Ahat-A).^2)/10000  
var_theory=((1-epsilon)*sig21+epsilon*sig22)/N
```

```
detection_demo.m
```

```
% detection_demo.m
%
% This program compares the performance of an autocorrelator detector
% with that of a matched filter and an energy detector by computing a
% receiver operating characteristic (ROC). The signal is
% a sinusoid of unknown frequency embedded in white Gaussian noise.
%
%
% Input parameters:
%
% f0 - frequency of sinusoidal signal (0<f0<0.25)
% N - number of samples in data record
% varw - variance of white Gaussian noise
% nreal - number of realizations used to generate ROC
%
% Output parameters:
%
% Pfa - array of dimension ngam x 1 containing Pfa values
% Pd - array of dimension ngam x 1 containing Pd values
%
% Verification of program:
```

```

%
% The exact values for the various detectors should be:
% for the autocorrelator, Pfa_ac(25)=0.0981, Pd_ac(25)=0.5069
% for the matched filter, Pfa_mf(25)=0.1575, Pd_mf(25)=0.9540
% for the energy detector, Pfa_ed(25)= 0.0428, Pd_ed(25)= 0.2270
%
% External programs required:
%
% roccurve.m - used to compute the ROC
% Q.m - used to compute Q function (see [1, pg. 50])
% Qinvg.m - used to compute inverse Q function (see [1, pg. 51])
% Qchipr2 - used to compute right-tail-probability for
%             central and noncentral chi-squared probability density
%             function (see [1, pg. 55])
%
clear all
close all
timestart=clock; % clock time when program begins
randn('state',0) % set random number generator to fixed initial state to
% generate same set of noise samples each time this program is run
rand('state',0)
f0=0.025;
% f0=0.15; % change to this frequency to show sensitivity to frequency
N=25;
n=[0:N-1]'; % integer time samples
s=cos(2*pi*f0*n); % generate signal
varw=2;
nreal=10000;

```

```

for i=1:nreal % compute realizations of detector test statistics
    w=sqrt(varw)*randn(N,1); % generate white Gaussian noise
%
%   w=sqrt(12*varw)*(rand(N,1)-0.5); % generate white uniform noise to show
%                                         % effect of noise statistics
    x=s+w;                                % generate signal plus noise
    Tac_0(i,1)=sum(w(1:N-1).*w(2:N)); % autocorrelation test statistic for
%                                         % noise only - see (7B.1)
    Tac_1(i,1)=sum(x(1:N-1).*x(2:N)); % autocorrelation test statistic for
%                                         % signal plus noise
    Tmf_0(i,1)=w'*s; % matched filter test statistic - see (7B.2)
    Tmf_1(i,1)=x'*s;
    Ted_0(i,1)=w'*w; % energy detector test statistic - see (7B.3)
    Ted_1(i,1)=x'*x;
end
ngam=50; % number of thresholds used to determine (Pfa,Pd) pairs
[Pfa_ac,Pd_ac,gam]=roccurve(Tac_0,Tac_1,ngam);
[Pfa_mf,Pd_mf,gam]=roccurve(Tmf_0,Tmf_1,ngam);
[Pfa_ed,Pd_ed,gam]=roccurve(Ted_0,Ted_1,ngam);
plot(Pfa_ac,Pd_ac,Pfa_mf,Pd_mf,Pfa_ed,Pd_ed,[0:0.01:1]',[0:0.01:1]')
xlabel('Probability of false alarm (Pfa)')
ylabel('Probability of detection (Pd)')
legend('autocorrelator','matched filter','energy detector', 'lower bound')
title('detection\_demo.m')
grid
d2=s'*s/varw; % This is deflection coefficient - see (7B.5)
Pfa_mft=0.1; Pd_mft=Q(Qinv(Pfa_mft)-sqrt(d2)) % matched filter theoretical
% Pd - see (7B.4)
gamma=61;epsilon=0.01;
Pfa_edt=Qchipr2(N,0,gamma/varw,epsilon) % energy detector theoretical
% Pfa - see (7B.6)
Pd_edt=Qchipr2(N,d2,gamma/varw,epsilon) % energy detector theoretical
% Pd - see (7B.7)
runtime=clock-timestart % total run time

```

```

N=500;
L=20; % length of data for each block
a=[0 0.9025]'; % AR filter parameters
sig2u=1; % AR excitation noise variance
for i=1:30
x=ARgendata(a,sig2u,N); % generate N points of data for each outcome
% In the PSD_est_avper.m subprogram replace
% Pper=(1/L)*abs(fftshift(fft(y,1024))) .^2; by
% Pper=(1/(hamming(L)'*hamming(L))...
% *abs(fftshift(fft(y.*hamming(L),Nfft))) .^2;
[freq,Pavper(:,i)]=PSD_est_avper(x,L,1024,0,0);
end
Pm=mean(Pavper'); % find mean of the outcomes
Pavper_mean=Pm'; % convert to column vector for plotting
[powsd_true,freq]=arpsd(a,sig2u,1024); % compute the true PSD
plot(freq,10*log10(Pavper_mean),freq,10*log10(powsd_true)) % plot the
% true PSD and the mean of the averaged periodogram
grid

```

```
randn('state',0) % initialize Gaussian random number generator  
N=500;  
a=[0 0.9025]'; % set AR filter parameters  
sig2u=1; % set AR excitation noise variance  
[powsd_true,freq]=ARpsd(a,sig2u,1024); % compute the true PSD  
x=ARgenda(a,sig2u,N); % generate N samples of data  
% put a loop here and plot each spectral estimate as L is increased  
%-----  
L=??; % length of data block  
[freq,Pavper]=PSD_est_avper(x,L,1024,0,0); % call to subprogram  
%-----
```

```
Fs=360; % sampling rate in samples/sec
Nfft=4096; % length of FFT
freq=[0:Nfft-1]'/Nfft; % frequencies at which FFT is computed,
% 0 <= freq <1
N=length(QRS_complex); % length of QRS complex
per=(1/N)*abs(fft(QRS_complex,Nfft)).^2; % compute periodogram
figure
plot(Fs*freq(1:Nfft/4),10*log10(per(1:Nfft/4))) % plot periodogram
% over discrete frequency range 0<= freq < 0.25 or analog frequency
% range 0 <= F < 90 Hz.
xlabel('F (Hz)')
ylabel('Periodogram (dB)')
grid
```

```

% matlabexample.m
%
% This program computes and plots samples of a sinusoid
% with amplitudes 1, 2, and 4. If desired, the sinusoid can be
% clipped to simulate the effect of a limiting device.
% The frequency is 1 Hz and the time duration is 10 seconds.
% The sample interval is 0.1 seconds. The code is not efficient but
% is meant to illustrate MATLAB statements.
%
clear all % clear all variables from workspace
delt=0.01; % set sampling time interval
F0=1; % set frequency
t=[0:delt:10]'; % compute time samples 0,0.01,0.02,...,10
A=[1 2 4]'; % set amplitudes
clip='yes'; % set option to clip
for i=1:length(A) % begin computation of sinusoid samples
    s(:,i)=A(i)*cos(2*pi*F0*t+pi/3); % note that samples for sinusoid
                                         % are computed all at once and
                                         % stored as columns in a matrix
    if clip=='yes' % determine if clipping desired
        for k=1:length(s(:,i)) % note that number of samples given as
                               % dimension of column using length
                               % command
            if s(k,i)>3 % check to see if sinusoid sample exceeds 3
                s(k,i)=3; % if yes, then clip
            elseif s(k,i)<-3 % check to see if sinusoid sample is less
                s(k,i)=-3; % than -3 if yes, then clip
            end
        end
    end
end
figure % open up a new figure window
plot(t,s(:,1),t,s(:,2),t,s(:,3)) % plot sinusoid samples versus time
                                    % samples for all three sinusoids
grid % add grid to plot
xlabel('time, t') % label x-axis
ylabel('s(t)') % label y-axis
axis([0 10 -4 4]) % set up axes using axis([xmin xmax ymin ymax])
legend('A=1','A=2','A=4') % display a legend to distinguish
                           % different sinusoids

```