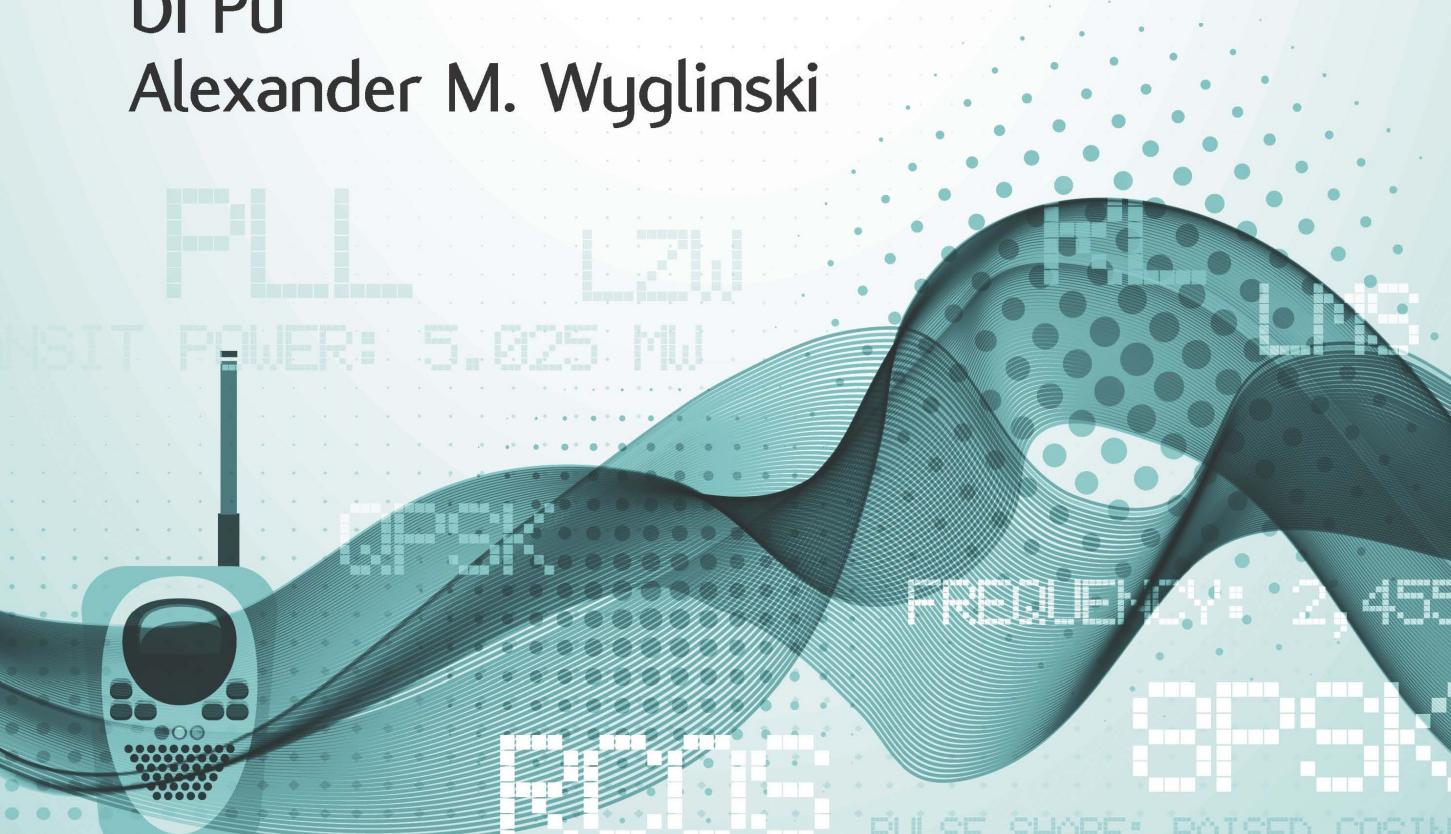




mobile communications series

Digital Communication Systems Engineering with Software-Defined Radio

Di Pu
Alexander M. Wyglinski



Digital Communication Systems

Engineering with

Software-Defined Radio

To my parents, Lingzhen and Xuexun
DP

*To my wife, Jen, my parents, Ted and Barbara,
and my sisters Laura and Joanne*
AMW

Digital Communication Systems Engineering with Software-Defined Radio

Di Pu
Alexander M. Wyglinski



**ARTECH
HOUSE**

BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data

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

British Library Cataloguing in Publication Data

A catalog record for this book is available from the British Library.

ISBN-13: 978-1-60807-525-6

Cover design by Adam Renvoize

© 2013 Artech House

Additional material may be found at <http://www.sdr.wpi.edu/>. All rights reserved. Printed and bound in the United States of America. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Artech House cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

10 9 8 7 6 5 4 3 2 1

To my parents, Lingzhen and Xuexun
DP

*To my wife, Jen, my parents, Ted and Barbara,
and my sisters Laura and Joanne*
AMW

Contents

Preface

xiii

CHAPTER 1

What Is an SDR?	1
1.1 Historical Perspective	1
1.2 Microelectronics Evolution and its Impact on Communications Technology	2
1.2.1 SDR Definition	3
1.3 Anatomy of an SDR	5
1.3.1 Design Considerations	6
1.4 Build it and they Will Come	7
1.4.1 Hardware Platforms	8
1.4.2 SDR Software Architecture	10
1.5 Chapter Summary	13
1.6 Additional Readings	13
References	13

CHAPTER 2

Signals and Systems Overview	15
2.1 Signals and Systems	15
2.1.1 Introduction to Signals	15
2.1.2 Introduction to Systems	16
2.2 Fourier Transform	18
2.2.1 Introduction and Historical Perspective	19
2.2.2 Definition	20
2.2.3 Properties	20
2.3 Sampling Theory	23
2.3.1 Uniform Sampling	24
2.3.2 Frequency Domain Representation of Uniform Sampling	24
2.3.3 Nyquist Sampling Theorem	26
2.3.4 Sampling Rate Conversion	27
2.4 Pulse Shaping	30
2.4.1 Eye Diagrams	32
2.4.2 Nyquist Pulse Shaping Theory	32
2.4.3 Two Nyquist Pulses	34
2.5 Filtering	39
2.5.1 Ideal Filter	39

2.5.2 Z-Transform	39
2.5.3 Digital Filtering	42
2.6 Chapter Summary	47
2.7 Problems	47
References	50

CHAPTER 3

Probability Review	53
3.1 Fundamental Concepts	53
3.1.1 Set Theory	53
3.1.2 Partitions	54
3.1.3 Functions	56
3.1.4 Axioms and Properties of Probability	57
3.1.5 Conditional Probability	57
3.1.6 Law of Total Probability and Bayes' Rule	58
3.1.7 Independence	59
3.2 Random Variables	59
3.2.1 Discrete Random Variables	60
3.2.2 Continuous Random Variables	65
3.2.3 Cumulative Distribution Functions	69
3.2.4 Central Limit Theorem	70
3.2.5 The Bivariate Normal	71
3.3 Random Processes	72
3.3.1 Statistical Characteristics of Random Processes	74
3.3.2 Stationarity	76
3.3.3 Gaussian Processes	77
3.3.4 Power Spectral Density and LTI Systems	78
3.4 Chapter Summary	79
3.5 Additional Readings	80
3.6 Problems	80
References	88

CHAPTER 4

Digital Transmission Fundamentals	89
4.1 What is Digital Transmission?	89
4.1.1 Source Encoding	91
4.1.2 Channel Encoding	92
4.2 Digital Modulation	94
4.2.1 Power Efficiency	94
4.2.2 Pulse Amplitude Modulation	95
4.2.3 Quadrature Amplitude Modulation	98
4.2.4 Phase Shift Keying	99
4.2.5 Power Efficiency Summary	104
4.3 Probability of Bit Error	105
4.3.1 Error Bounding	107

4.4	Signal Space Concept	108
4.5	Gram-Schmidt Orthogonalization	110
4.6	Optimal Detection	113
4.6.1	Signal Vector Framework	114
4.6.2	Decision Rules	116
4.6.3	Maximum Likelihood Detection in an AWGN Channel	117
4.7	Basic Receiver Realizations	119
4.7.1	Matched Filter Realization	120
4.7.2	Correlator Realization	122
4.8	Chapter Summary	124
4.9	Additional Readings	125
4.10	Problems	125
	References	130

CHAPTER 5

	Basic SDR Implementation of a Transmitter and a Receiver	131
5.1	Software Implementation	131
5.1.1	Repetition Coding	132
5.1.2	Interleaving	134
5.1.3	BER Calculator	135
5.1.4	Receiver Implementation over an Ideal Channel	136
5.2	USRP Hardware Implementation	137
5.2.1	Frequency Offset Compensation	138
5.2.2	Finding Wireless Signals: Observing IEEE 802.11 WiFi Networks	140
5.2.3	USRP In-phase/Quadrature Representation	141
5.3	Open-Ended Design Project: Automatic Frequency Offset Compensator	145
5.3.1	Introduction	145
5.3.2	Objective	146
5.3.3	Theoretical Background	147
5.4	Chapter Summary	149
5.5	Problems	149
	References	152

CHAPTER 6

	Receiver Structure and Waveform Synthesis of a Transmitter and a Receiver	153
6.1	Software Implementation	153
6.1.1	Observation Vector Construction	153
6.1.2	Maximum-Likelihood Decoder Implementation	157
6.1.3	Correlator Realization of a Receiver in Simulink	159
6.2	USRP Hardware Implementation	162
6.2.1	Differential Binary Phase-Shift Keying	163
6.2.2	Differential Quadrature Phase-Shift Keying	166
6.2.3	Accelerate the Simulink Model that Uses USRP Blocks	166
6.3	Open-Ended Design Project: Frame Synchronization	167
6.3.1	Frame Synchronization	167

6.3.2	Barker Code	168
6.3.3	Simulink Models	168
6.3.4	Hints for Implementation	172
6.3.5	Hints for Debugging	172
6.4	Chapter Summary	172
6.5	Problems	173
	Reference	175

CHAPTER 7

	Multicarrier Modulation and Duplex Communications	177
7.1	Theoretical Preparation	177
7.1.1	Single Carrier Transmission	177
7.1.2	Multicarrier Transmission	181
7.1.3	Dispersive Channel Environment	183
7.1.4	OFDM with Cyclic Prefix	185
7.1.5	Frequency Domain Equalization	186
7.1.6	Bit and Power Allocation	187
7.2	Software Implementation	189
7.2.1	MATLAB Design of Multicarrier Transmission	189
7.2.2	Simulink Design of OFDM	192
7.3	USRP Hardware Implementation	194
7.3.1	Eye Diagram	194
7.3.2	Matched Filter Observation	195
7.4	Open-Ended Design Project: Duplex Communication	197
7.4.1	Duplex Communication	197
7.4.2	Half-Duplex	198
7.4.3	Time-Division Duplexing	198
7.4.4	Useful Suggestions	199
7.4.5	Evaluation and Expected Outcomes	200
7.5	Chapter Summary	201
7.6	Problems	201
	References	204

CHAPTER 8

	Spectrum Sensing Techniques	207
8.1	Theoretical Preparation	207
8.1.1	Power Spectral Density	207
8.1.2	Practical Issues of Collecting Spectral Data	209
8.1.3	Hypothesis Testing	214
8.1.4	Spectral Detectors and Classifiers	218
8.2	Software Implementation	222
8.2.1	Constructing Energy Detector	222
8.2.2	Observing Cyclostationary Detector	226
8.3	USRP Hardware Experimentation	227
8.4	Open-Ended Design Project: CSMA/CA	230
8.4.1	Carrier Sense Multiple Access	230

8.4.2	Collision Avoidance	231
8.4.3	Implementation Approach	231
8.4.4	Useful Suggestions	232
8.4.5	Evaluation and Expected Outcomes	233
8.5	Chapter Summary	234
8.6	Problems	234
	References	236

CHAPTER 9

	Applications of Software-Defined Radio	239
9.1	Cognitive Radio and Intelligent Wireless Adaptation	239
9.1.1	Wireless Device Parameters	241
9.2	Vehicular Communication Networks	242
9.2.1	VDSA Overview	243
9.2.2	Transmitter Design	244
9.2.3	Receiver Design	245
9.2.4	VDSA Test-Bed Implementation	245
9.3	Satellite Communications	246
9.4	Chapter Summary	250
	References	250

APPENDIX A

	Getting Started with MATLAB and Simulink	253
A.1	MATLAB Introduction	253
A.2	Edit and Run a Program in MATLAB	253
A.3	Useful MATLAB Tools	254
A.3.1	Code Analysis and M-Lint Messages	254
A.3.2	Debugger	255
A.3.3	Profiler	256
A.4	Simulink Introduction	257
A.5	Getting Started in Simulink	257
A.5.1	Start a Simulink Session	257
A.5.2	Start a Simulink Model	257
A.5.3	Simulink Model Settings	258
A.6	Build a Simulink Model	259
A.6.1	Obtain the Blocks	260
A.6.2	Set the Parameters	260
A.6.3	Connect the Blocks	262
A.7	Run Simulink Simulations	264
	References	266

APPENDIX B

	Universal Hardware Driver (UHD)	267
B.1	Setting Up Your Hardware	267
B.2	Installing UHD-Based USRP I/O Blocks	267
B.3	Burning the Firmware to an SD Card	268

B.4 Configure the Ethernet Card	268
B.5 Modify the Iptables	269
B.6 Each Time You Use	269
B.7 Problems with Unicode	269
References	270

APPENDIX C

Data Flow on USRP	271
C.1 Receive Path	271
C.1.1 Situation 1	272
C.1.2 Situation 2	274
C.2 Transmit Path	274
References	276

APPENDIX D

Quick Reference Sheet	277
D.1 LINUX	277
D.1.1 Helpful Commands	277
D.1.2 Modify the Iptables	277
D.2 MATLAB	278
D.2.1 How to Start MATLAB	278
D.2.2 The MATLAB Environment	278
D.2.3 Obtaining Help	278
D.2.4 Variables in MATLAB	278
D.2.5 Vectors and Matrices in MATLAB	279
D.3 USRP2 Hardware	279
D.3.1 XCVR2450 Daughtercard	279
D.3.2 Sampling	280
D.3.3 Clocking	281
D.3.4 DDC and DUC	281
D.4 Differential Phase-Shift Keying (DPSK)	282
Reference	282

APPENDIX E

Trigonometric Identities	283
About the Author	285
Index	287

Preface

The communications sector has witnessed a significant paradigm shift over the past several decades with respect to how wireless transceivers and algorithms are implemented for a wide range of applications. Until the 1980s, almost all wireless communication systems were based on relatively static, hard-wired platforms using technologies such as application-specific integrated circuits (ASICs). Nonetheless, with numerous advances being made in the areas of digital processing technology, digital-to-analog converters (DACs), analog-to-digital converters (ADCs), and computer architectures, communication engineers started rethinking how wireless transceivers could be implemented entirely in the digital domain, resulting in both digital communications and digital signal processing algorithms being constructed completely in programmable digital logic and software modules. Subsequently, this rethinking by the communications sector has given rise to *software-defined radio* (SDR) technology, where all the baseband processing of the wireless transceiver is entirely performed in the digital domain (e.g., programmable logic, software source code). Nevertheless, despite the fact that SDR technology has been around for nearly three decades, undergraduate and graduate courses focusing on digital communications have been mostly taught using educational methodologies and techniques that were either based on transmission technologies from half a century ago or entirely lacked any physical hardware context whatsoever. Consequently, there exists a significant need by the wireless community for an educational approach that leverages SDR technology in order to provide individuals with the latest insights on digital communication systems; this book was primarily written in order to satisfy this need.

The objective of this book is to provide the reader with a “hands-on” educational approach employing software-defined radio (SDR) experimentation to facilitate the learning of digital communication principles and the concepts behind wireless data transmission, and provide an introduction to wireless access techniques. Based on four years of educational experiences by the authors with respect to the instruction of a digital communications course for senior undergraduate and junior graduate students at Worcester Polytechnic Institute (WPI) based on SDR experimentation, it is expected that enabling individuals to prototype and evaluate actual digital communication systems capable of performing “over-the-air” wireless data transmission and reception will provide them with a first-hand understanding of the design tradeoffs and issues associated with these systems, as well as provide them with a sense of actual “real world” operational behavior. Given that the learning process associated with using an actual SDR platform for prototyping digital communication systems is often lengthy, tedious, and complicated, it is difficult to employ this tool within a relatively short period of time, such as a typical undergraduate course (e.g., 7–14 weeks). Consequently, this book contains a collection of preassembled Simulink experiments along with several software examples to

enable the reader to successfully implement these designs over a short period of time, as well as simultaneously synthesize several key concepts in digital communication system theory. Furthermore, in order to provide the reader with a solid grasp of the theoretical fundamentals needed to understand and implement digital communication systems using SDR technology, three overview chapters near the beginning of this book provides a quick summary of essential concepts in signals and systems, probability theory, and basic digital transmission.

The intended readers for this book are senior undergraduate and entry-level graduate students enrolled in courses in digital communication systems, industry practitioners in the telecommunications sector attempting to master software-defined radio technology, and wireless communication researchers (government, industry, academia) seeking to verify designs and ideas using SDR technology. It is assumed that the reader possesses an electrical and computer engineering background with a concentration in signals, systems, and information. This book, along with its corresponding collection of software examples and 28 sets of lecture slides (which can be viewed at <http://www.artechhouse.com> and at <http://www.sdr.wpi.edu/>), is designed to introduce the interested reader to concepts in digital communications, wireless data transmission, and wireless access via a structured, “hands-on” approach.

In order to efficiently utilize this book for learning digital communications with SDR technology, it is recommended that the interested individual reads this book in a sequential manner. Starting with an overview of SDR technology in Chapter 1, the reader will then be provided a quick refresher of signals and systems, probability theory, and basic digital communications in Chapter 2, Chapter 3, and Chapter 4. Following this review of the fundamentals, the next four chapters, namely, Chapters 5 through 8, focus on performing basic wireless data transmission, designing different types of wireless receivers, introducing the concept of multicarrier modulation, and devising spectrum sensing techniques. Finally, Chapter 9 provides an overview of the various advanced applications where SDR technology has made a significant impact. A collection of appendixes at the end of this book provides the reader with instant access to information in order to help them quickly overcome the relatively steep learning curve associated with SDR implementations. In addition to the hands-on experiments, end-of-chapter problems also provide the readers with an opportunity to strengthen their theoretical understanding of the concepts covered in this book. Note that the 28 sets of lecture slides are closely related to the sequence of topics presented in this book, where each set of slides is intended for a lecture of approximately 60–90 minutes in duration. As a result, this book and its associated materials is well-suited for a senior-level undergraduate course on digital communications taught over a 7-week academic term or a 14-week academic semester.

This book was made possible by the extensive support of numerous individuals and organizations throughout the duration of this project. First, we are deeply indebted to our collaborators and sponsors at the MathWorks in Natick, MA, for their constant support. In particular, we would like to sincerely thank Don Orofino, Kate Fiore, Mike McLernon, and Ethem Sozer for their endless guidance and assistance throughout the entire project. Second, we would like to sincerely thank Daniel Cullen for his assistance in getting this educational approach “off the ground” in 2010 with the first offering of the SDR-based digital communications course on which this book is based. Third, we would like to express our deepest gratitude

to the WPI ECE department, especially its department head, Fred Looft, for being a unwavering supporter in making the course that this book is based on become a reality and a permanent addition to the WPI ECE curriculum. Finally, we would like to thank our families for their support and encouragement.

Di Pu
Worcester Polytechnic Institute, USA

Alexander M. Wyglinski
Worcester Polytechnic Institute, USA

What Is an SDR?

Modern society as we know it today is becoming increasingly dependent on the reliable and seamless wireless exchange of information. For instance, a rapidly growing number of individuals are using smartphones in order to access websites, send and receive emails, view streaming multimedia content, and engage in social networking activities anywhere on the planet at any time. Furthermore, numerous drivers on the road today are extensively relying upon global positioning system (GPS) devices and other wireless navigation systems in order to travel in a new neighborhood or city without the need for a conventional map. Finally, in many hospitals and medical centers across the nation and around the world, the health of a patient is now being continuously monitored using an array of medical sensors attached to him/her that send vital information wirelessly to a computer workstation for analysis by a medical expert.

The enabling technology for supporting any form of wireless information exchange is the *digital transceiver*, which is found in every cell phone, WiFi-enabled laptop, BlueTooth device, and other wireless appliances designed to transmit and receive digital data. Digital transceivers are capable of performing a variety of baseband operations, such as modulation, source coding, forward error correction, and equalization. Although digital transceivers were initially implemented using integrated circuits and other forms of nonprogrammable electronics, the advent of programmable baseband functionality for these digital transceivers is the result of numerous advancements in microprocessor technology over the past several decades. Consequently, there exists a plethora of digital transceiver solutions with a wide range of capabilities and features. In this chapter, we will provide an introduction to software-defined radio (SDR) hardware platforms and software architectures, as well as study how it has evolved over the past several decades to become a transformative communications technology.

1.1 HISTORICAL PERSPECTIVE

The term “software-defined radio” was first coined by Joseph Mitola [1], although SDR technology was available since the 1970s and the first demonstrated SDR prototype was presented in 1988 [2]. However, the key milestone for the advancement of SDR technology took place in the early 1990s with the first publicly funded SDR development initiative called *SpeakEasy I/II* by the U.S. military [3]. The SpeakEasy project was a significant leap forward in SDR technology since it used a collection of programmable microprocessors for implementing more than 10 military communication standards, with transmission carrier frequencies ranging from 2 MHz to 2 GHz, which at that time was a major advancement in communication systems engineering. Additionally, the SpeakEasy implementation allowed for software

upgrades of new functional blocks, such as modulation schemes and coding schemes. The first generation of the SpeakEasy system initially used a Texas Instruments TMS320C40 processor (40 MHz). However, the SpeakEasy II platform was the first SDR platform to involve field programmable gate array (FPGA) modules for implementing digital baseband functionality. Note that given the microprocessor technology at the time, the physical size of the SpeakEasy prototypes were large enough to fit in the back of a truck.



Why were the first SDR platforms physically very large in size?

Following this initial SDR project, research and development activities in this area continued to advance the current state-of-the-art in SDR technology, with one of the outcomes being the *Joint Tactical Radio System* (JTRS), which is a next-generation voice-and-data radio used by the U.S. military and employs the *software communications architecture* (SCA) [4]. Initially developed to support avionics [5] and dynamic spectrum access [6] applications, JTRS is scheduled to become the standard communications platform for the U.S. Army by 2012.

1.2 MICROELECTRONICS EVOLUTION AND ITS IMPACT ON COMMUNICATIONS TECHNOLOGY

The microelectronic industry has rapidly evolved over the past six decades, resulting in numerous advances in microprocessor systems that have enabled many of the applications we take for granted every day. The rate at which this evolution has progressed over time has been characterized by the well-known *Moore's Law*, which defines the long-term trend of the number of transistors that can be accommodated on an integrated circuit. In particular, Moore's Law dictates that the number of transistors per integrated circuit approximately doubles every two years, which subsequently affects the performance of microprocessor systems such as processing speed and memory. One area that the microelectronics industry has significantly influenced over the past half century is the digital communication systems sector, where microprocessor systems have been increasingly employed in the implementation of digital transceivers, yielding more *versatile*, *powerful*, and *portable* communication system platforms capable of performing a growing number of advanced operations and functions. With the latest advances in microelectronics and microprocessor systems, this has given rise to *software-defined radio* (SDR) technology, where baseband radio functionality can be *entirely* implemented in digital logic and software, as illustrated in Figure 1.2.

There exists several different types of microprocessor systems for SDR implementations. For instance, one popular choice is the *general purpose microprocessor*, which is often used in SDR implementations and prototypes due to its high level of flexibility with respect to reconfigurability, as well as due to its ease of implementation regarding new designs. On the other hand, general purpose microprocessors are not specialized for mathematical computations, and they can be potentially power ineffi-

ficient. Another type of microprocessor system, called a *digital signal processor* (DSP), is specialized for performing mathematical computations, implementation of new digital communication modules can be performed with relative ease, and the processor is relatively power efficient (e.g., DSPs are used in cellular telephones). On the other hand, DSPs are not well suited for computationally intensive processes and can be rather slow. Alternatively, *field programmable gate arrays* (FPGAs) are computationally powerful, but power inefficient, and it is neither flexible nor easy to implement new modules. Similarly, *graphics processing units* (GPUs) are extremely powerful computationally but difficult to use and it is difficult to implement new modules as well.

1.2.1 SDR Definition

Given these microprocessor hardware options, let us now proceed with formulating a definition for an SDR platform. An SDR is a class of reconfigurable/reprogrammable radios whose physical layer characteristics can be significantly modified via software changes. It is capable of implementing different functions at different times on the same platform, it defines in software various baseband radio features, (e.g., modulation, error correction coding), and it possesses some level of software control over RF front-end operations, (e.g., transmission carrier frequency). Since all of the baseband radio functionality is implemented using software, this implies that potential design options and radio modules available to the SDR platform can be readily stored in memory and called upon when needed for a specific application, such as a specific modulation, error correction coding, or other functional block needed to ensure reliable communications. Note that due to the programmable nature of the SDR platform and its associated baseband radio modules, these functional blocks can potentially be changed in real-time and the operating parameters of functional blocks can be adjusted either by a human operator or an automated process.

Although definitions may vary to a certain degree regarding what constitutes an SDR platform, several key characteristics that generally define an SDR can be summarized by the following list [7]:

- *Multifunctionality*: Possessing the ability to support multiple types of radio functions using the same digital communication system platform.
- *Global mobility*: Transparent operation with different communication networks located in different parts of the world (i.e., not confined to just one standard).
- *Compactness and power efficiency*: Many communication standards can be supported with just one SDR platform.
- *Ease of manufacturing*: Baseband functions are a software problem, not a hardware problem.
- *Ease of upgrading*: Firmware updates can be performed on the SDR platform to enable functionality with the latest communication standards.

One of the most commonly leveraged SDR characteristic is that of multifunctionality, where the SDR platform employs a particular set of baseband radio modules based on how well the communication system will perform as a result of that configuration. To illustrate how multifunctionality might work on an SDR platform, suppose we employ

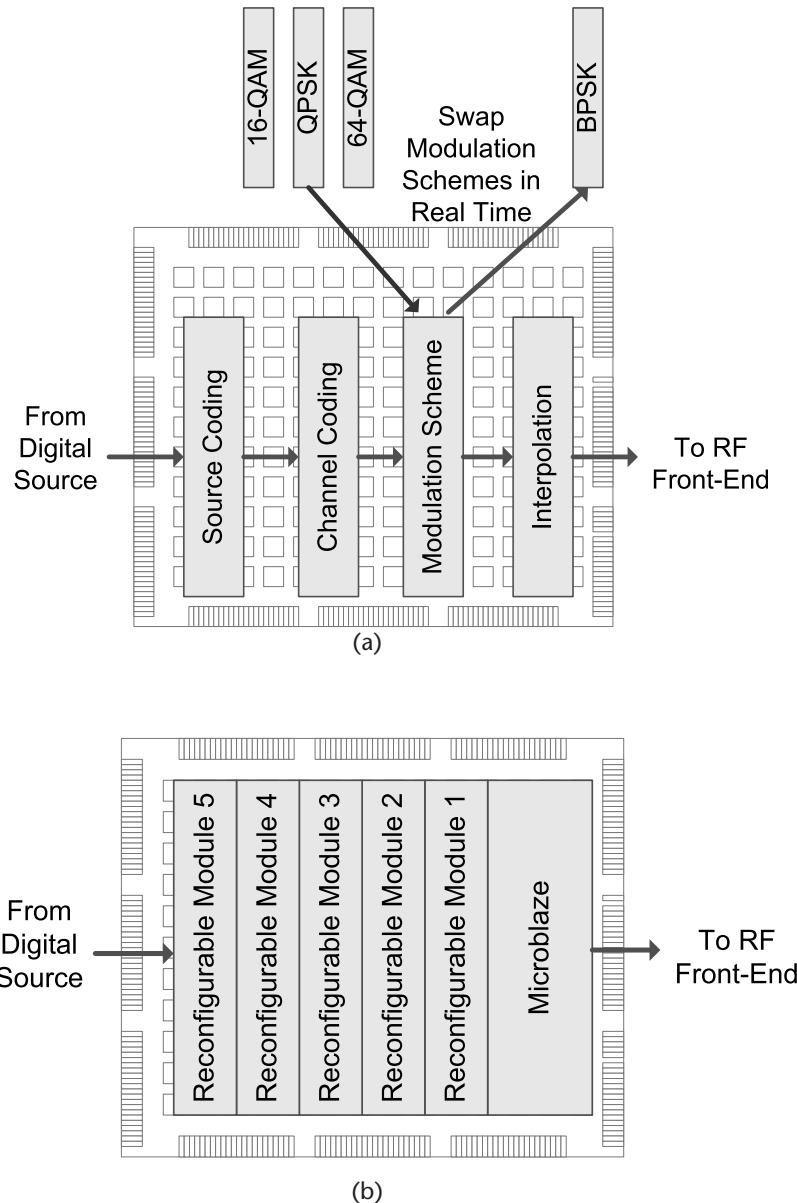


Figure 1.1 An illustration of multifunctionality in an FPGA-based SDR platform. (a) Swapping different modulation schemes on the same FPGA. (b) Digital communication transmitter chain on an FPGA using the *multiprocessor system-on-chip* (MPSoC) concept.

an FPGA-based implementation consisting of a baseband radio architecture shown in Figure 1.1(a). Now, let us assume that the transmission environment has changed and the SDR platform has determined that a change in the modulation scheme is needed to continue guaranteeing sufficient transmission performance. Consequently, using the *multiprocessor system-on-chip* (MPSoC) concept, the SDR platform quickly swaps out the BPSK modulation software module and replaces it with a QPSK modulation software module, allowing for the continued operation of the SDR platform with minimal interruption. The way that the MPSoC concept works is shown in Figure 1.1(b), where

a *microblaze* on the FPGA acts as a mini-microprocessor that makes decisions on what parts of the FPGA to activate, thus enabling these specific baseband radio modules.

1.3 ANATOMY OF AN SDR

An SDR system is a complex device that performs several complicated tasks simultaneously in order to enable the seamless transmission and reception of data. In general, digital communications systems consists of an interdependent sequence of operations responsible for taking some type of information, whether it is human speech, music, or video images, and transmit it over the air to an awaiting receiver for processing and decoding into a reconstructed version of the original information signal. If the original information is analog, it must first be digitized using techniques such as quantization in order for us to obtain a binary representation of this information. Once in a binary format, the transmitter digitally processes this information and converts it into an electromagnetic sinusoidal waveform that is uniquely defined by its physical characteristics, such as its signal amplitude, carrier frequency, and phase. At the other end of the communications link, the receiver is tasked with correctly identifying the physical characteristics of the intercepted electromagnetic waveform transmitted across a potentially noisy and distortion-filled channel, and ultimately returning the intercepted signal back into the correct binary representation. The basic building block of a digital communication system is shown in Figure 1.2.

We see in Figure 1.2 that the input to the transmitter and output of the receiver originate from or are fed into a *digital source* and *digital sink*, respectively. These two blocks represent the source and destination of the digital information to be communicated between the transmitter and receiver. Once the binary information

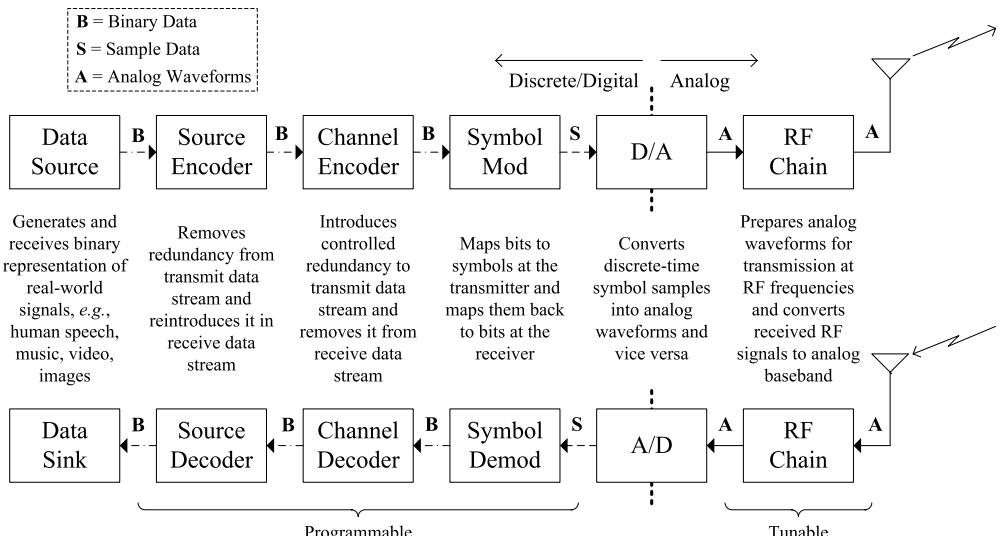


Figure 1.2 An illustration describing some of the important components that constitute a modern digital communications system. Note that for a SDR-based implementation, those components indicated as “programmable” can be realized in either programmable logic or software.

is introduced to the transmitter, the first task performed is to remove all redundant/repeating binary patterns from the information in order to increase the efficiency of the transmission. This is accomplished using the *source encoder* block, which is designed to strip out all redundancy from the information. Note that at the receiver, the *source decoder* reintroduces the redundancy in order to return the binary information back to its original form. Once the redundancy has been removed from the binary information at the transmitter, a *channel encoder* is employed to introduce a controlled amount of redundancy to the information stream in order to protect it from potential errors introduced during the transmission process across a noisy channel. A *channel decoder* is used to remove this controlled redundancy and return the binary information back to its original form. The next step at the transmitter is to convert the binary information into unique electromagnetic waveform properties such as amplitude, carrier frequency, and phase. This is accomplished using a mapping process called *modulation*. Similarly, at the receiver the *demodulation* process converts the electromagnetic waveform back into its respective binary representation. Finally, the discrete samples outputted by the modulation block are resampled and converted into a baseband analog waveform using a *digital-to-analog* (D/A) converter before being processed by the *radio frequency* (RF) front-end of the communication system and upconverted to an RF carrier frequency. At the receiver, the reverse operation is performed, where the intercepted analog signal is downconverted by the RF front-end to a baseband frequency before being sampled and processed by an *analog-to-digital* (A/D) converter.

1.3.1 Design Considerations

Given the complexity of an SDR platform and its respective components, as described in the previous section as well as in Figure 1.2, it is important to understand the limitations of a specific SDR platform and how various design decisions may impact the performance of the resulting prototype. For instance, it is very desirable to have real-time baseband processing for spectrum sensing and agile transmission operations with high computational throughput and low latency. However, if the microprocessor being employed by the SDR platform is not sufficiently powerful enough in order to support the computational operations of the digital communication system, one needs to reconsider either the overall transceiver design or the requirements for low latency and high throughput. Otherwise, the SDR implementation will fail to operate properly, yielding transmission errors and poor communication performance. An example of when the microprocessor is not capable of handling the computational needs of the SDR platform is shown in Figure 1.3, where the transmission being produced by the SDR platform is occurring in bursts, yielding several periods of transmitted signal interspersed with periods of no transmissions. Note that such a bursty transmission would be extremely difficult to handle at the receiver due to the intermittent nature of the signal being intercepted.

Other design considerations to think about when devising digital communication systems based on an SDR platform include the following:

- The integration of the physical and network layers via a real-time protocol implementation on an embedded processor. Note that most communication

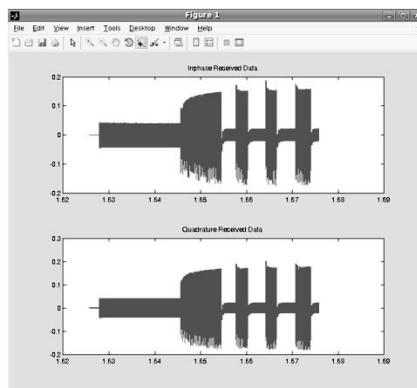


Figure 1.3 An example of sampling rate errors when the SDR platform is unable to keep up with the transmission of the data. Note that the bursts can result in significant difficulty at the receiver with respect to the interception and decoding of the received signal.

systems are divided into logically separated layers in order to more readily facilitate the design of the communication system. However, it is imperative that each layer is properly designed due to the strong interdependence between all the layers.

- Ensuring that a sufficiently wide bandwidth radio front-end exists with agility over multiple subchannels and scalable number of antennas for spatial processing. Given how many of the advanced communication system designs involve the use of multiple antennas and wideband transmissions, it is important to know what the SDR hardware is capable of doing with respect to these physical attributes.
- Many networks employing digital communication systems possess a centralized architecture for controlling the operations of the overall network (e.g., control channel implementation). Knowing your radio network architecture is important since it will dictate what sort of operations are essential for one digital transceiver to communicate with another.
- The ability to perform controlled experiments in different environments (e.g., shadowing and multipath, indoor and outdoor environments) is important for the sake of demonstrating the reliability of a particular SDR implementation. In other words, if an experiment involving an SDR prototype system is conducted twice in a row in the exact same environment and using the exact same operating parameters, it is expected that the resulting output and performance should be the same. Consequently, being able to perform controlled experiments provides the SDR designer with a “sanity check” capability.
- Reconfigurability and fast prototyping through a software design flow for algorithm and protocol description.

1.4 BUILD IT AND THEY WILL COME

Given our brief overview of SDR technology and its definition, as well as a survey of various microprocessor design options and constraints available when implementing an SDR platform, we will now focus our attention on several well-known

SDR hardware implementations published in the open literature. Note that when designing a complete SDR system from scratch, it is very important to have both a hardware platform that is both sufficiently programmable and computationally powerful, as well as a software architecture that can allow a communication system designer to implement a wide range of different transceiver realizations. In this section, we will first study some of the well-known SDR hardware platforms before talking about some of the available SDR software architectures.

1.4.1 Hardware Platforms

Exploration into advanced wireless communication and networking techniques require highly flexible hardware platforms. As a result, SDR is very well suited due to its rapidly reconfigurable attributes, which allows for controlled yet realistic experimentation. Thus, the use of real-time test bed operations enables a large set of experiments for various receiver settings, transmission scenarios, and network configurations. Furthermore, SDR hardware provides an excellent alternative for comprehensive evaluation of communication systems operating within a networked environment, whereas Monte Carlo simulations can be computationally exhaustive and are only as accurate as the devised computer model. In this section, we will study several well-known SDR hardware platforms used by the wireless community for research and experimentation.

One of the most well-known of all SDR hardware platforms is the *Universal Software Radio Peripheral* (USRP) concept that was introduced by Matt Ettus, founder and president of Ettus Research LLC, which is considered to be a relatively inexpensive hardware for enabling SDR design and development [8]. All the base-band digital communication algorithms and digital signal processing are conducted on a computer workstation “host,” where the USRP platform acts as a radio peripheral allowing for over-the-air transmissions and the `libusrp` library file defines the interface between the USRP platform and the host computer workstation. Note that the USRP design is open source, which allows for user customization and fabrication. Furthermore, USRP platform design is modular in terms of the supported RF front-ends, referred to as *daughtercards*. We will now talk about two types of USRP platforms: the *USRP1* and *USRP2*.

The *Universal Software Radio Peripheral-Version 1* (USRP1) was designed and manufactured by Ettus Research LLC for a variety of different communities interested in an inexpensive SDR platform. The USRP1 consists of a USB interface between host computer workstation and USRP1 platform, which resulted in a data bottleneck due to the low data rates supported by the USB connection. The USRP1 supports up to two RF transceiver daughtercards, possesses an Altera Cyclone EP1C12Q240C8 FPGA for performing sampling and filtering, contains four high-speed analog-to-digital converters, each capable of 64 MS/s at a resolution of 12 bits, with an 85 dB SFDR (AD9862), and contains four high-speed digital-to-analog converters, each capable of 128 MS/s at a resolution of 14 bits, with 83 dB SFDR (AD9862).

Following the success of the USRP1, Ettus Research LLC officially released the *Universal Software Radio Peripheral-Version 2* (USRP2) platform in September 2008, as shown in Figure 1.4(a). The USRP2 platform provides a more capable

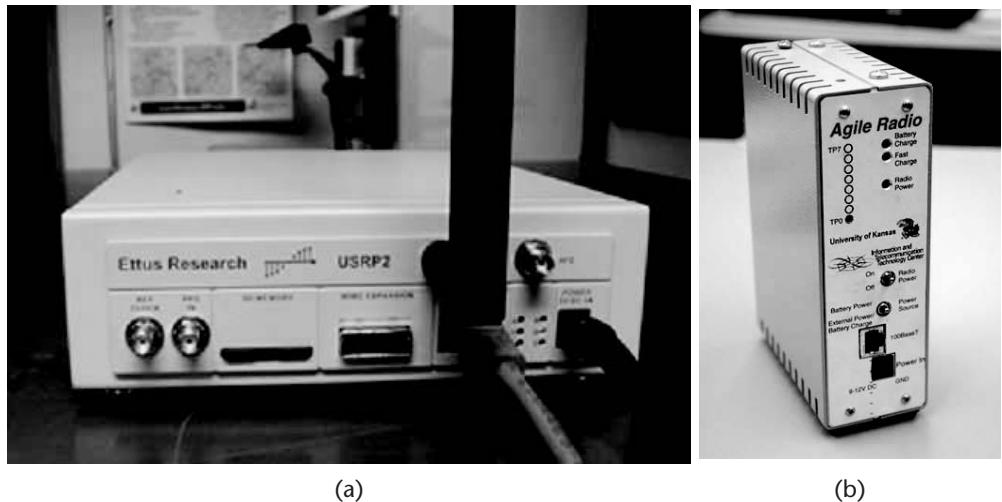


Figure 1.4 Examples of software-defined radio platforms. (a) Front view of a Universal Software Radio Peripheral-Version 2 (USRP2) software-defined radio platform by Ettus Research LLC. (b) Front view of a Kansas University Agile Radio (KUAR) software-defined radio platform.

SDR device for enabling digital communication system design and implementation. The USRP2 features include a gigabit Ethernet interface between host computer workstation and USRP2 platform, supports only one RF transceiver daughter card, possesses a Xilinx Spartan 3-2000 FPGA for performing sampling and filtering, contains two 100 MS/s, 14 bits, analog-to-digital converters (LTC2284), with a 72.4 dB SNR and 85 dB SFDR for signals at the Nyquist frequency, contains two 400 MS/s, 16 bits, digital-to-analog converters (AD9777), with a 160 MS/s without interpolation, and up to 400 MS/s with 8x interpolation, and is MIMO-capable for supporting the processing of digital communication system designs employing multiple antennas.

The radio frequency (RF) front-ends are usually very difficult to design and are often limited to a narrow range of transmission carrier frequencies. This is due to the fact that the properties of the RF circuit and its components change across different frequencies and that the RF filters are constrained in the sweep frequency range. Consequently, in order to support a wide range of transmission carrier frequencies, both the USRP1 and USRP2 platforms can use an assortment of modular RF daughter cards, such as the following:

- *BasicTX*: A transmitter that supports carrier frequencies within 1–250 MHz;
- *BasicRX*: A receiver that supports carrier frequencies within 1–250 MHz;
- *RFX900*: A transceiver that supports carrier frequencies within 800–1000 MHz with a 200+mW output;
- *RFX2400*: A transceiver that supports carrier frequencies within 2.3–2.9 GHz with a 20+mW output;
- *XCVR2450*: A transceiver that supports carrier frequencies within two bands, namely, 2.4–2.5 GHz with an output of 100+mW and 4.9–5.85 GHz with an output of 50+mW.

Another well-known SDR hardware platform was the *Kansas University Agile Radio* (KUAR), which is a small form factor SDR platform containing a Xilinx Virtex-II Pro FPGA board and a PCI Express 1.4 GHz Pentium-M microprocessor, as shown in Figure 1.4(b) [9, 10]. For its size and capability, the KUAR was one of the leading SDR implementations in its day, incorporating substantial computational resources as well as wideband frequency operations. Around the same time period, the Berkeley BEE2 was designed as a powerful reconfigurable computing engine with five Xilinx Virtex-II Pro FPGAs on a custom-built emulation board [11]. The Berkeley Wireless Research Center (BWRC) cognitive radio test-bed hardware architecture consists of the BEE2, several reconfigurable 2.4-GHz radio modems, and fiber link interfaces for connections between the BEE2 and the radios modems. The software architecture consists of Simulink-based design flow and BEE2 specific operating system, which provides an integrated environment for implementation and simple data acquisition during experiments.

With respect to compact SDR platforms, Motorola developed and built a 10-MHz, 4-GHz CMOS-based small form factor cognitive radio platform prototype [12]. Fundamentally flexible, with a low-power transceiver radio frequency integrated circuit (RFIC) at the core of this experimental platform, this prototype can receive and transmit signals of many wireless protocols, both standard and experimental. Carrier frequencies from 10 MHz to 4 GHz with channel bandwidths from 8 kHz to 20 MHz were supported. Similarly, the *Maynooth Adaptable Radio System* (MARS) is a custom-built small form factor SDR platform [13]. The MARS platform had the original objectives of being a personal computer connected radio front-end where all the signal processing is implemented on the computer's general-purpose processor. The MARS platform was designed to deliver performance equivalent to that of a future base station and the wireless communication standards in the 1700-MHz to 2450-MHz frequency range. Furthermore, the communication standards GSM1800, PCS1900, IEEE 802.11b/g, and UMTS (TDD and FDD) are also supported.

Rice University *Wireless Open Access Research Platform* (WARP) radios include a Xilinx Virtex-II Pro FPGA board as well as a MAX2829 transceiver [14], while the Lyrtech Small Form Factor SDR is developed by a company from the Canadian Province of Québec that leverages industrial collaborations between Texas Instruments and Xilinx in order to produce these high-performance SDR platforms that consist of an array of different microprocessor technology [15]. Finally, Epiq Solutions recently released the MatchStiq SDR platform, which is a powerful yet very compact form factor SDR platform capable of being deployed in the field to perform a variety of wireless experiments, including their inclusion onboard vehicles such as automobiles and unmanned aerial vehicles [16].

1.4.2 SDR Software Architecture

Given the programmable attributes of an SDR platform, it is vitally important to also develop an efficient and reliable software architecture that would operate on these platforms in order to perform the various data transmission functions we expect from a wireless communications system. In this section, we will review some of the SDR software architectures currently available for use with a wide variety of SDR hardware platforms.

One of the first Simulink interfaces to the USRP2 platform was implemented as part of an MS thesis at WPI and generously sponsored by the MathWorks [17]. In this research project, the focus was on creating a Simulink blockset capable of communicating with the USRP2 libraries, which can then allow for communications with the USRP2 platform itself. The resulting blocksets from this thesis research are shown in Figure 1.5. By creating a Simulink interface to this SDR hardware, it is expected that the existing signal processing libraries provided by the MathWorks can be extensively leveraged in order to create actual digital communications systems capable of performing over-the-air data transmission with other USRP2 platforms.

The architecture of the Simulink transmit and receiver blocks are shown in Figure 1.6. In Figure 1.6(b), we observe how the Simulink transmitter block calls the functions implemented by the S-function at different parts of the simulation. While the simulation is initializing, it calls on `mdlStart` such that a data handler object and first-in first-out (FIFO) register are created and a USRP2 object is instantiated. Once the USRP2 object has been created, the operating parameters set in the mask are passed down to the USRP2 hardware while the model is in the process of initializing. Furthermore, the data handler object loads data into the FIFO and, while the simulation is running, Simulink is repeatedly calling `mdlOutputs` such that a frame of data is read from the FIFO, converted to a Simulink data type, and sent to the output port to be received in the simulation. Note that when the simulation has finished, the FIFO and data handler are deallocated. The transmitter shown in Figure 1.6(a) possesses a similar mode of operation.

From the MS thesis and the development of the first Simulink prototype blockset interface with the USRP2 SDR platform, the MathWorks built upon the lessons learned from this experience and ultimately created the SDRu blockset, which can be downloaded from the MathWorks website and installed with MATLAB R2011a or later along with the Communications Toolbox [18]. After several years of development, the SDRu blocks are at the core of numerous SDR implementations using Simulink and the USRP2, as well as educational activities such as those to be discussed later in this book.

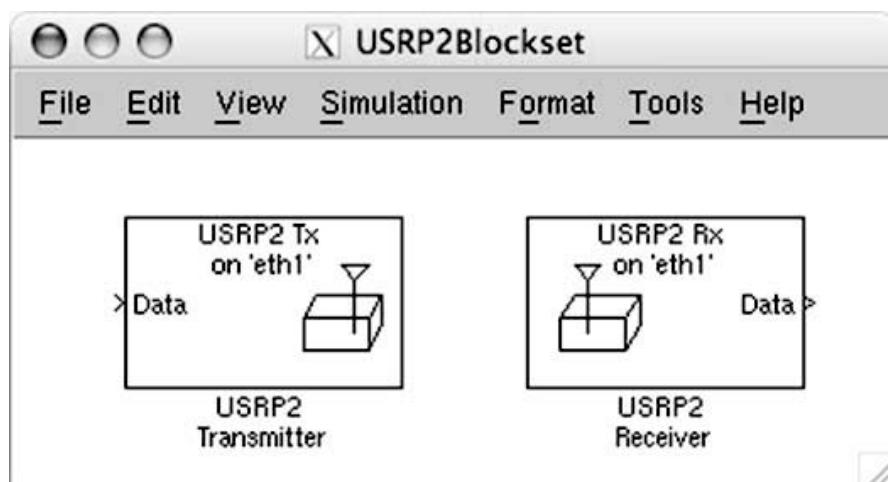


Figure 1.5 The initial prototype Simulink transmitter and receiver interfaces for the USRP2 platform [17].

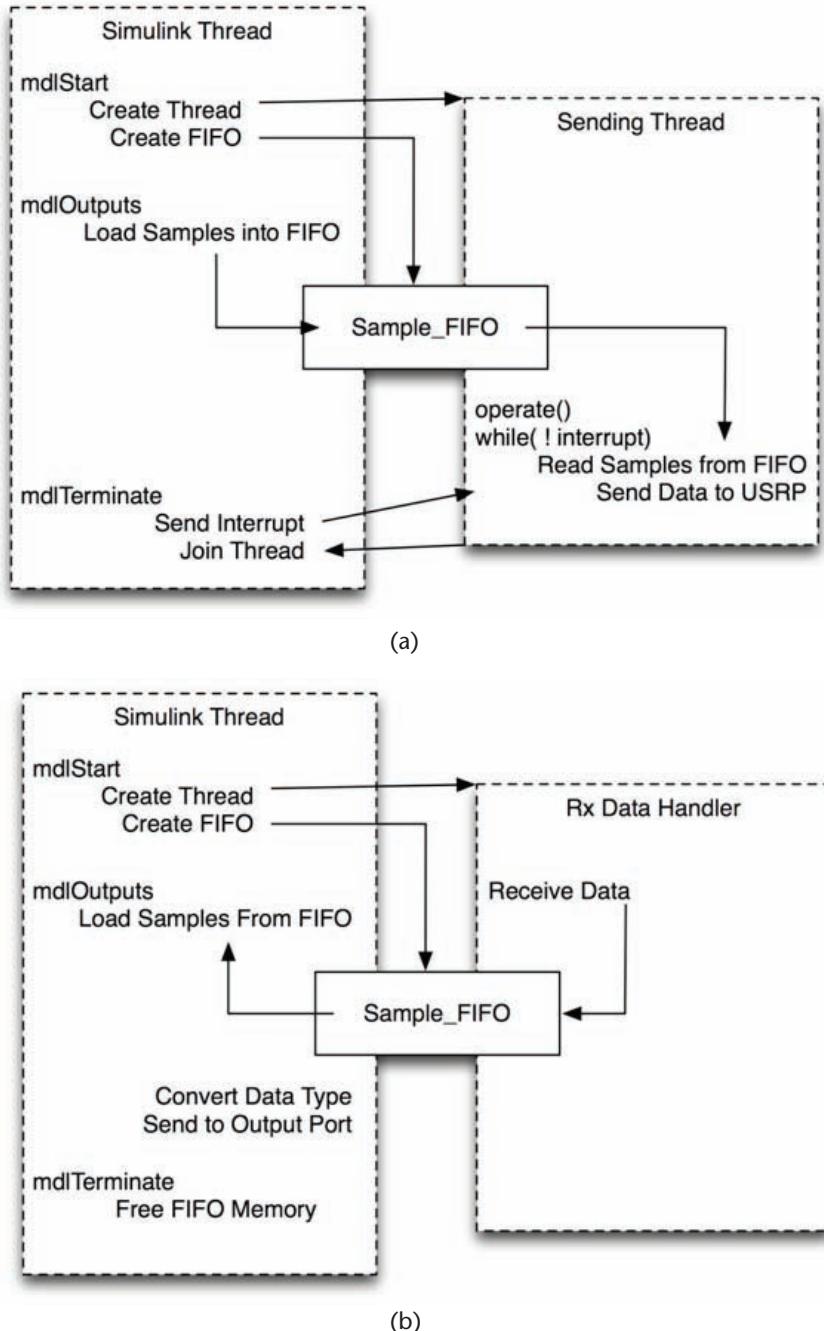


Figure 1.6 Architecture of the initial prototype interfaces to the USRP2 platform [17]. (a) Initial prototype Simulink transmitter interface. (b) Initial prototype Simulink receiver interface.

Other SDR software architectures include the popular open-source GNU Radio software [19], which is a community based effort for devising an SDR software architecture capable of interfacing with any SDR hardware platform, especially the USRP family of products, and enabling them to reliably and seamlessly communicate with other SDR platforms as well as conventional wireless systems. Given the large

open-source community supporting the GNU Radio software architecture, several community members have posted their customized solutions that were not incorporated into GNU Radio directly on the Comprehensive GNU Radio Archive Network (CGRAN) website for download by the rest of the community [20]. Finally, another SDR software interface is the *Implementing Radio in Software* (IRIS) project [21], which is led by the researchers at the Centre for Telecommunications Value-Chain Research (CTVR) at Trinity College Dublin and used by many researchers across Europe and around the world.

1.5 CHAPTER SUMMARY

In this chapter, we studied how the development of digital communication systems and SDR technology is closely linked to the evolution of microprocessor technology. Furthermore, we obtained some insight on the various microprocessor technologies that are currently available to be implemented as part of an SDR hardware prototype. Finally, we concluded this chapter with a literature survey of several SDR hardware platforms and software architectures, and getting a better understanding of the design decisions involved in implementing this systems. By providing some insights on how this technology functions, it is expected that when we eventually begin implementing SDR prototype systems we will know the limitations and capabilities of our SDR platform.

1.6 ADDITIONAL READINGS

Although this chapter gave a brief introduction to the expanding area of SDR technology, several books available in the open literature can provide a more detailed viewpoint of this topic. For instance, the book by Reed extensively covers many of the issues associated with the software architecture of an SDR platform [7], while many of the design considerations and approaches used to construct SDR hardware prototype and their RF front-ends are covered in the book by Kensington [22]. For a decent overview of some of the SDR prototype platforms covered in this chapter, namely, the BEE2, the MARS project, and the Motorola SDR platform, check out Chapter 19 in [23]. Finally, for a clean-slate viewpoint about SDR technology and wireless systems in general, the interested reader is encourage to check out the conference publication by Farrell et al. [24].

References

- [1] Mitola III, J. "Software Radios: Survey, Critical Evaluation and Future Directions," *IEEE Aerospace and Electronic Systems Magazine*, April 1993.
- [2] Hoher, P. and H. Lang, "Coded-8PSK Modem for Fixed and Mobile Satellite Services based on DSP," *Proceedings of the First International Workshop on Digital Signal Processing Techniques Applied to Space Communications*, Noordwijk, The Netherlands, 1988.
- [3] Lackey, R. J., and D. W. Upmal, "Speakeasy: The Military Software Radio," *IEEE Communications Magazine*, May 1995.

- [4] White, B. E., "Tactical Data Links, Air Traffic Management, and Software Programmable Radios," *Proceedings of the 18th Digital Avionics Systems Conference*, St. Louis, MO, 1999.
- [5] Eyermann, P. A., "Joint Tactical Radio Systems—a Solution to Avionics Modernization," *Proceedings of the 18th Digital Avionics Systems Conference*, St. Louis, MO, 1999.
- [6] Bergstrom, C. S. Chuprun and D. Torrieri, "Adaptive Spectrum Exploitation Using Emerging Software Defined Radios," *Proceedings of the IEEE Radio and Wireless Conference*. Denver, CO, 1999.
- [7] Reed, J. H., *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall PTR, 2002.
- [8] Ettus Research LLC, "USRP Networked Series". https://www.ettus.com/product/category/USRP_Networked_Series.
- [9] Minden, G. J., J. B. Evans, L. Searl, D. DePardo, V. R. Petty et al., "KUAR: A Flexible Software-Defined Radio Development Platform," *Proceedings of the IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Dublin, Ireland, 2007.
- [10] Minden, G. J., J. B. Evans, and L. Searl, D. DePardo, V. R. Petty et al., "Cognitive Radio for Dynamic Spectrum Access—An Agile Radio for Wireless Innovation," *IEEE Communications Magazine*, May, 2007.
- [11] Chang Chen, J. Wawrzynek., and R W. Brodersen., "BEE2: A High-End Reconfigurable Computing System," *IEEE Design and Test of Computers Magazine*, March/April, 2005.
- [12] Shi Q., D. Taubenheim, S. Kyerountas, P. Gorday, N. Correal et al., "Link Maintenance Protocol for Cognitive Radio System with OFDM PHY," *Proceedings of the IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Dublin, Ireland, 2007.
- [13] Farrell, R., M. Sanchez, and G. Corley, "Software-Defined Radio Demonstrators: An Example and Future Trends" *International Journal of Digital Multimedia Broadcasting*, 2009.
- [14] Rice University WARP, WARP: *Wireless Open-Access Research Platform*, <http://warp.rice.edu/>.
- [15] Lyrtech RD Incorporated, *Lyrtech RD Processing Systems: Software-Defined Radios*, <http://lyrtechrd.com/en/products/families/+processing-systems+software-defined-radio>.
- [16] Epiq Solutions, *MatchStiq: Handheld Reconfigurable RF Transceiver*, <http://epiqsolutions.com/matchstiq/>.
- [17] Leferman M. J., "Rapid Prototyping Interface for Software Defined Radio Experimentation," Masters Thesis of Worcester Polytechnic Institute, Worcester, MA, 2010.
- [18] The MathWorks, *USRP Hardware Support from MATLAB and Simulink*, <http://www.mathworks.com/discovery/sdr/usrp.html>.
- [19] GNU Radio, *Welcome to GNU Radio!*, <http://gnuradio.org/>.
- [20] Nychis, *The Comprehensive GNU Radio Archive Network*. <http://www.cgran.org/>.
- [21] Sutton P. D., J. Lotze, H. Lahlou, S. A. Fahmy, and K. Nolan et al., "Iris: An Architecture for Cognitive Radio Networking Testbeds" *IEEE Communications Magazine*, September, 2010.
- [22] Kensington P., *RF and Baseband Techniques for Software Defined Radio*, Norward, MA: Artech House, 2005.
- [23] Cabric, D., D. Taubenheim, G. Cafaro, R. Farrell, "Cognitive Radio Platforms and Test-beds" *Cognitive Radio Communications and Networks: Principles and Practice*, Academic Press, 2009.
- [24] Farrell, R., R. Villling, A. M. Wyglinski, C. R. Anderson, J. H. Reed et al., "Rationale for a Clean Slate Radio Architecture," *Proceedings of the SDR Forum Technical Conference*, Washington, DC, 2008.

Signals and Systems Overview

This chapter covers the theory of linear time-invariant (LTI) signals and systems, which forms the fundamental basis for the latter chapters. Sections 2.3 to 2.5 introduce several more advanced topics, such as sampling, pulse shaping, and filtering. However, this chapter does not discuss these topics at length, preferring to direct the reader to more comprehensive books on these subjects.

2.1 SIGNALS AND SYSTEMS

2.1.1 Introduction to Signals

A *signal* is defined as any physical quantity that varies with time, space, or any other independent variable or variables. Mathematically, we describe a signal as a function of one or more independent variables [1]. For example, a cosine wave signal is defined as:

$$s(t) = A \cos(2\pi ft) \quad (2.1)$$

where the signal amplitude A and carrier frequency f are constants. Then, this signal is a function of t . In other words, this signal varies with time. We can classify signals based on different criteria.

In terms of distribution, signals can be classified as deterministic signals and random signals. For example, the cosine wave signal defined in (2.1) is a deterministic signal because for each variable t , there is one and only one corresponding $s(t)$. However, the temperature of a location at each hour of a day is a random variable. Even if we know the temperature for past hours, we cannot predict what the temperature would be for the next hour.

With respect to the value of independent variable t , signals can be classified as either a continuous time signal or a discrete time signal. If t can pick up any value from $- \infty$ to ∞ , the signal is a continuous time signal. Otherwise, if t can only pick up discrete values from $- \infty$ to ∞ , the signal is a discrete time signal. In this case, we usually use n instead of t to denote the time variable, and the resulting signal is written as $s[n]$ instead of $s(t)$. Therefore, a discrete time cosine wave signal can be expressed as:

$$s[n] = A \cos(2\pi fn), \quad n = 1, \dots, N \quad (2.2)$$

where n are the discrete values we can pick up for time variable. Figure 2.1(a) shows an example of continuous time triangular wave, and Figure 2.1(b) shows an example of discrete time triangular wave.

In this chapter, we will discuss converting a signal from continuous-time domain to discrete-time domain using sampling, as well as from discrete-time domain

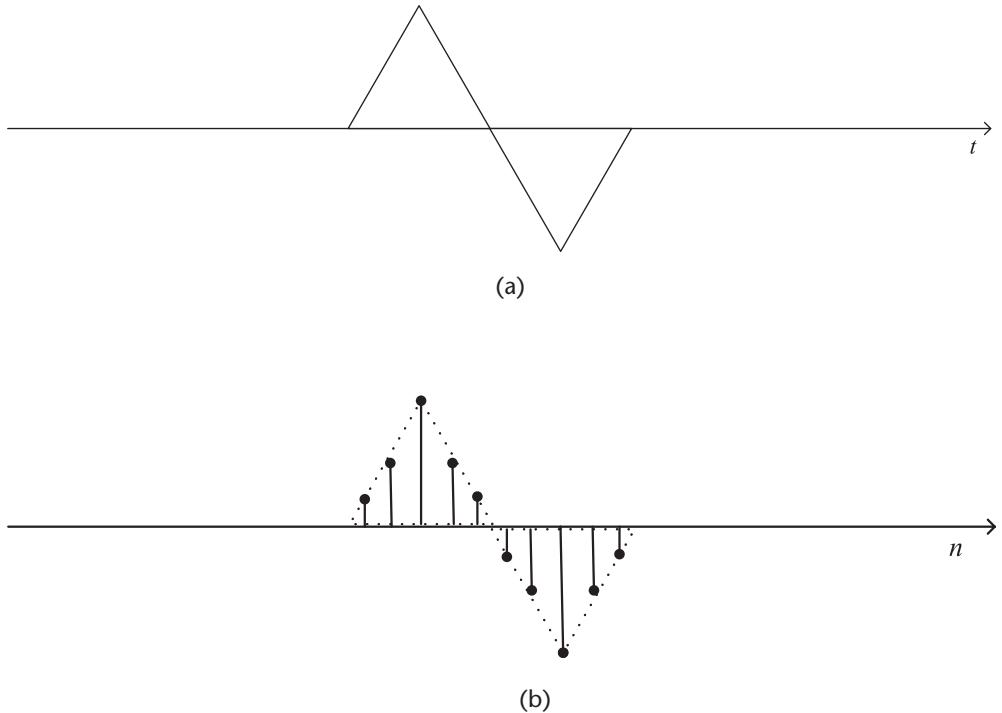


Figure 2.1 Use triangular wave as an example to illustrate continuous time signal and discrete time signal. (a) Continuous time signal. (b) Discrete time signal.

to continuous-time domain, using reconstruction. These processes are referenced to as analog-to-digital and digital-to-analog conversions.

Signals can be classified as either periodic signals or aperiodic signals. For a constant time T_0 , if the signal $s(t)$ satisfies:

$$s(t) = s(t + kT_0), \quad k = N, \dots, 1, \dots, N \quad (2.3)$$

then $s(t)$ is a periodic signal of period T_0 ; otherwise $s(t)$ is an aperiodic signal. Similarly, for discrete time signal $s[n]$, if it satisfies:

$$s[n] = s[n + kT_0], \quad k = N, \dots, 1, \dots, N \quad (2.4)$$

then $s[n]$ is a periodic signal of period T_0 ; otherwise $s[n]$ is an aperiodic signal. Figure 2.2(a) shows an example of periodic triangular wave, and Figure 2.2(b) shows an example of aperiodic triangular wave.

2.1.2 Introduction to Systems

A *system* may be defined as a physical device that performs an operation on a signal [1]. We usually denote a system using letter S . For example, assume the input signal to a system is a continuous-time signal $x(t)$, after some operations of system S , the output signal is a continuous-time signal $y(t)$, as shown in Figure 2.3, then this continuous-time system can be defined as:

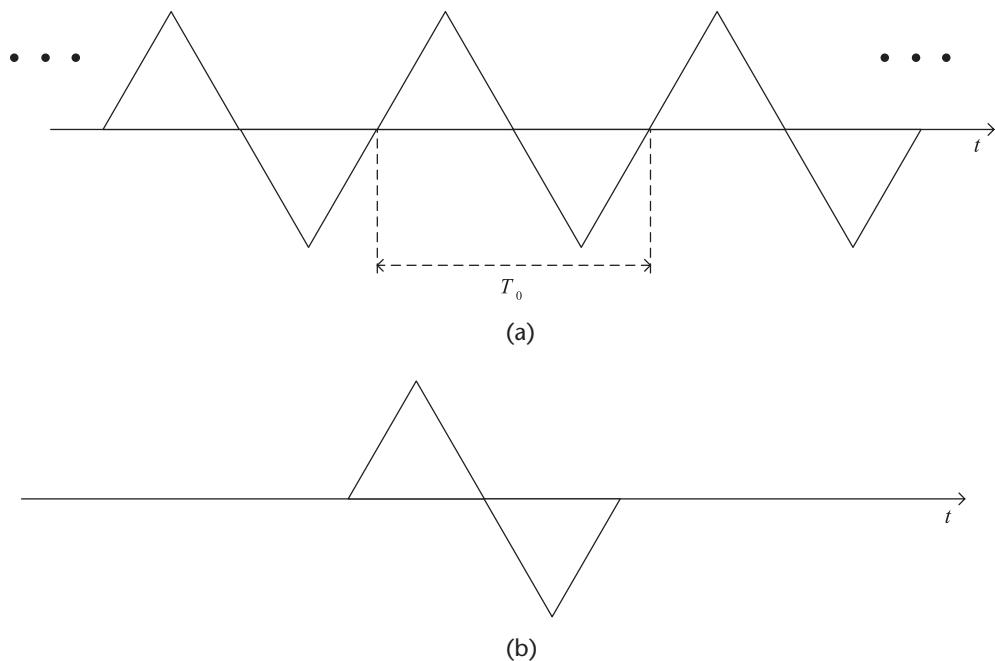


Figure 2.2 Use triangular wave as an example to illustrate periodic signal and aperiodic signal.
(a) Periodic triangular wave. (b) Aperiodic triangular wave.

$$y(t) = S[x(t)] \quad (2.5)$$

where the output signal at time t depends on the input signal $x(t)$, as well as the system S .

Similarly, we can define a discrete-time system as:

$$y[n] = S[x[n]] \quad (2.6)$$

For most situations, the LTI systems are most frequently studied and used. This type of system has two most important properties, linearity and time-invariance, which are specified as follows:

- **Linearity:** Linearity property includes two aspects, namely, *superposition property* and *homogeneity*. Superposition property means when there are several inputs to the system, the resulting output equals to the summation of each independent output. Homogeneity means if the input signal is multiplied by a constant a , the resulting output should also be multiplied by a constant a . Therefore, linearity property can be expressed as:

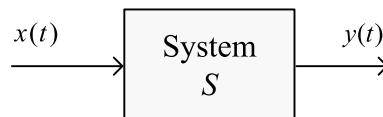


Figure 2.3 A continuous system S , with input signal $x(t)$ and output signal $y(t)$.

If

$$S[x_1(t)] = y_1(t), \quad S[x_2(t)] = y_2(t) \quad (2.7)$$

then

$$S[a_1x_1(t) + a_2x_2(t)] = a_1S[x_1(t)] + a_2S[x_2(t)] \quad (2.8)$$

- *Time-invariant:* Time-invariant can be expressed as:

If

$$S[x(t)] = y(t) \quad (2.9)$$

then

$$S[x(t - t_0)] = y(t - t_0) \quad (2.10)$$

It means for a time-invariant system, when the input signal is delayed by a time period of t_0 , the output signal will also be delayed by the same amount of time t_0 .

There are several other important properties of systems. Since this chapter focuses on continuous-time systems, the following properties are discussed in the continuous-time context:

- *Causal:* A system is causal if for $t < t_0$, the input signal is 0, then the corresponding output signal for $t < t_0$ is also 0. In other words, the output signal cannot exist before the input signal. Otherwise, the system is called a non-causal system.
- *Memoryless:* A system is memoryless if the output value at any time t depends and only depends on the input signal value at that same time t .

In order to analyze a linear time-invariant system, we usually need to build a mathematical model for the system. There are two frequently used models, namely, input-output model and state variable model. Then, two methods can be used to solve these models. They are time domain analysis and transform domain analysis. The latter leads us to the next section, which introduces one of the most important transforms, the Fourier transform.

2.2 FOURIER TRANSFORM

Electrical and computer engineers often deal with physical phenomena that are structured as some form of energy transmission via wave emission. Whether it is sound waves, wireless signals, or ultrasound pulses, it is very important to possess a collection of tools that can analyze and study these forms of transmission. Throughout history, numerous scientists and engineers have studied and developed mathematical tools that can be used to characterize these wave emissions.

One form of mathematical tools developed during the 18th century is Fourier analysis, which can be used to analyze signals and systems in frequency domain.

From a human perspective, we are often more used to analyzing signals and systems in the time domain. However, the perspectives in the frequency domain can provide us with more insight on the various features of a particular signal and/or system, as well as its manipulation by its combination with other signals and processing using other tools.

Fourier analysis family mainly includes Fourier transform (FT), Fourier series (FS), discrete time Fourier transform (DTFT), and discrete Fourier transform (DFT). Each has its own applications. In this section, we will focus on Fourier transform, but we will also briefly cover the other three.

2.2.1 Introduction and Historical Perspective

Fourier analysis is named after Jean Baptiste Joseph Fourier (March 21, 1768–May 16, 1830). Fourier was a French mathematician and physicist best known for initiating the investigation of the Fourier series and their applications to problems of heat transfer and vibrations. Fourier's Law is also named in his honor. Fourier is also generally credited with the discovery of the greenhouse effect [2].

Besides Fourier, many other scientists and mathematicians have also contributed to this work. For example, Daniel Bernoulli and Leonhard Euler introduced trigonometric representations of functions, and Lagrange gave the Fourier series solution to the wave



Figure 2.4 Jean Baptiste Joseph Fourier, 1768–1830 (from [3]).

equation. The subsequent development of the field is known as harmonic analysis and is also an early instance of representation theory. The first fast Fourier transform (FFT) algorithm for the DFT was discovered around 1805 by Carl Friedrich Gauss when interpolating measurements of the orbit of the asteroids Juno and Pallas [4].

2.2.2 Definition

As mentioned at the beginning of this section, each tool in the Fourier family has its own application, which is listed in Table 2.1. These applications are classified based on the properties of input signals, as shown in Figure 2.5.

The *Fourier transform*, in essence, decomposes or separates a continuous, aperiodic waveform or function, as shown in Figure 2.5(a), into sinusoids of different frequency that sum to the original waveform. It identifies or distinguishes the different frequency sinusoids and their respective amplitudes [5].

The Fourier transform of $x(t)$ is defined as [1]:

$$X(\omega) = \int x(t)e^{-j\omega t} dt \quad (2.11)$$

where t is the time variable in seconds across the time domain, and ω is the frequency variable in radian per second across frequency domain.

Applying the similar transform to $X(\omega)$ yields the inverse Fourier transform [1]:

$$x(t) = \frac{1}{2\pi} \int X(\omega)e^{j2\pi\omega t} d\omega \quad (2.12)$$

where we write $x(t)$ as a weighted sum of complex exponentials.

The previous Fourier transform pair can be denoted as [6]:

$$x(t) \xleftrightarrow{\mathcal{F}} X(\omega) \quad (2.13)$$

where the left hand side of the symbol $\xleftrightarrow{\mathcal{F}}$ is before Fourier transform, while the right hand side of the symbol $\xleftrightarrow{\mathcal{F}}$ is after Fourier transform. The most frequently used Fourier transform pairs are listed in Table 2.2 for your quick reference.

The mathematical definitions of the other three Fourier tools are listed in Table 2.3, but we will not introduce them in detail. Please note that for Fourier series, $x(t)$ is a periodic signal with fundamental period $T_p = l/F_0$, and $\{c_k\}$ are the coefficients.

2.2.3 Properties

Several properties of Fourier transform are useful when studying signals and systems in the Fourier transform domain:

Table 2.1 The Applications of Fourier Family Tools

Name	Continuous/Discrete	Periodic/Aperiodic
Fourier transform	Continuous	Aperiodic
Fourier series	Continuous	Periodic
Discrete time Fourier transform	Discrete	Aperiodic
Discrete Fourier transform	Discrete	Periodic

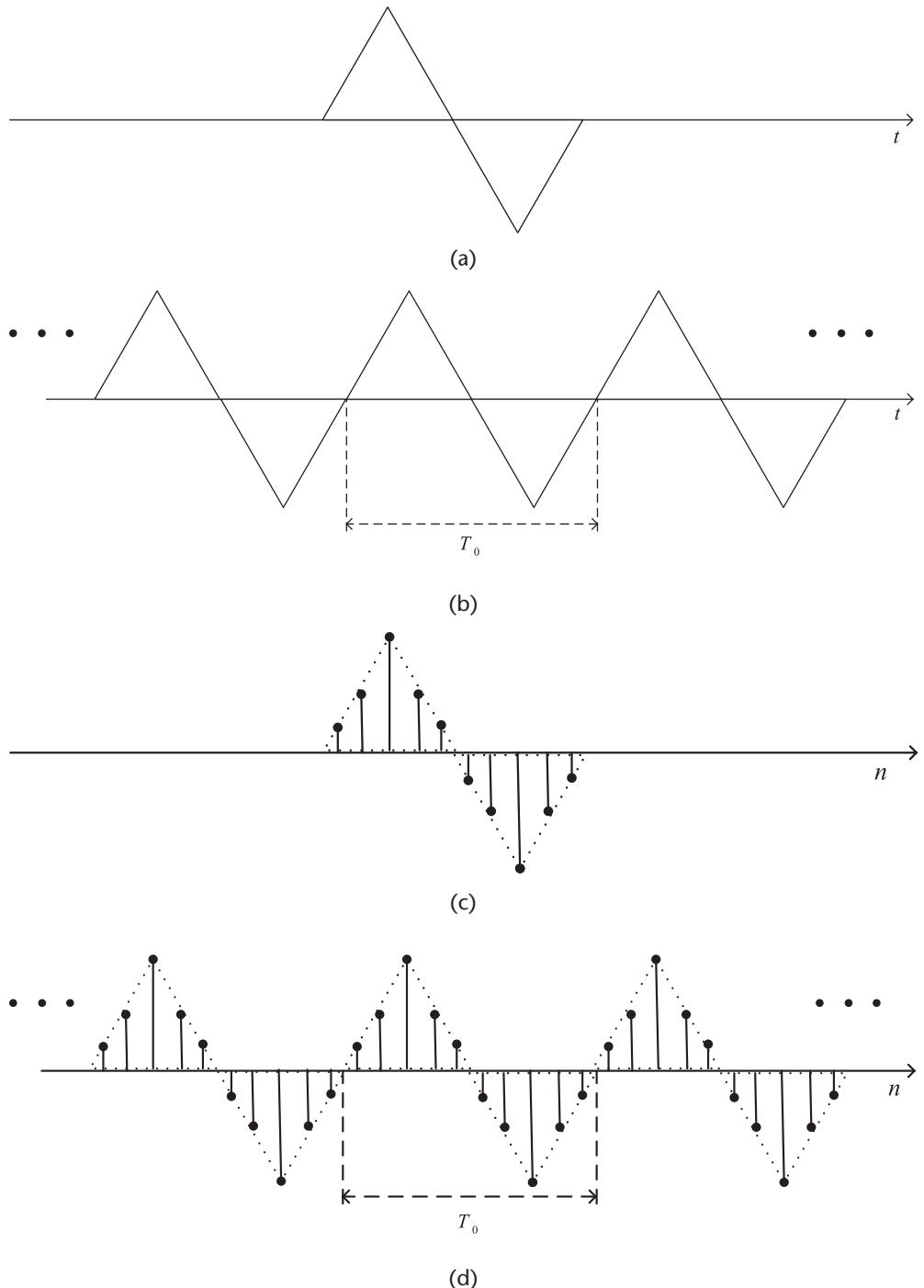


Figure 2.5 Use triangular wave as an example to illustrate four different types of signals. (a) Continuous, aperiodic signal. (b) Continuous, periodic signal. (c) Discrete, aperiodic signal. (d) Discrete, periodic signal.

Table 2.2 Fourier Transform Table: Selected Pairs [6]

$x(t)$	$X(\omega)$
$u(t)$	$\pi\delta(\omega) + \frac{1}{j\omega}$
$\delta(t)$	1
1	$2\pi\delta(\omega)$
$e^{j\omega_0 t}$	$2\pi\delta(\omega - \omega_0)$
$\cos(\omega_0 t)$	$\pi[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$
$\sin(\omega_0 t)$	$\pi j[\delta(\omega - \omega_0) - \delta(\omega + \omega_0)]$
$\cos(\omega_0 t)u(t)$	$\pi/2[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \frac{j\omega}{\omega_0^2 - \omega^2}$
$\sin(\omega_0 t)u(t)$	$\pi/2j[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)] + \frac{j\omega}{\omega_0^2 + \omega^2}$
$e^{-at}u(t), a > 0$	$\frac{1}{a + j\omega}$
$te^{-at}u(t), a > 0$	$\frac{1}{a + j\omega}^2$
$e^{-a t }u(t), a > 0$	$\frac{2a}{a^2 + \omega^2}$

- **Linearity:** Since Fourier transform is a linear transform, it has linearity. According to Section 2.1.2, for any constant $a_n (n = 1, \dots, N)$, if the time-domain signals $x(t) = \sum_{n=1}^N a_n x_n(t)$, then its Fourier transform is $\sum_{n=1}^N a_n X_n(\omega)$.
- **Symmetry:** If a signal $x(t)$ is symmetric in time domain, then its Fourier transform $X(\omega)$ is also symmetric in frequency domain.
- **Shifting property:** Shifting property includes time-shifting property and frequency-shifting property, namely:
 - **Time-shifting property:** If a signal $x(t)$ is shifted by t_0 on time domain, then its Fourier transform $X(\omega)$ will be multiplied by $e^{-j\omega t_0}$ on frequency domain. Suppose the time signal is $x(t)$ and its Fourier transform signal is $X(\omega)$.
 - **Frequency-shifting property:** If a signal $X(\omega)$ is shifted by ω_0 on frequency domain, then its inverse Fourier transform $x(t)$ will be multiplied by $e^{j\omega_0 t}$ on time domain.

Table 2.3 The Mathematical Definitions of the Fourier Tools [1]

Name	Direct Transform	Inverse Transform
Fourier series	$c_k = \frac{1}{T_p} \int_{-T_p/2}^{T_p/2} x(t) e^{-j2\pi k f_0 t} dt$	$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k f_0 t}$
Discrete time Fourier transform	$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$	$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega$
Discrete Fourier transform	$c_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi k n / N}$	$x[n] = \sum_{k=0}^{N-1} c_k e^{j2\pi k n / N}$

Table 2.4 Fourier Transform Properties, based on [6]

Property	Time Signal	Fourier Transform Signal
Linearity	$\sum_{n=1}^N a_n x_n(t)$	$\sum_{n=1}^N a_n X_n(\omega)$
Symmetry	$x(-t)$	$X(-\omega)$
Time shift	$x(t - t_0)$	$X(\omega)e^{-j\omega t_0}$
Frequency shift	$x(t)e^{j\omega_0 t}$	$X(\omega - \omega_0)$
Scaling	$x(\alpha t)$	$\frac{1}{ \alpha } X\left(\frac{\omega}{\alpha}\right)$
Derivative	$\frac{d^n}{dt^n} x(t)$	$(j\omega)^n X(\omega)$
Integration	$\int_{-\infty}^{\infty} x(\tau) d\tau$	$\frac{X(\omega)}{j\omega} + \pi X(0)\delta(\omega)$
Time convolution	$x(t) h(t)$	$X(\omega)H(\omega)$
Frequency convolution	$X(t) h(t)$	$\frac{1}{2\pi} X(\omega) H(\omega)$

Suppose the time signal is $x(t)$ and its Fourier transform signal is $X(\omega)$.

- *Scaling property:* If a signal $x(t)$ is scaled by a factor of α on time domain, then its Fourier transform $X(\omega)$ will be scaled by a factor of $1/\alpha$ on frequency domain.
- *Derivative and integration:* If a time-domain signal $x(t)$ is taking derivative or integration operation, then its Fourier transform $X(\omega)$ will be multiplied or divided by $j\omega$.
- *Convolution theorem:* Convolution theorem includes time-domain convolution and frequency-domain convolution, namely:
 - *Time-domain convolution:* If two signals $x(t)$ and $h(t)$ are convoluted on time domain, then their Fourier transform $X(\omega)$ and $H(\omega)$ will be multiplied on frequency domain.
 - *Frequency-domain convolution:* If two signals $X(\omega)$ and $H(\omega)$ are convoluted on frequency domain, then their inverse Fourier transform $x(t)$ and $h(t)$ will be multiplied on time domain.

The Fourier transform properties introduced here are summarized in Table 2.4 for your quick reference.

2.3 SAMPLING THEORY

As mentioned in Section 2.1.1, a continuous-time signal can be converted to a discrete-time signal using sampling. At the same time, sampling is the most important operation in analog-to-digital conversion, as shown in Figure 2.6, where an analog input signal $x_a(t)$ is converted to a digital output signal. *Sampling* is the conversion of a continuous-time signal into a discrete-time signal obtained by taking “samples” of the continuous-time signal at discrete-time instants [1]. Section 2.3.1 will introduce a frequently used sampling method, namely, uniform sampling.

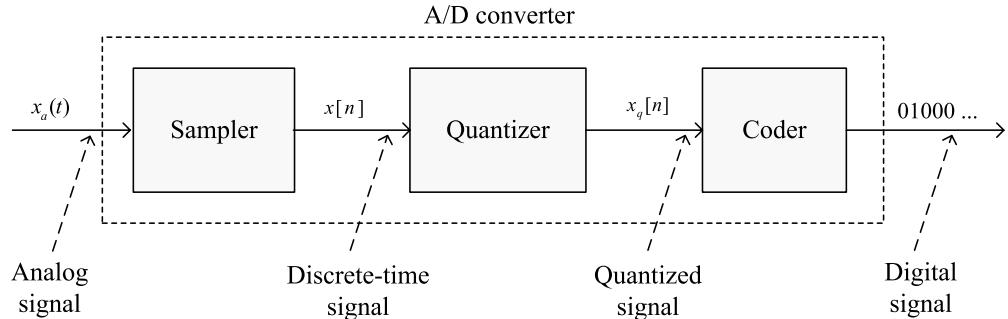


Figure 2.6 Basic parts of an analog-to-digital converter (ADC) [1]. Sampling takes place in the sampler block.

Similarly, a discrete-time signal can also be converted to a continuous-time signal using reconstruction. However, reconstruction is not always successful. Sometimes, the reconstructed signal is not the same as the original signal. Since for a given sampled signal, we can fit infinite continuous-time signals through the samples. However, if the sampling satisfies certain criterion, the signal can be reconstructed perfectly. This criterion, called Nyquist sampling theorem, will be introduced in Section 2.3.3.

2.3.1 Uniform Sampling

Concerning sampling interval, there are infinite ways to do sampling. However, if we specify the sampling interval as a constant number T_s , we get the most widely used sampling, called *uniform sampling*, or *periodic sampling*. Using this method, we are taking “samples” of the continuous-time signal every T_s seconds, which can be defined as:

$$x[n] = x(nT_s), \quad n < \dots \quad (2.14)$$

where $x(t)$ is the input continuous-time signal, $x[n]$ is the output discrete-time signal, T_s is the sampling period, and $f_s = 1/T_s$ is the sampling frequency.

An equivalent model for the uniform sampling operation is shown in Figure 2.7(a), where the continuous-time signal $x(t)$ is multiplied by an impulse train $p(t)$ to form the sampled signal $x_s(t)$, which can be defined as:

$$x_s(t) = x(t)p(t) \quad (2.15)$$

where the signal $p(t)$ is referred to as the sampling function.

The sampling function is assumed to be a series of narrow pulses, which is either zero or one. Thus, $x_s(t) = x(t)$ when $p(t) = 1$, and $x_s(t) = 0$ when $p(t) = 0$. Since $p(t) = 1$ only at time instants T_s , the resulting $x_s(t) = x(nT_s) = x[n]$, which proves that this is indeed an equivalent model for the uniform sampling operation. This model will help us to obtain the frequency domain representation of uniform sampling in Section 2.3.2.

2.3.2 Frequency Domain Representation of Uniform Sampling

Since it is easier for us to observe and derive the Nyquist sampling theorem in frequency domain, in this section, we will try to represent the uniform sampling process in frequency domain.

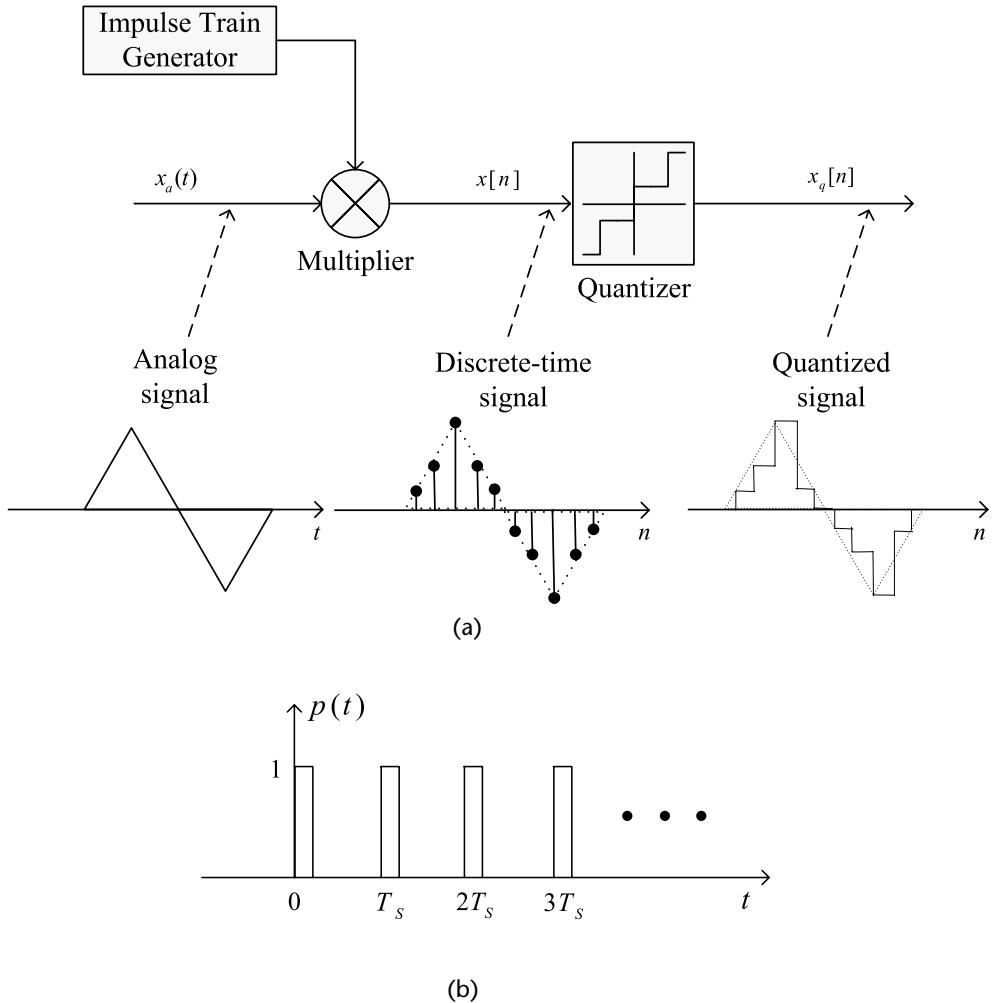


Figure 2.7 An equivalent model for the uniform sampling operation. (a) A continuous-time signal $x(t)$ is multiplied by a periodic pulse $p(t)$ to form the sampled signal $x_s(t)$. (b) A periodic pulse $p(t)$.

According to Figure 2.7(b), we can define the sampling function $p(t)$ as:

$$p(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s), \quad k = 0, 1, 2, \dots \quad (2.16)$$

where at time instants kT_s , we have $p(t) = 1$. According to [7], $p(t)$ is a Dirac comb constructed from Dirac delta functions.

Substitution of (2.16) into (2.15) gives:

$$x_s(t) = x(t)p(t) = x(t) \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \quad (2.17)$$

In order to learn the sampling process in frequency domain, we need to take the Fourier transform of $x_s(t)$. According to frequency-domain convolution property in

Section 2.2.3, multiplication in time domain will lead to convolution in frequency domain. Therefore, multiplication of $x(t)$ and $p(t)$ will yield the convolution of $X(\omega)$ and $P(\omega)$:

$$X_s(\omega) = \frac{1}{2\pi} X(\omega) * P(\omega) \quad (2.18)$$

where $X(\omega)$ is the Fourier transform of $x(t)$, and $P(\omega)$ is the Fourier transform of $p(t)$.

The Fourier transform of a Dirac comb is also a Dirac comb [7], namely:

$$P(\omega) = \frac{\sqrt{2\pi}}{T_s} \sum_{k=-\infty}^{\infty} \delta\left(\omega - k \frac{2\pi}{T_s}\right) = \frac{\sqrt{2\pi}}{T_s} \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_s) \quad (2.19)$$

where $\omega_s = 2\pi f_s$ is the sampling frequency.

Substitute (2.19) into (2.18). Since convolution with delta function pulses replicas at pulse location we have:

$$X_s(\omega) = \frac{1}{\sqrt{2\pi T_s}} \sum_{k=-\infty}^{\infty} X(\omega - k\omega_s) \quad (2.20)$$

Equation (2.20) tells us that the uniform sampling creates images of the Fourier transform of the input signal, and images are periodic with sampling frequency f_s .

2.3.3 Nyquist Sampling Theorem

Based on (2.20), we draw the spectrum of original signal $x(t)$ and the sampled signal $x_s(t)$ on frequency domain, as shown in Figure 2.8. Assume the bandwidth of the original signal is $[-f_b, f_b]$, as shown in Figure 2.8(a). For now, we do not pay attention to the signal amplitude, so we use A and A_s to represent them. Assume the sampling frequency is f_s , then the sampled signal will have replicas at location kf_s . In order to reconstruct the original signal from the sampled signal, we will apply a lowpass filter on the sampled signal, trying to extract the $n = 0$ term from $X_s(f)$, as shown in Figure 2.8(b). Therefore, accomplishing reconstruction without error requires that the portion of the spectrum of $X_s(f)$ at $f = \pm f_s$ does not overlap with the portion of the spectrum at $f = 0$. In other words, this requires that $f_s - f_b > f_b$ or $f_s > 2f_b$, which leads to the Nyquist sampling theorem.

Nyquist sampling theorem applies for the *bandlimited signal*, which is a signal $x(t)$ that has no spectral components beyond a frequency B Hz [8]. That is:

$$X(\omega) = 0, \quad |\omega| > 2\pi B \quad (2.21)$$

The Nyquist sampling theorem states that a real signal, $x(t)$, which is bandlimited to B Hz can be reconstructed without error from samples taken uniformly at a rate $R > 2B$ samples per second. This minimum sampling frequency, $F_s = 2B$ Hz, is called the *Nyquist rate* or the *Nyquist frequency*. The corresponding sampling interval, $T = 1/2B$, is called the *Nyquist interval* [1]. A signal bandlimited to B Hz which is sampled at less than the Nyquist frequency of $2B$, (i.e., which was sampled at an interval $T > 1/2B$) is said to be *undersampled*.

When a signal is undersampled, its spectrum has overlapping tails, where $X_s(f)$ no longer has complete information about the spectrum and it is no longer possible

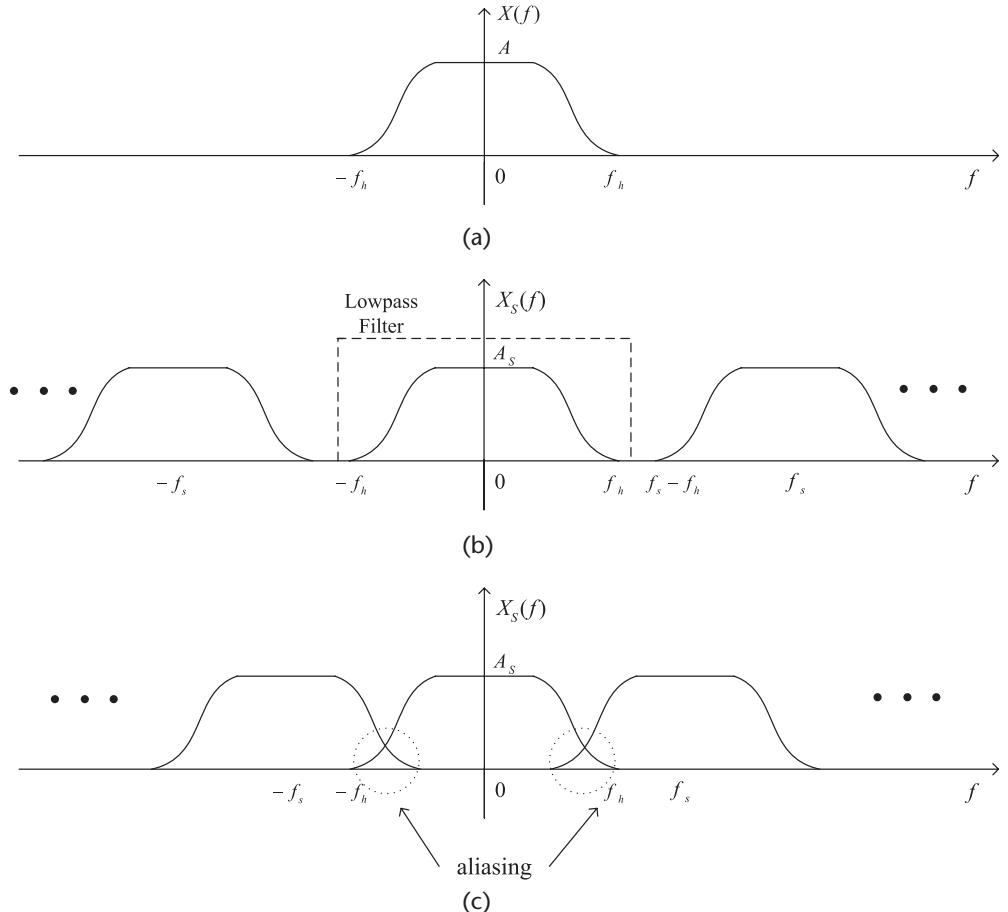


Figure 2.8 The spectrum of original signal $x(t)$ and the sampled signal $x_s(t)$ on frequency domain. (a) The spectrum of original signal $x(t)$, with bandwidth $-f_h$ to f_h , and amplitude A . (b) The spectrum of sampled signal $x_s(t)$, which satisfies Nyquist sampling theorem. (c) The spectrum of sampled signal $x_s(t)$, which *does not* satisfies Nyquist sampling theorem and has aliasing.

to recover $x(t)$ from the sampled signal. In this case, the tailing spectrum does not go to zero, but is folded back onto the apparent spectrum. This inversion of the tail is called *spectral folding* or *aliasing*, as shown in Figure 2.8(c) [5].

2.3.4 Sampling Rate Conversion

In real-world applications, sometimes, we would like to lower sampling rate because it reduces storage and computation requirements. Sometimes, however, we prefer a higher sampling rate, because it preserves fidelity. Sampling rate conversion is a general term for the process of changing the time interval between the adjacent elements in a sequence consisting of samples of a continuous-time function [5].

The process of lowering the sampling rate is called *decimation*, which is achieved by ignoring all but every D th sample. In time domain, it can be defined as:

$$y[n] = x[nD] \quad (2.22)$$

where $x[n]$ is the original signal, $y[n]$ is the decimated signal, and D is the decimation rate.

According to (2.22), the sampling rates of the original signal and the decimated signal can be expressed as:

$$F_y = \frac{F_x}{D} \quad (2.23)$$

where F_x is the sampling rates of the original signal, and F_y is the sampling rates of the decimated signal.

Since the frequency variables in radians, ω_x and ω_y , can be related to sampling rate, F_x and F_y , by:

$$\omega_x = 2\pi F T_x = \frac{2\pi F}{F_x} \quad (2.24)$$

and

$$\omega_y = 2\pi F T_y = \frac{2\pi F}{F_y} \quad (2.25)$$

it follows that ω_x and ω_y are related by:

$$\omega_y = D\omega_x \quad (2.26)$$

which means that the frequency range of ω_x is stretched into the corresponding frequency range of ω_y by a factor of D .

In order to avoid aliasing of the decimated sequence $y[n]$, it is required that $0 < \omega_y < \pi/D$. Based on (2.26), it implies that the spectrum of the original sequence should satisfy $0 < |\omega_x| < \pi/D$. Therefore, in reality, decimation is usually a two-step process, consisting of a lowpass anti-aliasing filter and a downampler, as shown in Figure 2.9. The lowpass anti-aliasing filter is used to constrain the bandwidth of the input signal to the downampler $x[n]$ to be $0 < |\omega_x| < \pi/D$.

In frequency domain, the spectrum of the decimated signal, $y[n]$, can be expressed as [1]:

$$Y(\omega_y) = \frac{1}{D} \sum_{k=0}^{D-1} H_D \left(\frac{\omega_y - 2\pi k}{D} \right) S \left(\frac{\omega_y - 2\pi k}{D} \right) \quad (2.27)$$

where $S(\omega)$ is the spectrum of the input signal $s[n]$, and $H_D(\omega)$ is the frequency response of the lowpass filter $h_D[n]$. With a properly designed filter $H_D(\omega)$, the aliasing is eliminated, and consequently, all but the first $k = 0$ term in (2.27) vanish [1]. Hence, (2.27) becomes:

$$Y(\omega_y) = \frac{1}{D} H_D \left(\frac{\omega_y}{D} \right) S \left(\frac{\omega_y}{D} \right) = \frac{1}{D} S \left(\frac{\omega_y}{D} \right) \quad (2.28)$$

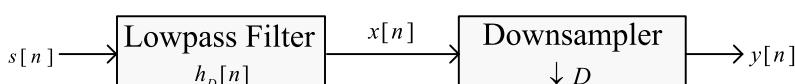


Figure 2.9 The structure of decimation, consisting of a lowpass anti-aliasing filter and a downampler.

for $0 \leq |\omega_y| \leq \pi$. The spectra for the sequence $x[n]$ and $y[n]$ are illustrated in Figure 2.10, where the frequency range of the intermediate signal is $0 \leq |\omega_x| \leq \frac{\pi}{D}$, and the frequency range of the decimated signal is $0 \leq |\omega_y| \leq \pi$.

The process of increasing the sampling rate is called *interpolation*, which can be accomplished by interpolating $I - 1$ new samples between successive values of signal. In time domain, it can be defined as:

$$y[n] = \begin{cases} x[n/I] & n = 0, \pm I, \pm 2I, \dots \\ 0 & \text{otherwise} \end{cases} \quad (2.29)$$

where $x[n]$ is the original signal, $y[n]$ is the interpolated signal, and I is the interpolation rate.

According to (2.29), the sampling rates of the original signal and the interpolated signal can be expressed as:

$$F_y = IF_x \quad (2.30)$$

where F_x is the sampling rates of the original signal, and F_y is the sampling rates of the interpolated signal. Since (2.24) and (2.25) also hold here, it follows that ω_x and ω_y are related by:

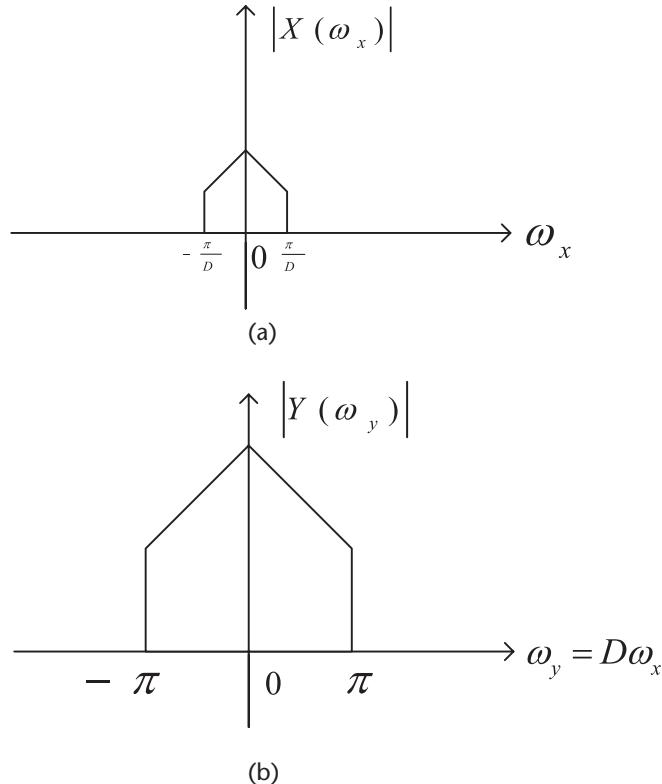


Figure 2.10 The spectra for the sequence $x[n]$ and $y[n]$ where the frequency range of ω_x is stretched into the corresponding frequency range of ω_y by a factor of D . (a) Spectrum of the intermediate sequence. (b) Spectrum of the decimated sequence.

$$\omega_y = \frac{\omega_x}{I} \quad (2.31)$$

which means that the frequency range of ω_x is compressed into the corresponding frequency range of ω_y by a factor of I . Therefore, after the interpolation, there will be I replicas of the spectrum of $x[n]$, where each replica occupies a bandwidth of π/I . Since only the frequency components of $y[n]$ in the range $0 < |\omega_y| < \pi/I$ are unique, (i.e., all the other replicas are the same as this one), the images of $Y(\omega)$ above $\omega_y = \pi/I$ should be rejected by passing it through a lowpass filter with the following frequency response:

$$H_I(\omega_y) = \begin{cases} C & 0 < |\omega_y| < \frac{\pi}{I} \\ 0 & \text{otherwise} \end{cases} \quad (2.32)$$

where C is a scale factor.

Therefore, in reality, interpolation is also a two-step process, consisting of an upsampler and a lowpass filter, as shown in Figure 2.11. The spectrum of the output signal $z[n]$ is:

$$Z(\omega_z) = \begin{cases} CX(\omega_z I) & 0 < |\omega_z| < \frac{\pi}{I} \\ 0 & \text{otherwise} \end{cases} \quad (2.33)$$

where $X(\omega)$ is the spectrum of the output signal $x[n]$.

The spectra for the sequence $x[n]$, $y[n]$, and $z[n]$ are illustrated in Figure 2.12, where the frequency range of the original signal is $0 < |\omega_x| < \pi$, and the frequency range of the decimated signal is $0 < |\omega_z| < \frac{\pi}{I}$.

2.4 PULSE SHAPING

In Section 2.3, we have shown that an analog signal can be converted to a digital signal using an analog-to-digital converter (ADC), as illustrated in Figure 2.6. These digital signals consisting of “1” and “0” values have significant advantages in the aspect of signal processing. However, they cannot be directly transmitted in a real-world situation. Instead, they must be first converted into an analog signal in order to be transmitted. This conversion is done by the “transmit” or “pulse-shaping” filter, which changes each symbol in the digital message into a suitable analog pulse [8]. This process is shown in Figure 2.13, where the impulse modulator is denoted by $p(t)$, and the impulse response of the transmit filter is denoted by $h_T(t)$.

Similar to the sampling function in Section 2.3.2, the impulse modulator can be defined as:

$$p(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT) \quad (2.34)$$

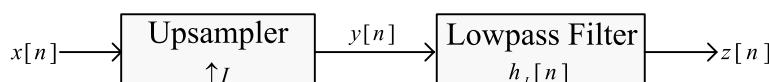


Figure 2.11 The structure of interpolation, consisting of an upsampler and a lowpass filter.

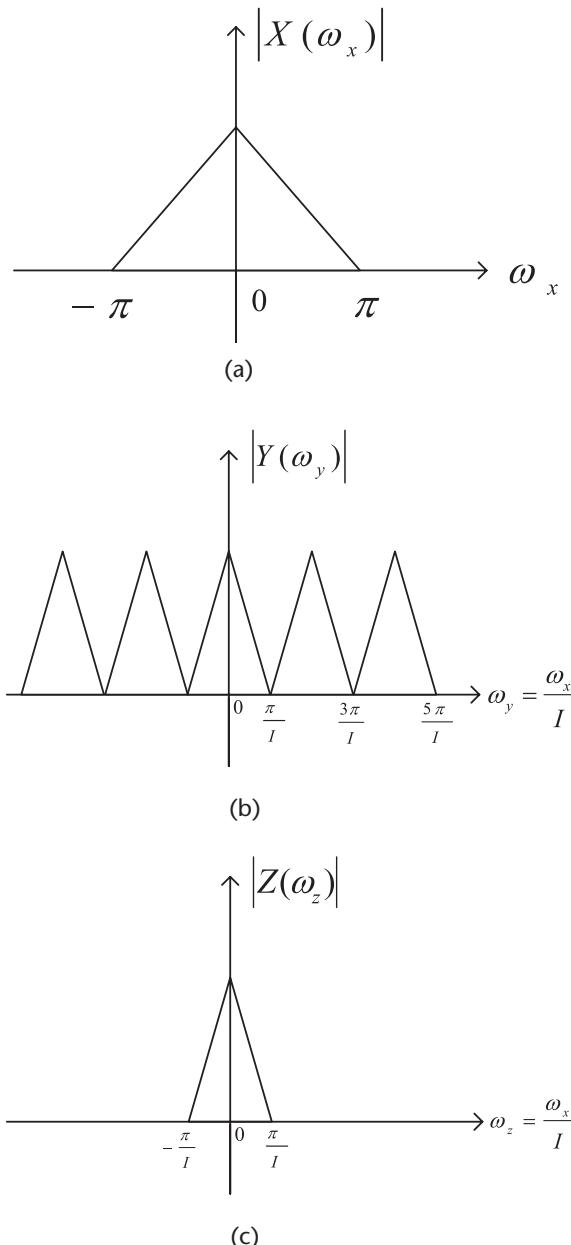


Figure 2.12 The spectra for the sequence $x[n]$, $y[n]$, and $z[n]$, where the frequency range of w_x is compressed into the corresponding frequency range of ω_y by a factor of I . (a) Spectrum of the original sequence. (b) Spectrum of the intermediate sequence. (c) Spectrum of the interpolated sequence.

where $p(t) = 1$ for $t = kT$, and $p(t) = 0$ for all the other time instants. Therefore, the analog pulse train after the impulse modulator is:

$$s_a(t) = s[n]p(t) = \sum_{k=-\infty}^{\infty} s(kT)\delta(t - kT) = \begin{cases} s(kT) & t = kT \\ 0 & t \neq kT \end{cases} \quad (2.35)$$

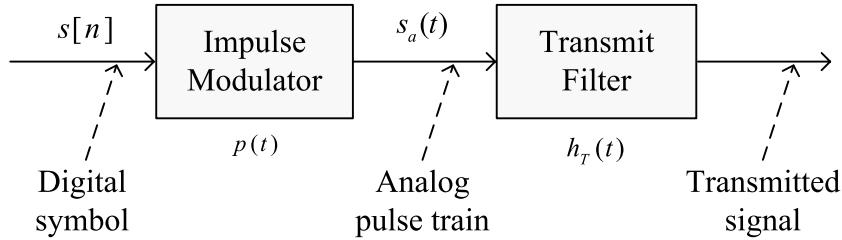


Figure 2.13 On the transmitter side, the impulse modulator and transmit filter convert the digital symbols into analog pulses for the sake of transmission.

where each digital symbol $s[n]$ initiates an analog pulse that is scaled by the value of the symbol.

If we do not have a transmit filter in our system, these pulses $s_a(t)$ will be transmitted through the channel right away. In theory, on the receiver side, we should be able to get the same pulse train, although with some delay. However, in reality, we actually cannot get it due to the interference between adjacent symbols. There are two situations when adjacent symbols may interfere with each other: when the pulse shape is wider than a single symbol interval T , and when there is a nonunity channel that “smears” nearby pulses, causing them to overlap. Both of these situations are called *intersymbol interference* (ISI) [8]. In order to solve this problem, pulse shaping filters are introduced to bandlimit the transmit waveform.

2.4.1 Eye Diagrams

In telecommunication, an *eye diagram*, also known as an *eye pattern*, is an oscilloscope display in which a digital data signal from a receiver is repetitively sampled and applied to the vertical input, while the data rate is used to trigger the horizontal sweep [8]. It is so called because, for several types of coding, the pattern looks like a series of eyes between a pair of rails.

Several system performance measures can be derived by analyzing the display, especially the extent of the ISI. As the eye closes, the ISI increases; as the eye opens, the ISI decreases. Besides, if the signals are too long, too short, poorly synchronized with the system clock, too high, too low, too noisy, or too slow to change, or have too much undershoot or overshoot, this can be observed from the eye diagram. For example, Figure 2.14 shows a typical eye pattern for the noisy QPSK signal.

2.4.2 Nyquist Pulse Shaping Theory

In a communication system, there are normally two pulse shaping filters. One on the transmitter side, and the other on the receiver side, as shown in Figure 2.15, where we use $h_T(t)$ and $h_R(t)$ to denote the impulse response of the transmit filter and receive filter. For simplicity, when considering the Nyquist pulse shaping theory, we usually use the concept of equivalent channel, which includes not only the channel itself, but also the two pulse shaping filters. Therefore, the impulse response of the equivalent channel is:

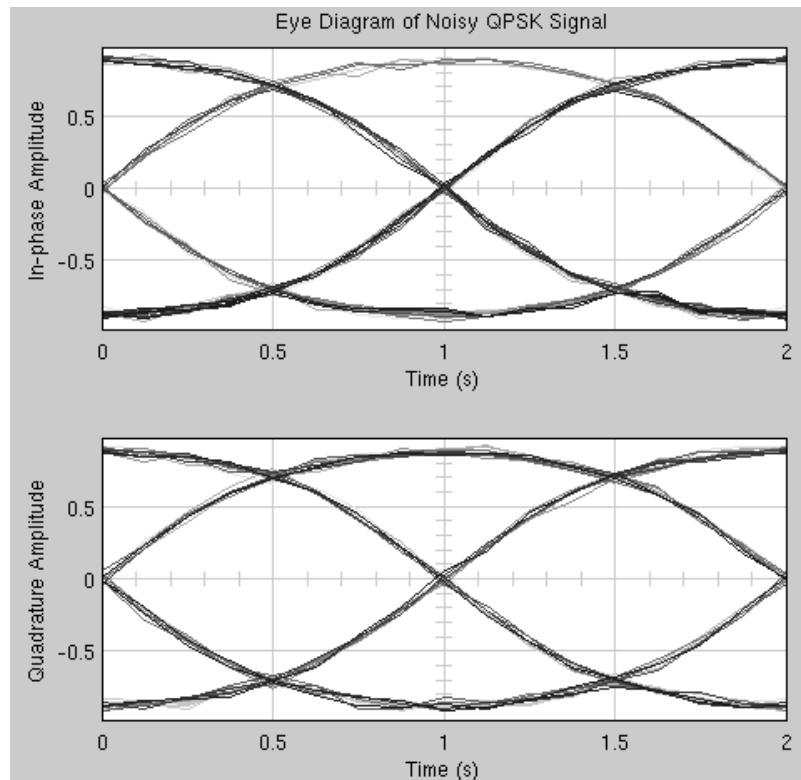


Figure 2.14 A typical eye pattern for the QPSK signal. The width of the opening indicates the time over which sampling for detection might be performed. The optimum sampling time corresponds to the maximum eye opening, yielding the greatest protection against noise. If there were no filtering in the system, then the system would look like a box rather than an eye.

$$h(t) = h_T(t) * h_C(t) * h_R(t) \quad (2.36)$$

Now, let us consider under which condition we can assure that there is no intersymbol interference between symbols. The input to the equivalent channel, $s_a(t)$, has been defined in (2.35). As mentioned before, each analog pulse is scaled by the value of the symbol, so we can express $s_a(t)$ in another way:

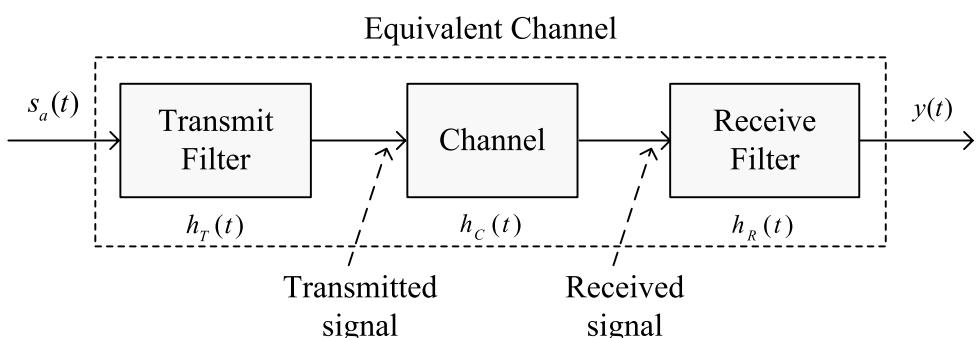


Figure 2.15 The equivalent channel of a communication system, which consists of the transmit filter, the channel, and the receive filter.

$$s_a(t) = \sum a_k \delta(t - kT) \quad (2.37)$$

where a_k is the value of the k th symbol. It yields the output to the equivalent channel, $y(t)$, which is:

$$y(t) = s_a(t) * h(t) = a_k [\delta(t - kT) * h(t)] = a_k h(t - kT) \quad (2.38)$$

Therefore, given a specific time instant, for example, $t = mT$, where m is a constant, the input $s_a(t)$ is:

$$s_a(mT) = \sum a_k (mT - kT) = a_m \quad (2.39)$$

Consequently, the output $y(t)$ becomes:

$$y(mT) = a_k h(mT - kT) = a_0 h(mT) + a_1 h(mT - T) + \dots + a_m h(mT - mT) \quad (2.40)$$

Since we do not want the interference from the other symbols, we would like the output to contain only the a_m term, which is:

$$y(mT) = a_m h(mT - mT) \quad (2.41)$$

Moreover, it means at a time instant $t = mT$, we need to have:

$$h(mt - kT) = \begin{cases} C & k = m \\ 0 & k \neq m \end{cases} \quad (2.42)$$

where C is some nonzero constant.

If we generalize (2.42) to any time instant t , we can get the Nyquist pulse shaping theory as follows.

The condition that one pulse does not interfere with other pulses at subsequent T -spaced sample instants is formalized by saying that $h(t)$ is a *Nyquist pulse* if it satisfies:

$$h(t) = h(kT) = \begin{cases} C & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (2.43)$$

for all integers k .

2.4.3 Two Nyquist Pulses

In this section, we will introduce two important Nyquist pulses, namely, sine pulse and raised cosine pulse. When considering (2.43), the most straightforward pulse we can think of is a rectangular pulse with time-width less than T , or any pulse shape that is less than T wide. But the bandwidth of the rectangular pulse (and other narrow pulse shapes) may be too wide. Narrow pulse shapes do not utilize the spectrum efficiently, but wide pulse shapes may cause ISI, so what is needed is a signal that is wide in time (and narrow in frequency) that also fulfills the Nyquist condition in (2.43) [8].

2.4.3.1 Sinc Pulse

One possible solution is the *sine pulse*. In mathematics, the sine function is defined as:

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \quad (2.44)$$

As shown in Figure 2.16, when variable x takes an integer value k , the value of the sine function will be:

$$\text{sinc}(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (2.45)$$

In other words, zero crossings of the sine function occur at nonzero integer values.

Another important property of sinc function is that sine pulse is the inverse Fourier transform of a rectangular signal, as shown in Figure 2.17(a). Suppose the rectangular signal is defined as [9]:

$$H(\omega) = \begin{cases} T & |\omega| < \frac{1}{2T} \\ 0 & \text{otherwise} \end{cases} \quad (2.46)$$

Take the inverse Fourier transform of the rectangular signal, we can obtain the sine signal as:

$$h(t) = \text{sinc} \frac{t}{T} \quad (2.47)$$

Make $t = kT$ in (2.47), and we will have:

$$h(t) = h(kT) = \text{sinc} \frac{kT}{T} = \text{sinc}(k) \quad (2.48)$$

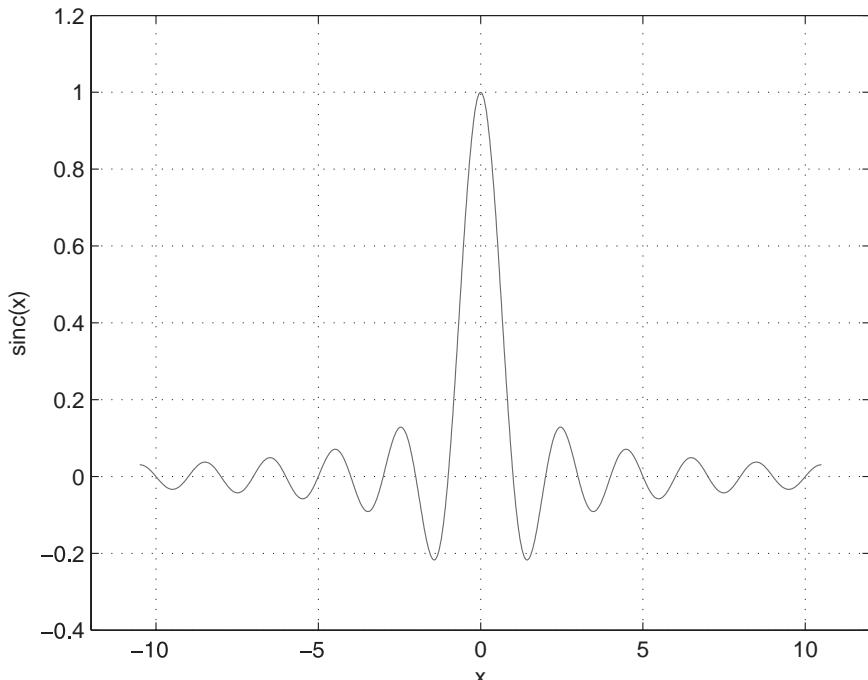


Figure 2.16 The plot of sine function as defined in (2.44). The x-axis is x , and the y-axis is $\text{sinc}(x)$.

Since k is an integer here, according to (2.45), we can continue writing (2.48) as:

$$h(t) = h(kT) = \text{sinc} \frac{kT}{T_{\div}} = \text{sinc}(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (2.49)$$

Comparing (2.49) with (2.43), we can easily find that if we make $t = kT$, the sine pulse exactly satisfies Nyquist pulse shaping theory in Section 2.4.2. In other words, by choosing sampling time at kT (sampling frequency equals $\frac{1}{T}$), our sampling instants are located at the equally spaced zero crossings, as shown in Figure 2.17(b), so there will be no intersymbol interference.

Recall the Nyquist sampling theorem in Section 2.3.3 states that a real signal, $x(t)$, which is bandlimited to B Hz can be reconstructed without error using a

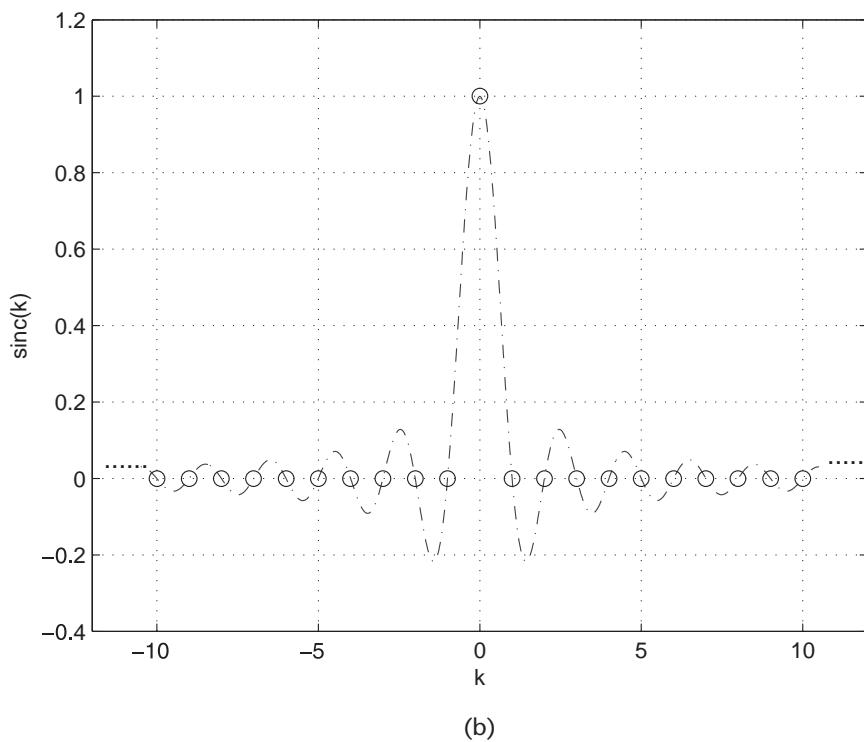
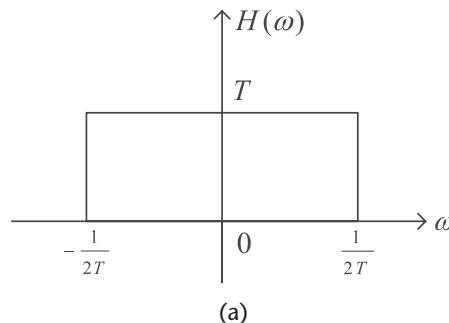


Figure 2.17 The sine pulse on time domain and its Fourier transform, rectangular pulse, on frequency domain. (a) The rectangular pulse on frequency domain, defined in (2.46). (b) The sine pulse defined in (2.49). The x-axis is k , where $k = \frac{t}{T}$, and the y-axis is $\text{sinc}(k)$.

minimum sampling frequency of $F_s = 2B$ Hz. In this case, the minimum sampling frequency is $F_s = \frac{1}{T}$ Hz. Therefore, the corresponding minimum bandwidth is:

$$B = \frac{F_s}{2} = \frac{1}{2T} \quad (2.50)$$

which is exactly the bandwidth of the rectangular signal defined in (2.46). Based on the previous discussion, this choice of sine pulse $h(t)$ yields the minimum bandwidth $B = B_{\min} = \frac{1}{2T}$, so it is called Nyquist-I pulse [10].

2.4.3.2 Raised Cosine Pulse

Nyquist-I pulse is a type of very attractive pulse, because it is wide in time and narrow in frequency, which means it has the advantages of both spectrum efficiency and no ISI. However, Nyquist-I pulse is not practical, as it has ISI sensitivity due to timing errors. Take sine pulse, for instance; for large t , the sine pulse defined in (2.47) has the following approximation:

$$h(t) \sim \frac{1}{t} \quad (2.51)$$

so it is obvious that timing error can cause large ISI.

Given that Nyquist-I pulse is not practical, we need to introduce Nyquist-II pulse, which has a larger bandwidth $B > B_{\min}$, but with less ISI sensitivity. Since this type of pulse is more practical, it is much more widely used in practice.

The raised cosine pulse is one of the most important types of Nyquist-II pulses, which has the frequency transfer function defined as:

$$H_{RC}(f) = \begin{cases} T & 0 \leq |f| \leq \frac{1-\beta}{2T} \\ \frac{T}{2} \left(1 + \cos \frac{\pi T}{\beta}\right) & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0 & |f| > \frac{1+\beta}{2T} \end{cases} \quad (2.52)$$

where β is the *rolloff factor*, which takes a value from 0 to 1, and $\frac{\beta}{2T}$ is the *excess bandwidth*.

The spectrum of raised cosine pulse is shown in Figure 2.18. In general, it has the bandwidth $\beta = 1/(2T)$. When $\beta = 0$, the bandwidth $\beta = 1/(2T)$, and there is no excess bandwidth, which is actually a rectangular pulse. On the other end, when $\beta = 1$, it reaches the maximum bandwidth $B = 1/T$.

By taking the inverse Fourier transform of $H_{RC}(f)$, we can obtain the impulse response of raised cosine pulse, defined as:

$$h_{RC}(t) = \frac{\cos \pi \frac{\beta t}{T}}{1 - 2 \frac{\beta t}{T}} \operatorname{sinc} \frac{\pi t}{T} \quad (2.53)$$

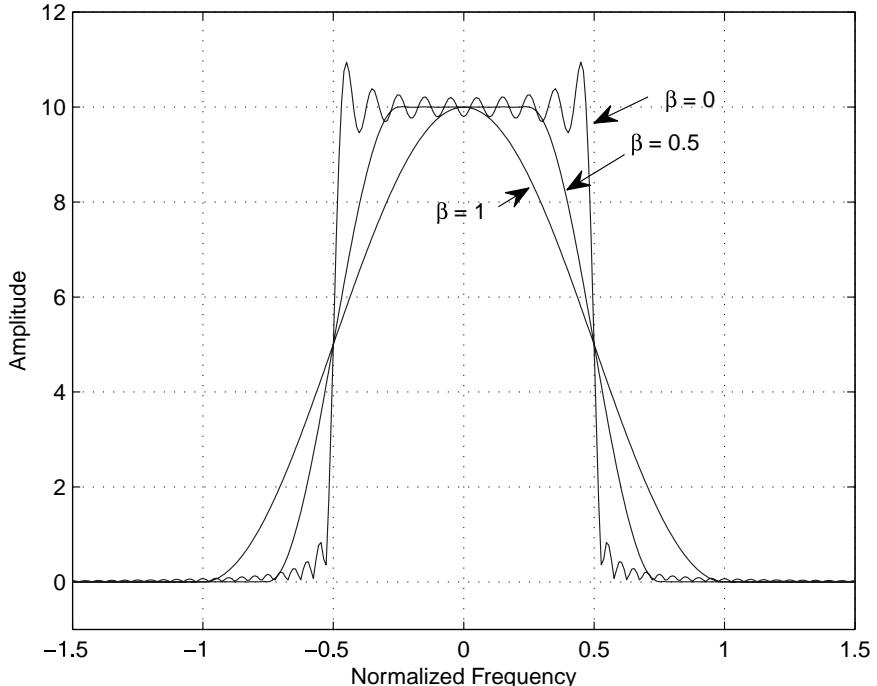


Figure 2.18 Spectrum of a raised cosine pulse, defined in (2.52), which varies by the rolloff factor β . The x-axis is the normalized frequency f_0 . The actual frequency can be obtained by f_0/T .

Nyquist-II pulses do not have an ISI sensitivity because their peak distortion, the tail of $h_{RC}(t)$, converges quickly, which can be expressed as:

$$D_p = \sum_{n=1}^{\infty} |h_{RC}(n + (n-k))| \sim \frac{1}{n^3} \quad (2.54)$$

Therefore, when timing error occurs, the distortion will not accumulate to infinity in a similar fashion related to Nyquist-I pulses [10].

Actually, in many practical communications systems, *root raised cosine filters* are used [11]. If we consider the communication channel as an ideal channel, and we assume that the transmit filter and the receive filter are identical, we can use root raised cosine filters for both of them, and their net response must equal to $H_{RC}(f)$ defined in (2.52). Since the impulse response of the equivalent channel can be expressed as:

$$h(t) = h_T(t) * h_C(t) * h_R(t) \quad (2.55)$$

where $h_C(t)$ is the impulse response of the communication channel, and $h_T(t)$ and $h_R(t)$ are the impulse responses of the transmit filter and the receive filter, it means on frequency domain, we have:

$$H_{RC}(f) = H_T(f)H_R(f) \quad (2.56)$$

Therefore, the frequency response of root raised cosine filter must satisfy:

$$|H_T(f)| = |H_R(f)| = \sqrt{|H_{RC}(f)|} \quad (2.57)$$

2.5 FILTERING

When doing signal processing, we usually need to get rid of the noise and extract the useful signal. This process is called *filtering*, and the device employed is called a *filter*, which discriminates, according to some attribute of the objects applied at its input, what passes through it [1]. Typical filter is a frequency-selective circuit, namely, if noise and useful signal have different frequency distributions, by applying this circuit, noise will be attenuated or even eliminated, while the useful signal is retained.

Filters can be classified from different aspects. For example, according to its frequency response, filters can be classified as lowpass, highpass, bandpass and bandstop. According to the signal it deals with, filters can be classified as analog filters and digital filters. Specifically, analog filters deal with continuous-time signals, while digital filter deal with discrete-time signals. This section will focus on digital filters.

2.5.1 Ideal Filter

As mentioned before, filters are usually classified according to their frequency-domain characteristics as lowpass, highpass, bandpass, and bandstop or band-eliminated filters [1]. The ideal magnitude response characteristics of these types of filters are shown in Figure 2.19.

According to Figure 2.19, the magnitude response characteristics of an ideal filter can be generalized as follows: In passband, the magnitude is a constant. While in stopband, the magnitude falls to zero. However, in reality, this type of ideal filter cannot be achieved, so a practical filter is actually the optimum approximation of an ideal filter.

2.5.2 Z-Transform

In Section 2.2, we have introduced the Fourier transform, which deals with continuous-time signals on the frequency domain. Since we are focusing on digital filter design in this section, where discrete-time signals are involved, we need to introduce a new type of transform, namely, the z-transform.

The z-transform of a discrete-time signal $x[n]$ is defined as the power series:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \quad (2.58)$$

where z is a complex variable [1].

The z-transform is used to analyze discrete-time systems. Its continuous-time counterpart is the Laplace transform, defined as follows:

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt \quad (2.59)$$

where t is the time variable in seconds across the time domain, and $s = \sigma + j\omega$ is a complex variable. When evaluated along the $j\omega$ axis (i.e., $\sigma = 0$), the Laplace transform reduces to the Fourier transform defined in (2.11). Thus, the Laplace

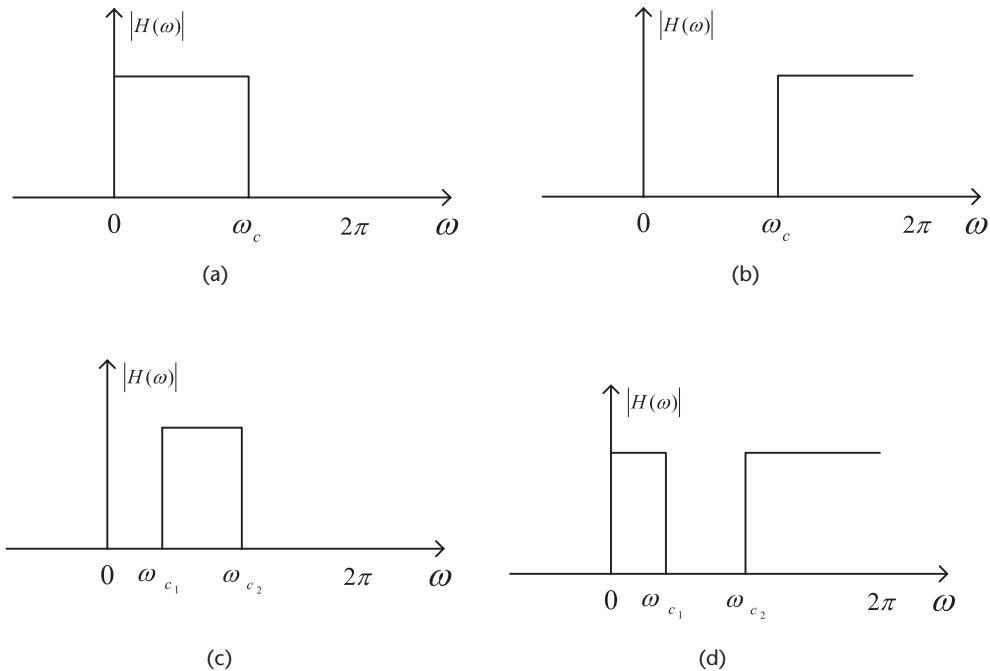


Figure 2.19 Ideal magnitude response characteristics of four types of filters on the frequency range $[0, 2\pi]$. (a) Lowpass filter. (b) Highpass filter. (c) Bandpass filter, where $(\omega_{c1}, \omega_{c2})$ is pass band. (d) Bandstop filter, where $(\omega_{c1}, \omega_{c2})$ is stop band.

transform generalizes the Fourier transform from the real line (the frequency axis $j\omega$) to the entire complex plane.

According to Section 2.3.2, we know that if a continuous-time signal $x(t)$ is uniformly sampled, its sampling signal $x_s(t)$ can be expressed as:

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT) \delta(t - nT) \quad (2.60)$$

where T is the sampling interval. If we take the Laplace transform of both sides, we will get:

$$X_s(s) = \int_{-\infty}^{\infty} x_s(t) e^{-st} dt = \sum_{n=-\infty}^{\infty} x(nT) \delta(t - nT) e^{-st} dt \quad (2.61)$$

Since integration and summation are both linear operators, we can exchange their order. Then, based on the sampling property of the delta function, we can further get:

$$X_s(s) = \sum_{n=-\infty}^{\infty} x(nT) \delta(t - nT) e^{-st} dt = \sum_{n=-\infty}^{\infty} x(nT) e^{-snT} \quad (2.62)$$

Let $z = e^{sT}$, or $s = \frac{1}{T} \ln z$; then (2.62) becomes:

$$X(z) = \sum_{n=-\infty}^{\infty} x(nT) z^{-n} \quad (2.63)$$

Since T is the sampling interval, $x(nT) = x[n]$. The previous equation can be further written as:

$$X(z) = \sum_{n=0}^{\infty} x[n]z^{-n} \quad (2.64)$$

which is exactly the definition of z-transform in (2.58). Therefore, the z-transform and the Laplace transform can be connected by:

$$z = e^{sT} \quad (2.65)$$

or

$$s = \frac{1}{T} \ln(z) \quad (2.66)$$

According to (2.58), we know that z-transform is the series of z^{-1} . Actually, the z-transform has no meaning unless the series converge. Given a limitary-amplitude sequence $x[n]$, the set of all the z values that makes its z-transform converge is called region of convergence (ROC). Based on the theory of series, these z-values must satisfy:

$$\left| \sum_{n=0}^{\infty} x[n]z^{-n} \right| < \infty \quad (2.67)$$

The frequently used z-transform pairs and their region of convergence are listed in Table 2.5.

When discussing a linear time-invariant system, the z-transform of its system impulse response can be expressed as the ratio of two polynomials:

$$H(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + \dots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \dots + a_1 z + a_0} = \frac{B(z)}{A(z)} \quad (2.68)$$

where the roots of $A(z) = 0$ are called the *poles* of the system, and the roots of $B(z) = 0$ are called the *zeros* of the system. It is possible that the system can have multiple poles and zeros.

If we factorize the numerator $B(z)$ and denominator $A(z)$, (2.68) can be written as:

$$H(z) = C \frac{(z - z_1)(z - z_2) \cdots (z - z_m)}{(z - p_1)(z - p_2) \cdots (z - p_n)} \quad (2.69)$$

where C is a constant, $\{p_k\}$ are all the poles, and $\{z_k\}$ are all the zeros. It will help us draw the pole-zero plot of $H(z)$.

Suppose we have a linear time-invariant system whose system impulse response is defined as:

$$h[n] = n^2 a^n u[n] \quad (2.70)$$

According to Table 2.5, its z-transform is as follows:

Table 2.5 Z-Transform Table: Selected Pairs [12]

$x[n]$	$X(z)$	<i>Region of Convergence</i>
$\delta[n]$	1	all z
$a^n u[n]$	$\frac{z}{z - a}$	$z > a$
$na^n u[n]$	$\frac{az}{(z - a)^2}$	$z > a > 0$
$n^2 a^n u[n]$	$\frac{az(z + a)}{(z - a)^3}$	$z > a > 0$
$\frac{1}{a^n} + \frac{1}{b^n} u[n]$	$\frac{az}{az - 1} + \frac{bz}{bz - 1}$	$ z > \max \frac{1}{ a }, \frac{1}{ b }$
$a^n u[n] \sin[-\omega_0 n]$	$\frac{az \sin \omega_0}{z^2 - 2az \cos \omega_0 + a^2}$	$z > a > 0$
$a^n u[n] \cos[-\omega_0 n]$	$\frac{z(z - a \cos \omega_0)}{z^2 - 2az \cos \omega_0 + a^2}$	$z > a > 0$
$e^{an} u[n]$	$\frac{z}{z - e^a}$	$z > e^{-a}$
$e^{an} u[n] \sin[-\omega_0 n]$	$\frac{ze^a \sin \omega_0}{z^2 e^{2a} - 2ze^a \cos \omega_0 + 1}$	$z > e^{-a}$
$e^{an} u[n] \cos[-\omega_0 n]$	$\frac{ze^a (ze^a \cos \omega_0)}{z^2 e^{2a} - 2ze^a \cos \omega_0 + 1}$	$z > e^{-a}$

$$H(z) = a \frac{z(z + a)}{(z - a)^3} \quad (2.71)$$

Comparing (2.71) with (2.69), we easily get that this system has two zeros, $z_1 = 0$ and $z_2 = -a$, and three poles, $p_1 = p_2 = p_3 = a$. Therefore, its pole-zero plot is shown in Figure 2.20.

There are several properties of the z-transform that are useful when studying signals and systems in the z-transform domain. Since these properties are very similar to those of the Fourier transform introduced in Section 2.2.3, we list them in Table 2.6 without further discussion.

2.5.3 Digital Filtering

In order to perform digital filtering, input and output of a digital system must both be discrete-time series. If the input series is $x[n]$, the impulse response of the filter is $h[n]$, then the output series $y[n]$ will be:

$$y[n] = x[n] * h[n] \quad (2.72)$$

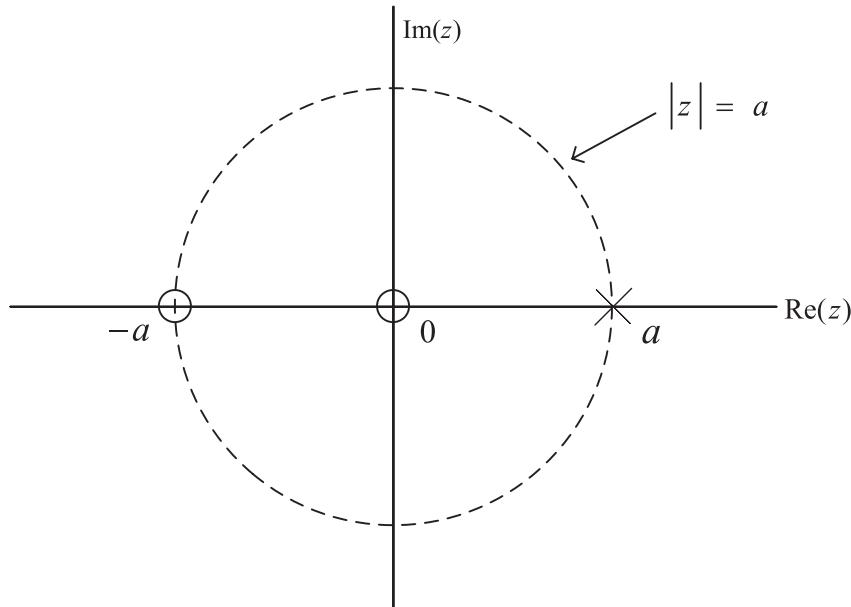


Figure 2.20 Pole-zero plot of the system defined in (2.71). Poles are denoted using crossings, and zeros are denoted using circles. The region of convergence of this system is the region outside the circle $|z| = |a|$.

According to the time convolution property in Table 2.6, on the frequency domain, (2.72) is equivalent to:

$$Y(z) = X(z)H(z) \quad (2.73)$$

where $X(z)$ and $Y(z)$ are the z-transforms of the input and output series, $x[n]$ and $y[n]$, and $H(z)$ is the z-transform of $h[n]$.

Table 2.6 Z-Transform Properties, Based on [12]

Property	Time Signal	Z-Transform Signal
Linearity	$\sum_{m=1}^N a_m x_m(t)$	$\sum_{m=1}^N a_m X_m(z)$
Symmetry	$x[-n]$	$X(z^{-1})$
Shifting	$x[n - m]$	$z^{-m} X(z)$
Scaling	$a^n x[n]$	$X \frac{a}{z}$
Derivative	$n x[n]$	$z \frac{d\lambda(z)}{dz}$
Integration	$\sum_{m=1}^n x[m]$	$\frac{z}{z - 1} X(z)$
Time convolution	$x[n] * h[n]$	$X(z)H(z)$
Frequency convolution	$x[n]h[n]$	$\frac{1}{2\pi j} X(v)H \frac{z}{v} \frac{dv}{v}$

Suppose the time signal is $x[n]$ and its z-transform signal is $X(z)$.

As mentioned in Section 2.5.1, the ideal filters are not achievable in practice. Consequently, we limit our attention to the class of linear time-invariant systems specified by the difference equation [1]:

$$y[n] = \sum_{k=1}^N a_k y[n-k] + \sum_{k=0}^M b_k x[n-k] \quad (2.74)$$

where $y[n]$ is the current filter output, the $y[n-k]$ are previous filter outputs, the $x[n-k]$ are current or previous filter inputs. This system has the following frequency response:

$$H(z) = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}} \quad (2.75)$$

where the $\{a_k\}$ are the filter's feedback coefficients corresponding to the poles of the filter, the $\{b_k\}$ are the filter's feed forward coefficients corresponding to the zeros of the filter, and N is the filter's order.

The basic digital filter design problem is to approximate any of the ideal frequency response characteristics with a system that has the frequency response (2.75) by properly selecting the coefficients $\{a_k\}$ and $\{b_k\}$ [1].

There are two basic types of digital filters, finite impulse response (FIR) and infinite impulse response (IIR) filters. When excited by an unit sample $\delta[n]$, the impulse response $h[n]$ of a system may last a finite duration, as shown in Figure 2.21(a), or forever even before the input is applied, as shown in Figure 2.21(b). In the former case, the system is finite impulse response, and, in the latter case, the system is infinite impulse response.

A FIR filter of length M with input $x[n]$ and output $y[n]$ can be described by the difference equation [1]:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_{M-1} x[n-M+1] = \sum_{k=0}^{M-1} b_k x[n-k] \quad (2.76)$$

where the filter has no feedback coefficients $\{a_k\}$, so $H(z)$ has only zeros.

IIR filter has been defined in (2.74), which has one or more nonzero feedback coefficients $\{a_k\}$. Therefore, once the filter has been excited with an impulse, there is always an output.

2.5.3.1 Case Study: Cascaded Integrator-Comb Filter

Since cascaded integrator-comb (CIC) filter plays an important role in the universal software radio peripheral (USRP) hardware, this section introduces this special filter. CIC filters, invented by Eugene B. Hogenauer, are a class of FIR filters used in multirate processing. The CIC filter finds applications in interpolation and decimation. Unlike most FIR filters, it has a decimator or interpolator built into the architecture [13].

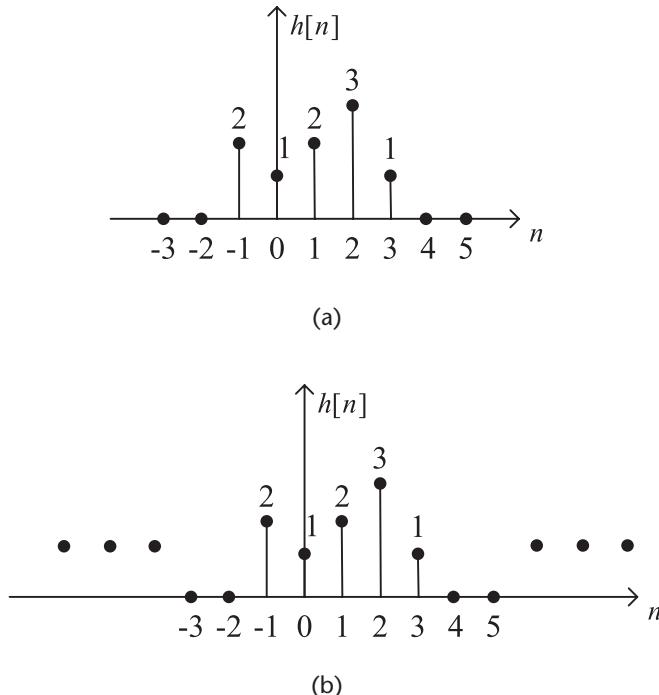


Figure 2.21 The impulse responses of an FIR filter and an IIR filter. (a) The impulse response of an FIR filter. (b) The impulse response of an IIR filter.

A CIC filter consists of one or more integrator and comb filter pairs. In the case of a decimating CIC, the input signal is fed through one or more cascaded integrators, then a down-sampler, followed by one or more comb sections whose number equals to the number of integrators. An interpolating CIC is simply the reverse of this architecture, with the down-sampler replaced with a up-sampler, as shown in Figure 2.22.

To illustrate a simple form of a comb filter, consider a moving average FIR filter described by the difference equation:

$$y[n] = \frac{1}{M+1} \sum_{k=0}^M x[n-k] \quad (2.77)$$

The system function of this FIR filter is:

$$H(z) = \frac{1}{M+1} \sum_{k=0}^M z^{-k} = \frac{1}{M+1} \frac{[1 - z^{-(M+1)}]}{(1 - z^{-1})} \quad (2.78)$$

Suppose we replace z by z^L , where L is a positive integer; then the resulting comb filter has the system function:

$$H_L(z) = \frac{1}{M+1} \frac{[1 - z^{L(M+1)}]}{(1 - z^L)} \quad (2.79)$$

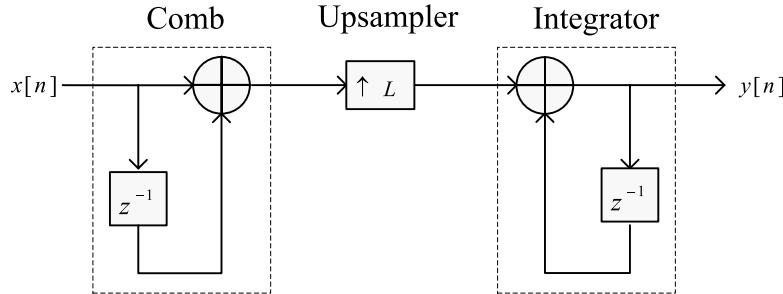


Figure 2.22 The structure of an interpolating cascaded integrator-comb filter [14], with input signal $x[n]$ and output signal $y[n]$. This filter consists of a comb and integrator filter pair, and an upsampler with interpolation ratio L .

where L is decimation or interpolation ratio, and M is number of samples per stage, usually 1 or 2. This filter has zeros on the unit circle at:

$$z_k = e^{j2\pi k/L(M+1)} \quad (2.80)$$

for all integer value of k except $k = 0, L, 2L, \dots, ML$, as shown in Figure 2.23.

The common form of the CIC filter usually consists of several stages, and the system function for the composite CIC filter is:

$$H(z) = H_L(z)^N = \frac{1}{M+1} \frac{1 - z^{-L(M+1)}}{1 - z^{-L}}^N \quad (2.81)$$

where N is number of stages in the composite filter.

Characteristics of CIC filters include linear phase response and utilizing only delay and addition and subtraction. In other words, it requires no multiplication operations. Therefore, it is suitable for hardware implementation.

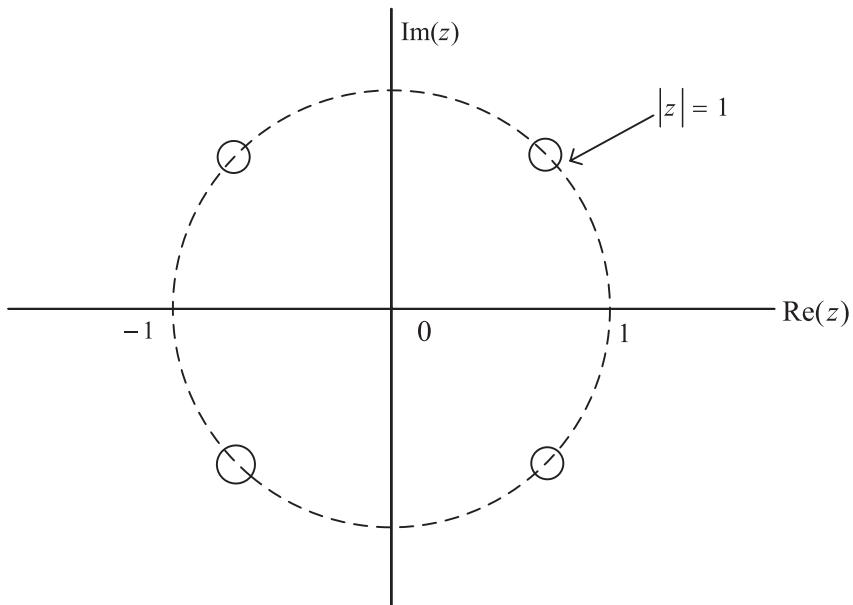


Figure 2.23 The zeros of a CIC filter defined in (2.79), where all the zeros are on the unit circle.

2.6 CHAPTER SUMMARY

Software defined radio experimentation requires both strong computer skills and extensive knowledge of digital signal processing. This chapter lists some useful topics including sampling, pulse shaping, and filtering. The purpose of this chapter is to help the readers get prepared for the experimentation chapters, especially for the USRP hardware. For example, on the USRP transmit path, there are analog-to-digital converters and digital down-converters, where the theory of sampling is needed. On the USRP receive path, besides digital-to-analog converters and digital up-converters, there are CIC filters, where filtering knowledge is useful. In addition, when you build a real communication system, you will need a transmit filter and receive filter, which requires expertise on pulse shaping.

2.7 PROBLEMS

1. [Periodic Signal] Suppose we have a discrete signal

$$x[n] = \cos\left(\frac{\pi}{2}n\right) + e^{j\frac{4\pi}{5}n} \quad (2.82)$$

- (a) Is it a periodic signal?
- (b) If yes, what is its period?

2. [LTI System] Suppose we have a discrete system

$$y[n] = nx[n] + x[n-1]. \quad (2.83)$$

Is it an LTI system? Please justify.

3. [Casual System] Please determine whether the following systems belong to causal systems or not and explain.

- (a) $y[n] = x[n] - x[n+1]$
- (b) $y[n] = \sum_{k=-\infty}^n x[k]$
- (c) $h(t) = e^{-2t} u(t)$

4. [Fourier Transform] Assume the Fourier transform of a signal $f(t)$ is $F(j\omega)$.

- (a) What is the Fourier transform of $f(2t - 3)$?
- (b) Suppose we have another signal $p(t) = \cos(t)$ and $f_s(t) = f(t)p(t)$, what is the Fourier transform of $f_s(t)$?

5. [Nyquist Sampling Theorem] Suppose we have a signal $x(t)$ whose frequency response is

$$X(j\omega) = \begin{cases} 1 & |\omega| < 2 \text{ rad/s} \\ 0 & \text{otherwise} \end{cases} \quad (2.84)$$

and assume we have another signal $f(t)$ defined as

$$f(t) = x(t)\cos(2t). \quad (2.85)$$

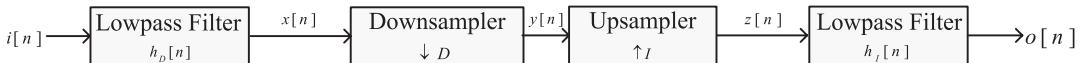
What is the Nyquist sampling frequency of $f(t)$?

6. [Sampling Rate Conversion] Let us cascade the decimator and the interpolator introduced in Section 2.3.4 to perform a sampling rate conversion by factor 3/4, as shown in Figure 2.24(a), where the input signal is $i[n]$ and the output signal is $o[n]$. Suppose the spectrum of the input signal is shown in Figure 2.24(b).

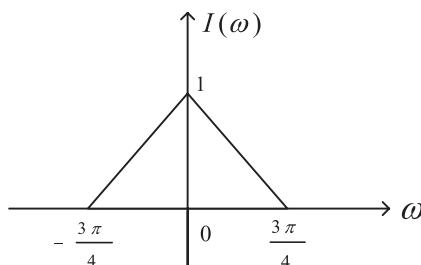
- (a) Design the system by specifying D , I , $H_D(\omega)$ and $H_I(\omega)$.
- (b) Plot the spectrum of all the signals in this system, namely $x[n]$, $y[n]$, $z[n]$ and $o[n]$.

7. [Sampling Rate Conversion] There is a cascaded system shown in Figure 2.25, where the input $x[n] = \frac{1}{3^n} u[n]$. Assume the upsampler and down-sampler are ideal (with ideal low-pass filters) and the overall system response $F(e^{j\omega})$ is LTI. Determine this overall system response, namely, find

$$F(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})}. \quad (2.86)$$



(a) A sampling rate conversion system.



(b) The spectrum of input signal.

Figure 2.24 Sampling rate conversion problem.

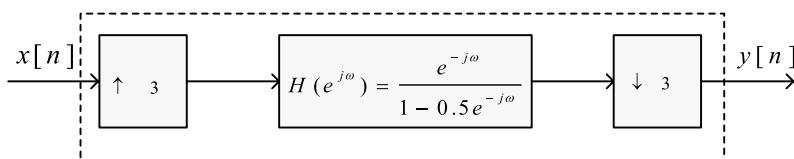


Figure 2.25 A cascaded system, with input signal $x[n]$ and output signal $y[n]$.

8. [Z Transform] Given a discrete signal

$$x[n] = 3\delta(n-2) + 2\delta(n-k), \quad (2.87)$$

what is its z transform? What is its region of convergence?

9. [Z Transform] Given a z transform

$$X(z) = \frac{5z}{z^2 - 3z + 2}, \quad (2.88)$$

what is its inverse z transform?

10. [Z Transform] Let $y[n] = x[n] * h[n]$ be the output of a discrete-time system, where $*$ is the convolution operator. Let $h[n] = \frac{1}{3^n} u[n]$ and

$$x[n] = \begin{cases} \left(\frac{1}{2}\right)^n & n \geq 0, \\ \left(\frac{1}{4}\right)^n & n < 0. \end{cases} \quad (2.89)$$

Find $Y(z)$ and specify its region of convergence.

11. [Stable System] Suppose a linear time-invariant system is specified by the following difference equation

$$y[n] + 0.2y[n-1] - 0.24y[n-2] = x[n] + x[n-1] \quad (2.90)$$

- (a) What is its frequency response $H(z)$?
- (b) What are its zeros and poles?
- (c) Is it a stable system?

12. [Ideal Filter] Suppose we have a system shown in Figure 2.26(a), where the input signals are

$$f(t) = \frac{\sin(2t)}{2\pi t}, \quad (2.91)$$

and

$$s(t) = \cos(1000t). \quad (2.92)$$

The frequency response of the bandpass filter is shown in Figure 2.26(b).

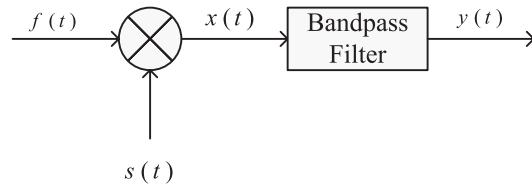
- (a) What is output signal $y(t)$?
- (b) Please draw the frequency response of $y(t)$.

13. [FIR Filter] Suppose we have a three order real-coefficient FIR filter, whose frequency response is

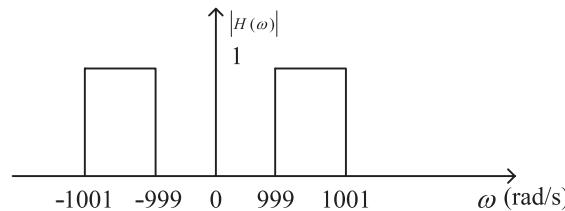
$$H(e^{j\omega}) = \sum_{n=0}^3 b[n]e^{-j\omega n} \quad (2.93)$$

Given $H(e^{j0}) = 2$, $H(e^{j\frac{\pi}{2}}) = 7$, $H(e^{j3\pi}) = 0$, what is $H(z)$?

14. [FIR Filter] Design a second-order FIR filter whose frequency response has a zero magnitude at $\omega = \frac{\pi}{2}$. Specify the design using time-domain difference equation.



(a) Block diagram of the system.



(b) Frequency response of the bandpass filter.

Figure 2.26 Ideal filter problem.

15. [Linear Filter] Draw a direct form realization of the linear filter with the following impulse response:

$$h[n] = \{-2, 5, 8, 3, -3\}. \quad (2.94)$$

References

- [1] Proakis, and J. Dimitris, G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd edition, Prentice Hall, 1996.
- [2] Chang, *Academic Genealogy of Mathematicians*, World Scientific, 2011.
- [3] Joseph Fourier, http://en.wikipedia.org/wiki/Joseph_Fourier.
- [4] Fulop, S. A., *Speech Spectrum Analysis*. New York: Springer-Verlag, LC, 2011.
- [5] Cavicchi, T. J., *Digital Signal Processing*, John Wiley Sons, 2000.
- [6] Elali, T. S., and M. A. Karim, *Continuous Signals and Systems with MATLAB*, CRC Press, 2001.
- [7] James, J. F., *A Student's Guide to Fourier Transforms: With Applications in Physics and Engineering*, 3rd edition, Cambridge University Press, 2011.
- [8] Johnson, C. R. and W. A. Sethares, *Telecommunications Breakdown: Concepts of Communication Transmitted via Software-Defined Radio*, Prentice Hall, 2004.
- [9] Johnson C. R., W. A. Sethares, and A. G. Klein, *Software Receiver Design: Build Your Own Digital Communications System in Five Easy Steps*, Cambridge University Press, 2011.
- [10] Wyglinski, A. M., Nekovee and Y. T. Hou, *Cognitive Radio Communications and Networks: Principles and Practice*, Academic Press, 2009.
- [11] Barry J., E. A. Lee, and D. G. Messerschmitt, *Digital Communication*, 3rd edition, Kluwer Academic Press, 2004.
- [12] Alan Jeffrey and Daniel Zwillinger, *Table of Integrals, Series, and Products*, 7th edition, Academic Press, 2007.

- [13] Hogenauer, E. B., “An Economical Class of Digital Filters for Decimation and Interpolation” *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 29, No. 2, 1981, pp. 155–162.
- [14] Lyons, G., *Streamlining Digital Signal Processing: A Tricks of the Trade Guidebook*, Wiley-IEEE Press, 2007.

Probability Review

We live in a world that possesses a significant degree of randomness. Whether it is the final scores for a sporting event, the precise arrival time of a friend, or the next time that your cellular phone rings, there exists a wide range of events occurring throughout our daily lives that have some level of uncertainty associated with them. However, despite the fact that these and many other phenomena appear “random” at first glance, it is possible to extract a quantitative description of the behavior for these occurrences and precisely characterize them using mathematical models. These models are referred to as *random variables* for static events and *random processes* for time-varying events. In this chapter, we will present the fundamental mathematical tools needed to understand, create, and analyze these types of models within the context of wireless data transmission.

3.1 FUNDAMENTAL CONCEPTS

To accurately quantify the behavior of a random variable or a random process, it is necessary to understand an assortment of fundamental mathematical tools that are needed in order to develop these models. One of these tools is the concept of *relative frequency*, which is used to provide a basic metric regarding the rate at which a specific outcome resulting from a random phenomenon occurs. Suppose we observe the outcome of M trials of a random phenomenon, such as the tossing of a fair, two-sided coin. Labeling each outcome as O_1, O_2, \dots, O_M , what we are interested in is the number of times a “heads” occur out of the total number of M coin tosses.

Defining $N_n(O_i)$ as the number of trials for which the O_i outcome occurred over a total of n trials, we can specify the relative frequency $N_n(O_i)/n$. Furthermore, in order to obtain the *probability measure* or *probability* of an event having an outcome O_i , one would take the limit of the relative frequency as it approaches infinity, namely:

$$P_i = \lim_n \frac{N_n(O_i)}{n} \quad (3.1)$$

Throughout the rest of this chapter and the this book, we will be using the probability measure to characterize the likelihood that a specific outcome will occur.

3.1.1 Set Theory

When dealing with the outcomes of random events and phenomena, one useful mathematical tool is *set theory*. Suppose we define the *sample space* as the set of all possible, distinct, indecomposable measurements that could be observed. The sample space is usually denoted as Ω and can either be continuous (e.g., $\Omega = [0, \infty)$) or discrete (e.g., $\Omega = \{0, 1, 2, \dots\}$). An *outcome* is an element of a point located within the sample space Ω , while an *event* is a collection of outcomes.

The idea of sample spaces, outcomes, and events can be described using *set theory*. For example, suppose we are given A and B as a collection of points, and ω is a point in Ω ; then:

1. $\omega \in A$: “ ω is an element of A .”
2. $A \subseteq B$: “ A is a subset of B .”
3. $A = B$: “ A is an equal set to B .”
4. $A \subset B$ and $A \neq B$: “ A is a proper subset of B .”

Graphically representing these relationships can be readily accomplished using *Venn diagrams*. An example of a Venn diagram for the case of $A \subseteq B$ is shown in Figure 3.1. Furthermore, a collection of commonly used set identities are summarized in Table 3.1, which can be generalized into:

$$B = \bigcup_{n=1}^{\infty} A_n \Leftrightarrow \bigcup_{n=1}^{\infty} (B - A_n) \text{ and } B = \bigcap_{n=1}^{\infty} A_n \Leftrightarrow \bigcap_{n=1}^{\infty} (B - A_n) \quad (3.2)$$

and

$$\left(\bigcap_{n=1}^{\infty} A_n \right)^C = \bigcup_{n=1}^{\infty} A_n^C \text{ and } \left(\bigcup_{n=1}^{\infty} A_n \right)^C = \bigcap_{n=1}^{\infty} A_n^C \quad (3.3)$$

Finally, a summary of possible set operations and their associated Venn diagram representations are shown in Table 3.2.

3.1.2 Partitions

Partitioning of a sample space is a very powerful tool in set theory since it allows an individual to “divide and conquer” all the possible events into mutually exclusive, complementary subsets. Suppose we have a family of nonempty sets B_n that are pairwise disjoint and form partitions of the entire sample space Ω , as shown in Figure 3.2(a), where the sets satisfy:

$$B_n \cap B_m = \emptyset, \forall n, m \text{ and } n \neq m,$$

$$\bigcup_{n=1}^N B_n = \Omega \quad (3.4)$$

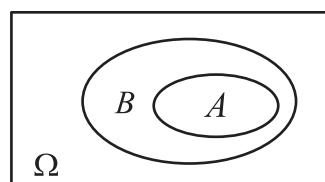


Figure 3.1 Illustration of a Venn diagram representing $A \subseteq B$.

Table 3.1 Commonly Used Set Identities

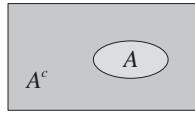
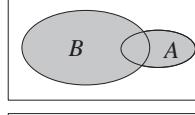
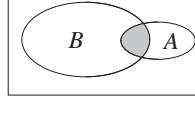
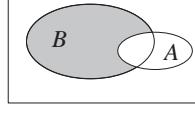
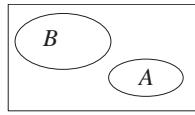
<i>Identity Name</i>	<i>Definition</i>
<i>Commutativity</i>	$A \cup B = B \cup A$ and $A \cap B = B \cap A$
<i>Associativity</i>	$A \cup (B \cup C) = (A \cup B) \cup C$ and $A \cap (B \cap C) = (A \cap B) \cap C$
<i>Distributivity</i>	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ and $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
<i>DeMorgan's Law</i>	$(A \cup B)^c = A^c \cap B^c$ and $(A \cap B)^c = A^c \cup B^c$

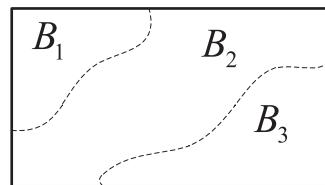
Consequently, we can carve up some set A using these partitions, such that:

$$\begin{aligned}
 A &= A \cap \Omega \\
 &= A \cap \left(\bigcup_{n=1}^N B_n \right) \\
 &= \bigcup_{n=1}^N (A \cap B_n)
 \end{aligned} \tag{3.5}$$

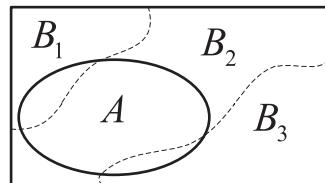
This partitioning of A is shown in Figure 3.2(b), where A is carved up into $A \cap B_1$, $A \cap B_2$, and $A \cap B_3$. As we will see later in this chapter, the ability to partition sets into several disjoint subsets will be very helpful in developing additional mathematical tools.

Table 3.2 Collection of Fundamental Set Operations

<i>Operation</i>	<i>Definition</i>	<i>Venn Diagram</i>
<i>Complement</i>	$A^c := \{\omega \in \Omega : \omega \notin A\}$	
<i>Empty set or null set</i>	$\emptyset =$	
<i>Union</i>	$A \cup B := \{\omega \in \Omega : \omega \in A \text{ or } \omega \in B\}$	
<i>Intersection</i>	$A \cap B := \{\omega \in \Omega : \omega \in A \text{ and } \omega \in B\}$	
<i>Set difference</i>	$B \setminus A := B \cap A^c$	
<i>Disjoint set or mutually exclusive</i>	$A \cap B = \emptyset$	



(a)



(b)

Figure 3.2 An illustration of the partitioning of a set A into several disjoint subsets based on the disjoint sets $\{B_n\}$. (a) Sample space partitioned into disjoint sets $\{B_n\}$. (b) Set A partitioned by disjoint sets $\{B_n\}$.

3.1.3 Functions

By definition, functions are used to transform or map various values into another quantitative representation. This mapping process can be readily represented in Figure 3.3, where the values of x located within the *domain* X are mapped by a function f to a collection of values y contained within a specific *range*, which forms part of a potentially larger co-domain Y .

Notice how the range contains all possible values of $f(x)$, where $\{f(x) : x \in X\}$. Furthermore, the range is a subset of the co-domain Y , and under specific circumstances the range and co-domain are one and in the same set of values. Whenever dealing with functions, there is a collection of associated terminology describing the mapping operations by functions such as f , such as:

- **Onto:** A function is “onto” if its range equals its co-domain such that for every $y \in Y$, $f(x) = y$ has a solution.
- **One-to-one:** The condition $f(x_1) = f(x_2)$ implies $x_1 = x_2$.
- **Invertible:** For every $y \in Y$, there is a unique $x \in X$ with $f(x) = y$, which means that the function is both onto and one-to-one, and that for every $y \in Y$ and $f(x) = y$ has a unique solution.

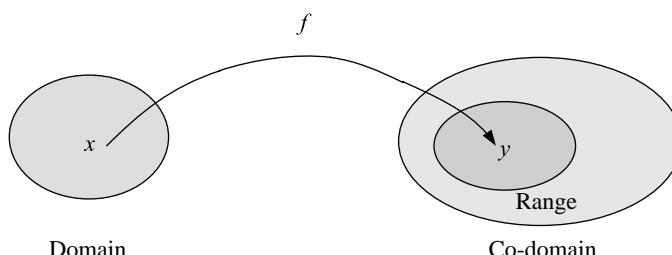


Figure 3.3 An illustration of how a function maps values from the domain to a range contained within a co-domain.

- *Inverse image:* If we have $f: X \rightarrow Y$ and if $B \subset Y$, then the inverse image of B is $f^{-1}(B) := \{x \in X : f(x) \in B\}$.

3.1.4 Axioms and Properties of Probability

Using the mathematical tools that we have developed thus far, it is now time to establish a fundamental set of rules for probability theory. Given a nonempty set Ω called the sample space, and a function P defined on the subset of Ω , we say P is a *probability measure* if the following four axioms are satisfied:

1. The empty set ϕ is called the *impossible event*. Moreover, $P(\phi) = 0$.
2. Probabilities are *nonnegative* (i.e., $P(A) \geq 0, \forall A$).
3. If A_1, A_2, \dots are events that are mutually exclusive or pairwise disjoint (i.e., $A_n \cap A_m = \phi$ for $n \neq m$), then $P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} P(A_n)$.
4. The *sure event* or *certain event* occurs when finding the probability measure of the entire sample space Ω (i.e., $P(\Omega) = 1$). Furthermore, the *almost-sure event* occurs when $P(A) = 1$ where $A \neq \Omega$.

Several useful properties based on several manipulations of the Axioms of Probability are presented in Table 3.3.



Using the Axioms of Probability, verify that the properties presented in Table 3.3 are indeed true.

3.1.5 Conditional Probability

We are often interested in determining whether two events, say A and B , are related in the sense that knowledge of one influences the occurrence of the other. Conse-

Table 3.3 Several Common Probability Properties

Property	Definition
Finite disjoint union	$P\left(\bigcup_{n=1}^N A_n\right) = \sum_{n=1}^N P(A_n)$
Probability of a complement	$P(A^c) = 1 - P(A)$
Monotonicity	$A \subset B \rightarrow P(A) \leq P(B)$
Inclusion-exclusion	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$
Limit properties	$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \lim_{N \rightarrow \infty} P\left(\bigcup_{n=1}^N A_n\right)$ $P\left(\bigcap_{n=1}^{\infty} A_n\right) = \lim_{N \rightarrow \infty} P\left(\bigcap_{n=1}^N A_n\right)$

quently, we employ the *conditional probability* mathematical tool in order to determine the relationship between two events, where this tool is defined as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \text{ for } P(B) > 0 \quad (3.6)$$

This definition can be interpreted using the concept of relative frequency that was discussed earlier in this chapter. Suppose that an experiment is performed n times, and that the event B occurs n_B times while the event A occurs n_A times and the event $A \cap B$ occurs $n_{A \cap B}$ times. Then, the ratio of the number of times that A and B occur given the number of times B occurs is defined as:

$$\frac{n_{A \cap B}}{n_B} = \frac{n_{A \cap B}/n}{n_B/n} \therefore \lim_{n \rightarrow \infty} \frac{n_{A \cap B}/n}{n_B/n} = \frac{P(A \cap B)}{P(B)} \quad (3.7)$$

In other words, the number of times that B occurs (i.e., n_B), will normalize the number of times that $A \cap B$ occurs. As we will see next, the concept of conditional probability will be very useful in developing some of the key concepts for probability theory.

3.1.6 Law of Total Probability and Bayes' Rule

Recall from Section 3.1.2 how we could partition a sample space into a collection of mutually exclusive events B_1, B_2, \dots, B_n . Let us proceed with partitioning the event A using this collection, such that:

$$\begin{aligned} A &= A \cap (B_1 \cup B_2 \cup B_3 \cup \dots \cup B_n), \\ &= (A \cap B_1) \cup (A \cap B_2) \cup \dots \cup (A \cap B_n) \end{aligned} \quad (3.8)$$

Now taking the definition for the conditional probability and isolating for $P(A \cap B_i)$, we get the following:

$$P(A \cap B_i) = P(A|B_i)P(B_i) \quad (3.9)$$

which becomes the definition for the *Law of Total Probability*, namely:

$$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \dots + P(A|B_n)P(B_n) \quad (3.10)$$

From this expression, suppose that B_1, B_2, \dots, B_n constitute partitioned components of the sample space and that we have defined an event A . We can now define the probability of event B_i using the following expression:

$$P(B_i|A) = \frac{P(A \cap B_i)}{P(A)} = \frac{P(A \cap B_i)}{\sum_{k=1}^n P(A|B_k)P(B_k)} \quad (3.11)$$

which is referred to as *Bayes' Rule*. With the Law of Total Probability and Bayes' Rule now defined, it is possible to solve complex probability models within the context of a set theory framework using these tools and their variants.

3.1.7 Independence

There are several occasions where a special relationship exists between two or more events. For instance, if knowledge of an occurrence of an event B does not alter the probability of some other event A , we can say that A is *independent* of B ; that is,

$$P(A) = P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (3.12)$$

By definition, two events A and B are considered to be independent when:

$$P(A \cap B) = P(A)P(B) \quad (3.13)$$

In general, when we have events A_i , $i = 1, 2, \dots$, we say that they are *mutually independent* when there are a finite number of events and satisfy:

$$P\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} P(A_i) \quad (3.14)$$

Note that if (3.14) holds for all subsets of I containing exactly two events but not necessarily three or more, we say that the A_i are *pairwise independent*.



Prove the definition for independence, where $P(A \cap B) = P(A)P(B)$.

3.2 RANDOM VARIABLES

In science and engineering, measurements and observations are expressed as numerical quantities. These tend to have a degree of uncertainty, and we refer to this uncertainty variability as a *random variable*. A random variable is a number assigned to every outcome of an experiment. This number could be the gain in a game of chance, the voltage of a random source, the cost of a random component, or any other numerical quantity that is of interest in the performance of the experiment.

A random variable is a mapping function whose domain is a sample space and whose range is some set of real numbers:

$$X = X(\omega) \quad (3.15)$$

where X is the random variable, and ω is the outcome of an experiment found in the sample space . An example of the mapping process relating some experiment outcome ω by the random variable X to some real number value is shown in Figure 3.4.

Note that the sample space is the domain of the function $X()$, while x is the range and the co-domain is \mathbb{R} .

Given this mapping function, let us now develop a notation for defining the probability of an event or an outcome. First, we will let upper case letters denote random variables, while lower case letters will denote the possible values of the

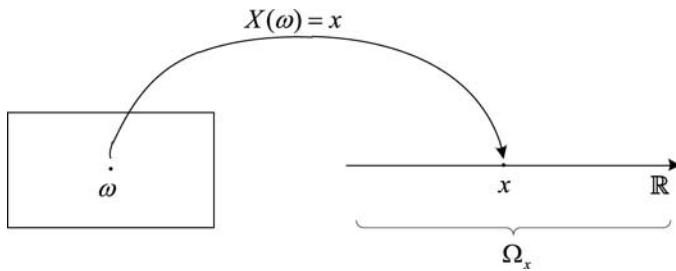


Figure 3.4 An example of a random variable X mapping an experimental outcome ω to a real number value.

random variables. Second, let us define the following expression for the probability of a specific event, as well as its shorthand representation on the right-hand side:

$$P(\{\omega \in \Omega : X(\omega) < x\}) \rightarrow P(X < x) \quad (3.16)$$

where $P(\cdot)$ denotes the probability measure, ω denotes the outcomes are elements of the sample space Ω , and $X(\omega)$ denotes the mapping function that produces an output value less than a deterministic quantity x . On the right-hand side, the shorthand version is interpreted as “the probability that the random variable X will produce an output that is less than the deterministic quantity x .”

Finally, some additional shorthand expressions and their equivalent full mathematical representations include the scenario where if B is a set of real numbers, then the probability that X produces an output belonging to that set is given by:

$$P(X \in B) = P(\{X \in B\}) = P(\{\omega \in \Omega : X(\omega) \in B\}) \quad (3.17)$$

Another expression is when B is an interval such that $B = (a, b]$, the probability that the output of the random variable X is given by:

$$P(a < X \leq b) = P(\{\omega \in \Omega : a < X(\omega) \leq b\}) \quad (3.18)$$

3.2.1 Discrete Random Variables

The first type of random variable we will be studying is discrete random variables. Suppose that X is a *discrete random variable*, in which case there exists distinct real numbers x_i such that:

$$\sum_i P(X = x_i) = 1 \quad (3.19)$$

Using the Law of Total Probability that we have studied in Section 3.1.6, we know that:

$$P(X \in B) = \sum_{i: x_i \in B} P(X = x_i) \quad (3.20)$$

where the set B is composed of a collection of values x_i . A specific form of discrete random variable is die *integer-valued random variable*, where its output values are the integers $x_i = i$; that is,

$$P(X \in B) = \sum_{i \in B} P(X = i) \quad (3.21)$$

Since each output of the random variable X possesses a different probability of occurrence, we usually define the probability of a specific discrete output x_i being generated by X as:

$$p_X(x_i) = P(X = x_i) \quad (3.22)$$

where $p_X(x_i)$ is referred to as the *probability mass function* (PMF). Note that the values of the PMF are constrained by:

$$0 \leq p_X(x_i) \leq 1 \text{ and } \sum_i p_X(x_i) = 1 \quad (3.23)$$

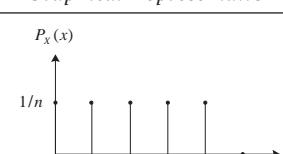
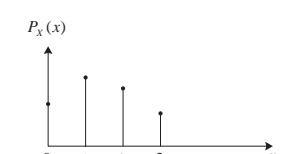
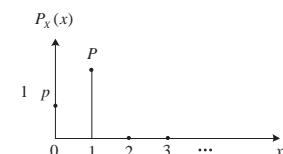
Several frequently used PMFs are specified in Table 3.4, including uniform, Poisson, and Bernoulli random variables.

When dealing with multiple discrete random variables X_1, \dots, X_n , the generalized definition for all of these random variables to be independent is given as:

$$p\left(\bigcap_{j=1}^n \{X_j \in B_j\}\right) = \prod_{j=1}^n P(X_j \in B_j), \forall \text{ choices of sets } B_1, \dots, B_n \quad (3.24)$$

Note that if X_1, \dots, X_n are independent, then so are any subset of them. Furthermore, if X_1, \dots, X_n, \dots is an infinite sequence of random variables, then they are independent if the generalized expression holds for every finite $n = 1, 2, \dots$. Finally, if for every B the probability $P(X_j \in B)$ does not depend on j , then we say that all the X_j are *identically distributed*. In the case when all the X_j are also independent with each other, we refer to these random variables as *independent and identically distributed* (i.i.d.).

Table 3.4 Several Frequently Used Probability Mass Functions (PMFs)

Random Variable	PMF Definition	Graphical Representation
Uniform	$p_X(k) = \frac{1}{n}, \quad k = 1, \dots, n$ 0, otherwise	
Poisson	$p_X(k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, \dots$	
Bernoulli	$p_X(k) = \begin{cases} p, & k = 1 \\ 1 - p, & k = 0 \\ 0, & \text{otherwise} \end{cases}$	

3.2.1.1 Expectation

One way to characterize a random variable is by its *expected value* or *mean*, which can be quantitatively determined using the expression:

$$m_X = E[X] = \sum_i x_i P(X = x_i) \quad (3.25)$$

where the sum of each value is weighted by its probability of occurrence. Consequently, using the definition of the PMF, we can rewrite (3.25) as:

$$E[X] = \sum_i x_i p_X(x_i) \quad (3.26)$$

Suppose we have a random variable X and a real-valued function $g(x)$ that maps it to a new random variable Z (i.e., $Z = g(X)$). Recall that X is actually the mapping of points from a sample space to a collection of real numbers. Therefore, we are performing a mapping of a mapping when solving for the random variable Z (i.e., $Z(\omega) = g(X(\omega))$). Thus, in order to compute the expectation of the random variable Z , namely $E[Z]$, we can employ the following expression:

$$E[Z] = E[g(X)] = \sum_i g(x_i) p_X(x_i) \quad (3.27)$$

which is referred to the *expectation of a function of a random variable*. One of the most important aspects of (3.27) is that the expectation of Z can be solved using the PMF for X rather than having to determine the PMF of Z . Table 3.5 presents several useful properties associated with the expectation operator.



Derive the Markov Inequality, $P(X - a) \geq \frac{E[X^r]}{a^r}$, using the definition for the expectation.

3.2.1.2 Conditional Probability and Conditional Expectation

For random variables, we define the *conditional probability* as:

$$\begin{aligned} P(X \in B | Y \in C) &= P(\{X \in B\} \cap \{Y \in C\}) \\ &= \frac{P(\{X \in B\} \cap \{Y \in C\})}{P(\{Y \in C\})} = \frac{P(X \in B, Y \in C)}{P(Y \in C)} \end{aligned} \quad (3.28)$$

Furthermore, we can define the conditional PMF as:

$$\begin{aligned} p_{X|Y}(x_i | y_j) &= P(X = x_i | Y = y_j) = \frac{P(X = x_i, Y = y_j)}{P(Y = y_j)} = \frac{p_{XY}(x_i, y_j)}{p_Y(y_j)} \\ p_{Y|X}(y_j | x_i) &= P(Y = y_j | X = x_i) = \frac{P(X = x_i, Y = y_j)}{P(X = x_i)} = \frac{p_{XY}(x_i, y_j)}{p_X(x_i)} \end{aligned} \quad (3.29)$$

Table 3.5 Several Useful Properties of Expectation

Name	Definition
Linearity	If a and b are deterministic constants and X and Y are random variables, then $E[aX + bY] = E[aX] + E[bY] = aE[X] + bE[Y]$.
Moment	The n th moment ($n \geq 1$) of a real-valued random variable X is defined as the expected value of X raised to the n th power (i.e., $\text{Moment}_n(X) = E[X^n]$).
Mean	The mean is the first moment of X (i.e., $\text{Moment}_1(X) = E[X] = \mu$).
Central Moment	The second moment of X with its mean subtracted is its variance (i.e., $\text{Moment}_2(X - \mu) = E[(X - \mu)^2] = \text{var}(X) = \sigma^2$).
Correlation	The correlation between two random variables X and Y is defined to be equal to $E[XY]$.
Covariance	The covariance between X and Y is defined as $\text{cov}(X, Y)$.
Correlation coefficient	The correlation coefficient of two random variables X and Y is given as $\rho_{XY} = E\left(\frac{X - \mu_X}{\sigma_X}\right)\left(\frac{Y - \mu_Y}{\sigma_Y}\right)$.
Markov inequality	If X is a nonnegative random variable, then for any $\alpha > 0$ we have $P(X \geq \alpha) \leq \frac{E[X]}{\alpha}$.
Chebyshev inequality	For any random variable Y and any $\alpha > 0$, we have $P(Y \geq \alpha) \leq \frac{E[Y^2]}{\alpha^2}$.
Cauchy-Schwarz inequality	The Cauchy-Schwarz Inequality states that $ E[XY] \leq \sqrt{E[X^2]E[Y^2]}$ which becomes an equality if and only if X and Y are linearly related.
Independence	The random variables are independent if and only if $E[h(X)k(Y)] = E[h(X)]E[k(Y)]$ for all functions $h(x)$ and $k(y)$.

which can then be expressed as:

$$p_{XY}(x_i, y_j) = p_{X|Y}(x_i|y_j)p_Y(y_j) = p_{Y|X}(y_j|x_i)p_X(x_i) \quad (3.30)$$

If Y is an arbitrary random variable, and we take $A = \{Y \in C\}$, where $C \subseteq \mathbb{R}$, then we define the *Law of Total Probability* as:

$$P(Y \in C) = \sum_i P(Y \in C | X = x_i)P(X = x_i) \quad (3.31)$$

If Y is a discrete random variable, taking on distinct values y_i and setting $C = \{y_i\}$ yields:

$$\begin{aligned} P(Y = y_j) &= \sum_i P(Y = y_j | X = x_i)P(X = x_i) \\ &= \sum_i p_{Y|X}(y_j|x_i)p_X(x_i) \end{aligned} \quad (3.32)$$

Often we encounter the situation where some random variable Z is a function of X and some other discrete random variable Y , say, $Z = g(X, Y)$. However, we are interested in determining $P(Z = z)$. We know that by the Law of Total Probability that:

$$\begin{aligned} P(Z = z) &= \sum_i P(Z = z | X = x_i)P(X = x_i) \\ &= \sum_i P(g(X, Y) = z | X = x_i)P(X = x_i) \end{aligned} \quad (3.33)$$

Looking at the resulting equation more closely, we notice that since we already have the entire expression conditioned on the fact that the random variable X produces a deterministic output x_i , namely, $X = x_i$, we can substitute the random variable X with the deterministic output x_i such that:

$$P(g(X, Y) = z | X = x_i) = P(g(x_i, Y) = z | X = x_i) \quad (3.34)$$

This operation is referred to as the *Substitution Law*, and it can be used to help solve expressions and operations involving multiple random variables.

Similarly, the *conditional expectation* is given as:

$$E[g(Y)|X = x_i] = \sum_j g(y_j)p_{Y|X}(y_j|x_i) \quad (3.35)$$

the *Substitution Law for Conditional Expectation* is given as:

$$E[g(X, Y)|X = x_i] = E[g(x_i, Y)|X = x_i] \quad (3.36)$$

and the *Law of Total Probability for Expectation* is defined as:

$$E[g(X, Y)] = \sum_i E[g(X, Y)|X = x_i] p_X(x_i) \quad (3.37)$$



Derive the resulting expressions for the Law of Total Probability for Expectation and the Substitution Law.

3.2.1.3 A Communications Example: Binary Channel Receiver Design

In digital communications and wireless data transmission, we can view the operations of receivers as attempting to “guess” what was sent for a specific transmitter. When guessing the value of a transmission whose values can be modeled as a random variable X , and once it has been observed at the receiver whose values can be modeled by a random variable Y , we need some sort of decision rule to figure out which value of X was originally transmitted. This is illustrated in Figure 3.5, where X can output values of either 0 or 1 while Y is observed at the receiver. Notice how the observation Y can be accidentally interpreted as a value other than the one transmitted (i.e., $X \neq Y$).

Although there exist several approaches for deciding on what values were transmitted based on the observation of the intercepted received signal, no decision rule possesses a smaller probability of error than the *maximum a posteriori* (MAP) rule, where having observed $Y = j$, the MAP rule says to decide on $X = 1$ if:

$$P(X = 1 | Y = j) \geq P(X = 0 | Y = j) \quad (3.38)$$

and to decide $X = 0$ otherwise. In other words, the MAP rule decides $X = 1$ if the posterior probability of $X = 1$ given the observation $Y = j$ is greater than the

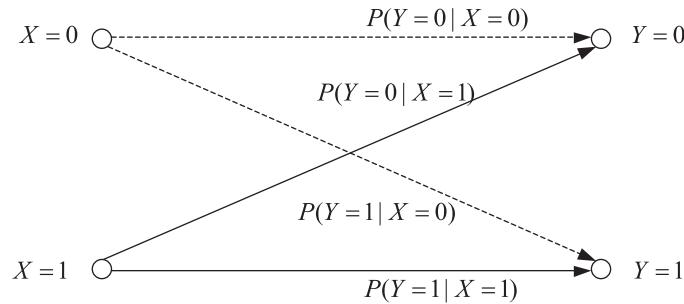


Figure 3.5 An example of a binary channel where transmission values generated by the random variable X are being observed at the receiver as the random variable Y .

posterior probability of $X = 0$ given the observation $Y = j$. Furthermore, we can observe that:

$$P(X = i|Y = j) = \frac{P(X = i, Y = j)}{P(Y = j)} = \frac{P(Y = j|X = i)P(X = i)}{P(Y = j)} \quad (3.39)$$

which we can then use to rewrite (3.38) as:

$$\begin{aligned} \frac{P(Y = j|X = 1)P(X = 1)}{P(Y = j)} &\geq \frac{P(Y = j|X = 0)P(X = 0)}{P(Y = j)} \\ P(Y = j|X = 1)P(X = 1) &\geq P(Y = j|X = 0)P(X = 0) \end{aligned} \quad (3.40)$$

If $X = 0$ and $X = 1$ are equally likely to occur, we can simplify this expression such that it yields the *maximum likelihood* (ML) rule:

$$P(Y = j|X = 1) \geq P(Y = j|X = 0) \quad (3.41)$$

Furthermore, in the general case, we can rearrange (3.40) such that it yields the *likelihood ratio*, namely:

$$\frac{P(Y = j|X = 1)}{P(Y = j|X = 0)} \geq \frac{P(X = 0)}{P(X = 1)} \quad (3.42)$$

where the right-hand side is referred to as the “threshold” since it does not depend on j .

3.2.2 Continuous Random Variables

So far we have looked at random variables that assume only a discrete set of values. We will now focus our attention on random variables that can produce a continuum of output values, which we will refer to as *continuous random variables*. One of the primary reasons continuous random variables are important to study is that they can readily model numerous random phenomena that occur in the real world.

However, before moving forward, it must be noted that some of the language used to describe discrete random variables will be slightly different when talking about continuous random variables. For example, instead of referring to the probability mass function of a random variable, we will call them *probability density functions* due to the fact that the possible outputs of a continuous random variable can assume any infinitesimal value rather than some discrete quantity.

3.2.2.1 Densities and Probabilities

As we have seen earlier in this chapter, it is possible to mathematically compute the probability of an event occurring when employing discrete random variables, namely:

$$P(a \leq X < b) = \sum_{i=a}^{b-1} p_X(i) \quad (3.43)$$

where X is the discrete random variable and both a and b are boundaries of a subset belonging to the sample space Ω . However, suppose now that X is a continuous random variable that can take on the entire continuum of values within the interval $[a, b]$. In order to compute the same probability in this case, we can start by realizing this scenario as the summation of an infinite number of points in $[a, b]$ with the space between samples equal to Δx .

Given an infinite number of samples, our Δx becomes so tiny that Δx ultimately converges to dx and our summation becomes an integral expression. Therefore, we can rewrite (3.43) in terms of continuous random variables, yielding:

$$P(a \leq X < b) = \int_a^b f(t)dt \quad (3.44)$$

where $f(t)$ is called the *probability density function* or PDF. Note that the PDF is the continuous version of the PMF that we discussed previously in this chapter. Moreover, generally, we can express the probability of a continuous random variable using the PDF by the expression:

$$P(X \in B) = \int_B f(t)dt = \int_{-\infty}^{+\infty} I_B(t)f(t)dt \quad (3.45)$$

where $I_B(t)$ is an *indicator function* that produces an output of unity whenever a value of t belongs to the set B and produces an output of zero otherwise. Note that $\int_B f(t)dt = 1$, $f(t)$ is nonnegative, $f(t)$ approximately provides the probability at a point t , and:

$$P(a \leq X \leq b) = P(a < X \leq b) = P(a \leq X < b) = P(a < X < b) \quad (3.46)$$

where the endpoints do not affect the resulting probability measure. Finally, a summary of several commonly used PDFs are presented in Table 3.6, including uniform, Gaussian, and exponential random variables.

3.2.2.2 Univariate Expectation

Similar to the expectation of a single discrete random variable, the expectation for a continuous random variable X with PDF $f(x)$ can be computed using the following expression:

$$E[g(X)] = \int_{-\infty}^{+\infty} g(x)f(x)dx \quad (3.47)$$

where $g(\cdot)$ is some real function that is applied to the random variable X .

Table 3.6 Several Frequently Used Probability Density Functions

Random Variable	PMF Definition	Graphical Representation
Uniform	$f(x) = \begin{cases} \frac{1}{(b-a)}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$	
Exponential	$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$	
Laplace	$f(x) = \frac{\lambda}{2} e^{-\lambda x }$	
Cauchy	$f(x) = \frac{\lambda/\pi}{\lambda^2 + x^2}, \lambda > 0$	
Gaussian	$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-0.5((x-\mu)/\sigma)^2}$	

3.2.2.3 Conditional Probability and Conditional Expectation

Many random variables of practical interest are not independent, where it is often the case that the outcome of one event may depend or be influenced by the result of another event. Consequently, it is sometimes necessary to compute the *conditional probability* and *conditional expectation*, especially in circumstances where we have to deal with problems involving more than one random variable.

Unlike the conditional probability for a discrete random variable, the conditional probability for a continuous random variable needs to be defined in an alternative manner since the probability of a single exact point occurring is zero (i.e., $P(X = x) = 0$). As a result, if we employ the definition for the conditional probability for a discrete random variable from (3.28), namely:

$$P(Y \in C | X = x) = \frac{P(Y \in C, X = x)}{P(X = x)} \quad (3.48)$$

we observe that the occurrence of $P(X = x) = 0$ would yield a “divide-by-zero” scenario. Consequently, we need to determine another definition for the conditional

probability (and conditional expectation) that would work within the continuous random variable framework.

It can be shown that in order to calculate the conditional probability, one must employ a *conditional density* [1], which can be defined as:

$$f_{Y|X}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)} \quad (3.49)$$

where $f_X(x) > 0$. Thus, leveraging the conditional density, we can now compute the conditional probability without concern for a “divide-by-zero” scenario by solving for the following expression:

$$P(Y \in C | X = x) = \int_C f_{Y|X}(y|x) dy \quad (3.50)$$

Furthermore, we can define the *Law of Total Probability* as the following:

$$P(Y \in C) = \int_{-\infty}^{+\infty} P(Y \in C | X = x) f_X(x) dx \quad (3.51)$$

where we weigh all the conditional probabilities by the PDF of X before integrating them together to form the overall probability of the event $Y \in C$. Finally, just as in the discrete random variable case, we can employ a form of *Substitution Law* for continuous random variables when dealing with conditional probability, which is defined by:

$$P((X, Y) \in A | X = x) = P((x, Y) \in A | X = x) \quad (3.52)$$

Note that if X and Y are independent, then the joint density factors, yielding the following expression for the conditional density:

$$\begin{aligned} f_{Y|X}(y|x) &= \frac{f_{X,Y}(x,y)}{f_X(x)} \\ &= \frac{f_X(x)f_Y(y)}{f_X(x)} \\ &= f_Y(y) \end{aligned} \quad (3.53)$$

which implies that when the two random variables are independent, we do not need to condition one event on the other.

Similarly, the *conditional expectation* when dealing with continuous random variables is defined as the following expression employing the conditional density, namely:

$$E[g(Y)|X=x] = \int_{-\infty}^{+\infty} g(y)f_{Y|X}(y|x)dy \quad (3.54)$$

Furthermore, the *Law of Total Probability* for a conditional expectation is given as:

$$E[g(X,Y)] = \int_{-\infty}^{+\infty} E[g(X,Y)|X=x]f_X(x)dx \quad (3.55)$$

and the *Substitution Law* for a conditional expectation is defined as:

$$E[g(X,Y)|X=x] = E[g(x,Y)|X=x] \quad (3.56)$$

3.2.3 Cumulative Distribution Functions

For both PDFs and PMFs, it is sometimes important to visualize the *cumulative distribution function* (CDF), especially since it provides another perspective on how the random variable behaves probabilistically. Furthermore, the CDF can sometimes be used to solve problems that would otherwise be difficult to access via some other definition.

Mathematically speaking, we can define the CDF by the following expression:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^x f(t)dt \quad (3.57)$$

which describes the probability that the outcome of an experiment described by the random variable X is less than or equal to the dummy variable x .

As an example, suppose that we want to calculate the probability of $P(a \leq X < b)$ using the PDF shown in Figure 3.6(a). One approach to quickly evaluating this probability is to leverage the *tail probabilities* of this distribution, namely, $P(X < a)$ (shown in Figure 3.6(b)) and $P(X < b)$ (shown in Figure 3.6(c)). Notice how the tail probabilities are actually the CDFs of X based on (3.57), where $F_X(a) = P(X < a)$ and $F_X(b) = P(X < b)$.

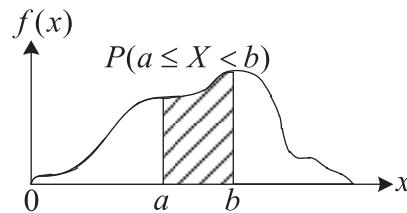
Consequently, given that we are only interested in the region of the PDF where these two tail probabilities do not intersect, we can compute the following probability:

$$P(a \leq X < b) = P(X < b) - P(X < a) = F_X(b) - F_X(a) \quad (3.58)$$

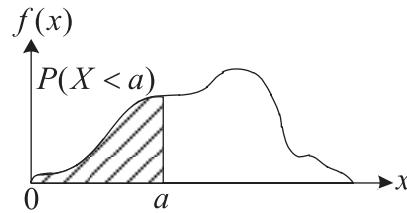
where all we really need are the values for the CDF of X at $x = a$ and $x = b$.

Several fundamental characteristics of the CDF include the fact that $F_X(x)$ is bounded between zero and one, and that $F_X(x)$ is a nondecreasing function (i.e., $F_X(x_1) \leq F_X(x_2)$ if $x_1 \leq x_2$). Furthermore, the PDF is the derivative of the CDF in terms of the dummy variable x , which we can define as:

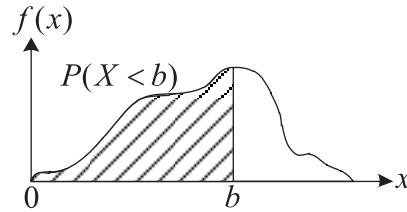
$$f_X(x) = \frac{d}{dx}F_X(x) \quad (3.59)$$



(a)



(b)



(c)

Figure 3.6 An example of how the CDF can be used to obtain the tail probabilities $P(X < a)$ and $P(X < b)$ in order to quickly calculate $P(a < X < b)$. (a) The region of the PDF of the random variable X that needs to be integrated in order to yield $P(a \leq X < b)$. (b) The region of the PDF of the random variable X that needs to be integrated in order to yield $P(X < a)$. (c) The region of the PDF of the random variable X that needs to be integrated in order to yield $P(X < b)$.

The *Q function* is a convenient way to express right-tail probabilities for Gaussian random variables, $P(X > x)$. Mathematically, this is equivalent to finding the complementary CDF of X , namely [2]:



$$Q(x) = 1 - F_X(x) = 1 - P(X \leq x)$$

$$= P(X > x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt$$

where $F_X(x)$ is the CDF of X .

3.2.4 Central Limit Theorem

There exists several instances where some sort of phenomenon is modeled as the summation of a collection of independently and identically distributed random variables.

In such cases, we can employ a useful theorem that is capable of helping us define the overall behavior of the summed random variables. Referred to as the *central limit theorem* (CLT), this theorem states the conditions under which the mean of a sufficiently large number of independent random variables, each with finite mean and variance, can be approximated by a Normal (i.e., Gaussian) distribution. Note that the CLT requires the random variables to be identically distributed. Since real-world quantities are often the balanced sum of many unobserved random events, this theorem provides a partial explanation for the prevalence of the normal probability distribution.

Suppose we have $X_1, X_2, X_3, \dots, X_n$ independent and identically distributed random variables with a common mean μ and common variance σ^2 . Let us define the sum of these random variables by the expression:

$$Y_n = \frac{1}{\sqrt{n}} \sum_{i=1}^n \left(\frac{X_i - \mu}{\sigma} \right) \quad (3.60)$$

where the random variable X_i is normalized by the standard deviation σ and its mean shifted to zero by μ . Consequently, if we take the limit of $n \rightarrow \infty$, then by the CLT the CDF of Y_n becomes:

$$\lim_{n \rightarrow \infty} F_{Y_n}(y) = \frac{1}{\sqrt{2\pi}} \int_y^{-\infty} e^{-t^2/2} dt \quad (3.61)$$

where the integrand is the definition for the standard normal PDF. The reason why this result for the CLT is so important is that if we take a collection of independent and identically distributed random variables possessing any distribution and sum them all together, we will obtain a resulting distribution that is Gaussian.



Given that a *binomial* random variable is formed by summing n independently and identically distributed *Bernoulli* random variables, show for a “large” value of n that the binomial PDF begins to appear similar to a Gaussian PDF by using die hist command in MATLAB. Explain this result using the CLT.

3.2.5 The Bivariate Normal

As mentioned previously, there are several random phenomena occurring frequently in the real world that can be characterized by a Gaussian (i.e., Normal) distribution. For example, a univariate Gaussian distribution possesses a PDF that is defined by:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad (3.62)$$

where μ is the mean of the Gaussian random variable X , and σ^2 is the variance of X . In the case that $\mu = 0$ and $\sigma^2 = 1$, we refer to X as a *standard Normal* random variable.

Although the univariate Normal distribution occurs rather frequently in the real world, there are several instances where a *bivariate Normal distribution* could also be defined to characterize a specific distribution involving two Gaussian

random variables possessing some degree of correlation between each other. An illustration of an example of a bivariate Normal distribution is shown in Figure 3.7. To visualize the impact of correlation between two Gaussian random variables forming a bivariate Normal distribution, let us examine the examples shown in Figure 3.8 and Figure 3.9, where the zero-mean Gaussian random variables X and Y produce 10,000 coordinate points in two-dimensional space. Since these random variables are zero-mean, the density of the resulting scatter plot is mostly concentrated around the origin at coordinates $(0, 0)$. We can see in Figure 3.8 that since X and Y are uncorrelated, the resulting density of coordinate outputs are *circularly symmetrical* around the means of the two random variables. On the other hand, if X and Y are correlated, the resulting density of output coordinates are heavily skewed along an axis within the two-dimensional plane, as shown in Figure 3.9.

Mathematically speaking, the general definition for a bivariate Gaussian density with parameters $\mu_X, \mu_Y, \sigma_X^2, \sigma_Y^2$, and correlation coefficient ρ is given by:

$$f_{XY}(x, y) = \frac{\exp \frac{1}{2(1-\rho^2)} \left(\frac{x-\mu_X}{\sigma_X} \right)^2 - 2\rho \left(\frac{x-\mu_X}{\sigma_X} \right) \left(\frac{y-\mu_Y}{\sigma_Y} \right) + \left(\frac{y-\mu_Y}{\sigma_Y} \right)^2}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \quad (3.63)$$

where the *correlation coefficient* is defined as:

$$\rho = E \frac{x-\mu_X}{\sigma_X} \div \frac{y-\mu_Y}{\sigma_Y} \quad (3.64)$$

3.3 RANDOM PROCESSES

As we have seen thus far, a random variable is simply a rule for assigning to every possible outcome ω from a sample space a number $X(\omega)$. Given this description,

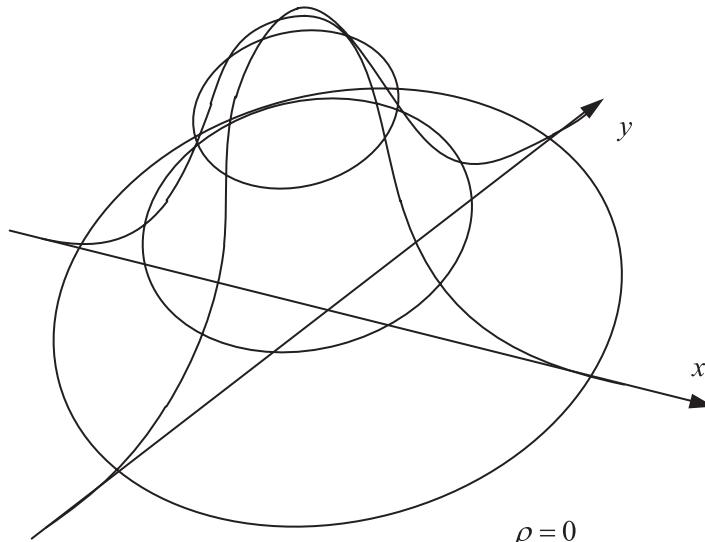


Figure 3.7 Density of a bivariate Normal distribution with no correlation (i.e., $\rho = 0$).

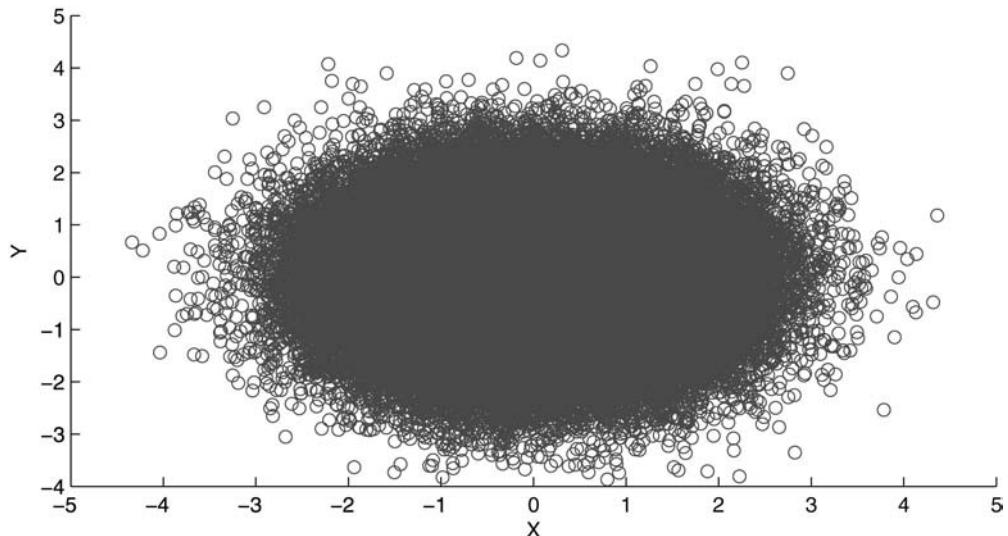


Figure 3.8 Scatter plot of a bivariate Normal distribution with no correlation (i.e., $\rho = 0$).

we will now provide a definition for a *random process*, which is also referred to as a *stochastic process*. A random process is a family of time-domain functions depending on the parameters t and ω (i.e., we can define a random process by the function).

$$X(t) = X(t, \omega) \quad (3.65)$$

where the left-hand side is the shortened representation of a random process that implicitly assumes the existence of a mapping of an outcome ω from the sample space to a real-valued number. Note that the domain of ω is while the domain of t is either \mathbb{R} for continuous-time random processes or \mathbb{Z} for discrete-time random processes.

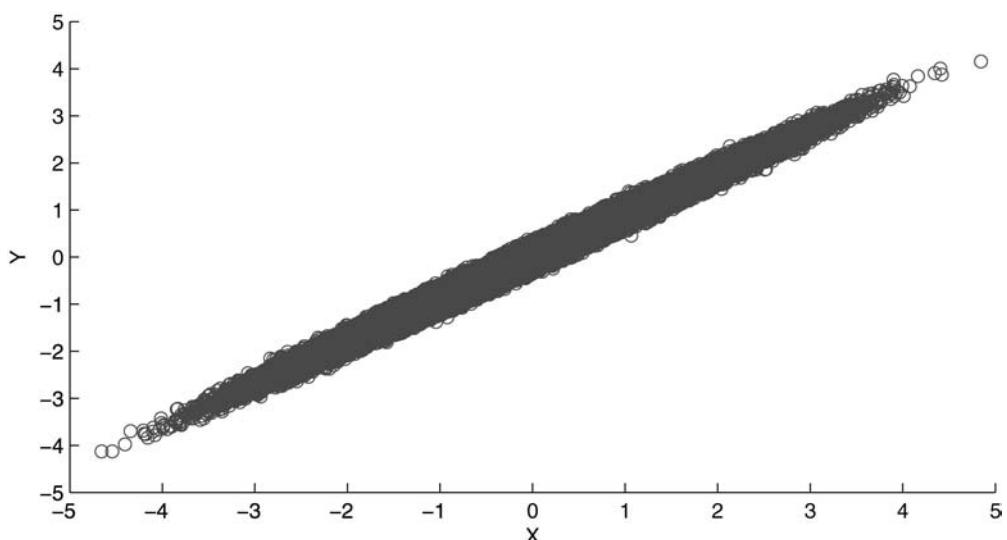


Figure 3.9 Scatter plot of a bivariate Normal distribution with a substantial amount of correlation between Gaussian random variables X and Y .

Given this additional degree of freedom, namely, the time parameter t , we should ask ourselves how we should interpret the behavior of a random process. Referring to the random process illustration shown in Figure 3.10, we see that a random process $X(t, \omega)$ consists of a collection of deterministic time-domain functions, each of which corresponds to an outcome ω_i from the sample space Ω . The “randomness” associated with the random process is derived from the fact that at every time instant, the real-valued output of the random process is generated by the completely random selection of one of the time-domain functions at that specific time instant. Furthermore, notice how the selection of the same deterministic time-domain function corresponding to the same outcome at two different time instants, say t_0 and t_1 , could yield a real-valued output that is different. In general, we can interpret the random process $X(t, \omega)$ as follows:

1. A *family* or *ensemble* of functions $X(t, \omega)$ when t and ω are variables;
2. A *single time function* when t is a variable and ω is fixed;
3. A *random variable* when t is fixed and ω is a variable;
4. A *number* when t and ω are fixed.

3.3.1 Statistical Characteristics of Random Processes

Suppose we have a random process that is noncountable infinite for each time instant t . Given this situation, we can define its first-order distribution $F(x, t)$ and first-order density $f(x, t)$ as:

$$F(x, t) = P(X(t) \leq x) \quad (3.66)$$

and

$$f(x, t) = \frac{\partial F(x, t)}{\partial x} \quad (3.67)$$

When studying a random process at two time instants, say, t_1 and t_2 , this is similar to studying two random variables, namely, $X(t_1, \omega)$ and $X(t_2, \omega)$. Therefore, the

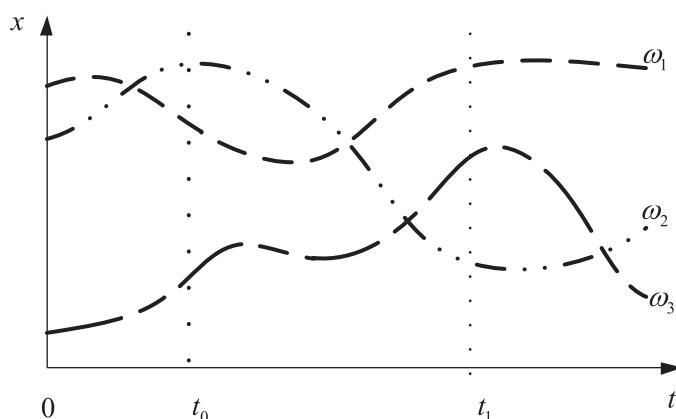


Figure 3.10 An illustration of a random process $X(t, \omega)$.

second-order distribution $F(x_1, x_2; t_1, t_2)$ and second-order density $f(x_1, x_2; t_1, t_2)$ are defined as:

$$F(x_1, x_2; t_1, t_2) = P(X(t_1) \leq x_1, X(t_2) \leq x_2) \quad (3.68)$$

and

$$f(x_1, x_2; t_1, t_2) = \frac{\partial^2 F(x_1, x_2; t_1, t_2)}{\partial x_1 \partial x_2} \quad (3.69)$$

Finally, if we extend these definitions to the n th-order distribution $F(x_1, \dots, x_n; t_1, \dots, t_n)$ and n th-order density $f(x_1, \dots, x_n; t_1, \dots, t_n)$, we get:

$$F(x_1, \dots, x_n; t_1, \dots, t_n) = P(X(t_1) \leq x_1, \dots, X(t_n) \leq x_n) \quad (3.70)$$

and

$$f(x_1, \dots, x_n; t_1, \dots, t_n) = \frac{\partial^n F(x_1, \dots, x_n; t_1, \dots, t_n)}{\partial x_1 \dots \partial x_n} \quad (3.71)$$

For determining the statistical properties of a random process, knowledge from the function $F(x_1, \dots, x_n; t_1, \dots, t_n)$ is required. However, in most applications only certain averages are actually needed. For instance, one of the mostly commonly used statistical characterizations for a random process is the *mean function*, where the mean function $\mu_x(t)$ of a random process $X(t, \omega)$ is the expectation of the random process at a specific time instant t . This can be mathematically represented by:

$$\mu_X(t) = E[X(t, \omega)] \quad (3.72)$$

Another useful statistical characterization tool for a random process $X(t, \omega)$ is the *autocorrelation function* $R_{XX}(t_1, t_2)$, where we evaluate the amount of correlation that the random process $X(t, \omega)$ possesses at two different time instants t_1 and t_2 . We can define this mathematically by the expression:

$$\begin{aligned} R_{XX}(t_1, t_2) &= E[X(t_1, \omega)X^*(t_2, \omega)] \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_1 x_2^* f(x_1, x_2; t_1, t_2) dx_1 dx_2 \end{aligned} \quad (3.73)$$

Note that the value of the diagonal for $R_{XX}(t_1, t_2)$ is the average power of $X(t, \omega)$, namely:

$$E[|X(t, \omega)|^2] = R_{XX}(t, t) \quad (3.74)$$

Several other useful properties and observations about the autocorrelation function include:

1. Since $R_{XX}(t_1, t_2) = E[X(t_1, \omega)X^*(t_2, \omega)]$, then $R_{XX}(t_2, t_1) = E[X(t_2, \omega)X^*(t_1, \omega)] = R_{XX}^*(t_1, t_2)$.
2. We have $R_{XX}(t, t) = E[|X(t, \omega)|^2] \geq 0$.
3. A random process for which $E[|X(t, \omega)|^2] < \infty$ for all t is called a *second-order process*.

4. For $R_{XX}(t,t) = E[X(t,\omega)|^2] \geq 0$ and given time instants t_1 and t_2 , we have the following inequality:

$$|R_{XX}(t_1, t_2)| \leq \sqrt{E[|X(t_1, \omega)|^2]E[|X(t_2, \omega)|^2]}$$

5. A normalized process occurs when $X(t, \omega)/\sqrt{C_{XX}(t, t)}$.

An extension of the definition for the autocorrelation function is the *autocovariance function* $C_{XX}(t_1, t_2)$ of $X(t, \omega)$, which is the covariance of the random process $X(t, \omega)$ at time instants t_1 and t_2 . Mathematically, we can represent the autocovariance function by the expression:

$$C_{XX}(t_1, t_2) = R_{XX}(t_1, t_2) - \mu_X(t_1)\mu_X^*(t_2) \quad (3.75)$$

Note that for $t_1 = t_2$, the autocovariance function produces the variance of $X(t, \omega)$. Furthermore, we can sometimes represent the autocovariance function of a random process $X(t, \omega)$ using a normalize metric called the *correlation coefficient*, which we define as:

$$\rho_{XX}(t_1, t_2) = \frac{C_{XX}(t_1, t_2)}{\sqrt{C_{XX}(t_1, t_1)C_{XX}(t_2, t_2)}} \quad (3.76)$$

3.3.2 Stationarity

A *stationary process* is a random process that exhibits the same behavior at any two time instants (i.e., the random process is time invariant). Two common forms of stationary processes are *strict-sense stationary* (SSS) random processes and *wide-sense stationary* (WSS) random processes. In this section, we will define these two types of random processes.

3.3.2.1 Strict-Sense Stationarity

A random process is SSS whenever its statistical properties are invariant to a shift of the origin (i.e., the random process $X(t, \omega)$ and $X(t + c, \omega)$ both possess the same statistics for any time shift c). Therefore, the n th-order density of an SSS random process would be equal to, by definition, the following expression:

$$f(x_1, \dots, x_n; t_1, \dots, t_n) = f(x_1, \dots, x_n; t_1 + c, \dots, t_n + c) \quad (3.77)$$

for any time shift c .

It follows that $f(x; t) = f(x; t + c)$ for any time shift c , which means that the first-order density of $X(t, \omega)$ is independent of the time t , namely:

$$f(x; t) = f(x) \quad (3.78)$$

Furthermore, $f(x_1, x_2; t_1, t_2) = f(x_1, x_2; t_1 + c, t_2 + c)$ is independent of c for any value of c . Consequently, this means that the density becomes:

$$f(x_1, x_2; t_1, t_2) = f(x_1, x_2; \tau), \text{ where } \tau = t_1 - t_2 \quad (3.79)$$

Thus, the joint density of the random process at time instants t and $t + \tau$ is independent of t and is equal to $f(x_1, x_2; \tau)$.

3.3.2.2 Wide-Sense Stationarity

Although SSS random processes can yield mathematically tractable solutions based on their useful time-invariant property, the occurrence of SSS random processes in the real world is not very frequent. As a result, another form of stationarity that does occur rather frequently in the real world will now be discussed.

A random process $X(t, \omega)$ is considered to be WSS whenever both of the following properties are true:

- The mean function $\mu_X(t)$ does not depend on time t (i.e., $\mu_X(t) = E[X(t, \omega)] = \mu_X]$.
- The autocorrelation function $R_{XX}(t + \tau, t)$ depends only on the relative difference between t and $t + \tau$ (i.e., $R_{XX}(t + \tau, t) = R_{XX}(\tau)$).

Several observations about WSS random processes include the following:

- The average power of a WSS random process is independent of time since $E[|X(t, \omega)|^2] = R_{XX}(0)$.
- The autocovariance function of a WSS random process is equal to $C_{XX}(\tau) = R_{XX}(\tau) - |\mu_X|^2$.
- The correlation coefficient of a WSS random process is given by $\rho_{XX}(\tau) = C_{XX}(\tau)/C_{XX}(0)$.
- Two random processes $X(t, \omega)$ and $Y(t, \omega)$ are jointly WSS if each is WSS and their cross-correlation depends on $\tau = t_1 - t_2$.
- If the random process $X(t, \omega)$ is WSS and uncorrelated, then $C_{XX}(\tau) = q\delta(\tau)$, where q is some multiplicative constant.

3.3.2.3 Cyclostationarity

There exists another form of stationarity characteristic that often occurs in wireless data transmission. A *cyclostationary* random process $Y(t)$ is defined by a mean function $\mu_Y(t)$ that is periodic across time t as well as an autocorrelation function $R_{YY}(\tau + \theta, \theta)$ that is periodic across θ for a fixed value of τ . Consequently, a cyclostationary random process $Y(t)$ with period T_0 can be described mathematically by:

$$\bar{R}_{YY}(\tau) = \frac{1}{T_0} \int_0^{T_0} R_{XX}(\tau + \theta, \theta) d\theta \quad (3.80)$$

3.3.3 Gaussian Processes

In probability theory and statistics, a *Gaussian process* is a stochastic process whose realizations consist of random values associated with every point in a range of times (or of space) such that each such random variable has a Normal distribution. Moreover, every finite collection of those random variables has a multivariate Normal distribution.

Gaussian processes are important in statistical modeling because of properties inherited from the Normal distribution. For example, if a random process is modeled as a Gaussian process, the distributions of various derived quantities can be obtained explicitly. Such quantities include the average value of the process over a range of times, and the error in estimating the average using sample values at a small set of times.

Given the following expression:

$$y = \int_0^T g(t)X(t)dt \quad (3.81)$$

we can say that $X(t)$ is a Gaussian process if:

- $E(y^2)$ is finite (i.e., does not blow up).
- Y is Gaussian-distributed for every $g(t)$.

Note that the random variable Y has a Gaussian distribution, where its PDF is defined as:

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma_Y^2}} e^{-\frac{(y-\mu_Y)^2}{2\sigma_Y^2}} \quad (3.82)$$

where μ_Y is the mean and σ_Y^2 is the variance. Such processes are important because they closely match the behavior of numerous physical phenomena, such as *additive white Gaussian noise* (AWGN).



Why is an uncorrelated random process referred to as “white” such as in the case of additive *white* Gaussian noise?

3.3.4 Power Spectral Density and LTI Systems

To analyze a signal in the frequency domain, the power spectral density (PSD), $S_{XX}(f)$, is often used to characterize the signal, which is obtained by taking the Fourier transform of the autocorrelation $R_{XX}(\tau)$ of the WSS random process $X(t)$. The PSD and the autocorrelation of a function, $R_{XX}(\tau)$, are mathematically related by the *Einstein-Wiener-Khinchin* (EWK) relations, namely:

$$S_{XX}(f) = R_{XX}(\tau)e^{j2\pi f\tau}d\tau \quad (3.83)$$

$$R_{XX}(f) = S_{XX}(\tau)e^{+j2\pi f\tau} df \quad (3.84)$$

Using the EWK relations, we can derive several general properties of the PSD of a stationary process, namely:

- $S_{XX}(0) = \int R_{XX}(\tau)d\tau$
- $E\{X^2(t)\} = \int_{-\infty}^{\infty} S_{XX}(f)df$
- $S_{xx}(f) > 0$ for all f .
- $S_{xx}(-f) = S_x(f)$.
- The power spectral density, appropriately normalized, has the properties usually associated with a probability density function:

$$p_x(f) = \frac{S_{XX}(f)}{\int_{-\infty}^{\infty} S_{XX}(f)df} \quad (3.85)$$

A very powerful consequence of the EWK relations is its usefulness when attempting to determine the autocorrelation function or PSD of a WSS random process that is the output of a linear time-invariant (LTI) system whose input is also a WSS random process. Specifically, suppose we denote $H(f)$ as the frequency response of an LTI system $h(t)$. We can then relate the power spectral density of input and output random processes by the following equation:

$$S_{YY}(f) = |H(f)|^2 S_{XX}(f) \quad (3.86)$$

where $S_{XX}(f)$ is the PSD of input random process and $S_{YY}(f)$ is the PSD of output random process. This very useful relationship is illustrated in Figure 3.11.

3.4 CHAPTER SUMMARY

In this chapter, a brief introduction to some of the key mathematical tools for analyzing and modeling random variables and random processes has been presented. Of particular importance, the reader should understand how to mathematically manipulate Gaussian random variables, Gaussian random processes, and bivariate Normal distributions since they frequently occur in digital communications and wireless data transmission applications. Furthermore, understanding how stationarity works and how to apply the EWK relations to situations involving random processes being filtered by LTI systems is vitally important, especially when dealing with the processing and treatment of received wireless signals by the communication system receiver.

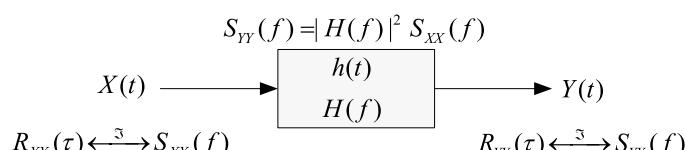


Figure 3.11 An example of how an LTI system $h(t)$ can transform the PSD between the WSS random process input $X(t)$ and the WSS random process output $Y(t)$.

3.5 ADDITIONAL READINGS

Although this chapter attempts to provide the reader with an introduction to some of the key mathematical tools needed to analyze and model the random variables and random processes that frequently occur in many digital communication systems, the treatment of these mathematical tools is by no means rigorous nor thorough. Consequently, the interested reader is encouraged to see some of the available textbooks that address this broad area in greater detail. For instance, the gold standard for any textbook on the subject of probability and random processes is by Papoulis and Pillai [1]. On the other hand, those individuals seeking to understand probability and random processes theory within the context of communication networks would potentially find the textbook by Leon-Garcia to be highly relevant [3].

For individuals who are interested in activities involving physical layer digital communication systems and digital signal processing, such as Wiener filtering, the textbook by Gubner would be a decent option given that many of the examples and problems included in this publication are related to many of the classic problems in communication systems [4]. Regarding textbooks that possess numerous solved examples and explanations, the books by Hsu [5] and Krishnan [6] would serve as suitable reading material. Finally, for those individuals who are interested in studying advanced topics and treatments of random processes, the textbook by Grimmett and Stirzaker would be a suitable publication [7].

3.6 PROBLEMS

1. [Probability of Event] A die is tossed twice and the number of dots facing up in each toss is counted and noted in the order of occurrence.
 - (a) Find the probabilities of the elementary events.
 - (b) Find the probabilities of events $A, B, C, A \cap B^c$, and $A \cap C$, where set A corresponds to the event “number of dots in first toss is not less than the number of dots in second toss,” set B corresponds to the event “number of dots in first toss is 6,” and set C corresponds to the event “number of dots in dice differs by 2.”
2. [Probability of Set] A number x is selected at random in the interval $[-3,2]$. Let the events, $A = \{x < 0\}$, $B = \{|x - 0.5| < 0.5\}$ and $C = \{x > 0.75\}$.
 - (a) Find the probabilities of $A, B, A \cap B$, and $A \cap C$.
 - (b) Find the probabilities of $A \cap B, A \cap C$, and $A \cap B \cap C$ first by directly evaluating the sets and then their probabilities, and second, by using appropriate axioms or corollaries.

HINT: Assume that the probability of any subinterval is proportional to its length.

3. [Probability of Event] An asymmetric communications channel shown in Figure 3.12 operates as follows. The input can take one of the two discrete values,

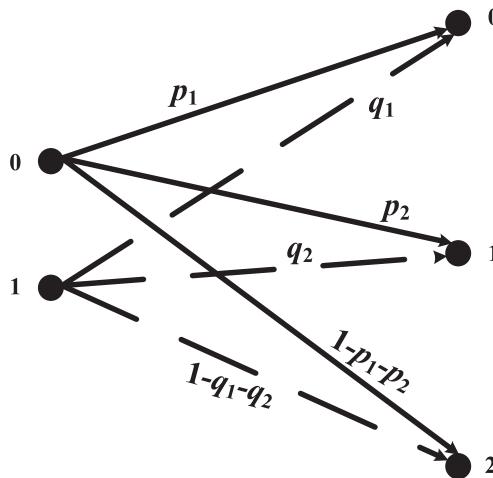


Figure 3.12 An asymmetric communications channel with input values (“0,” “1”) and output values (“0,” “1,” “2”).

namely a value of “0” with a probability α , and a value of “1” with a probability $1 - \alpha$. The output can take one of the three discrete values: “0,” “1,” and “2.” Use the notation,

$$T_i = \{i \text{ is transmitted}\}, i = 0, 1 \text{ with } P[T_0] = \alpha, P[T_1] = 1 - \alpha, \text{ and}$$

$$Rj = \{j \text{ is received}\}, j = 0, 1, 2.$$

The conditional probabilities for the occurrence of the output symbols are also shown in the figure. For example, $P[R_0|T_0] = p_1$ (read as: “probability that a ‘0’ is received *given* that a ‘0’ was transmitted equals p_1 ”).

- (a) Find the probability that the output is “0” (i.e., find $P[R_0]$).
- (b) Find the probability that the input was “0” *given* that the output is “0” (i.e., find $P[T_0|R_0]$).
- (c) Find the probability that the input was “1” *given* that the output is “2” (i.e., find $P[T_1|R_2]$).

4. [MATLAB TOOL] Referring to the asymmetric channel of Figure 3.12:

- (a) Generate a $1 \times 1,000,000$ vector containing “0” and “1” with probabilities α and $1 - \alpha$ (NOTE: Use in this problem $\alpha = 0.1, 0.5$, and 0.8). Show as a function of vector index, n , the accumulation of the probabilities of “0” and “1” as n increases from 1 to $1,000,000$. Do these values correspond to the theoretical (expected) outcome?
- (b) Take the vector generated in part (a) and feed it to the asymmetric channel of Figure 3.12. Plot the accumulated values of $P[R_0]$, $P[T_0|R_0]$, and $P[T_1|R_2]$ as n increases from 1 to $1,000,000$. Does the asymptotic behavior of these three scenarios correspond to the theoretical results in question 3?

One way of writing a MATLAB script is explained as follows. Notice that the following steps need to be implemented cumulatively as n increases from 1 to 1,000,000.

- Generate the transmit symbol vector using `randsrc()` function. The `randsrc()` function allows you to generate the input vector such that each entry of the alphabet occurs with the desired probability.
 - Select the indices associated with zeros (ones) in the transmit symbol vector using `find()` function.
 - Randomize the indices associated with zeros (ones) in the transmit symbol vector using `randperm()` function.
 - Select the first $100 \times p_1\%$ ($100 \times q_1\%$) elements from the vector containing randomized zero (one) locations and store them as zeros, select the next $100 \times p_2\%$ ($100 \times q_2\%$) elements from the same vector containing randomized zero (one) locations and store them as ones, and so on. (You may have to use `round()` function to select an integer number of zeros (ones).)
 - Count the appropriate values, and store $P[R_0]$ as a function of n , $P[T_0|R_0]$ as a function of n and $P[T_1|R_2]$ as a function of n .
 - Substitute the numerical values in your result from question 3. Verify that the plots of $P[R_0]$ versus n , $P[T_0|R_0]$ versus n and $P[T_1|R_2]$ versus n settle at these numerically obtained values as n increases.
5. [Probability Mass Function] Consider an information source that indefinitely produces binary pairs that we designate as $S_X = \{1, 2, 3, 4\}$. Find and plot the probability mass function in the following cases:
- $p_k = p_1/k$ for all k in S_X .
 - $p_k = p_{k-1}/2$ for $k = 2, 3, 4$.
 - $p_k = p_{k-1}/2^k$ for $k = 2, 3, 4$.
 - Can the random variables in parts (a), (b), and (c) be extended to take on values in the set $\{1, 2, 3, \dots\}$? If yes, specify the probability mass function of the resulting random variables. If no, explain why not.
6. [Bernoulli Random Numbers] Suppose you have just been hired by the Acme Randomness Company, Inc. and your first assignment is to deal with their RNG3000 random number generator, as shown in Figure 3.13. This random number generator operates as follows: Three Bernoulli random number generators are continuously producing sequences of “1” and “0” values, where the i th Bernoulli random generator has a probability of p_i of producing a “1” and a probability of $1 - p_i$ of producing a “0.” Although these three random number generators are simultaneously producing binary values, only one is actually being selected by a switch for a duration of N samples, and the switch randomly selects a random number generator after every N samples. Note that the value for N is selected at random from a range [1000, 10000] only once when the RNG3000 starts up. Finally, the output of the switch is then passed through a binary symmetric channel. The binary symmetric channel possesses a probability of p for the case when the input samples remains the same at the output of the channel, and a probability $1 - p$ when the sample value is *flipped* to the other binary

value. Note that the RNG3000 is capable of producing $L = 10\,000\,000$ random samples in a single operation.

Given that $p = 0.995$, $p_1 = 0.9$, $p_2 = 0.5$, $p_3 = 0.1$, and that we are not informed about the value of N generated at the start of the process, design a device that can be connected to the output of the binary symmetric channel capable of determining the approximate value of N based solely on the continuous stream of output samples from the binary symmetric channel, as well as determine which Bernoulli random number generator is being selected by the switch for each segment of N samples. Note that you will need to implement Figure 3.13 in the first place such that you can test your proposed design.

Several hints to get you in the right direction:

- Store in a single vector all the L samples produced at the output of the RNG3000 that you have implemented for a fixed value of N (note that N is generated only once via a uniform random number generator).
- To determine the size of N , apply a window of size N starting with $N = 1000$ and slide it across the vector of length L , recording the statistics of “0” and “1” values per window. Based on the statistics, and iteratively repeating this procedure for incrementally increasing window sizes, one can then determine based on the statistics the appropriate size of N .
- Once N has been determined, one can use the statistics per window in order to ascertain which one of the three Bernoulli random number generators were employed.

7. [Random Variable] A random variable X has the following probability density function (PDF):

$$f_X(x) = \begin{cases} c(1 - x^2) & 1 \leq x \leq 1 \\ 0 & \text{elsewhere} \end{cases} \quad (3.87)$$

- (a) Find c and plot the PDF.
(b) Find $P[X = 0]$, $P[0 < X < 0.5]$, and $P[|X - 0.5| < 0.25]$.

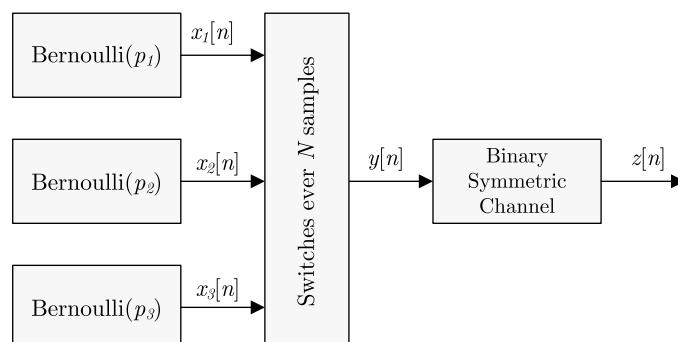


Figure 3.13 Schematic of the Acme Corporation’s RNG3000 random number generator.

8. [Zero-mean Additive Noise] In this problem, we need to extract a *secret message* for a noisy transmission possessing several unknown and random characteristics. The noisy transmission is saved in the file `secretmsg.mat`.

The objective of this problem is nullify the impact of the zero-mean additive noise and attempt to locate all the actual transmission samples that have been broadcast in bursts of various lengths rather than in a single continuous signal. The transmitter and the channel have the following properties:

- A secret message, contained within the intercepted signal, which consists of 26 lowercase letters of the alphabet. Each letter has been translated into a 5-digit binary word based its order in the alphabet (i.e., we have decimal representation “a”=“0,” “b” = “1,” “c” = “2,” . . . , “z”=“25” that is translated into binary words with the least significant bit located at the left-hand side; for example, “3” = “1 1 0 0 0.” Note that since it takes 5 bits to represent these 26 letters, and since there are $2^5 = 32$ possible binary words, the remaining $32 - 26 = 6$ binary words are equal to a blank space.
- Once the stream of binary digits have been formed from the secret message, all binary digits equal to “1” are replaced with a “+1” and all binary digits equal to “0” are replaced with a “−1,” thus forming a sample train of “+1” and “−1” values.
- Once this transmission stream of “+1” and “−1” values have been created, a *sample-and-hold* technique is employed per sample, repeating it $R = 1000$ times.
- The transmitter does not send out the output of the sample-and-hold operation continuously but rather in bursts, where the duration of each burst in terms of letters can be characterized by a discrete uniform random variable distributed across $[5, 20]$.
- The interburst duration in terms of the number of samples between the end of the last burst and the start of the successive burst is characterized by an exponential random variable of parameter λ .
- The noise added to each sample of the transmitted signal in the channel can be characterized by a zero-mean Gaussian random variable with variance σ^2 .
- Since noise is introduced into the channel, the original secret message prior to binary encoding is repeated $N = 3$ times such that it is possible to recover the message even in the presence of some corrupted bits.
- At the receiver, an *integrate-and-dump* approach can be used across each of the R samples in order to determine whether a “+1” or a “−1” has been transmitted. Note that this procedure simply adds up all the R samples corresponding to a specific binary digit and decides which value is more likely after normalizing by R .

Once your receiver has been constructed, please answer the following questions:

- (a) What is the approximate value of the parameter λ of the exponential random variable describing the interburst time?

- (b) What is the approximate value of the variance σ^2 of the Gaussian noise added to the transmitted message?
 (c) Concatenating all the transmission bursts together, what is the *secret message*?

9. [Random Variable] A random variable X has the following cumulative distribution function (CDF):

$$F_X(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 - \frac{1}{4}e^{-2x} & \text{for } x \geq 0 \end{cases} \quad (3.88)$$

- (a) Plot the CDF and identify the type of random variable.
 (b) Find $P[X = 2]$, $P[X = 0]$, $P[X < 0]$, $P[2 < X < 6]$, and $P[X > 10]$.
 (c) Find the mean and variance of X .

10. [Random Variable] A random variable X has the following probability density function (PDF):

$$f_X(x) = \begin{cases} c(1-x^2) & 1 \leq x \leq 1 \\ 0 & \text{elsewhere} \end{cases} \quad (3.89)$$

- (a) Find c and plot the PDF.
 (b) Plot the CDF of X .
 (c) Find $P[X = 0]$, $P[0 < X < 0.5]$, and $P[|X - 0.5| < 0.25]$.

11. [RLC Circuit] In this problem, suppose the Acme Corporation is focusing their efforts on the design of an RLC circuit similar to the one shown in Figure 3.14. The voltage transfer function between the source and the capacitor is:

$$H(\omega) = \frac{1}{(1-\omega^2 LC) + j\omega RC} \quad (3.90)$$

Suppose that the resonant frequency of the circuit, ω_0 , is the value of ω that maximizes $|H(\omega)|^2$.

- (a) Solve for the expression of the resonant frequency in terms of R , L , and C .
 (b) Given that these circuits are mass produced, then the actual values of R , L , and C in a particular device may vary somewhat from their design specifica-

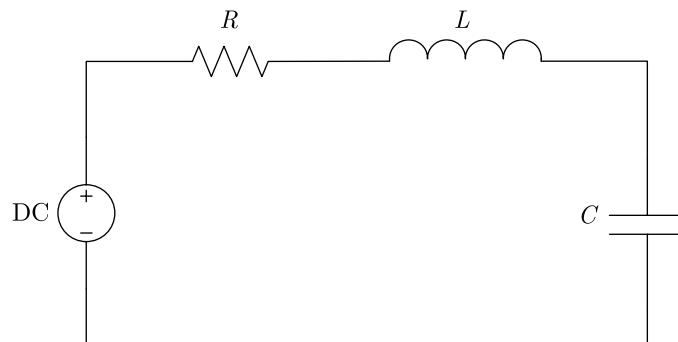


Figure 3.14 Schematic of the Acme Corporation's RLC-1975 circuit.

tions, and, hence, so does the resonant frequency. Assuming $L = 1$, $C = 1$, and $R \sim \text{uniform}[0, \sqrt{2}]$, plot the probability density function of the resonant frequency ω_0 by testing $N = 100,000,000$ components.

- (c) Assess the accuracy of your answer in part (b) by deriving and comparing with the theoretical probability density function for the resonant frequency ω_0 .
 - (d) Assuming $L = 1$, $C = 1$, and $R \sim \text{uniform}[0, \sqrt{2}]$, plot the probability density function of the resonant peak $|H(\omega_0)|^2$ by testing $N = 100,000,000$ components.
 - (e) Assess the accuracy of your answer in part (d) by deriving and comparing with the theoretical probability density function for the resonant peak $|H(\omega_0)|^2$.
12. [Bivariate Normal] In this problem, you have just been assigned another project by the Acme Randomness Company, with respect to the characterization of some test data that was recently collected during a measurement campaign. This test data is saved in the file `testdata.mat`.

An initial analysis by the company has shown that this data is a concatenation of five bivariate Normal vector pairs, (X, Y) , where the length for each pair of vectors is derived from a discrete uniform random variable that produces values between 100,000 and 1,000,000. Note that all of these bivariate Normal vectors are zero-mean and unit variance.

Furthermore, each of these vector pairs possess a different positive correlation factor ρ_{XY} between them, where ρ_{XY} is defined as:

$$\rho_{XY} = E \left[\frac{X - m_X}{\sigma_X} \right] \left[\frac{Y - m_Y}{\sigma_Y} \right]^T \quad (3.91)$$

where m_X and m_Y are the mean values of X and Y , while σ_X and σ_Y are the standard deviation values of X and Y .

Your task is to determine the approximate lengths of these five vectors and to find the approximate correlation factor ρ_{XY} for each pair of vectors.

Several helpful suggestions:

- Use the plotting command `scatter` in order to show the correlation between X and Y . Please include these scatter plots in your solutions and make sure to explain them thoroughly.
- Employ a sliding window approach in order to test the correlation of a specific segment. Changes in the correlation could be indicative of a transition from one vector to another.

13. [Autocorrelation] Prove the following two properties of the autocorrelation function $R_{xx}(\tau)$ of a random process $X(t)$:

- (a) If $X(t)$ contains a DC component equal to A , then $R_{xx}(\tau)$ will contain a constant component equal to A^2 .
- (b) If $X(t)$ contains a sinusoidal component, then $R_{xx}(\tau)$ will also contain a sinusoidal component of the same frequency.

14. [Cross-correlation] Consider a pair of stationary processes $X(t)$ and $Y(t)$. Show that the cross-correlations $R_{XY}(\tau)$ and $R_{YX}(\tau)$ of these processes have the following properties:

$$(a) R_{XY}(\tau) = R_{YX}(-\tau)$$

$$(b) |R_{XY}(\tau)| \leq \frac{1}{2} [R_X(0) + R_Y(0)]$$

15. [Advanced Topic] A sequence of L random variables $\{X_n, n \geq 1\}$ is said to be a *discrete martingale* with respect to another sequence $\{Y_n, n \geq 1\}$ if X_n is a function of Y_1, Y_2, \dots, Y_n and if:

$$E[X_{n+1} | Y_n, \dots, Y_1] = X_n, \quad n \geq 1 \quad (3.92)$$

Moreover, if $E[X_{n+1}|Y_n, \dots, Y_1] \geq X_n$ for $n \geq 1$, then X_n is said to be a *discrete submartingale*. On the other hand, if $E[X_{n+1}|Y_n, \dots, Y_1] \leq X_n$ for $n \geq 1$, then X_n is said to be a *discrete supermartingale*.

Suppose we let $\{Z_i\}$ be some sort of Bernoulli process defined as:

$$Z_n = \begin{cases} 1, & \text{for success in the } n\text{th trial} \\ -1, & \text{for failure in the } n\text{th trial} \end{cases} \quad (3.93)$$

and

$$\begin{aligned} P(Z_n = 1) &= p \\ P(Z_n = -1) &= 1 - p = q \end{aligned} \quad (3.94)$$

where $\mu_z = p - q$ and $\sigma_z^2 = 4pq$. Let us define the following random walk process:

$$W_n = \sum_{i=1}^n Z_i = \sum_{i=1}^{n-1} Z_i + Z_n = W_{n-1} + Z_n \quad (3.95)$$

The mean of W_n is $\mu_W = n(p - q)$, while its variance is $\sigma_W^2 = 4npq$.

Prove that $Y_n = W_n - n(p - q)$ is a discrete martingale with respect to $\{Z_k, 1 \leq k \leq n\}$.

16. [Advanced Topic] We define a random process $\{Y(t), t \geq 0\}$ as a *continuous martingale* with respect to the random process $\{X(s), 0 \leq s \leq t\}$ when:

$$\begin{aligned} E[Y(t)] &< \infty, \\ E[Y(t) | X(s), 0 \leq s \leq t] &= Y(s) \end{aligned} \quad (3.96)$$

Moreover, when $E[Y(t)|X(s), 0 \leq s \leq t] \leq Y(s)$, then $\{Y(t), t \geq 0\}$ is a *continuous submartingale* with respect to $\{X(s), 0 \leq s \leq t\}$. Finally, when $E[Y(t)|X(s), 0 \leq s \leq t] \geq Y(s)$, then $\{Y(t), t \geq 0\}$ is a *continuous supermartingale* with respect to $\{X(s), 0 \leq s \leq t\}$.

Suppose that a Wiener process $\{W(t), t \geq 0\}$ possesses a mean of zero and a variance equal to $\sigma^2 t$. Moreover, let us define the random process:

$$Y(t) = e^{kW(t) - k^2(\sigma^2 t/2)}, \quad t > 0 \quad (3.97)$$

where k is an arbitrary constant. Show that $Y(t)$ is a martingale with respect to $\{W(t), t \geq 0\}$.

17. [Advanced Topic] Suppose that we define a random process as $Y(t) = W^2(t)$, where $W(t)$ is a Wiener process with variance $\sigma^2 t$. Show that $Y(t)$ is a submartingale.
18. [Advanced Topic] Suppose that we have a Wiener process $X(t)$. Using the following arbitrary linear combination:

$$\sum_{i=1}^n a_i X(t_i) = a_1 X(t_1) + a_2 X(t_2) + \dots + a_n X(t_n) \quad (3.98)$$

where $0 < t_1 < \dots < t_n$ and a_i are real constants, show that it can also be a Gaussian process. [15 marks]

19. [Advanced Topic] Suppose we define the term *continuous in probability* for a random process $\{X(t), t \geq T\}$ when for every $\epsilon > 0$ and $t \geq T$:

$$\lim_{h \rightarrow 0} P(|X(t+h) - X(t)| > \epsilon) = 0 \quad (3.99)$$

Show that a Wiener process $X(t)$ is continuous in probability.

20. [Advanced Topic] Suppose we define $X(\omega)$ as the Fourier transform of a random process $X(t)$. If $X(\omega)$ is nothing more than white noise with zero mean and an autocorrelation function that is equal to $q(\omega_1)\delta(\omega_1 - \omega_2)$, then show that $X(t)$ is wide-sense stationary with a power spectral density equal to $q(\omega)/2\pi$.

References

- [1] Papoulis A., and S. Unnikrishna Pillai, *Probability, Random Variables and Stochastic Processes*, 4th Edition, McGraw-Hill Higher Education, 2002.
- [2] Abramowitz, M., and I. Stegun, *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*, Dover Publications, 1965.
- [3] Leon-Garcia, A., *Probability, Statistics, and Random Processes for Electrical Engineering*, 3rd Edition., Prentice Hall, 2008.
- [4] Gubner J. A., *Probability and Random Processes for Electrical and Computer Engineers*, Cambridge University Press, 2006.
- [5] Hsu H., *Schaum's Outline of Probability, Random Variables, and Random Processes*, McGraw-Hill, 1996.
- [6] Krishnan V., *Probability and Random Processes*, Wiley-Interscience, 2006.
- [7] Grimmett G., and Stirzaker D., *Probability and Random Processes*, 3rd Edition, Oxford University Press, 2001.

Digital Transmission Fundamentals

In this chapter, we will provide an overview of several key fundamental concepts employed in the transmission of digital data. Starting with an understanding of how binary data can be used to manipulate the physical characteristics of an electromagnetic waveforms, we will then look at the basic anatomy of a digital communication system before concentrating our attention on several different approaches for modulating electromagnetic waveforms using binary data. Once we have established how a digital data transmission process operates, we will then explore one of the key analytical tools for assessing the quantitative performance of such systems: the *bit error rate*. Finally, this chapter will conclude with an introduction to the design of digital receivers via a signal vector space perspective.

4.1 WHAT IS DIGITAL TRANSMISSION?

A digital transceiver is a system composed of a collection of both digital and analog processes that work in concert with each other in order to handle the treatment and manipulation of binary information. The purpose of these processes is to achieve data transmission and reception across some sort of medium, whether it is a twisted pair of copper wire, a fiber-optic cable, or a wireless environment. At the core of any digital transceiver system is the *binary digit* or *bit*, which for the purposes of this book is considered the fundamental unit of information used by a digital communication system.

Therefore, a digital transceiver is essentially responsible for the translation between a stream of digital data represented by bits and electromagnetic waveforms possessing physical characteristics that uniquely represent those bits. Since electromagnetic waveforms are usually described by sine waves and cosine waves, several physical characteristics of electromagnetic waveforms commonly used to represent digital data per time interval T include the amplitude, phase, and carrier frequency of the waveform, as shown in Figure 4.1. Notice how different combinations of bits represent different amplitude levels or different phase values or different carrier frequency values, where each value uniquely represents a particular binary pattern. Note that in some advanced mapping regimes, binary patterns can potentially be represented by two or more physical quantities.

However, there is much more going on in a digital transceiver than just a mapping between bits and waveforms, as shown in Figure 4.2. In this illustration of the basic anatomy of a digital transceiver, we observe that there are several functional blocks that constitute a communication system. For instance, the mapping between bits and electromagnetic waveform characteristics is represented by the *modulation* and *demodulation* blocks. Additionally, there are the *source encoding* and *source*

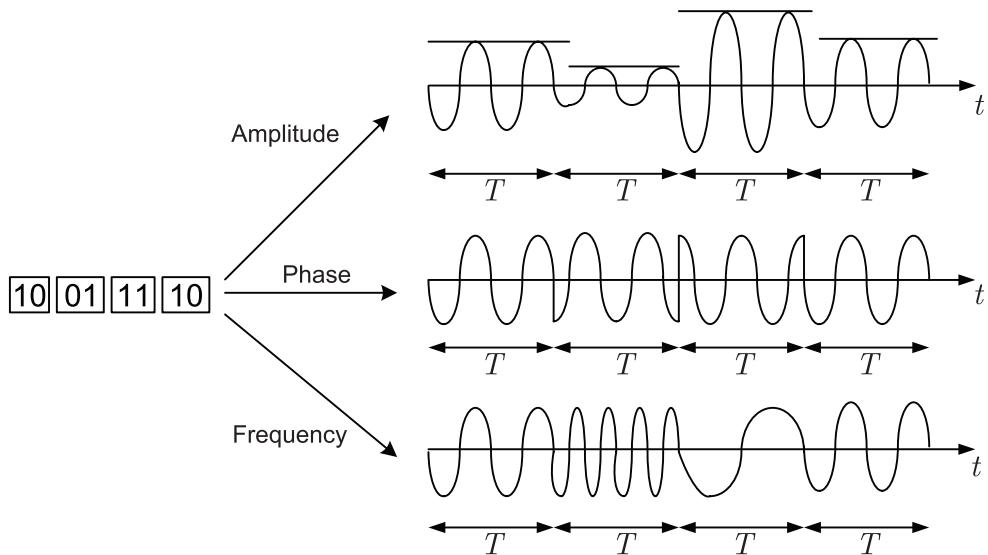


Figure 4.1 Possible mappings of binary information to EM wave properties.

decoding blocks that handle the removal of redundant information from the binary data, *channel encoding* and *channel decoding* blocks that introduce a controlled amount of redundant information to protect the transmission for potential errors, and the *radio frequency front-end* or *RF front-end* blocks that handle the conversion of baseband waveforms to higher carrier frequencies.

One may ask the question: *Why do we need all these blocks in our digital communication system?* Notice in Figure 4.2 the presence of a *channel* between the *transmitter* and the *receiver* of the digital transmission system. The main reason that the design of a digital communication system tends to be challenging, and that so many blocks are involved in its implementation, is due to this channel. If the channel was an ideal medium, where the electromagnetic waveforms from the transmitter are clearly sent to the receiver

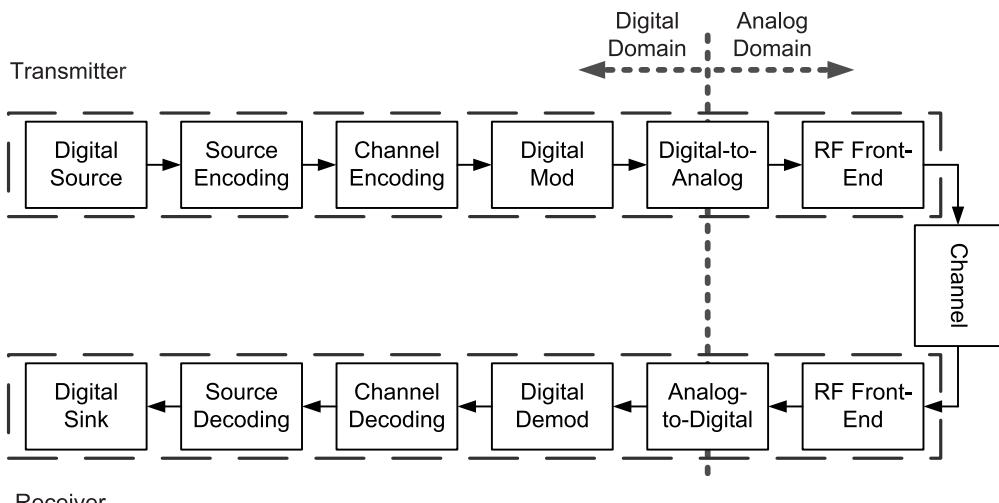


Figure 4.2 Generic representation of a digital communication transceiver.

without any sort of distortion or disturbances, then the design of digital communication systems would be trivial. However, in reality a channel introduces a variety of random impairments to a digital transmission that can potentially affect the correct reception of waveforms intercepted at the receiver. For instance, a channel may introduce some form of noise that can obfuscate some of the waveform characteristics. Furthermore, in many real-world scenarios many of these nonideal effects introduced by the channel are time-varying and thus difficult to deal with, especially if they vary rapidly in time.

Thus, under real-world conditions, the primary goal of any digital communication system is to transmit a binary message $\hat{m}(t)$ and have the reconstructed version of this binary message $\hat{m}(t)$ at the output of the receiver be equal to each other. In other words, our goal is to have $P(\hat{m}(t) \neq m(t))$ as small as needed for a particular application. The metric for quantitatively assessing the error performance of a digital communication system is referred to as the *probability of error or bit error rate* (BER), which we define as $Pe = P(\hat{m}(t) \neq m(t))$. Note that several data transmission applications possess different Pe requirements due in part to the data transmission rate. For instance, for digital voice transmission, a BER of $Pe \sim 10^{-3}$ is considered acceptable, while for an average data transmission application a BER of $Pe \sim 10^{-5} - 10^{-6}$ is deemed sufficient. On the other hand, for a very high data rate application such as those that employ fiber-optic cables, a BER of $Pe \sim 10^{-9}$ is needed since any more errors would flood a receiver given that the data rates can be extremely high.

To help mitigate errors that may occur due to the impairments introduced by the channel, we will briefly study how source encoding and channel encoding works before proceeding with an introduction to modulation.

4.1.1 Source Encoding

One of the goals of any communication system is to efficiently and reliably communicate information across a medium from a transmitter to a receiver. As a result, it would be ideal if all the redundant information from a transmission could be removed in order to minimize the amount of information that needs to be sent across the channel, which would ultimately result in a decrease in the amount of time, computational resources, and power being expended on the transmission. Consequently, *source encoding* is a mechanism designed to remove redundant information in order to facilitate more efficient communications.

The way source encoding operates is by taking a sequence of source symbols \underline{u} and mapping them to a corresponding sequence of source encoded symbols $\underline{v}, v_i \in \underline{v}$ as close to random as possible, and the components of \underline{v} are uncorrelated, (i.e., unrelated). Thus, by performing this source encoding operation we hope to achieve some level of redundancy minimization in $v_i \in \underline{v}$, thus limiting the amount of wasted radio resources employed in the transmission of otherwise “predictable symbols” in \underline{u} . In other words, a source encoder removes redundant information from the source symbols in order to realize efficient transmission. Note that in order to perform source encoding, the source symbols need to be *digital*.



A single analog television channel occupies 6 MHz of frequency bandwidth. On the other hand, up to eight digitally encoded television channels can fit within the same frequency bandwidth of 6 MHz.

4.1.2 Channel Encoding

To protect a digital transmission from the possibility of its information being corrupted, it is necessary to introduce some level of controlled redundancy in order to reverse the effects of data corruption. Consequently, *channel encoding* is designed to correct for channel transmission errors by introducing *controlled* redundancy into the data transmission. As opposed to the redundancy that is removed during the source encoding process, which is random in nature, the redundancy introduced by a channel encoding is specifically designed to combat the effects of bit errors in the transmission, (i.e., the redundancy possesses a specific structure known to both the transmitter and receiver).

In general, channel encoding operates as follows: Each vector of a source encoded output of length K , namely, v_l where $l = 1, 2, \dots, 2^K$, is assigned a unique codeword such that the vector $v_l = (101010 \dots)$ is assigned a unique codeword $c_l \in \mathbb{C}$ of length N , where \mathbb{C} is a *codebook*. During this process, the channel encoder has introduced $N - K = r$ controlled number of bits to the channel encoding process. The *code rate* of a communications system is equal to the ratio of the number of information bits to the size of the codeword, (i.e., the code rate is equal to k/N).

When designing a codebook for a channel encoder, it is necessary to quantitatively assess how well or how poorly the codewords will perform in a situation involving data corruption. Consequently, the *Hamming distance* is often used to determine the effectiveness of a set of codewords contained within a codebook by evaluating the relative difference between any two codewords. Specifically, the Hamming distance $d_H(c_i, c_j)$ between any two codewords, say, c_i and c_j , is equal to the number of components in which c_i and c_j are different. When determining the effectiveness of a codebook design, we often are looking for the minimum Hamming distances between codewords, that is,

$$d_{H,\min} = \min_{c_i, c_j \in \mathbb{C}, i \neq j} d_H(c_i, c_j) \quad (4.1)$$

since our goal is to maximize the minimum Hamming distance in a given codebook to ensure that the probability of accidentally choosing a codeword other than the correct codeword is kept to a minimum. For example, suppose we have a codebook consisting of $\{101, 010\}$. We can readily calculate the minimum Hamming distance to be equal to $d_{H,\min} = 3$, which is the best possible result. On the other hand, a codebook consisting of $\{111, 101\}$ possesses a minimum Hamming distance of $d_{H,\min} = 1$, which is relatively poor in comparison to the previous codebook example.

In the event that a codeword is corrupted during transmission, *decoding spheres* (also known as *Hamming spheres*) can be employed in order to make decisions on the received information, as shown in Figure 4.3, where codewords that are cor-

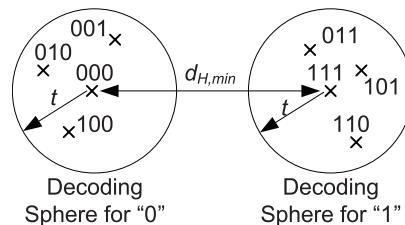


Figure 4.3 An example of decoding spheres.

rupted during transmission are mapped to the nearest eligible codeword. Note that when designing a codebook, the decoding spheres should not overlap in order to enable the best possible decoding performance at the receiver, (i.e., $\rightarrow d_{H,\min} = 2t + 1$).

4.1.2.1 Shannon's Channel Coding Theorem

In digital communications, it is sometimes necessary to determine the upper limit of the data rate for a specific digital transceiver design. Consequently, in 1949 Claude Shannon published his seminar paper that addressed this topic, “Communication in the Presence of Noise” [1]. In this paper, he defined

A rate 1/3 *repetition code* with no source encoding would look like:

$$1 \rightarrow 111 = c_1 \text{ (1st codeword)}$$



$$0 \rightarrow 000 = c_2 \text{ (2nd codeword)}$$

$$\therefore C = \{000, 111\}$$

What are the Hamming distances for the codeword pairs $d_H(111,000)$ and $d_H(111,101)$?

a quantitative expression that described the limit on the data rate, or *capacity*, of a digital transceiver in order to achieve error-free transmission.

Suppose one considers a channel with capacity C and we transmit data at a fixed code rate of K/N , which is equal to R_c (a constant). Consequently, if we increase N , then we must increase K in order to keep R_c equal to a constant. What Shannon states is that there exists a code such that for $R_c = K/N < C$ and as $N \rightarrow \infty$, we have the probability of error $P_e \rightarrow 0$. Conversely, for $R_c = K/N \geq C$, Shannon indicated that no such code exists. Hence, C is the limit in rate for reliable communications, (i.e., C is the *absolute limit* that you cannot go any faster than this amount without causing errors).

So why is the result so important? First, the concept of *reliability* in digital communications is usually expressed as the probability of bit error, which is measured at the output of the receiver. As a result, it would be convenient to know what this capacity is given the transmission bandwidth, B , the received signal-to-noise ratio (SNR) using mathematical tools rather than empirical measurements. Thus, Shannon derived the *information capacity of the channel*, which turned out to be equal to:

$$C = B \log_2(1 + \text{SNR}) \quad [\text{b/s}] \quad (4.2)$$

where this information capacity tells us the achievable data rate. Note that Shannon provided us with only the theoretical limit for the achievable capacity of a data transmission, but he does not tell us how to build a transceiver to achieve this limit.

Second, the information capacity of the channel is useful since this expression provides us with a bound on the achievable data rate given bandwidth B and received SNR, employed in the ratio $\eta = R/C$, where R is the signaling rate and C is

the channel capacity. Thus, as $\eta \rightarrow 1$, the system becomes more efficient. Therefore, the capacity expression provides us with a basis for tradeoff analysis between B and SNR, and it can be used for comparing the noise performance of one modulated scheme versus another.

4.2 DIGITAL MODULATION

In analog modulation schemes, the analog message signal modulates a continuous wave prior to transmission across a medium. Conversely, digital modulation involves having a digital message signal modulating a continuous waveform. As we have seen earlier in this chapter, this can be accomplished by uniquely manipulating the amplitude and phase information of a signal during each symbol period T based on a specific pattern of bits. However, most digital modulation techniques possess an intermediary step that we will focus on in this section, where collections of b bits forming a binary message m_b are mapped to a *symbol*, which is then used to define the physical characteristics of a continuous waveform in terms of amplitude and phase. In particular, for each of the possible 2^b values of m_b , we need a unique signal $s_i(t), 1 \leq i \leq 2^b$ that can then be used to modulate the continuous waveform, as shown in Figure 4.4.

In this section, we will study several different families of approaches for mapping binary data into symbols that can then be used to modulate continuous waveforms. These modulation scheme families are defined by which physical characteristic or combination of characteristics are manipulated by the mapping process in order to uniquely represent a specific bit pattern. However, there exist various tradeoffs between these different families, including how efficiently a bit is mapped to a symbol in terms of the transmit power expended. Consequently, we will first explore how we assess this tradeoff before studying three major families of digital modulation schemes and how they compare to each other.

4.2.1 Power Efficiency

In order to assess the effectiveness of mapping a bit to a symbol in terms of the transmit power expended per symbol, we can employ the *power efficiency* metric. Suppose we define the energy of a symbol $s(t)$ as:

$$E_s = \int_0^T s^2(t) dt \quad (4.3)$$

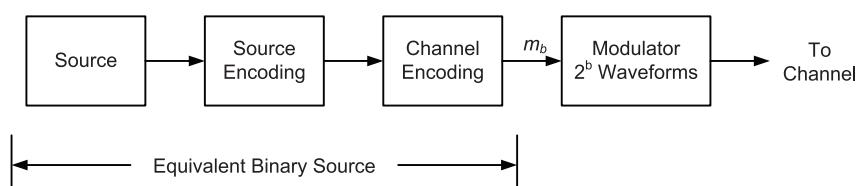


Figure 4.4 Modulation process of equivalent binary data.

where T is the period of the symbol. Then, for a modulation scheme consisting of M symbols, we can define the average symbol energy via the following weighted average:

$$\bar{E}_s = P(s_1(t)) \cdot \int_0^T s_1^2(t) dt + \dots + P(s_M(t)) \cdot \int_0^T s_M^2(t) dt \quad (4.4)$$

where $P(s_i(t))$ is the probability that the symbol $s_i(t)$ occurs. Furthermore, if we would like to calculate the average energy per bit, we can approximate this using E_s and dividing this quantity by $b = \log_2(M)$ bits per symbol, yielding:

$$\bar{E}_b = \frac{\bar{E}_s}{b} = \frac{\bar{E}_s}{\log_2(M)} \quad (4.5)$$

To quantitatively assess the similarity between two symbols in terms of their physical characteristics, we define the *Euclidean distance* as:

$$d_{ij}^2 = \int_0^T (s_i(t) - s_j(t))^2 dt = E_{\Delta s_{ij}} \quad (4.6)$$

where $\Delta s_{ij}(t) = s_i(t) - s_j(t)$. Since we are often interested in the worst-case scenario when assessing the performance of a modulation scheme, we usually compute the minimum Euclidean distance, namely:

$$d_{min}^2 = \min_{s_i(t), s_j(t), i \neq j} \int_0^T (s_i(t) - s_j(t))^2 dt \quad (4.7)$$

Thus, the *power efficiency* of a signal set used for modulation is given by the expression:

$$\epsilon_p = \frac{d_{min}^2}{\bar{E}_b} \quad (4.8)$$

4.2.2 Pulse Amplitude Modulation

Of the various physical characteristics of a signal waveform that can be manipulated in order to convey digital information, the most obvious choice is the signal amplitude level. Leveraging this physical characteristic, *pulse amplitude*

Suppose we would like to find the ϵ_p given the following waveforms:

$$s_1(t) = A \times [u(t) - u(t - T)] = s(t)$$

$$s_2(t) = -A \times [u(t) - u(t - T)] = -s(t)$$



where $u(t)$ is the unit step function. Compute the following:

- The minimum Euclidean distance d_{min}^2 .
- The average bit energy \bar{E}_b .
- The power efficiency ϵ_p .

modulation (PAM) is a digital modulation scheme where the message information is encoded in the amplitude of a series of signal pulses. Furthermore, demodulation of a PAM transmission is performed by detecting the amplitude level of the carrier at every symbol period.

The most basic form of PAM is binary PAM (B-PAM), where the individual binary digits are mapped to a waveform $s(t)$ possessing two amplitude levels according to the following modulation rule:

- “1” $\rightarrow s_1(t)$;
- “0” $\rightarrow s_2(t)$.

where $s_1(t)$ is the waveform $s(t)$ possessing one unique amplitude level, while $s_2(t)$ is also based on the waveform $s(t)$ but possesses another unique amplitude level. Note that the waveform $s(t)$ is defined across a time period T and is zero otherwise. Since the duration of the symbols is equivalent to the duration of the bits, the *bit rate* for a B-PAM transmission is defined as $R_b = 1/T$ bits per second.

The energy of the waveform $s(t)$ is defined as:

$$E_s = \int_0^T s^2(t) dt \quad [\text{Joules}] \quad (4.9)$$

Suppose we define $s(t)$ as a rectangular waveform, namely:

$$s(t) = A \cdot [u(t) - u(t - T)] \quad (4.10)$$

where $u(t)$ is the unit step function and A is the signal amplitude. Furthermore, suppose that the bit “1” is defined by the amplitude A while the bit “0” is defined by the amplitude $-A$. We can subsequently write our modulation rule to be equal to:

- “1” $\rightarrow s(t)$;
- “0” $\rightarrow -s(t)$.

Therefore, the symbol energy is given by $E_s = E_s = A^2 T = \frac{A^2}{R_b}$. From this result, we can define the energy per bit for a B-PAM transmission as:

$$\bar{E}_b = P(1) \times \int_0^T s_1^2(t) dt + P(0) \times \int_0^T s_2^2(t) dt \quad (4.11)$$

where $P(1)$ is the probability that the bit is a “1,” and $P(0)$ is the probability that the bit is a “0.” Thus, if we define $s_1(t) = s(t)$ and $s_2(t) = -s(t)$, then the average energy per bit is equal to:

$$\bar{E}_b = E_s \{P(1) + P(0)\} = E_s = \int_0^T s^2(t) dt = A^2 T \quad (4.12)$$

Calculating the minimum Euclidean distance, we get:

$$d_{\min}^2 = \int_0^T (s(t) - (-s(t)))^2 dt = \int_0^T (2s(t))^2 dt = 4A^2 T \quad (4.13)$$

which is then plugged into (4.8) in order to yield a power efficiency result for a B-PAM transmission of:

$$p = \frac{d_{\min}^2}{\bar{E}_b} = \frac{4A^2 T}{A^2 T} = 4 \quad (4.14)$$

As we will observe throughout the rest of this section, a power efficiency result of 4 is the *best possible* result that you can obtain for any digital modulation scheme when all possible binary sequences are each mapped to a unique symbol.

Suppose we now generalize the B-PAM results obtained for the average bit energy, the minimum Euclidean distance, and the power efficiency and apply them to the case when we try mapping binary sequences to one of M possible unique signal amplitude levels, referred to as M -ary pulse amplitude modulation (M-PAM). First, let us express the M-PAM waveform as:

$$s_i(t) = A_i \cdot p(t), \text{ for } i = 1, 2, \dots, M/2 \quad (4.15)$$

where $A_i = A(2i - 1)$, $p(t) = u(t) - u(t - T)$, and $u(t)$ is the unit step function. Graphically speaking, the M-PAM modulation scheme looks like the signal constellation shown in Figure 4.5.

In order to compute the power efficiency of M-PAM, $\mathcal{E}_{p,M\text{-PAM}}$, we select the d_{\min}^2 pair $s_1(t) = A \cdot p(t)$ and $s_2(t) = -A \cdot p(t)$ since this pair of signal waveforms are the closest to each other in terms of shape. Thus, using this selection of waveforms, we can solve for the difference between them:

$$\Delta s(t) = 2A \cdot p(t) \quad (4.16)$$

which yields a minimum Euclidean distance of:

$$d_{\min}^2 = 4A^2T \quad (4.17)$$

In order to calculate the average symbol energy, \bar{E}_s , we can simplify the mathematics by exploiting the symmetry of signal constellation, which yields:

$$\begin{aligned} \bar{E}_s &= \frac{2}{M} A^2 T \sum_{i=1}^{M/2} (2i - 1)^2 \\ &= A^2 T \frac{(M^2 - 1)}{3} \quad \text{which is simplified via table:} \\ \rightarrow \bar{E}_b &= \frac{\bar{E}_s}{\log_2(M)} = \frac{A^2 T (2^{2b} - 1)}{3b} \end{aligned} \quad (4.18)$$

Finally, solving for the power efficiency yields:

$$\eta_{p,M\text{-PAM}} = \frac{12b}{2^{2b} - 1} \quad (4.19)$$

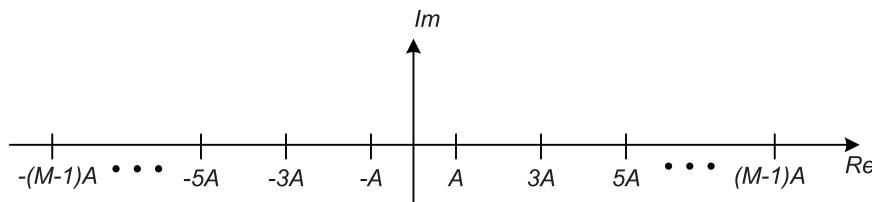


Figure 4.5 M-PAM Signal constellation.

4.2.3 Quadrature Amplitude Modulation

Similar to PAM, quadrature amplitude modulation (QAM) implies some sort of amplitude modulation. However, QAM modulation is a two-dimensional signal modulation scheme as opposed to PAM modulation. The two dimensions of the QAM modulation, namely, the *in-phase* and *quadrature* components, are orthogonal to each other, which implies that one can essentially double the transmission data rate “for free.” Furthermore, rectangular QAM can be thought of as two orthogonal PAM signals being transmitted simultaneously.

Mathematically, if a rectangular QAM signal constellation consists of M unique waveforms, this could potentially be represented as \sqrt{M} -PAM transmissions operating simultaneously in orthogonal dimensions. Note that QAM signal constellations could also take the form of nested circles (called circular QAM), or any other geometric pattern that involves orthogonal modulators. Rectangular QAM is a popular modulation scheme due to its relatively simple receiver structure, as shown in Figure 4.6, where each dimension employs a \sqrt{M} -ary PAM detector.

In order to determine the power efficiency of M-QAM, let us first define the mathematical representation of a signal waveform belonging to this form of modulation:

$$s_{ij}(t) = A_i \cdot \cos(\omega_c t) + B_j \cdot \sin(\omega_c t) \quad (4.20)$$

where ω_c is the carrier frequency, and A_i and B_j are the in-phase and quadrature amplitude levels. Notice how the cosine and sine functions are used to modulate these amplitude levels in orthogonal dimensions. Visualizing M-QAM as a signal constellation, the signal waveforms will assume positions in both the real and imaginary axes, as shown in Figure 4.7.

To compute the power efficiency of M-QAM, $\epsilon_{p,1M\text{-QAM}}$, we first need to calculate the minimum Euclidean distance, which becomes:

$$d_{\min}^2 = \int_0^T \Delta s^2(t) dt = 2A^2 T \quad (4.21)$$

where we have selected the following signal waveforms without loss of generality:

$$\begin{aligned} s_1(t) &= A \cdot \cos(\omega_c t) + A \cdot \sin(\omega_c t) \\ s_2(t) &= 3A \cdot \cos(\omega_c t) + A \cdot \sin(\omega_c t) \end{aligned} \quad (4.22)$$

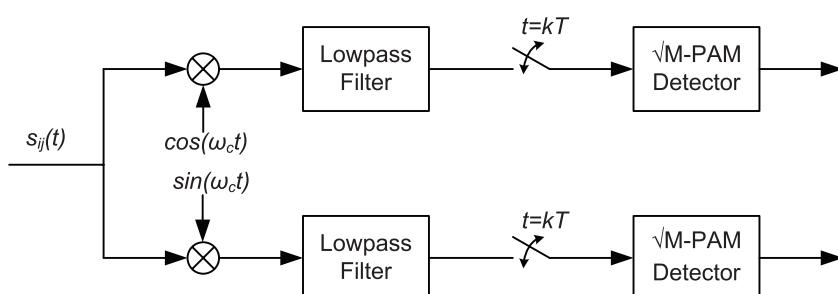


Figure 4.6 M-QAM receiver structure.

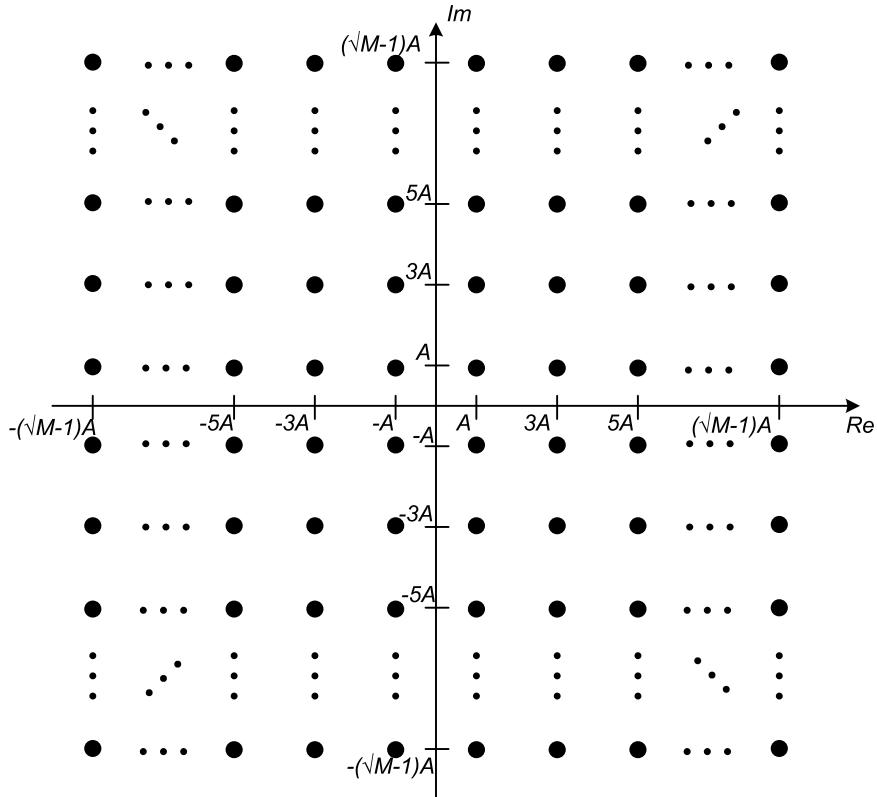


Figure 4.7 M-QAM signal constellation.

In order to derive the average symbol energy, \bar{E}_s , we leverage the expression from M-ary PAM by replacing M with \sqrt{M} such that:

$$\bar{E}_s = A^2 T \frac{M-1}{3} \quad (4.23)$$

which can then be used to solve:

$$\bar{E}_b = \frac{\bar{E}_s}{\log_2(M)} = A^2 T \frac{2^b - 1}{3b} \quad (4.24)$$

Thus, the power efficiency is equal to:

$$p_{M\text{-QAM}} = \frac{3!b}{2^b - 1} \quad (4.25)$$

4.2.4 Phase Shift Keying

Phase shift keying (PSK) is a digital modulation scheme that conveys data by changing, or modulating, the phase of a reference signal, (i.e., the carrier wave). Any digital modulation scheme uses a finite number of distinct signals to represent digital data. PSK uses a finite number of phases, each assigned a unique pattern of binary digits. Usually, each phase encodes an equal number of bits. Each pattern of bits forms the symbol that is represented by the particular phase. The demodulator,

which is designed specifically for the symbol-set used by the modulator, determines the phase of the received signal and maps it back to the symbol it represents, thus recovering the original data. This requires the receiver to be able to compare the phase of the received signal to a reference signal; such a system is termed *coherent*.

Phase shift keying (PSK) characterizes symbols by their phase. Mathematically, a PSK signal waveform is represented by:

$$s_i(t) = A \cos\left(2\pi f_c t + (2i - 1)\frac{\pi}{m}\right), \quad \text{for } i = 1, \dots, \log_2 m \quad (4.26)$$

where A is the amplitude, f_c is carrier frequency, and $(2i - 1)\frac{\pi}{m}$ is the phase offset of each symbol. PSK presents an interesting set of tradeoffs with PAM and QAM. In amplitude modulation schemes, channel equalization is an important part of decoding the correct symbols. In PSK schemes, the phase of the received signal is much more important than the amplitude information.

There are several types of PSK modulation schemes based on the number of M possible phase values a particular PSK waveform can be assigned. One of the most popular and most robust is binary PSK, or B-PSK, modulation, whose signal constellation is illustrated in Figure 4.8. In general, the modulation rule for B-PSK modulation is the following:

$$\begin{aligned} "1" \rightarrow s_1(t) &= A \cdot \cos(\omega_c t + \theta) \\ "0" \rightarrow s_2(t) &= -A \cdot \cos(\omega_c t + \theta) \\ &= A \cdot \cos(\omega_c t + \theta + \pi) \\ &= -s_1(t) \end{aligned} \quad (4.27)$$

In other words, the two signal waveforms that constitute a B-PSK modulation scheme are separated in phase by θ .

In order to derive the power efficiency of a B-PSK modulation scheme, $\epsilon_{p,\text{BPSK}}$, we first need to compute the minimum Euclidean distance d_{\min}^2 by employing the definition and solving for:

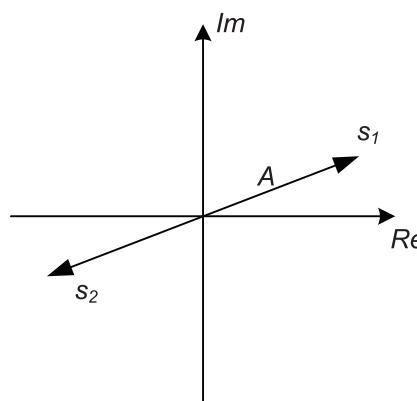


Figure 4.8 BPSK signal constellation.

$$\begin{aligned}
d_{\min}^2 &= \int_0^T (s_1(t) - s_2(t))^2 dt \\
&= 4A^2 \int_0^T \cos^2(\omega_c t + \theta) dt \\
&= \frac{4A^2 T}{2} + \frac{4A^2}{2} \int_0^T \cos(2\omega_c t + 2\theta) dt \\
&= 2A^2 T
\end{aligned} \tag{4.28}$$

Notice in (4.28) how the second term disappeared from the final result. This is due to the fact that the second term possessed a carrier frequency that was twice that of the original signal. Since the carrier signal is a periodic sinusoidal waveform, integrating such a signal possessing as high frequency would result in the positive portion of the integration canceling out the negative portion of the integration, yielding an answer than is close to zero. Consequently, we refer to this term as a *double frequency term*. Also note that many communication systems filter their received signals, which means the probability of filtering out the double frequency term is also quite high.



Note that another way for computing d_{\min}^2 is to use the concept of *correlation*, which describes the amount of similarity between two different signal waveforms. In this case, we can express the minimum Euclidean distance as:

$$d_{\min}^2 = \int_0^T (s_2(t) - s_1(t))^2 dt = E_{s_1} + E_{s_2} - 2\rho_{12} \tag{4.29}$$

where the symbol energy for symbol i , E_{s_i} , and the correlation between symbols 1 and 2, ρ_{12} , are given by:

$$E_{s_i} = \int_0^T s_i^2(t) dt \quad \text{and} \quad \rho_{12} = \int_0^T s_1(t)s_2(t) dt$$

Employing the definition for \bar{E}_b , we can now solve for the average bit energy of the B-PSK modulation scheme by first solving for the symbol energies of the two signal waveforms and then averaging them; that is,

$$\begin{aligned}
E_{s_1} &= \int_0^T s_1^2(t) dt = A^2 \int_0^T \cos^2(\omega_c t + \theta) dt \\
&= \frac{A^2 T}{2} + \frac{A^2}{2} \int_0^T \cos(2\omega_c t + 2\theta) dt \\
&= \frac{A^2 T}{2} \\
E_{s_2} &= \frac{A^2 T}{2} \\
\bar{E}_b &= P(0) \cdot E_{s_2} + P(1) \cdot E_{s_1} = \frac{A^2 T}{2}
\end{aligned} \tag{4.30}$$

Note that since the number of bits represented by a single symbol is equal to one, both the bit energy and symbol energy are equivalent.

Finally, applying the definition for the power efficiency, we get the following expression:

$$\varepsilon_p, \text{BPSK} = \frac{d_{\min}^2}{\bar{E}_b} = 4 \quad (4.31)$$

This is suppose to be the largest possible value for ε_p for a modulation scheme employing all possible signal representations, (i.e., $M = 2^b$ waveforms). Notice when using the correlation approach to calculate the minimum Euclidean distance, in order to get a large ε_p , we need to maximize d_{\min}^2 , which means we want $\rho_{12} < 0$. Thus, to achieve this outcome, we need the following situation:

$$E_{s_1} = E_{s_2} = E = A^2 T / 2 \quad (4.32)$$

which means $d_{\min}^2 = 2(E - \rho_{12})$ and consequently $\rho_{12} = -E$.

Show that for the following signal waveforms:



$$s_1(t) = A \cdot \cos(\omega_c t + \phi)$$

$$s_2(t) = 0$$

the power efficiency is equal to $\varepsilon_p = 2$.

Show that for the following signal waveforms:



$$s_1(t) = A \cdot \cos(\omega_c t + \phi)$$

$$s_2(t) = A \cdot \sin(\omega_c t + \phi)$$

the power efficiency is equal to $\varepsilon_p = 2$.

So far we have studied a PSK modulation scheme that consist of only just one of two waveforms. We will now expand our PSK signal constellation repertoire to include four distinct waveforms per modulation scheme. In quadrature PSK (QPSK) modulation, a signal waveform possesses the following representation:

$$s_i(t) = \pm A \cdot \cos(\omega_c t + \phi_i) \pm A \cdot \sin(\omega_c t + \phi_i) \quad (4.33)$$

where each signal waveform possesses the same amplitude but one of four possible phase values. This kind of phase modulation is illustrated by the signal constellation diagram shown in Figure 4.9, where each waveform is located at a different phase value.

In order to derive the power efficiency of QPSK, $\varepsilon_{p,\text{QPSK}}$, we first need to solve for the minimum Euclidean distance, d_{\min}^2 , which is equal to:

$$d_{\min}^2 = \int_0^T \Delta s^2(t) dt = 2A^2 T \quad (4.34)$$

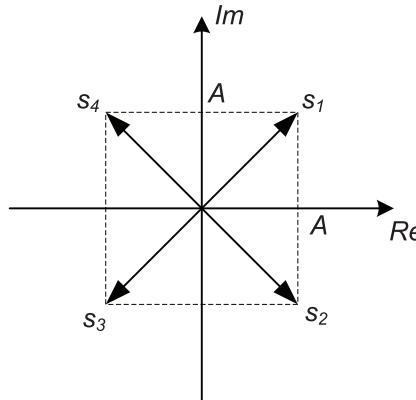


Figure 4.9 QPSK signal constellation.

Next, we would like to find \bar{E}_b , which requires us to average over all the signal waveforms. Consequently, this is equal to:

$$\bar{E}_b = \frac{(E_{s1} + E_{s2} + E_{s3} + E_{s4})/4}{\log_2(M)} = \frac{A^2 T}{2} \quad (4.35)$$

where the symbol energy of all four symbols is equal to $E_{s1} = E_{s2} = E_{s3} = E_{s4} = A^2 T$. Finally, solving for the power efficiency using (4.8), we get:

$$\rho_{\text{QPSK}} = \frac{d_{\min}^2}{\bar{E}_b} = 4 \quad (4.36)$$

which is the same as BPSK but with 2 bits per symbol, making this a fantastic result!

Finally, let us study the general case when a PSK modulation scheme has a choice of M possible phase values, where the distance of a signal constellation point to the origin is always a constant and the signal constellation consists of M equally spaced points on a circle. Referred to as M -ary PSK (M -PSK), a signal waveform can be mathematically represented as:

$$s_i(t) = A \cdot \cos\left(-\omega_c t + \frac{2\pi i}{M}\right), \text{ for } i=0,1,2,\dots,M-1 \quad (4.37)$$

Note that there are several advantages and disadvantages with this modulation scheme. For instance, as M increases the spacing between signal constellation points decreases, thus resulting in a decrease in error robustness. Conversely, having the information encoded in the phase results in constant envelope modulation, which is good for nonlinear power amplifiers and makes the transmission robust to amplitude distortion channels.

Regarding the derivation of the power efficiency for an M -PSK modulation scheme, $\varepsilon_{p,M-\text{PSK}}$, suppose we define two adjacent M -PSK signal waveforms as $s_1(t) = A \cdot \cos(\omega_c t)$ and $s_2(t) = A \cdot \cos(\omega_c t + 2\pi/M)$. Calculating the minimum Euclidean distance using:

$$d_{\min}^2 = E_{s1} + E_{s2} - 2\rho_{12} \quad (4.38)$$

where we define the symbol energy as:

$$E_{s_i} = \int_0^T s_i^2(t) dt = \frac{A^2 T}{2}, \text{ for } i = 1, 2 \quad (4.39)$$

and the correlation between the two signal waveforms as:

$$\rho_{12} = \int_0^T s_1(t)s_2(t) dt = \frac{A^2 T}{2} \cos\left(\frac{2}{M}\right) \quad (4.40)$$

this yields:

$$d_{\min}^2 = A^2 T \left(1 - \cos\left(\frac{2}{M}\right)\right) \quad (4.41)$$

The average bit energy \bar{E}_b is equal to $\bar{E}_b = \frac{\bar{E}_s}{\log_2(M)} = \frac{\bar{E}_s}{b}$ where $\bar{E}_s = A^2 T / 2$. Using the definition for the power efficiency from (4.8), we see that:

$$\epsilon_{p,M-PSK} = 2b \left(1 - \cos\left(\frac{2\pi}{M}\right)\right) = 4b \sin^2\left(\frac{\pi}{2^b}\right) \quad (4.42)$$

4.2.5 Power Efficiency Summary

After examining the power efficiency performance of several different modulation schemes, it is important to assess the tradeoffs between the different schemes such that we can make the appropriate design decisions in the future when implementing a digital communication system. To determine how much power efficiency we are losing relative to $p,QPSK$, which possesses the best possible result, we use the following expression:

$$\text{SNR} = 10 \cdot \log_{10} \left(\frac{p_{,QPSK}}{p_{,other}} \right) \quad (4.43)$$

Using this expression, we created a table of δSNR values for the modulation schemes studied in this chapter, as shown in Table 4.1.

From Table 4.1, notice how the two-dimensional modulation schemes perform better than the one-dimensional modulation schemes. Furthermore, notice how all of the modulation schemes studied are linear modulation schemes, which means they possess a similar level of receiver complexity. Given these insights on the power efficiency performance of these modulation schemes, we now turn our attention to the robustness of a modulation technique in the presence of noise.

Table 4.1 δSNR Values of Various Modulation Schemes

M	b	M-ASK	M-PSK	M-QAM
2	1	0	0	0
4	2	4	0	0
8	3	8.45	3.5	(??)
16	4	13.27	8.17	4.0
32	5	18.34	13.41	(??)
64	6	24.4	18.4	8.45

4.3 PROBABILITY OF BIT ERROR

One of the most commonly used quantitative metrics for measuring the performance of a digital communication system is the *probability of bit error* or BER, which is the probability that a bit transmitted will be decoded incorrectly. This metric is very important when assessing whether the design of a digital communication system meets the specific error robustness requirements of the application to be supported (e.g., voice, multimedia, data). Furthermore, having a metric that quantifies error performance is helpful when comparing one digital communication design with another. Consequently, in this section we will provide a mathematical introduction to the concept of BER.

Suppose that a signal $s_i(t)$, $i = 1, 2$, was transmitted across an AWGN channel with noise signal $n(t)$, and that a receiver intercepts the signal $r(t)$. The objective of the receiver is to determine whether either $s_1(t)$ or $s_2(t)$ was sent by the transmitter. Given that the transmission of either $s_1(t)$ or $s_2(t)$ is a purely random event, the only information that the receiver has about what was sent by the transmitter is the observed intercepted signal $r(t)$, which contains either signal in addition to some noise introduced by the AWGN channel.

Given this situation, we employ the concept of *hypothesis testing* [2] in order to set up a framework by which the receiver can decide on whether $s_1(t)$ or $s_2(t)$ was sent based on the observation of the intercepted signal $r(t)$. Thus, let us employ the following hypothesis testing framework:

$$\mathcal{H}_1 : r(t) = s_1(t) + n(t), 0 \leq t \leq T$$

$$\mathcal{H}_0 : r(t) = s_2(t) + n(t), 0 \leq t \leq T$$

where \mathcal{H}_0 and \mathcal{H}_1 are *hypothesis 0* and *hypothesis 1*.

Leveraging this framework, we next want to establish a *decision rule* at the receiver such that it can select which waveform was sent based on the intercept signal. Suppose we assume that $s_1(t)$ was transmitted. In general, we can determine the level of correlation between two signals $x(t)$ and $y(t)$ over the time interval $0 \leq t \leq T$ using the expression:

$$\int_0^T x(t)y(t)dt$$

Consequently, our decision rule on whether $s_1(t)$ or $s_2(t)$ was transmitted given that we observe $r(t)$ is defined as:

$$\int_0^T r(t)s_1(t)dt \geq \int_0^T r(t)s_2(t)dt \quad (4.44)$$

where we assume that $s_1(t)$ was transmitted. Recall that correlation tells us how similar one waveform is to another waveform. Therefore, if the receiver knows the appearance of $s_1(t)$ and $s_2(t)$, we can then determine which of these two waveforms is more correlated to $r(t)$. Since $s_1(t)$ was assumed to be transmitted, ideally the received signal $r(t)$ should be more correlated to $s_1(t)$ than $s_2(t)$.

On the other hand, what happens if some distortion, interference, and/or noise is introduced in the transmission channel such that the transmitted signal waveforms

are corrupted? In the situation where a transmitted signal waveform is sufficiently corrupted such that it appears to be more correlated to another possible signal waveform, the receiver could potentially select an incorrect waveform, thus yielding an *error event*. In other words, assuming $s_1(t)$ was transmitted, an error event occurs when:

$$\int_0^T r(t)s_1(t)dt \leq \int_0^T r(t)s_2(t)dt \quad (4.45)$$

Since $r(t) = s_1(t) + n(t)$, we can substitute this into the error event in order to obtain the decision rule:

$$\begin{aligned} \int_0^T s_1^2(t)dt + \int_0^T n(t)s_1(t)dt &\leq \int_0^T s_1(t)s_2(t)dt + \int_0^T n(t)s_2(t)dt \\ E_{s_1} - \rho_{12} &\leq \int_0^T n(t)(s_2(t) - s_1(t))dt \\ E_{s_1} - \rho_{12} &\leq z \end{aligned}$$

From this expression, we observe that both E_{s_1} and ρ_{12} are deterministic quantities. On the other hand, z is based on the noise introduced by the transmission channel, and thus it is a random quantity that requires some characterization. Since $n(t)$ is a Gaussian random variable, then z is also a Gaussian random variable. This is due to the fact that the process of integration is equivalent to a summation across an infinite number of samples, and since we are summing up Gaussian random variables, the result is also a Gaussian random variable. With $z \sim N(0, \sigma^2)$, we now need to calculate the variance of z , σ^2 , which can be solved as follows:

$$\begin{aligned} \sigma^2 = E\{z^2\} &= \frac{N_0}{2} \int_0^T (s_1(t) - s_2(t))^2 dt \\ &= \frac{N_0}{2}(E_{s_1} + E_{s_2} - 2\rho_{12}) \rightarrow \text{Assume } E_{s_1} = E_{s_2} = E \\ &= N_0(E - \rho_{12}) \end{aligned}$$

where $E = E_i = \int_0^T s_i^2(t)dt$ and $\rho_{12} = \int_0^T s_1(t)s_2(t)dt$. Note that we are assuming that the channel is introducing zero-mean noise, which means the sum of these noise contributions, (i.e., z , will also be zero-mean).

With both deterministic and random quantities characterized, we can now proceed with the derivation for the probability of bit error. The probability of an error occurring given that a “1” was transmitted, (i.e., $P(e|1)$), is equal to:

$$\begin{aligned} P(z \geq E - \rho_{12}) &= Q\left(\frac{E - \rho_{12}}{\sigma}\right) \rightarrow \begin{array}{l} \text{Since } z \sim N(0, \sigma^2) \\ \text{and } E - \rho_{12} \text{ is constant} \end{array} \\ &= Q\left(\sqrt{\frac{(E - \rho_{12})^2}{\sigma^2}}\right) \rightarrow \text{Use } \sigma^2 = N_0(E - \rho_{12}) \\ &= Q\left(\sqrt{\frac{E - \rho_{12}}{N_0}}\right) \end{aligned}$$

where the Q-function is defined as:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt \quad (4.46)$$

The next step is to optimize the probability of bit error by minimizing $P(e|1)$, which can be achieved by optimizing the correlation term ρ_{12} . Intuitively, the best choice we can make is when $s_2(t) = -s_1(t)$, which gives us $\rho_{12} = -E$. Consequently, this result yields:

$$P(e|1) = Q\left(\sqrt{\frac{2\bar{E}_b}{N_0}}\right) \quad (4.47)$$

Note that when $E_{s_1} \neq E_{s_2}$, we can then use $d_{\min}^2 = E_{s_1} + E_{s_2} - 2\rho_{12}$, which yields the following expression for the probability of bit error:

$$P_e = Q\left(\sqrt{\frac{d_{\min}^2}{2N_0}}\right) \quad (4.48)$$

Show that the total probability of bit error is equal to:



$$P_e = P(e|1)P(1) + P(e|0)P(0) = Q\left(\sqrt{\frac{E - \rho_{12}}{N_0}}\right) \quad (4.49)$$

When dealing with a large number of signal waveforms that form a modulation scheme, the resulting probability of error, P_e , is expressed as a sum of pairwise error probabilities (i.e., the probability of one received symbol being another, specific received symbol). The pairwise error probability of $s_i(t)$ being decoded when $s_j(t)$ was transmitted is given as:

$$Q\left(\frac{d_{ij}^2}{2N_0}\right) \quad (4.50)$$

where N_0 is the variance of the noise. An important note here is that we are assuming the noise is AWGN. since Q functions apply specifically to Gaussian random variables. Therefore, the complete expression for P_e can be expressed as:

$$Q\left(\frac{d_{\min}^2}{2N_0}\right) \div P_e = Q\left(\frac{d_{1j}^2}{2N_0}\right) + \dots + Q\left(\frac{d_{Mj}^2}{2N_0}\right), \quad i \neq j \quad (4.51)$$

where the second half of the relationship is the summation of every single pairwise error probability.

4.3.1 Error Bounding

Computing each pairwise error probability is not always practical. It is possible to create an upper and lower bound on P_e by computing only the pairwise errors of points that are within one degree of the point of interest. Consider the behavior of the Q function $Q(.)$. As the input to $Q(.)$ increases, the resulting output of the Q function

approaches zero. You will find that computing the pairwise error probability of points farther away yields negligible contributions to the total P_e , but can save a significant amount of time as well as cycles. Thus, an accurate estimate of $P(e)$ can be computed from the following bounds. These upper and lower bounds can be expressed as:

$$\mathcal{Q}\left(\frac{d_{\min}^2}{2N_0}\right) \leq P(e) \leq \mathcal{Q}\left(\frac{d_{ij}^2}{2N_0}\right) \quad (4.52)$$

where I is the set of all signal waveforms within the signal constellation that are immediately adjacent to the signal waveform j . In order to accurately assess the performance of a communications system, it must be simulated until a certain number of symbol errors are confirmed [3]. In most cases, 100 errors will give a 95 percent confidence interval, which should be employed later on in this book in order to characterize the bit error rate of any digital communication system under evaluation.

4.4 SIGNAL SPACE CONCEPT

Until this point we have studied digital communication systems from a signal waveform perspective. Leveraging this perspective, we have developed mathematical tools for analyzing the power efficiency and BER of different modulation schemes. However, there are several instances where the use of a signal waveform framework can be tedious or somewhat cumbersome. In this section, we will introduce another perspective on how to characterize and analyze modulation scheme using a different mathematics representation: signal vectors.

Suppose we define $\phi_i(t)$ as an orthonormal set of functions over the time interval $[0, T]$ such that:

$$\int_0^T \phi_i(t) \phi_j(t) dt = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

Given that $s_i(t)$ is the i^{th} signal waveform, we would like to represent this waveform as a sum of several orthonormal functions; that is,

$$s_i(t) = \sum_{k=1}^N s_{ik} \phi_k(t) \quad (4.53)$$

which can be equivalently represented by the vector:

$$\mathbf{s}_i = (s_{i1}, s_{i2}, s_{i3}, \dots, s_{iN}) \quad (4.54)$$

where the elements of the vector \mathbf{s}_i define the amplitude scaling of each orthonormal function used to represent the waveform. An illustration of a signal waveform represented by three orthonormal functions is shown in Figure 4.10. Consequently, given this relationship between the signal waveform and the orthonormal functions, where the former can be represented as the weighted sum of the latter, we can readily describe the signal waveform as a simple vector, which we will see next possesses the advantage of enabling us to employ relatively straightforward mathematical operations based on linear algebra.

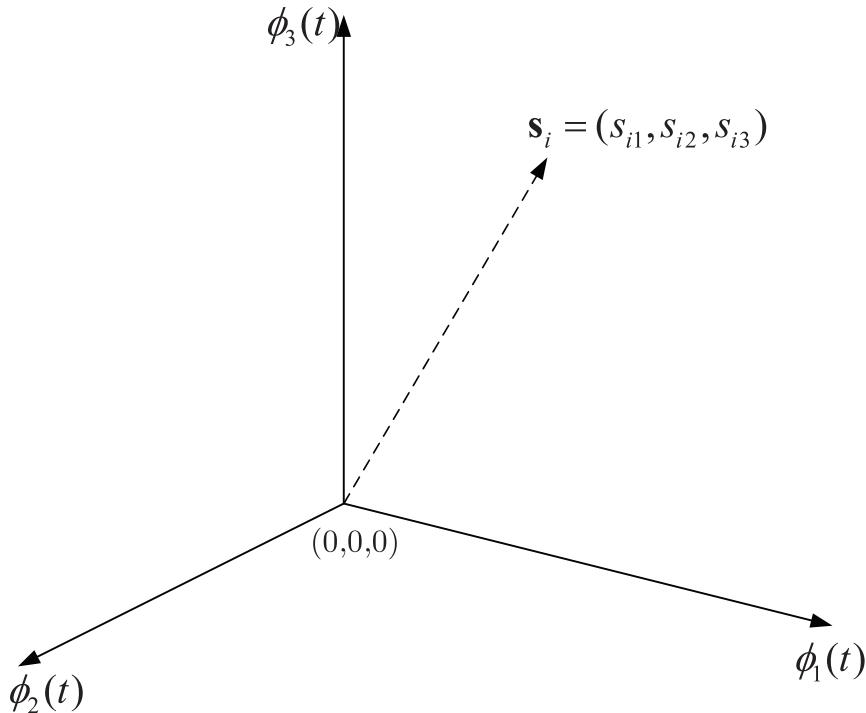


Figure 4.10 Sample vector representation of $s_i(t)$ in three dimensional space using basis functions $\phi_1(t)$, $\phi_2(t)$, and $\phi_3(t)$.

In order to find the vector elements, s_{il} , we need to solve the expression:

$$\int_0^T s_i(t)\phi_l(t)dt = \sum_{k=1}^N s_{ik} \int_0^T \phi_k(t)\phi_l(t)dt = s_{il} \quad (4.55)$$

which is essentially a *dot product* or *projection* of the signal waveform $s_i(t)$ on the orthonormal function $\phi_l(t)$. At the same time, if we perform the vector dot product between the signal waveforms $s_i(t)$ and $s_j(t)$, we get a correlation operation that is equal to:

$$\int_0^T s_i(t)s_j(t)dt = \mathbf{s}_i \cdot \mathbf{s}_j = \rho_{ij} \quad (4.56)$$

while the energy of a signal $s_i(t)$ is equal to:

$$E_{s_i} = \int_0^T s_i^2(t)dt = \mathbf{s}_i \cdot \mathbf{s}_i = \|\mathbf{s}_i\|^2 \quad (4.57)$$

All of these mathematical operations will be employed when determining the power efficiency of a modulation scheme or deriving the optimal decision rule for a receiver.

Suppose we would like to compute the power efficiency for a modulation scheme using a signal vector approach rather than a signal waveform approach. The first step would be to calculate the minimum Euclidean distance, which can be solved using the following:

$$\begin{aligned}
d_{\min}^2 &= \int_0^T \Delta s_{ij}^2(t) dt = \int_0^T (s_i(t) - s_j(t))^2 dt \\
&= \|s_i - s_j\|^2 = (s_i - s_j) \cdot (s_i - s_j) \\
&= E_{s_i} + E_{s_j} - 2\rho_{ij}
\end{aligned}$$

where the correlation term between signal waveforms $s_i(t)$ and $s_j(t)$ is given by:

$$\rho_{ij} = \int_0^T s_i(t)s_j(t) dt = s_i \cdot s_j \quad (4.58)$$

In order to solve for the power efficiency, we choose a set of orthonormal basis functions $\phi_i(t)$, $i = 1, 2, \dots, k$, where k is the dimension of the signal vector space. Given this set of functions, we can now represent the vector s_i , $i = 1, 2, \dots, M$ where $s_i = (s_{i1}, s_{i2}, \dots, s_{ik})$ and:

$$s_{ij} = \int_0^T s_i(t)\phi_j(t) dt \quad (4.59)$$

Consequently, using the vector representations for the signals and the orthonormal functions, we can calculate the minimum Euclidean distance:

$$d_{\min}^2 = \min_{i \neq j} \|s_i - s_j\|^2 \quad (4.60)$$

the average symbol and bit energy values:

$$\begin{aligned}
\bar{E}_s &= \frac{1}{M} \sum_{i=1}^M \|s_i\|^2 \\
\bar{E}_b &= \bar{E}_s / \log_2(M)
\end{aligned} \quad (4.61)$$

and the power efficiency:

$$\epsilon_p = d_{\min}^2 / \bar{E}_b \quad (4.62)$$

4.5 GRAM-SCHMIDT ORTHOGONALIZATION

In mathematics, particularly linear algebra and numerical analysis, the Gram-Schmidt orthogonalization process is a method for creating an orthonormal set of functions in an inner product space such as the Euclidean space \mathbb{R}^n . The Gram-Schmidt orthogonalization process takes a finite set of signal waveforms $\{s_1(t), \dots, s_M(t)\}$ and generates from it an orthogonal set of functions $\{\phi_1(t), \dots, \phi_k(t)\}$ that spans the space \mathbb{R}^n . Note that an orthonormal function possesses the following property:

$$\int_0^T \phi_i(t) \phi_j(t) dt = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, it is possible to represent a signal waveform $s_i(t)$ as the weighted sum of these orthonormal basis functions; that is,

$$s_i(t) = \sum_{k=1}^N s_{ik}\phi_k(t) \quad (4.63)$$

However, what we need now is an approach to generate the set of orthonormal basis functions $\{\phi_j(t)\}$.

To derive a set of orthogonal basis functions $\{\phi_1(t), \dots, \phi_i(t)\}$ from set of signal waveforms denoted by $\{s_1(t), \dots, s_M(t)\}$, let us first start with $s_1(t)$ and normalize it:

$$\phi_1(t) = \frac{s_1(t)}{\sqrt{E_{s_1}}}$$

where E_{s_1} is the energy of the signal $s_1(t)$. This normalized version of the signal waveform $s_1(t)$ will serve as our first orthonormal basis function from which we will construct the rest of our orthonormal basis function set. In other words, we are effectively “bootstrapping” a set of orthonormal basis functions based on the existing signal waveforms of the modulation scheme to be used. Note that we can represent $s_1(t)$ as:

$$s_1(t) = \sqrt{E_{s_1}}\phi_1(t) = s_{11}\phi_1(t)$$

where the coefficient $s_{11} = \sqrt{E_{s_1}}$ and the orthonormal function $\phi_1(t)$ satisfy the unit energy constraint as required.

Next, we would like to create the second orthonormal function, $\phi_2(t)$. In order to accomplish this task, we use the signal waveform $S_2(t)$ as a starting point. However, $S_2(t)$ may contain elements of $\phi_1(t)$, and thus we need to remove this component before normalizing it in order to transform this waveform into $\phi_2(t)$. To achieve this, we first determine how much of $\phi_1(t)$ is contained within $s_2(t)$ by taking the dot product between these two functions and determining how much $s_2(t)$ projects onto $\phi_1(t)$; that is,

$$s_{21} = \int_0^T s_2(t)\phi_1(t)dt$$

To help in getting the basis function $\phi_2(t)$, we define the intermediate function:

$$g_2(t) = s_2(t) - s_{21}\phi_1(t)$$

which is orthogonal to $\phi_1(t)$ over the interval $0 \leq t \leq T$ by virtue of the fact that we have removed the $\phi_1(t)$ component from $s_2(t)$. Finally, normalizing $g_2(t)$ yields the basis function $\phi_2(t)$:

$$\phi_2(t) = \frac{g_2(t)}{\sqrt{\int_0^T g_2^2(t)dt}} \quad (4.64)$$

which can be expanded to:

$$\phi_2(t) = \frac{s_2(t) - s_{21}\phi_1(t)}{\sqrt{E_{s_2} - s_{21}^2}} \quad (4.65)$$

where E_{s_2} is the energy of the signal $s_2(t)$. A quick sanity check clearly shows that the orthonormal basis function $\phi_2(t)$ satisfies the constraint:

$$\int_0^T \phi_2^2(t) dt = 1 \quad \text{and} \quad \int_0^T \phi_1(t)\phi_2(t) dt = 0$$

In general, we can define the following functions that can be employed in an iterative procedure for generating a set of orthonormal basis functions:

$$\begin{aligned} g_i(t) &= s_i(t) - \sum_{j=1}^{i-1} s_{ij}\phi_j(t) \\ s_{ij} &= \int_0^T s_i(t)\phi_j(t) dt, \quad j = 1, 2, \dots, i-1 \\ \phi_i(t) &= \frac{g_i(t)}{\sqrt{\int_0^T g_i^2(t) dt}}, \quad i = 1, 2, \dots, N \end{aligned} \quad (4.66)$$

We will now work out an example that deals with the Gram-Schmidt orthogonalization process.

4.5.1 An Example

Suppose we want to perform the Gram-Schmidt orthogonalization procedure of the signals shown in Figure 4.11 in the order $s_3(t), s_1(t), s_4(t), s_2(t)$ and obtain a set of orthonormal functions $\{\phi_m(t)\}$. Note that the order in which the signal waveforms are employed to generate the orthonormal basis functions is very important, since each ordering of signal waveforms can yield a potentially different set of orthonormal basis functions.

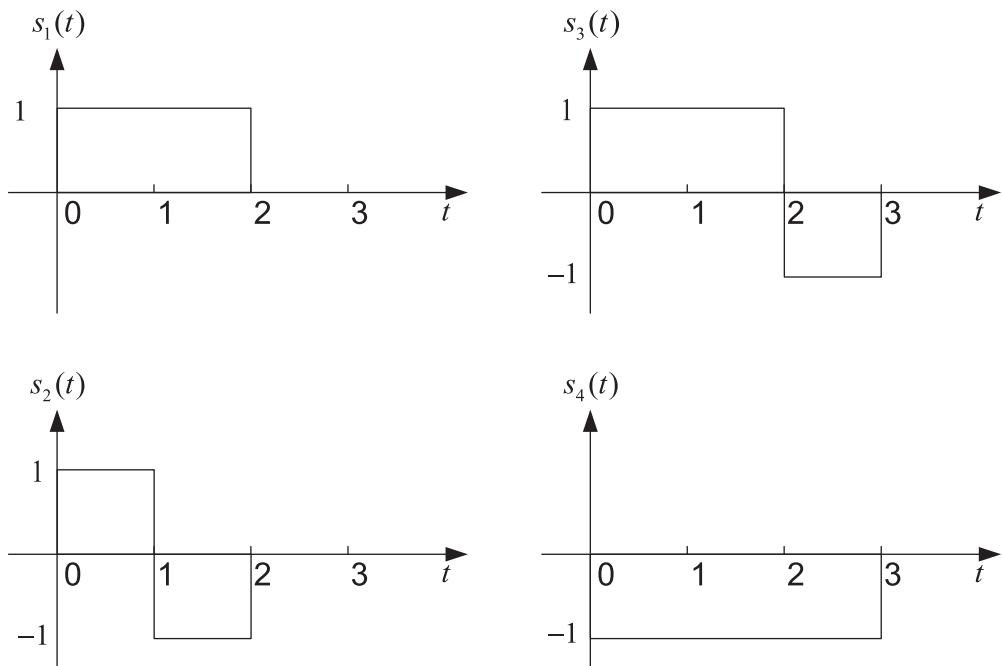


Figure 4.11 Example signal waveforms.

Starting with $s_3(t)$, we get:

$$\phi_1(t) = \frac{s_3(t)}{\sqrt{E_{s_3}}} = \frac{s_3(t)}{\sqrt{3}} \quad (4.67)$$

Then, leveraging the result for $\phi_1(t)$, we then derive the orthonormal basis function $\phi_2(t)$ using $S_1(t)$:

$$g_2(t) = s_1(t) - s_{12}\phi_1(t) = s_1(t) - \frac{2}{3}s_3(t) = \begin{cases} 1/3, & 0 \leq t < 2 \\ 2/3, & 2 \leq t < 3 \\ 0, & t \geq 3 \end{cases}$$

$$\therefore \phi_2(t) = \frac{g_2(t)}{\sqrt{\int_0^T g_2^2(t)dt}} = \begin{cases} \sqrt{6}/6, & 0 \leq t < 2 \\ 2/\sqrt{6}, & 2 \leq t < 3 \\ 0, & t \geq 3 \end{cases} \quad (4.68)$$

We subsequently repeat this operation for $s_4(t)$:

$$g_3(t) = s_4(t) - \sum_{j=1}^2 s_{4j}\phi_j(t) = 0$$

$$\therefore \phi_3(t) = 0 \quad (4.69)$$

but we notice the resulting $\phi_3(t)$ is equal to zero. This implies that the signal waveform $s_4(t)$ can be entirely characterized by only $\phi_1(t)$ and $\phi_2(t)$. Finally, for $s_2(t)$, we get the following:

$$g_4(t) = s_2(t) - \sum_{j=1}^3 s_{2j}\phi_j(t) = 0$$

$$\therefore \phi_4(t) = \frac{g_4(t)}{\sqrt{\int_0^T g_4^2(t)dt}} = \frac{s_2(t)}{\sqrt{2}} \quad (4.70)$$

Consequently, with the orthonormal basis functions $\{\phi_1(t), \phi_2(t), \phi_4(t)\}$ defined, we can now express the four signal waveforms as:

- $s_1 = (2/\sqrt{3}, \sqrt{6}/3, 0);$
- $s_2 = (0, 0, \sqrt{2});$
- $s_3 = (\sqrt{3}, 0, 0);$
- $s_4 = (-1/\sqrt{3}, -4/\sqrt{6}, 0).$

4.6 OPTIMAL DETECTION

Detection theory, or signal detection theory, is used in order to discern between signal and noise [2]. Using this theory, we can explain how changing the decision threshold will affect the ability to discern between two or more scenarios, often exposing how adapted the system is to the task, purpose, or goal at which it is aimed.

4.6.1 Signal Vector Framework

Let us assume a simple digital transceiver model as shown in Figure 4.12. As mentioned previously, the receiver only observes the corrupted version of $s_i(t)$ by the noise signal $n(t)$, namely, $r(t)$. The noise signal $n(t)$ usually represents the culmination of all noise sources into a single variable. Therefore, our detection problem in this situation can be summarized as: given $r(t)$ for $0 \leq t \leq T$, determine which $s_i(t)$, $i = 1, 2, \dots, M$, is present in the intercepted signal $r(t)$.

Suppose we decompose the waveforms $s_i(t)$, $n(t)$, and $r(t)$ into a collection of weights applied to a set of orthonormal basis functions, namely:

$$s_i(t) = \sum_{k=1}^N s_{ik} \phi_k(t), \quad r(t) = \sum_{k=1}^N r_k \phi_k(t), \quad n(t) = \sum_{k=1}^N n_k \phi_k(t)$$

Given that all of these signal waveforms use the same orthonormal basis functions, we can rewrite the waveform model expression $r(t) = s_i(t) + n(t)$ into:

$$\sum_{k=1}^N r_k \phi_k(t) = \sum_{k=1}^N s_{ik} \phi_k(t) + \sum_{k=1}^N n_k \phi_k(t)$$

$$\mathbf{r} = \mathbf{s}_i + \mathbf{n}$$

Since \mathbf{r} consists of a combination of the deterministic waveform s_i and probabilistic signal \mathbf{n} , our attention now turns to mathematically characterizing \mathbf{n} . Since the noise signal $n(t)$ is assumed to be a Gaussian random variable, we need to determine how the characteristics of this random variable translates into a signal vector representation. We know that the noise vector element n_k is equal to:

$$n_k = \int_0^T n(t) \phi_k(t) dt \quad (4.71)$$

which is the projection of the noise signal waveform on the orthonormal basis function $\phi_k(t)$. Since the noise signal $n(t)$ is a Gaussian random variable and the integration process is a linear operation, this means that n_k is a Gaussian random variable as well. Thus, the noise signal vector \mathbf{n} is a Gaussian vector. Let us now proceed with determining the statistical characteristics of \mathbf{n} in order to employ this knowledge in signal waveform detection.

First, we would like to calculate the mean of these vector elements. Thus, by applying the definition for the expectation, this yields:

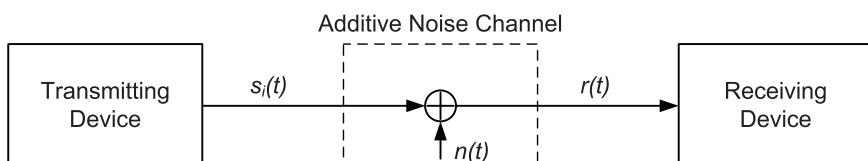


Figure 4.12 Simple digital transceiver model.

$$\begin{aligned}
E\{n_k\} &= E \left\{ \int_0^T n(t)\phi_k(t)dt \right\} \\
&= \int_0^T E\{n(t)\}\phi_k(t)dt \\
&= 0
\end{aligned} \tag{4.72}$$

since since $E\{n(t)\} = 0$, which ultimately means that the mean of the noise signal vector is $E\{n\} = 0$.

The next step is to calculate the variance of these vector elements. Suppose we let $(nn^T)_{kl} = n_k n_l$ be equal to the $(k, l)^{\text{th}}$ element of nn^T . Therefore, in order to determine $E\{n_k n_l\}$, where n_k and n_l are defined by:

$$n_k = \int_0^T n(t)\phi_k(t)dt, \quad n_l = \int_0^T n(\rho)\phi_l(\rho)d\rho$$

we can apply the definition for $E\{n_k n_l\}$, which yields:

$$\begin{aligned}
E\{n_k n_l\} &= E \int_0^T n(t)\phi_k(t)dt \cdot \int_0^T n(\rho)\phi_l(\rho)d\rho \\
&= E \int_0^T \int_0^T n(t)n(\rho)\phi_k(t)\phi_l(\rho)dtd\rho
\end{aligned}$$

Solving $E\{n_k n_l\}$ yields:

$$\begin{aligned}
E\{n_k n_l\} &= \int_0^T \int_0^T E\{n(t)n(\rho)\}\phi_k(t)\phi_l(\rho)dtd\rho \\
&= \int_0^T \int_0^T \frac{N_0}{2} \delta(t - \rho)\phi_k(t)\phi_l(t)dtd\rho \\
&= \frac{N_0}{2} \int_0^T \phi_k(t)\phi_l(t)dt \\
&= \frac{N_0}{2} \delta(k - l),
\end{aligned} \tag{4.73}$$

where the integration of the product of the two orthonormal functions $\phi_k(t)$ and $\phi_l(t)$ yields a delta function since only when $k = l$ do these two functions project onto each other. As a result, the matrix equivalent of this outcome is equal to:

$$E\{\mathbf{n}\mathbf{n}^T\} = \frac{N_0}{2} \mathbf{I}_{N \times N} \tag{4.74}$$

Given the vector representation of the Gaussian random variable obtained in (4.74), we need to define the joint probability density function of this representation in order to characterize the individual elements of this vector. Leveraging the assumption that the noise elements are independent to each other, we can express the joint probability density function as the product of the individual probability density functions for each element, yielding:

$$\begin{aligned} p(\mathbf{n}) &= p(n_1, n_2, \dots, n_N) = \frac{1}{(2\pi\sigma^2)^{N/2}} \prod_{i=1}^N e^{-n_i^2/2\sigma^2} \\ &= p(n_1)p(n_2)\dots p(n_N) \end{aligned}$$

where $p(n_i) = \frac{1}{\sigma\sqrt{2\pi}}e^{-n_i^2/2\sigma^2}$ is the probability density function for the vector element n_i . Since we know that $E[n_k n_l] = \frac{N_0}{2}\delta(k-l)$ we can then solve $E[n_k^2] = \frac{N_0}{2} = \sigma^2$. Additionally, we know that the dot product of a vector can be written as the summation of the squared elements, namely:

$$\sum_{i=1}^N n_i^2 = \|\mathbf{n}\|^2 \quad (4.75)$$

which can then be used to yield the following expression for the joint probability density function:

$$p(\mathbf{n}) = p(n_1, n_2, \dots, n_N) = \frac{1}{(2\pi\sigma^2)^{N/2}}e^{-\|\mathbf{n}\|^2/2\sigma^2} \quad (4.76)$$

4.6.2 Decision Rules

With the formulation of the joint probability density function derived in (4.76), we can now define a rule for the receiver that can be used to determine which signal waveform is being intercepted given the presence of some noise introduced by the channel. Suppose we define the following criterion for the receiver as:

$$\text{Minimize } P(\text{error}) \rightarrow P(\hat{m}_i \neq m_i)$$

$$\text{Maximize } P(\text{correct}) \rightarrow P(\hat{m}_i = m_i) \quad (4.77)$$

where the probability of error is $P(e) = P(\text{error})$, the probability of correct reception is $P(c) = P(\text{correct})$, and $P(e) = 1 - P(c)$ is the complementary relationship between these two probabilities. Then, using the law of total probability, the overall probability of correct detection is equal to:

$$P(c) = \int_V P(c|\mathbf{r} = \rho)p(\rho)d\rho \quad (4.78)$$

where $P(\text{clr} = 0) = 0$ and $p(\rho) = 0$. Therefore, we observe that when $P(c)$ attains a maximum value, this occurs when $P(\text{clr} = \rho)$ also possesses a maximum value.

In order to maximize $P(\text{clr} = \rho)$, we use the following *decision rule* at the receiver:

$$P(s_k|\rho) \geq P(s_i|\rho), \text{ for } i = 1, 2, \dots, M \text{ and } i \neq k \quad (4.79)$$

for $i = 1, 2, \dots, M$ and $i \neq k$. Note that for this decision rule we are assuming that s_k is present in ρ such that:

$$\rho = s_k + \mathbf{n} \rightarrow \hat{m} = m_k \quad (4.80)$$

Employing a mixed form of *Bayes Rule* that is composed of probability density functions and probabilities, namely:

$$P(s_i | r = \rho) = \frac{p(\rho | s_i)P(s_i)}{p(\rho)} \quad (4.81)$$

we would like to manipulate this decision rule into a formulation that can be employed by a receiver. Specifically, recall how we wanted to maximize $P(c|r = \rho)$ earlier in this section. By employing the mixed Bayes Rule formulation, the optimal detector can be rewritten such that it is equal to:

$$\max_{s_i} P(s_i | r = \rho) = \max_{s_i} \frac{p(\rho | s_i)P(s_i)}{p(\rho)} \quad (4.82)$$

for $i = 1, 2, \dots, M$. Since $p(\rho)$ does not depend on s_i , we can simplify the optimal detector expression such that:

$$\max_{s_i} p(\rho | s_i)P(s_i) \quad (4.83)$$

for $i = 1, 2, \dots, M$.

Based on our result in (4.83), two types of detectors can be derived based on this expression. The first type of detector is referred to as a *maximum a posteriori* (MAP) detector, which can be expressed as:

$$P(s_i | r = \rho) = \max_{s_i} p(\rho | s_i)P(s_i) \quad (4.84)$$

for $i = 1, 2, \dots, M$. However, in the event that $P(s_i) = \frac{1}{M}$, which implies that $P(s_i)$ does not depend on s_i , we can omit the $P(s_i)$ term from the optimal detector expression, yielding the second type of detector, referred to as a *maximum likelihood* (ML) detector:

$$P(s_i | r = \rho) = \max_{s_i} p(\rho | s_i) \quad (4.85)$$

for $i = 1, 2, \dots, M$. In the next section, we will mathematically derive the maximum likelihood detector given the optimal decision rule for data transmissions being performed across AWGN channels.

4.6.3 Maximum Likelihood Detection in an AWGN Channel

Maximum likelihood detection is a popular statistical method employed for fitting a statistical model to data and for identifying model parameters. In general, for a fixed set of data and underlying probability model, a maximum likelihood approach selects values of the model parameters that produce the distribution that are most likely to have resulted in the observed data (i.e., the parameters that maximize the likelihood function).

Suppose that a data transmission is operating across an AWGN channel prior to interception by the receiver. Recall that the transmission model for this scenario is given by:

$$r = s_i + n \quad (4.86)$$

where s_i is the i th signal waveform sent by the transmitter, n is the noise introduced to the data transmission by the AWGN channel, and r is the intercepted signal

waveform by the receiver. Given that s_i is a deterministic quantity, and \mathbf{n} is a random entity that has just been characterized by a joint probability density function, what is needed now is a characterization of \mathbf{r} , which can be derived from the characterization of \mathbf{n} coupled with the deterministic properties of s_i .

Suppose we consider the conditional probability of a single element of the received vector $\mathbf{r} = \rho$, say the k th element, given that the signal waveform s_i was assumed to be transmitted:

$$p(\rho_k | s_{ik}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\rho_k - s_{ik})^2/2\sigma^2} \quad (4.87)$$

where the k th element of the noise vector is equal to $n_k = \rho_k - s_{ik}$. Since we assume that the AWGN vector elements are uncorrelated, (i.e., independent), we can rewrite this conditional probability expression as:

$$\prod_{k=1}^N p(\rho | s_i) = \prod_{k=1}^N p(\rho_k | s_{ik}), \text{ for } i = 1, 2, \dots, M \quad (4.88)$$

Consequently, this product of multiple elemental probability density functions will ultimately yield the following expression:

$$p(\rho | s_i) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\|\rho - s_i\|^2/2\sigma^2} \quad (4.89)$$

Notice how we now have a formulation for the conditional probability that is entirely represented in terms of s_i, ρ , and their respective elements. Leveraging this expression, we can proceed with mathematically determining the maximum likelihood detector.

Since we would like to solve for $\max_{s_i} p(\rho | s_i)$, suppose we take the expression for $p(\rho | s_i)$, apply it to the detector, and take the natural logarithm of the resulting expression. Performing these operations would yield the following:

$$\ln(p(\rho | s_i)) = \frac{N}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{\|\rho - s_i\|^2}{2\sigma^2} \quad (4.90)$$

Note that the natural logarithm was employed in order to get rid of the exponential base in the expression, thus yielding a linear expression for the optimal decision rule. Furthermore, since natural logarithms are monotonic functions (i.e., if $x_2 > x_1$ then $\ln(x_2) > \ln(x_1)$), the decision rule would still remain valid when the inequality is employed.

Solving for this linear decision rule, and given the monotonic behavior of the natural logarithm, we can derive the following:

$$\begin{aligned} \max_{s_i} \ln(p(\rho | s_i)) &= \max_{s_i} \left(\frac{N}{2} \ln \frac{1}{2\pi\sigma^2} - \frac{\|\rho - s_i\|^2}{2\sigma^2} \right) \\ &= \max_{s_i} \frac{\|\rho - s_i\|^2}{2\sigma^2} \\ &= \min_{s_i} \|\rho - s_i\|^2 \quad (4.91) \end{aligned}$$

Since we are interested in the choice of s_i that yields the maximum value for the decision rule, we can rewrite this decision rule as:

$$s_k = \arg \min_{s_i} \|\rho - s_i\| \quad \hat{m} = m \quad (4.92)$$

Note that one of the advantages of employing a vector representation for these decision rules is that the entire scenario can be interpreted in terms of distance. Specifically, the term $\|\rho - s_i\|$ actually represents the distance between the heads of two vectors, ρ and s_i , whose tails are located at the origin. Thus, a maximum likelihood detector is the equivalent of a minimum distance detector.

Q

Referring to the signal constellation diagram shown in Figure 4.13, implement a simple QPSK transceiver operating in an AWGN channel and implement a maximum likelihood detector. Does this decision rule match the decision regions in Figure 4.13? Does this decision rule adequately declare s_i as the transmitted signal based on which quadrant ρ appears in? What is the impact of the AWGN channel for different values for the variance given that the noise is zero-mean?

4.7 BASIC RECEIVER REALIZATIONS

The fundamental challenge of digital communications is recovering what was transmitted after it has passed through a channel and been corrupted by noise. The first receiver structure we will examine is based on filtering the received signal with a static filter that maximizes the SNR of the channel, which will subsequently minimize the bit error rate. However, one of the disadvantages of a matched filtering approach is that it requires a priori knowledge of all the possible signal waveforms sent by the transmitter.

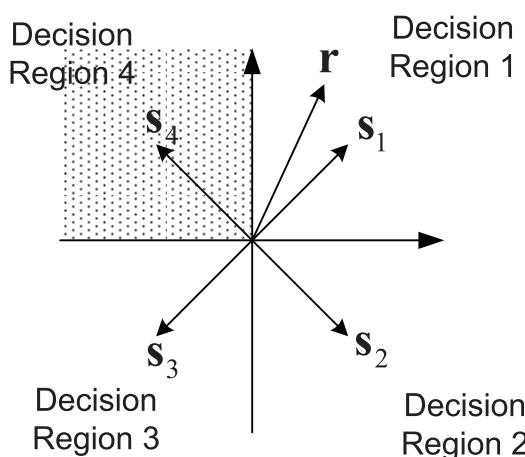


Figure 4.13 Decision regions for QPSK signal constellation.

4.7.1 Matched Filter Realization

When designing a receiver, we are interested in a decision rule where the receiver yields the correct result more often than any other decision rule that can be employed by the receiver. In other words, we are interested in detecting a pulse transmitted over a channel corrupted by noise.

Suppose we employ the following transmission model:

$$x(t) = g(t) + w(t), 0 \leq t \leq T \quad (4.93)$$

where $g(t)$ is a pulse signal, $w(t)$ is a white noise process with mean $= 0$ and power spectral density equal to, $\frac{N_0}{2}$ and $x(t)$ is the observed received signal. Assuming the receiver knows all the possible waveforms of $g(t)$ produced by the transmitter, the objective of the receiver is to detect the pulse signal $g(t)$ in an optimum manner based on an observed received signal $x(t)$. Note that the signal $g(t)$ may represent a “1” or a “0” in a digital communication system.

In order to enable the receiver to successfully detect the pulse signal $g(t)$ in an optimal manner given the observed received signal $x(t)$, let us filter $x(t)$ such that the effects of the noise are minimized in some statistical sense so the probability of correct detection is enhanced. Suppose we filter $x(t)$ using $h(t)$ such that the output of this process yields:

$$y(t) = g_0(t) + n(t) \quad (4.94)$$

where $n(t)$ is the result of the noise signal $w(t)$ filtered by $h(t)$ and $g_0(t)$ is the filtered version of $g(t)$ by $h(t)$. The transmission model and filtering operation by $h(t)$ is illustrated in Figure 4.14.

Let us rewrite this filtering operation in the frequency domain, where the time-domain convolution operations become frequency-domain products. Thus, taking the inverse Fourier transform of $H(f)G(f)$, which is equivalent to a convolution of $h(t)$ and $g(t)$, we get the following expression for the filtered version of $g(t)$:

$$g_0(t) = \int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi ft} df \quad (4.95)$$

where the inverse Fourier transform returns the filtering operation back to the time domain.

Let us now calculate the instantaneous power of the filtered signal $g_0(t)$, which is given as:

$$|g_0(t)|^2 = |H(f)G(f)e^{j2\pi ft} df|^2 \quad (4.96)$$

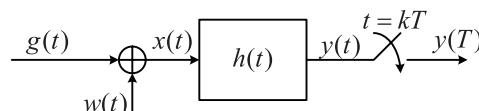


Figure 4.14 Filtering process for detecting $g(t)$.

In order to determine a quantitative metric that would indicate when we have achieved the goal of maximizing $g_0(t)$ relative to $n(t)$, let us employ the peak pulse SNR, which is defined as:

$$\eta = \frac{|g_0(T)|^2}{E\{n^2(t)\}} \quad (4.97)$$

where $|g_0(T)|^2$ is the instantaneous power of the output signal at sampling instant T , and $E\{n^2(t)\}$ is the average power of the output noise. Thus, goal of this matched filter realization is to maximize $g_0(t)$ with respect to $n(t)$ using the peak pulse SNR metric, which can be achieved by designing a filter $h(t)$ that can yield the largest possible value for η .

In order to design $h(t)$, we need to mathematically solve for $h(t)$, which consists of evaluating the expression:

$$|g_0(t)|^2 = \left| \int H(f)G(f)e^{j2\pi f t} df \right|^2 \quad (4.98)$$

which is the magnitude squared of the inverse Fourier transform of $H(f)G(f) = \mathcal{F}\{h(t)^*g(t)\}$. Since $w(t)$ is a white Gaussian process with power spectral density $\frac{N_0}{2}$, we know that by the EWK Theorem that the power spectral density of the filtered noise signal $n(t)$ is equal to $S_N(f) = \frac{N_0}{2} |H(f)|^2$. Therefore, applying the definition for η and including these expressions will yield:

$$\eta = \frac{\int |H(f)G(f)e^{j2\pi f T}|^2 df}{\frac{N_0}{2} \int |H(f)|^2 df} \quad (4.99)$$

From this resulting expression, we see that we need to solve for frequency response $H(f)$ such that it yields the largest possible value for the peak pulse SNR η . In order to obtain a closed-form solution let us employ Schwarz's Inequality. Suppose that we have two complex functions, say, $\phi_1(x)$ and $\phi_2(x)$, such that:

$$|\phi_1(x)|^2 dx < \quad \text{and} \quad |\phi_2(x)|^2 dx < \quad (4.100)$$

Then, by Schwarz's Inequality we can rewrite the following integral expression as an inequality:

$$\left| \int \phi_1(x)\phi_2(x) dx \right|^2 \leq \int |\phi_1(x)|^2 dx \times \int |\phi_2(x)|^2 dx \quad (4.101)$$

with this expression becoming an equality when $\phi_1(x) = K \cdot \phi_2^*(x)$.

Therefore, leveraging Schwarz's Inequality in our expression for the peak pulse SNR, it can be shown that the numerator of (4.99) can be rewritten as:

$$\left| \int H(f)G(f)e^{j2\pi f T} df \right|^2 \leq \int |H(f)|^2 df \times \int |G(f)|^2 df \quad (4.102)$$

which then yields the following inequality for η :

$$\eta = \frac{2}{N_0} |G(f)|^2 df \quad (4.103)$$

Thus, in order to make this expression an equality, the optimal value for $H(f)$ should be equal to:

$$H_{\text{opt}}(f) = K \cdot G^*(f) e^{-j2\pi f T} \quad (4.104)$$

whose time-domain representation can be mathematically determined using the inverse Fourier transform:

$$h_{\text{opt}}(t) = K \times G(f) e^{-j2\pi f T} e^{-j2\pi f t} df = K \times g(T-t) \quad (4.105)$$

Notice that when we are performing a matched filtering operation, we are convolving the time-flipped and time-shifted version of the transmitted pulse with the transmitted pulse itself in order to *maximize the SNR*.



The reason that we call these filters *matched filters* is because when we convolve the time-flipped and time-shifted version of the transmitted pulse signal with itself, the process is SNR maximizing. Consequently, if a receiver intercepts some unknown noise-corrupted signal, it can readily identify which one was sent by a transmitter by *matching* this intercepted signal to all the available signal waveforms known at the receiver using an implementation illustrated in Figure 4.15.



Referring to Figure 4.16, suppose we have a signal $g(t)$. Show that $h(t)$ and $g_0(t)$ are the corresponding matched filter and filtered output signals.

4.7.2 Correlator Realization

Recall that a matched filter realization assumes some sort of knowledge regarding the transmitted data. However, if the receiver possesses this information about the

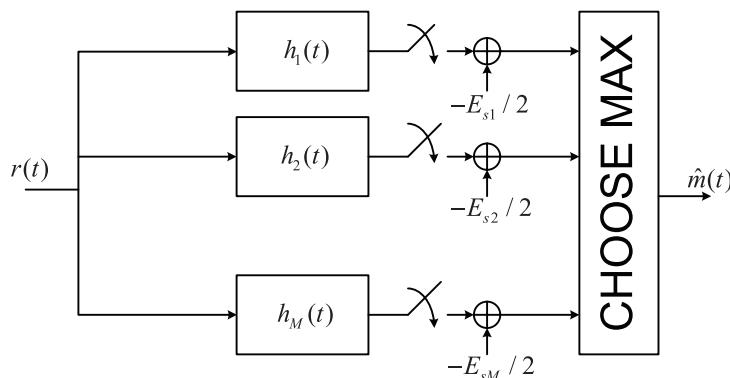


Figure 4.15 Schematic of matched filter realization of receiver structure.

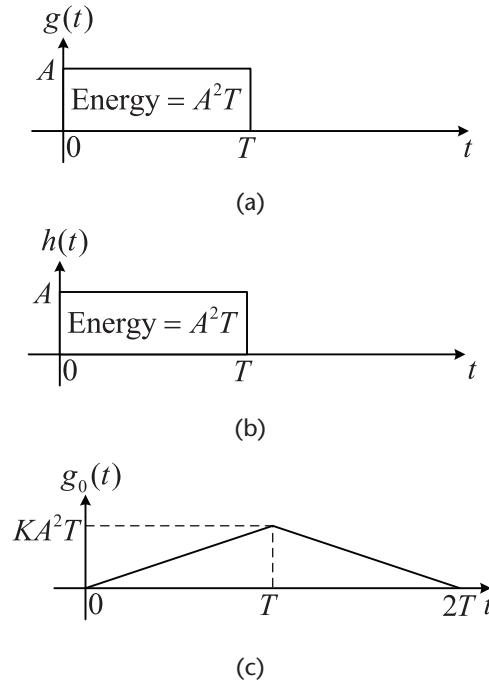


Figure 4.16 An example of the time-domain input and output signal waveforms processed by a matched filter. (a) Time-domain representation of the input signal to the matched filter. (b) Time-domain impulse response of the matched filter. (c) Time-domain representation of the output signal of the matched filter.

transmitter and its signal characteristics, it is also possible to employ a more statistical approach for determining which signal waveforms have been sent, even in the presence of a noisy, corruption-inducing channel. Specifically, we can employ the concept of *correlation* such that we only need to assume knowledge about the waveforms themselves.¹

Suppose we start with the decision rule derived at the beginning of this section and expand it such that:

$$\begin{aligned} \min_{\mathbf{s}_i} \|\rho - \mathbf{s}_i\|^2 &= \min_{\mathbf{s}_i} (\rho - \mathbf{s}_i) \times (\rho - \mathbf{s}_i) \\ &= \rho \times \rho - 2\rho \times \mathbf{s}_i + \mathbf{s}_i \times \mathbf{s}_i. \end{aligned} \quad (4.106)$$

Since $\rho \cdot \rho$ is common to all the decision metrics for different values of the signal waveforms \mathbf{s}_i , we can conveniently omit it from the expression, thus yielding:

$$\min_{\mathbf{s}_i} (-2\rho \cdot \mathbf{s}_i + \mathbf{s}_i \cdot \mathbf{s}_i) = \max_{\mathbf{s}_i} (2\rho \cdot \mathbf{s}_i - \mathbf{s}_i \cdot \mathbf{s}_i) \quad (4.107)$$

where $\rho \cdot \mathbf{s}_i$ and $\mathbf{s}_i \cdot \mathbf{s}_i$ are defined by:

- For a matched filtering implementation, knowledge of both the transmission signal waveforms and the statistical characteristics of the noise introduced by the channel is needed by the receiver.

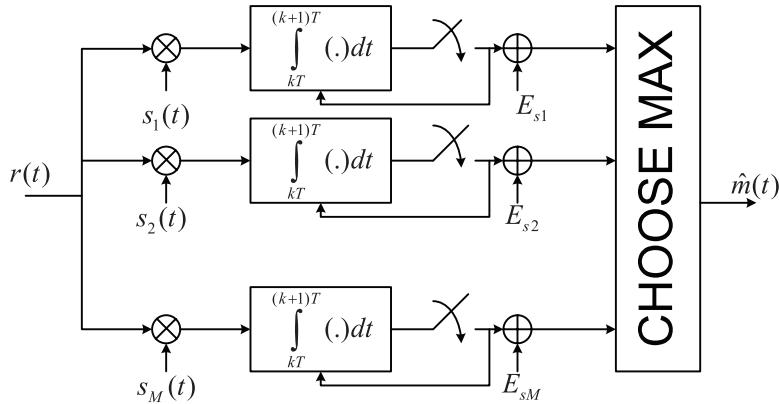


Figure 4.17 Correlator realization of a receiver structure assuming perfect synchronization.

$$\rho \cdot s_i = \int_0^T \rho(t)s_i(t)dt \quad s_i \cdot s_i = \int_0^T s_i^2(t)dt = E_{s_i}$$

We can observe that the waveform representation of $\rho \cdot s_i$ is equal to the correlation of $r(t) = p(t)$ with respect to $s_i(t)$. Thus, when $s_k(t)$ is present in $r(t)$, the optimal detector is equal to:

$$s_k = \arg \max_i \left(\int_0^T \rho(t)s_i(t)dt - \frac{E_{s_i}}{2} \right) \quad (4.108)$$

Based on this result, we can design a receiver structure that leverages correlation in order to decide which signal waveform was sent by the transmitter based on the observed intercepted signal at the receiver. A schematic of a correlation-based implementation is shown in Figure 4.17. Given $r(t) = s_i(t) + n(t)$ and we observe only $r(t) = p(t)$ at the input to the receiver, we first correlate $r(t)$ with $s_i(t)$ across all i . Next, we normalize the correlation result by the corresponding signal energy E_{s_i} in order to facilitate a fair comparison. Note that if all energy values are the same for each possible signal waveform, we can dispense with the energy normalization process since this will have no impact on the decision making. Finally, the resulting decision values for each of the branches are compared against each other, and the branch with the largest resulting value is selected.

4.8 CHAPTER SUMMARY

A deep and thorough understanding of digital communication theory is vitally essential when designing and evaluating software-defined radio implementations. In this chapter, an overview of several useful topics, including several different types of modulation schemes, the derivation of the probability of error, Gram-Schmidt orthogonalization, formulation of the optimal decision rule, and two receiver structures were studied in order to provide the reader with the fundamentals needed in order to master these versatile yet complex systems. In the subsequent chapters, the fundamental knowledge obtained in this chapter as well as the two previous

chapters will be leveraged extensively when implementing digital communication systems and networks based on software-defined radio technology.

4.9 ADDITIONAL READINGS

Given the introductory nature of this chapter with respect to the topic of digital communications, the interested reader is definitely encouraged to explore the numerous books that provide a substantially more detailed and advanced treatment of this topic. For instance, the latest edition of the seminal digital communications textbook by Proakis and Salehi [4] provides a rigorous, mathematical treatment of many of the concepts covered in this chapter, in addition to many other topics not presented, such as spread spectrum technologies, equalization, and RAKE receiver implementations. To complement this mathematically terse document, the authors also published a companion textbook that treats digital communications from a more applied perspective, including examples in MATLAB and Simulink [5].

As for introductory textbooks on digital communications, Sklar wrote an excellent document that provides the reader with a balance of mathematical rigor, detailed explanations, and several well-crafted examples [6]. The textbook by Couch is also in the same category as Sklar, but it treats both analog and digital communications [7], which is well suited for individuals who do not possess a background in the communications field. Rice wrote his introductory textbook on digital communications from a discrete-time perspective, which is suited for an individual possessing a background in discrete-time signal and systems [8]. The textbook also provides numerous end-of-chapter problems as well as MATLAB examples available online, providing the reader with many opportunities to practice the theoretical concepts covered within this text.

The classic digital communications book by Barry, Messerschmitt, and Lee [9] is an excellent reference textbook for those individuals who possess some understanding about digital communications but need convenient and immediate access to detailed information. Similarly, the textbooks by Madhow [10] and Pursley [11] both provide readers with a more advanced treatment of digital communication theory. Finally, the book by Hsu [12] is an excellent reference textbook that consists mostly of a popular collection of solved problems.

4.10 PROBLEMS

1. *Power efficiency:* Find and compare the power efficiency for the following three binary signal sets. Give the relative performance in decibels. Assume the $s_1(t)$ and $s_2(t)$ are equally likely.
 - (a) $s_1(t) = B \sin(\omega_0 t + \phi)$ and $s_2(t) = B \sin(\omega_0 t - \phi)$ for $0 \leq t \leq T$ and where $\bar{E}_b \leq \frac{A^2 T}{2}$. Find the best (B, ϕ) .
 - (b) $s_1(t) = A \sin(\omega_0 t + \theta)$ and $s_2(t) = B \sin(\omega_0 t)$ for $0 \leq t \leq T$ and where $\bar{E}_b \leq \frac{A_0^2 T}{2}$ and A_0 is known. Find the best (A, B, θ) .

- (c) $s_1(t) = A\sin(2 - \pi f_0 t + \theta)$ and $s_2(t) = A\cos(2 - \pi f_0 t - \theta)$ for $0 \leq t \leq T$.
 Find the best peak frequency deviation, Δf , in terms of T .

2. *Decision boundaries:* Three different eight-point constellations are proposed as shown in Figure 4.18. Draw the appropriate decision boundaries for each technique in two dimensions, and express the probability of symbol error P_s in terms of the *peak* energy-to-noise density ratio. Repeat for an *average* energy normalization. Which of the demodulators would be easier to implement? Justify.
3. *Power efficiency:* The signal constellation for a communications system with 16 equiprobable symbols is shown in Figure 4.19. The channel is AWGN with noise power spectral density of $\frac{N_0}{2}$. Compute and compare the power efficiency of this system with a 16-level PAM system with adjacent amplitude differences of $2A$.
4. *Matched filtering:* Consider the set of $M = 4$ waveforms shown in Figure 4.20.
- Apply the *Gram-Schmidt orthogonalization procedure* to obtain a set of K basis functions that spans the set of signals.
 HINT: Use the order $s_3(t), s_0(t), s_2(t)$, and $s_1(t)$.
 - What is the dimension of the signal space that contains these four signals?
 - Plot the signal constellation corresponding to the signal set.

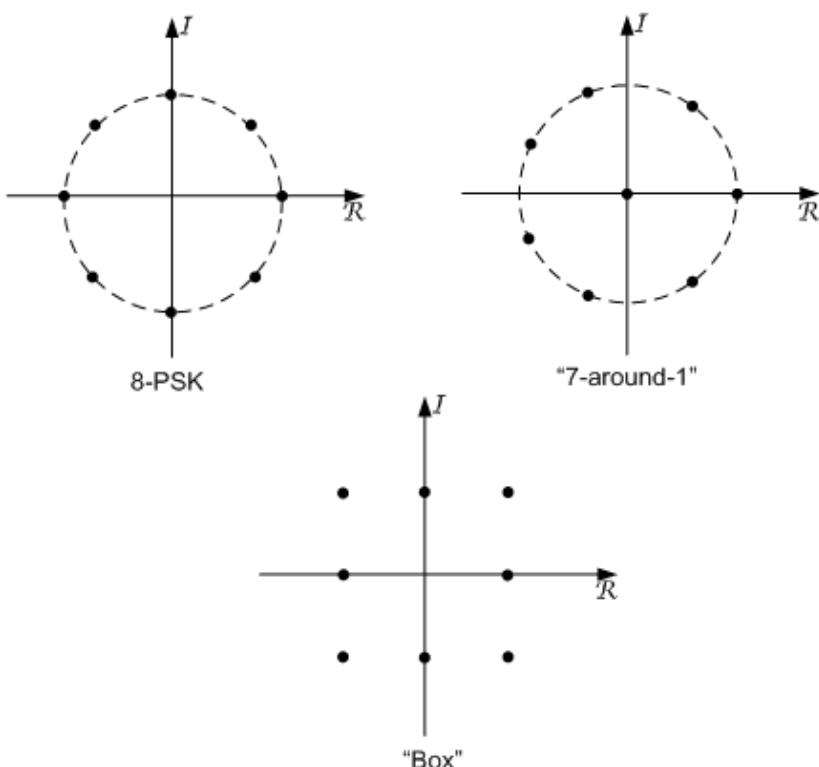


Figure 4.18 Three eight-point signal constellations (8-PSK, "7-around-1," and "Box").

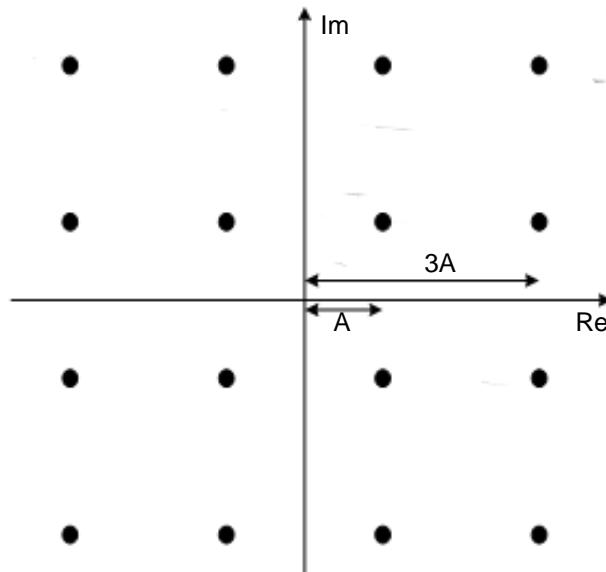


Figure 4.19 16-QAM signal constellation.

- (d) Suppose that the symbol period is $T = 1$; sketch the optimal matched filter designs $h_0(t)$, $h_1(t)$, $h_2(t)$, and $h_3(t)$ for each of these signals.

- (e) Sketch the matched filter realization when the receiver intercepts a signal $r(t)$.

NOTE: The received signal $r(t)$ can be composed of $s_0(t)$, $s_1(t)$, $s_2(t)$, and $s_3(t)$ signal waveforms that have been transmitted through an additive white Gaussian noise (AWGN) channel.

- (f) Sketch the output of the matched filters $h_0(t)$, $h_1(t)$, $h_2(t)$, and $h_3(t)$ when the input is equal to $h_0(t)$.

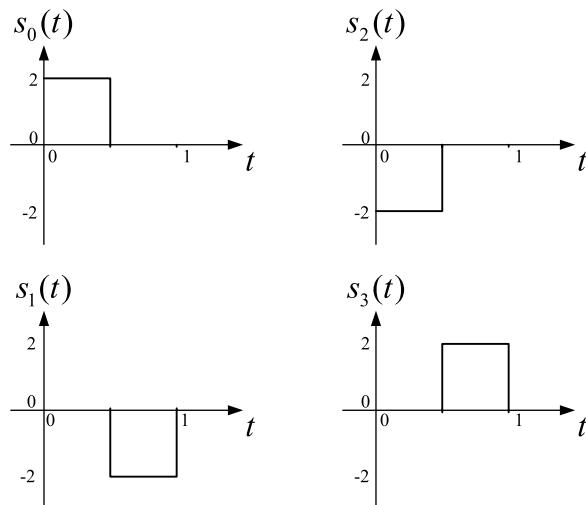


Figure 4.20 Collection of $M = 4$ waveforms employed in a digital transceiver.

5. *Error analysis:* A ternary communication system transmits one of three equiprobable signals $s(t)$, 0, or $-s(t)$ every T seconds. The received signal is $r(t) = s(t) + z(t)$, $r(t) = z(t)$, or $r(t) = -s(t) + z(t)$, where $z(t)$ is white Gaussian noise with $E[z(t)] = 0$ and $R_z(\tau) = E[z(t)z^*(t)] = 2N_0\delta(t - \tau)$.

The optimum receiver computes the correlation metric:

$$U = \operatorname{Re} \left[\int_0^T r(t)s^*(t)dt \right] \quad (4.109)$$

and compares U with a threshold A and a threshold $-A$. If $U > A$, the decision is made that $s(t)$ was sent. If $U < -A$, the decision is made in favor of $-s(t)$. If $-A < U < A$, the decision is made in favor of 0.

- (a) Suppose we define $E = \frac{1}{2} \int_0^T s(t)s^*(t)dt$ and $N = \operatorname{Re} \left[\int_0^T z(t)s^*(t)dt \right]$, where N is a Gaussian random variable. Solve for the mean and variance of N .

- (b) Determine the three conditional probabilities of error: P_e given that $s(t)$ was sent, P_e given that $-s(t)$ was sent, and P_e given that 0 was sent.

HINT: Solve in terms of E , N , and A , as well as express the answer using the Q -function:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-u^2/2} du \quad (4.110)$$

- (c) Determine the average probability of error \bar{P}_e as a function of the threshold A , assuming that the three symbols are equally probable a priori.
(d) Determine the value of A that minimizes \bar{P}_e .

HINT: Use *Leibnitz Rule*:

$$\frac{d}{dx} \left(\int_{f(x)}^{\infty} g(a)da \right) = -\frac{df}{dx} g(f(x)) \quad (4.111)$$

6. *Maximum a posteriori detection:* Suppose our transmitter is operating in a zero-mean additive white Gaussian noise (AWGN) channel of variance σ_n^2 such that the received signal vector is $\mathbf{r} = \mathbf{s}_i + \mathbf{n}$, for $i = 0, 1, 2, 3$.

- (a) What is the mathematical expression for the conditional probability density function (CPDF) $p(\mathbf{r}|\mathbf{s}_i)$?

HINT: Use the mathematical expression for a zero-mean Gaussian random variable.

- (b) What is the general expression for the *maximum a posteriori* (MAP) detector?

HINT: Recall hybrid presentation of Bayes Rule with respect to the usage of both PDFs and probabilities.

- (c) Using the answer from part (a), give the correlator realization of this receiver structure based on part (b) when the probabilities of the transmitted signals are $P(s_0) = P(s_1) = 2P(s_2) = 3P(s_3)$. Draw the receiver structure.

HINT: Use log-likelihood functions with the expression from part (a).

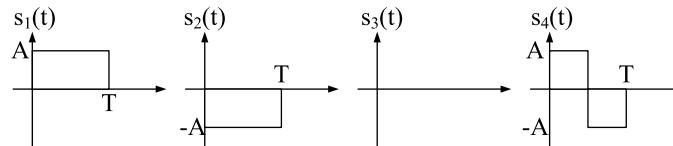


Figure 4.21 Four signaling waveforms employed in binary pulse amplitude modulation scheme.

7. *Modulation and power efficiency:* Suppose we are given the four waveforms shown in Figure 4.21 as candidates for a binary pulse amplitude modulation (PAM) modulation scheme. Let us consider the signal pairs $P1 : (s_1(t), s_2(t))$, $P2 : (s_1(t), s_3(t))$, and $P3 : (s_1(t), s_4(t))$.

- Determine the minimum Euclidean distance, d_{\min}^2 , for signal pairs $P1$, $P2$, and $P3$.
- Calculate the average energy per bit, \bar{E}_b , for signal pairs $P1$, $P2$, and $P3$. Note that the pair of signals represent either a binary 1 or a binary 0, each of which has an equal probability of occurring in the transmission.
- Of the three signal waveform pairs, which possesses the best power efficiency ϵ_p and explain in your own words why this is the case.

8. *Modulation and power efficiency:*

- Calculate the average symbol energy, \bar{E}_s , for the tertiary signal constellation shown in Figure 4.22. Assume all three symbols are equiprobable.
- Calculate the minimum Euclidean distance, d_{\min}^2 , for the tertiary signal constellation shown in Figure 4.22.
- Suppose that each signal constellation point in Figure 4.22 requires $b = 2$ bits in order to achieve unique representation. Find the power efficiency, ϵ_p .

9. *Modulation and power efficiency:*

- Given the four waveforms in Figure 4.23, compute the Euclidean distance between all possible pairs of waveforms:

$$d_{ij}^2 = \int_0^2 (s_i(t) - s_j(t))^2 dt, \quad i \neq j \quad (4.112)$$

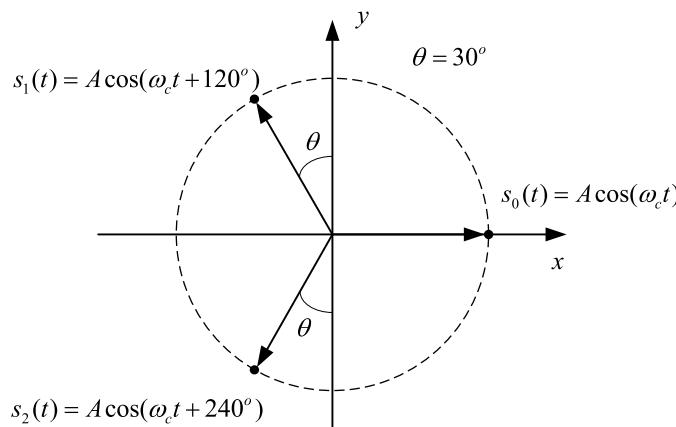


Figure 4.22 Tertiary signal constellation.

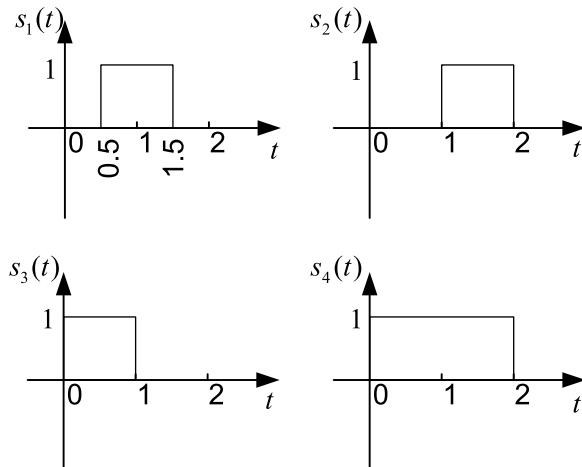


Figure 4.23 Four potential waveforms available for modulation.

- (b) Suppose that you are designing a modulator that maps $b = 1$ bit into a waveform and that you are given the responsibility of choosing two waveforms from the available four waveforms shown in Figure 4.23. To achieve the best possible result when operating in an additive noise channel, which two waveforms would you choose? Justify your answer. HINT: Use your results in part (d).

References

- [1] Shannon, C. E., "Communication in the Presence of Noise" *Proceedings of the Institute of Radio Engineers*, Vol. 37, No. 1, January, 1949. p. 1021.
- [2] Poor, *An Introduction to Signal Detection and Estimation*, Springer, 2010.
- [3] Wyglinski, A. M., "Physical Layer Loading Algorithms for Indoor Wireless Multicarrier Systems". PhD Thesis of McGill University. Montreal, Canada, 2004.
- [4] Proakis, J. and M. Salehi, *Digital Communications*, 5th Edition, McGraw-Hill, 2007.
- [5] Proakis, J. G., M. Salehi, and G. Bauch, *Contemporary Communication Systems Using MATLAB*, 2nd Edition. Brooks/Cole, 2003.
- [6] Sklar, B., *Digital Communications: Fundamentals and Applications*, 2nd Edition, Prentice Hall PTR, 2001.
- [7] Couch, W., *Digital and Analog Communication Systems*, 7th Edition, Prentice Hall, 2006.
- [8] Rice, M., *Digital Communications: A Discrete-Time Approach*, 3rd Edition, Pearson/Prentice Hall, 2009.
- [9] Barry, J. R., D. G. Messerschmitt, and E. A. Lee, *Digital Communication*, 3rd Edition, Springer, 2003.
- [10] Madhow, U., *Fundamentals of Digital Communication*, Cambridge University Press, 2008.
- [11] Pursley, M. B., *Introduction to Digital Communications*, Prentice Hall, 2004.
- [12] Hsu, H. P., *Schaum's Outline of Theory and Problems of Analog and Digital Communications*, 2nd Edition, McGraw-Hill, 2002.

Basic SDR Implementation of a Transmitter and a Receiver

This chapter will serve as an introduction to understanding how communication algorithms operate under real-world conditions using software-defined radio (SDR) technology. This chapter will also provide the foundation for subsequent digital communication algorithms and their implementation in SDR, which will be discussed later in this book. In this chapter, we will get our first hands-on experience with SDR by implementing a bare bones communication system in Simulink. For information about the theoretical background for the experiments in this chapter, please refer to Section 4.1 through Section 4.3, which cover the concepts of encoding, modulation, and probability of bit error. For information about MATLAB and Simulink products, please refer to Appendix A.

5.1 SOFTWARE IMPLEMENTATION

Before implementing any designs in SDR, it is imperative to first understand any design or implementation using a computer simulation package in order to better characterize the behavior of the implementation, as well as provide a set of benchmark results that can be used when constructing these prototypes in actual hardware. Furthermore, many communications engineers and researchers approach the implementation of a communication system design via a methodological sequence of computer simulation followed by hardware implementation.

In order to get things started, let us first open the `txbasic.mdl` file,¹ as shown in the Figure 5.1, which contains a very basic digital transmitter model in Simulink. Note that there is no channel in this model, and thus there are no impairments introduced to the data stream being produced by this transmitter. Random “1” and “0” values are generated by the transmitter, then sent using 36 bit long frames, which are processed by an encoding scheme to make them more robust to noise. Following the encoding, this bit stream is then modulated to a 16 QAM symbol format prior to transmission.

As mentioned before, although we can see all of the blocks of this system, it is not entirely complete. Specifically, the user-defined functions are left empty, and throughout this chapter we will proceed to fill in the remaining blocks in order to complete this system as part of the experiment defined in this chapter. Before we start writing code, we should examine the assigned parameters of the defined blocks in the mask parameters and attempt to understand why we have selected those values. For example, Figure 5.2 shows the mask parameters of the *Bernoulli Binary Generator* block, which can be accessed by double-clicking the block.

1. The experiment files used in this book are available at [1].

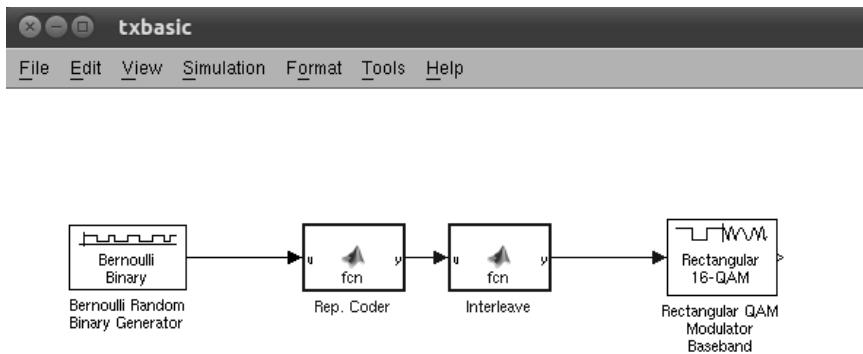


Figure 5.1 A very basic digital transmitter model in Simulink.

When working in Simulink, the format of the data is extremely important. For example, if a $[36 \quad 1]$ signal is sent, to an input port expecting a $[37 \quad 1]$ signal, as highlighted in Figure 5.2, not only will the model not run, the software will produce numerous error messages, as shown in Figure 5.3. Understanding how each block affects the data format will greatly simplify the design process of a communication system when we use Simulink.

5.1.1 Repetition Coding

One of the key building blocks of any communication system is the forward error correction (FEC), where redundant data is added to the transmitted stream to make

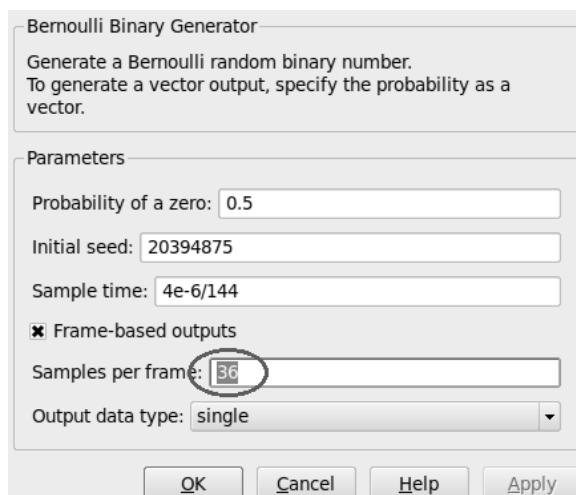


Figure 5.2 Mask parameters of the *Bernoulli Binary Generator* block, where the samples per frame is 36.

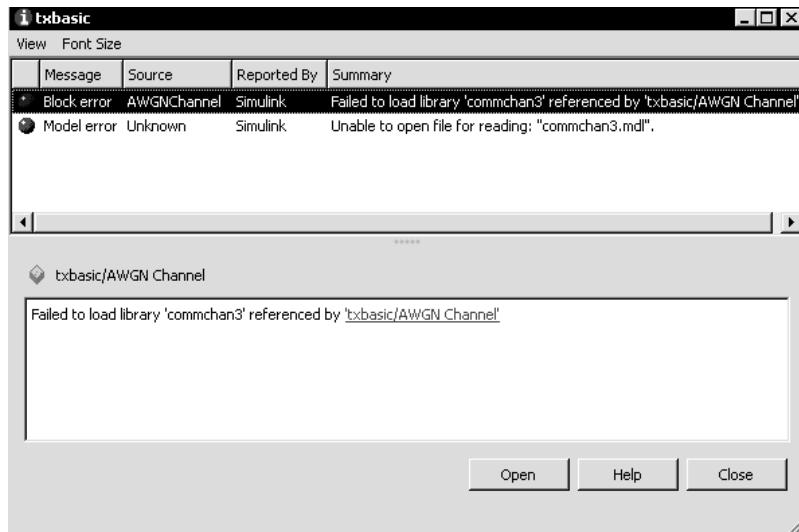


Figure 5.3 Error messages from Simulink. By clicking on the error message, a detailed description of the source of this error will be provided in the lower window to help with the debugging process.

it more robust to channel errors. There are many types of FEC techniques, such as the *repetition coding* approach, where each transmitted bit is repeated multiple times. For example, Problem 1 in Section 5.5 introduces a repetition code with repetition factor $R = 5$ in a binary pulse amplitude modulation (BPAM) transmission system. In this section, we will explore together one technique for combating the introduction of errors to data transmissions by implementing a simple repetition coder (repetition factor $R = 4$). So what does it mean by a repetition coder with repetition factor $R = 4$? A simple definition is that if a “0” symbol is to be transmitted, this “0” symbol will be repeated four times by the repetition coder, such that the output would be “0000”.

Let us first start by double clicking on the repetition coder block, which will result in a MATLAB function block editor opening, in which we can write customized MATLAB code. As mentioned previously, setting break points is a great way for understanding and debugging M-files. For more information about break points and how they can be used to debug and evaluate the code, please refer to Appendix A.



The *repmat* function in MATLAB can be used to realize a simplistic repetition coding scheme. For example, to repeat a vector u for 4 times, the following expression can obtain this result:

```
y=repmat(u,4,1);
```



What are the trade-offs to consider when choosing between a high or a low repetition factor?

5.1.2 Interleaving

A repetition code is one of several useful tools for a communication systems engineer in order to enhance a robust data transmission. However, it is sometimes not enough, since it does not address the issue when a large quantity of data is corrupted in contiguous blocks. For instance, if a transmitter sends the data stream “101101”, a repetition coder with a repetition factor of 4 will yield:

$$11110000111111100001111,$$

where each input bit is repeated four times. While this encoding scheme may appear robust to error, it is still possible during a data transmission that a significant noise burst occurs over many consecutive bits, corrupting numerous binary digits in the transmission, and yields the following outcome:

$$111100 \text{ ----- } 1100001111,$$

where some of the original data is completely irretrievable.



Why is it that even with repetition coding, our data transmission can still be severely affected? What could be done to make it even more robust?

Interleaving is an approach where binary data is *reordered* such that the correlation existing between the individual bits within a specific sequence is significantly reduced. Since errors usually occur across a consecutive series of bits, interleaving a bit sequence prior to transmission and de-interleaving the intercepted sequence at the receiver allows for the dispersion of bit errors across the entire sequence, thus minimizing its impact on the transmitted message. A simple interleaver will mix up the repeated bits to make the redundancy in the data even more robust to error. It reorders the duplicated bits amongst each other to ensure that at least one redundant copy of each will arrive even if a series of bits are lost. For example, if we use an interleaving step of 4, it means we reorder the vector by index [1, 5, 9, ..., 2, 6, 10, ...]. As a result, running “11110000111111100001111” through such an interleaver will yield the following output:

$$101101101101101101101101.$$

The interleaving step can be any of the factoring numbers of the data length. However, different mixing algorithms will change the effectiveness of the interleaver.



The *reshape* function in MATLAB can be used to realize the interleaving.

Once we have implemented the interleaver, let us combine the repetition coder and the interleaver into a single FEC sub-system. Although the simple interleaving technique introduced above is sufficient for our implementation, there are

various other forms of interleaving, that we will investigate in the next two subsections.

5.1.2.1 Block Interleaving

The first approach to interleaving is to employ a block interleaver, as shown in Figure 5.4. Block interleaving is one method for reordering a bit sequence, where $N \times M$ bits fill an N column by M row matrix on a column basis, and then each resulting row is concatenated with each other in serial and outputted from the interleave block. At the transmitter side, the block interleaver is loaded column by column with N codewords, each of length M bits. These N codewords are then transmitted row by row until the interleaver is emptied. Then the interleaver is loaded again and the cycle repeats. The main drawback of block interleavers is the delay introduced with each column-by-column fill of the interleaver [2].

5.1.2.2 Convolutional Interleaving

Another approach to interleaving is to employ a convolutional interleaver [3], as shown in Figure 5.5. At the transmitter, the bit sequence is shifted into a bank of N registers, each possessing an increasing amount of buffer memory. The bits in the bank of registers are then recombined via a commutator and transmitted across the channel. At the receiver, the reverse process is performed in order to recover the original sequence. Compared with block interleavers, convolutional interleavers reduce memory requirements by about one-half [4]. However, the delay problem associated with the initial fill still exists.

5.1.3 BER Calculator

After examining several techniques for protecting the reliability of a data transmission subject to interference and noise, we now need an approach to measure

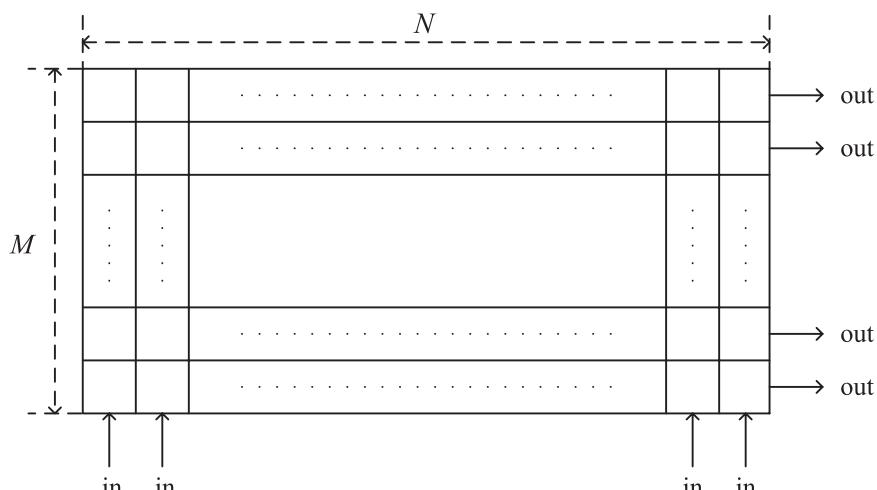


Figure 5.4 Schematic of a block interleaver.

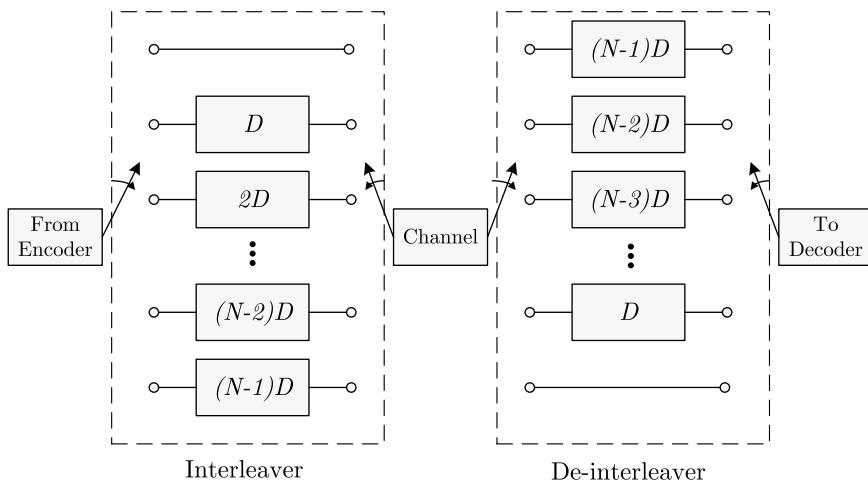


Figure 5.5 Schematic of a convolutional block interleaver (from [3]).

how well these techniques perform quantitatively. Referring back to Chapter 4, we saw that bit error rate (BER) is a commonly used metric for the evaluation and comparison of digital communication systems. One straightforward way of calculating the BER is to count the number of received bits of information and then determine which ones are received in error. In other words, the ratio of bit errors to the total number of bits received can provide us with an approximate BER calculation. Note that the more bits that are received, the more accurate this metric becomes.

In this model, the alignment block is accounting for the delay of the signal as it propagates through the other blocks. As the model is configured, the BER should be zero. Check this by connecting both the input ports of the alignment block to the initial bit sequence.

5.1.4 Receiver Implementation over an Ideal Channel

The next step is to implement the receiver for the transmitter we have just constructed. We can now remove the AWGN *Channel* block for this step and assume this is an ideal channel. Therefore, there is no noise and therefore no impairments to account for in the data transmission. The receiver design is simply the inverse of the transmitter design, as shown in Figure 5.6. To test out our implementation, let us feed the decoded signal into the BER calculator. Since we are using an ideal channel, we should expect that the resulting BER is equal to zero because no interference nor distortion is present during transmission.



1. Note the two persistent variables - what does persistent mean?
2. Once you believe you have achieved success, intentionally invert one of the bits. Does your BER calculated behave as expected?
3. What effect might different signal constellations have on the bit error rate?

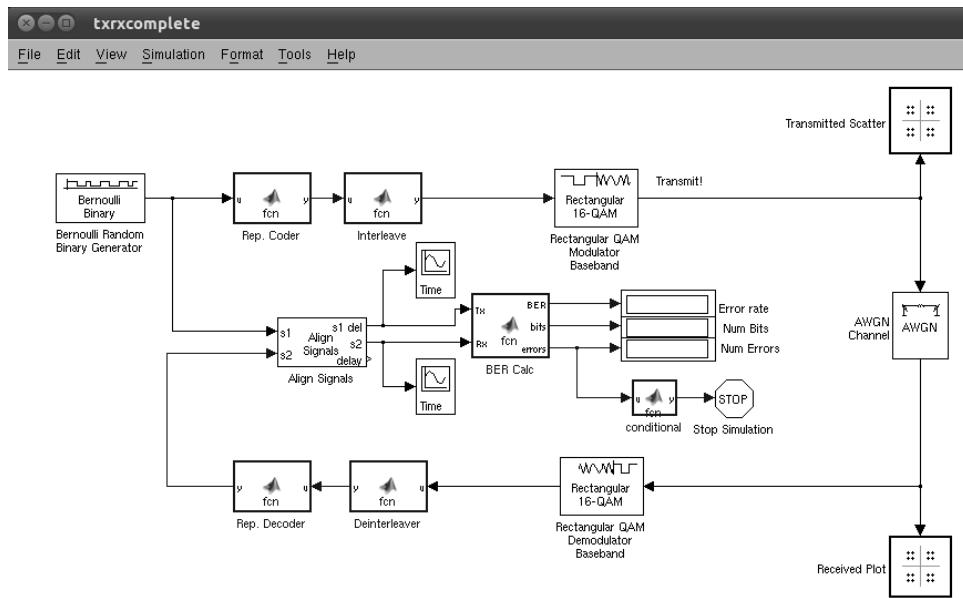


Figure 5.6 Approximate appearance of final Simulink transceiver model. This model implements a digital communication system using 16-QAM and repetition coding. We can simulate the communication channel by adding the *AWGN Channel* block.

- When configuring the user functions, please remember that formatting is important. We may need to set the expected port sizes in the model explorer.
- By default, the output of a user-defined function is sample based. Certain blocks may expect frame based inputs. We can use the *To frame* block to rectify this. A double-lined connection indicates the signal is frame based; a single line connection indicates the signal is sample based.

5.2 USRP HARDWARE IMPLEMENTATION

In this section, we will deal with several fundamental but important concepts in digital communication system designs via SDR using the ethernet-based USRP platforms.² First is frequency offset compensation. Due to the imperfection of devices, frequency offset exists in every communication system, which prevents the ideal signal reception. Therefore, users need to find out the amount of frequency offset and try to compensate for it. The other important concept is in-phase/quadrature (I/Q) representation of the signals. Suppose we have a signal $s(t)$ with the phase $\phi(t)$, we can represent this signal using its I and Q component, namely,

$$s_I(t) = s(t)\cos(\phi(t)) \quad (5.1)$$

and

$$s_Q(t) = s(t)\sin(\phi(t)) \quad (5.2)$$

- Simulink and MATLAB support most of the Ethernet-based USRP platforms, including USRP2, USRP N210 rev2 and rev 4.

Since the default data type of USRP is I/Q, understanding how I and Q data flows in USRP will greatly help us with the SDR design and implementation. In addition, we will be familiarized with some of the useful Simulink tools for digital communication system.

5.2.1 Frequency Offset Compensation

Most of the signal processing techniques introduced so far happen in baseband, where the center frequency is around 0 Hz. However, in order to enable radio transmission or broadcasting, baseband signals need to be modulated to a higher frequency band, which is usually referred to as a radio frequency (RF) band. The carrier wave used in this modulation process is provided by a circuit called oscillator circuit, where oscillator is the main component. Figure 5.7 shows a photograph of a local oscillator on an SDR PCB. However, oscillator crystals are not perfect, since they do not oscillate at exactly the specified frequency. To measure this imperfection, oscillators and other frequency control devices specify their peak variation in units of parts per million (ppm). When operating on a certain frequency, it will lead to a peak frequency variation, which can be calculated using the following expression:

$$\Delta f = \frac{f \times \text{ppm}}{10^6} \quad (5.3)$$

where ppm is the peak variation (expressed as \pm), f is the center frequency (in Hz), and Δf is the peak frequency variation (in Hz). Due to imperfections and tolerances in the manufacturing process, oscillator crystals typically have inaccuracies of about 20 or 50 parts per million (ppm), which are guaranteed by the manufacturer.



Figure 5.7 A 10.7 Mhz local oscillator on the SDR PCB (from [5]), which is used to generate the carrier wave in order to modulate signals from baseband to RF band.

Inaccuracies in the oscillator crystal frequency cause errors in the RF carrier frequency. Note that even relatively small differences in frequency between the transmitter and the receiver can prevent the receiver from correctly decoding the transmission. It is also worth noting that oscillator circuits provide the clock source for the digital electronics. In particular, the clocks for the analog-to-digital and digital-to-analog converters affect their sampling rates. Therefore, accurate oscillators are essential for both the RF and digital systems.

The GNU Radio FAQ [6] states that the “USRP2 reference clock stability” is “about 20 ppm unless you lock to an external reference.” Note that the USRP2 has an input connector for an external frequency reference, which can provide a highly precise timing signal for USRP2. Suppose that we select a carrier frequency of 2.45 GHz for our USRP2. An offset of 20 ppm may sound small, but it is quite significant at high frequencies. According to (5.3), the peak frequency variation in this scenario can be:

$$f = \frac{2.45 \cdot 10^9}{10^6} \cdot 20 = 49,000 \text{ Hz} \quad (5.4)$$

In other words, when using a carrier frequency of 2.45 GHz, the worst case frequency offset could be as much as about 50 kHz. Note that if there are two USRP2s trying to communicate with each other, where one has an offset of -50kHz and the other has an offset of +50kHz, the overall difference in frequency will be 100kHz.

It is necessary to compensate for this frequency offset in order to achieve successful digital communication between the USRPs. For example, we have measured some of the USRP2s to have frequency offsets of as much as 45kHz ($45\text{kHz}/2.45\text{GHz} = 18\text{ppm}$). With frequency offset compensation, the USRP2 works fine, but without it the digital communication system will not work at all.

In order to devise a software-based approach for frequency offset compensator, let us perform the following steps to measure and compensate this phenomenon. First, arbitrarily select a carrier somewhere within either of the supported bands of the XCVR2450 daughterboard (2.4 to 2.5 GHz and 4.9 to 5.9 GHz). For example, suppose we choose 2.45 GHz. Note that it is not recommended to choose this exact frequency since other individuals using wireless equipment including software-defined radio might be transmitting all at the same time and frequency.

Then, let us run `siggen.mdl` on one USRP and `observeFFT.mdl` on another USRP, as shown in Figure 5.8 and Figure 5.9. By setting the Center frequency parameter in both *SDRu Transmitter* and *SDRu Receiver* blocks, we can use the carrier frequency selected in the previous step, as shown in Figure 5.10(a) and Figure 5.10(b). Please note that these two blocks can be obtained by typing `sdrulib` in the command window.

As long as these two models are running, we can now use the FFT plot from *spectrum scope* to measure the frequency offset between the received signal and the desired carrier. In order to have a better observation from the scope, we can right click on the plot from the scope, and choose “Autoscale”. An example of the output from the FFT scope is shown in Figure 5.11(a). Once the value of the frequency offset is obtained, we can proceed to stop the models, and adjust the carrier frequency of the transmitter by the amount of this offset. For example, by visually

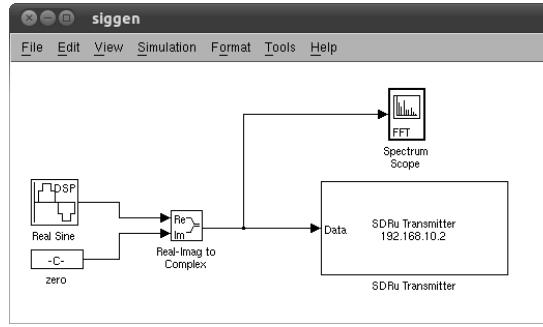


Figure 5.8 The structure of siggen.mdl. In this model, since the *SDRU Transmitter* block requires the complex input, the *Real-Imag to Complex* block converts real and imaginary inputs to a complex-valued output signal.

inspection, we determine the value at which the peak occurred is f Hz, as shown in Figure 5.11(b). Then, we can set the center frequency of the transmitter to be $2.45 \text{ GHz} + f \text{ Hz}$. By iteratively adjusting the carrier frequency and observing the result, we should be able to determine the carrier frequency that we will use on the transmitter in order to observe the correct carrier frequency on the receiver.

Keep in mind that both the transmitter and the receiver have frequency offsets. For example, 2.45 GHz on the receiver is somewhere within $2.45 \text{ GHz} \pm 20\text{ppm}$ and is not exactly 2.45 GHz. When tuning the transmitter to eliminate the frequency offset on the receiver, we are essentially compensating for the offset between them. It is the relative offset between the transmitter and receiver that matters. Note that we can adjust either the transmitter or the receiver (or both) to compensate for the offset. The basic idea is to get the two devices synchronized in carrier frequency with each other. If there are more than two USRPs communicating with each other, it is important to make sure that each pair of transmitters and receivers are all tuned to each other.

5.2.2 Finding Wireless Signals: Observing IEEE 802.11 WiFi Networks

Let us now use the knowledge that we have so far picked up in this chapter to look at a commercially available network. Specifically, we can use USRP to observe the activities of electromagnetic spectrum of commercial wireless local area networks (WLANs) within the geographical vicinity. Most high population density areas em-

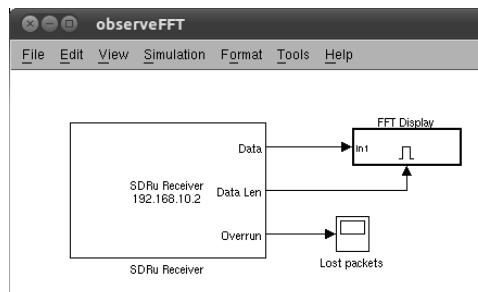


Figure 5.9 The structure of observeFFT.mdl. In this model, *FFT Display* is an *enabled subsystem*. We use the *Data Len* parameter to qualify the execution of this part. Specifically, when *Data Len* contains a zero value, there is no data, so *FFT Display* cannot be enabled.

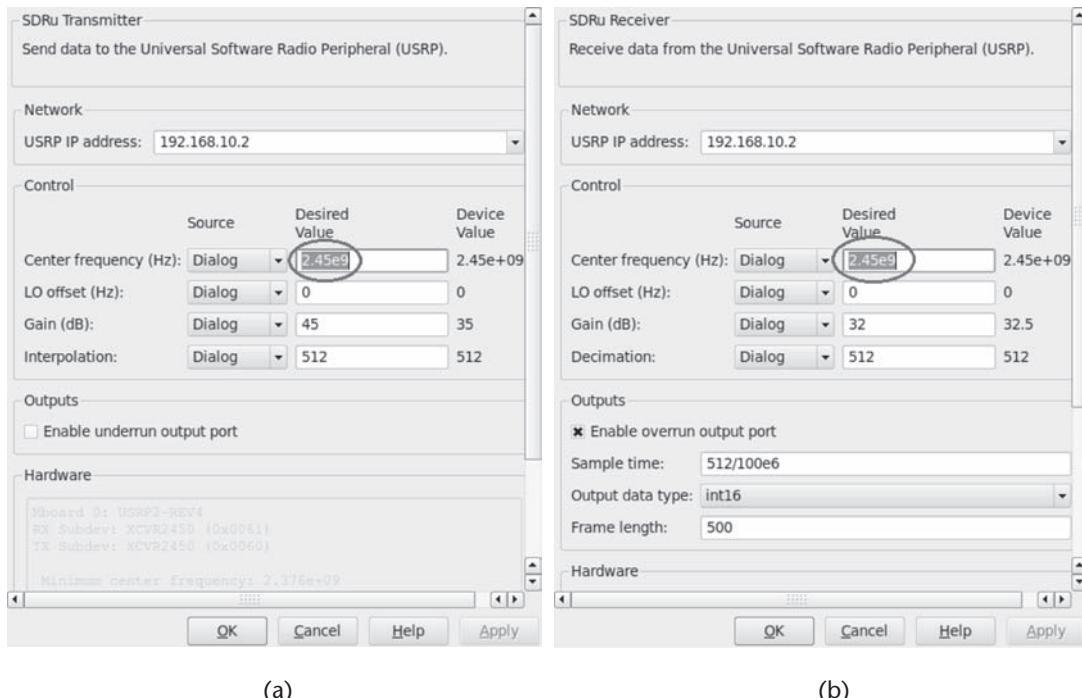


Figure 5.10 Set Center frequency of the *SDRu Transmitter* block and the *SDRu Receiver* block. (a) Set Center frequency of the *SDRu Transmitter* block to be 2.45 GHz; (b) Set Center frequency of the *SDRu Receiver* block to be 2.45 GHz.

ploy numerous wireless communication networks for a variety of applications, such as the wireless network at school, company, or home. Consequently, we can use the USRP experimentation platform to plot their magnitude spectrum.

IEEE 802.11 [7] is one type of WLAN standard that possesses a list of carrier frequencies for a collection of Wi-Fi channels. For example, The IEEE 802.11 standard defines Channel 1 of the 2.4 GHz band to be centered at 2.412 GHz. Now, specify the Center frequency parameter of *SDRu Receiver* with this carrier frequency, and use the *Spectrum Scope* to plot their magnitude spectrums. Since the XCVR2450 daughtercard supports the 5 GHz band as well, use the *Spectrum Scope* to plot spectrum in the 5 GHz band.



1. It is recommended to turn on “Autoscale” (right click the scope display) and adjust the amplitude, since the peaks of these wireless signals could be rather low.
2. It might be useful to change the “Decimation” parameter of *SDRu Receiver* to adjust the frequency resolution for a better graph of the spectrum.
3. In order to capture the full bandwidth of a Wi-Fi signal, we may need to decrease the downsampling of the received USRP data.

5.2.3 USRP In-phase/Quadrature Representation

Understanding how a signal is represented can greatly enhance our ability to analyze and design baseband digital communication systems. As mentioned in

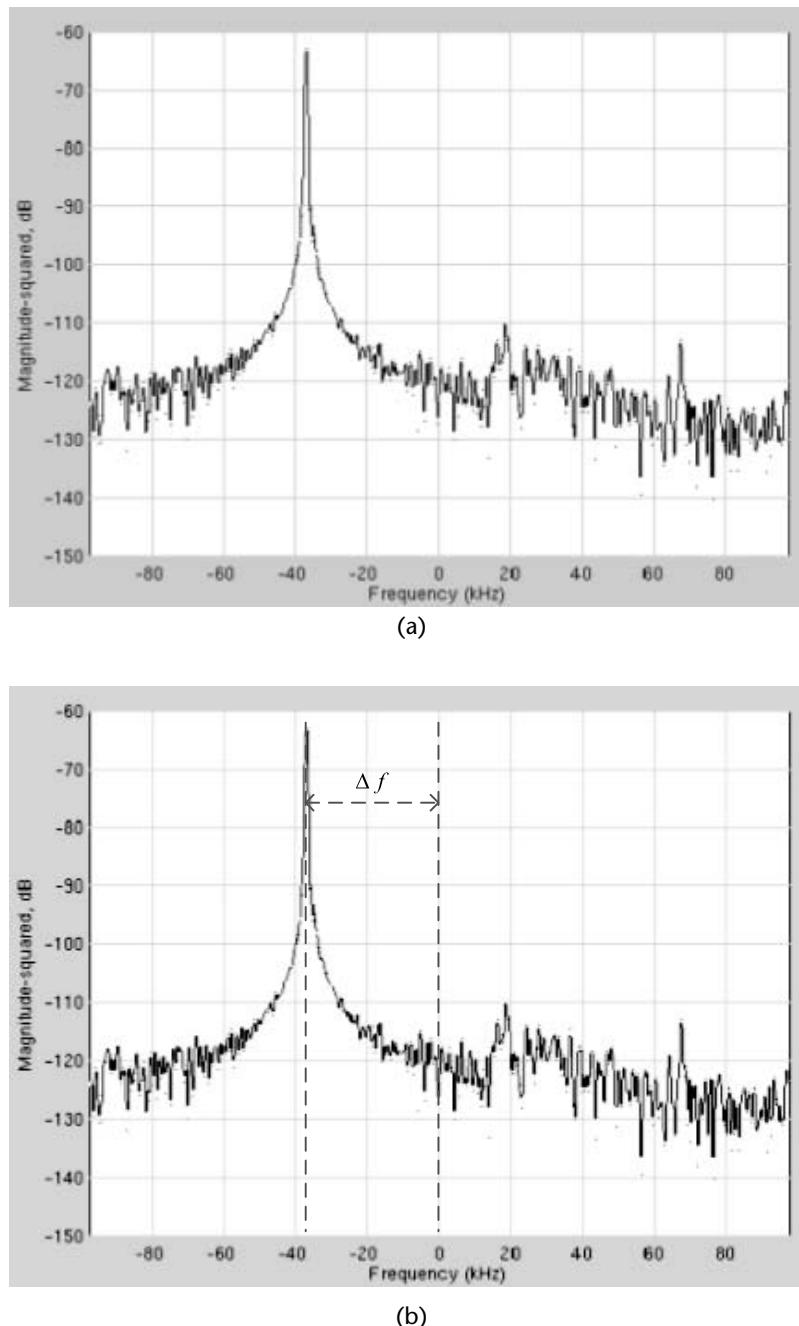


Figure 5.11 Determine the offset f based on the output from the FFT scope. (a) An example of the output from the FFT scope; (b) by visual inspection, we can determine the offset f , the value at which peak occurred.

Section 4.2.3, a bandpass signal can be represented by the sum of its in-phase (I) and quadrature (Q) components. Moreover, the data input to the USRP board is complex, which includes I and Q . Since I and Q play an important role in digital communications, we are going to observe the I and Q data on transmitter and receiver sides.

5.2.3.1 Observing In-phase and Quadrature Data

For the transmitter side, let us use `siggen.mdl` from Section 5.2.1. For the receiver side, let us use the `observeiq.mdl` model, where the main subsystem of this model is shown in Figure 5.12. The model consists of two *Vector Scope* blocks that are attached to the real and imaginary output. With these two scopes, we can observe the real and imaginary parts of a complex number separately.

By changing the two blocks attached to the *Real-Image to Complex* block, we can specify the sources of `siggen.mdl` using six sets of values defined in Table 5.1. For each source input, let us record the corresponding output in `observeiq.mdl` by saving the plots of *Vector Scope*. For example, Figure 5.13 shows the sample plots from the two scopes when both I and Q inputs are sine waves.



Is the output the same as the input you have specified?

5.2.3.2 Mathematical Derivation

The behavior of the in-phase and quadrature data is closely related to the structure of USRP board, especially the digital up converter (DUC) and digital down converter (DDC), which are used by the field-programmable gate array (FPGA) to convert between baseband and intermediate frequency (IF).

At the transmitter path, a baseband I/Q complex signal is sent to the USRP board. The digital up converter will interpolate the signal, upconvert it to the IF band, and finally send it through the DAC. The structure of DUC is shown in Figure 5.14, which is named as *complex multiplier*. In this structure, the numerically controlled oscillator (NCO) is a digital signal generator which creates a synchronous (i.e., clocked), discrete-time, discrete-valued representation of a waveform, usually sinusoidal. NCOs are often used in conjunction with a digital-to-analog converter (DAC) at the output to create a direct digital synthesizer (DDS) [8].

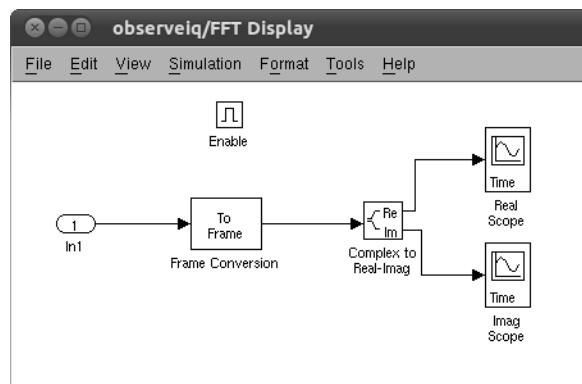
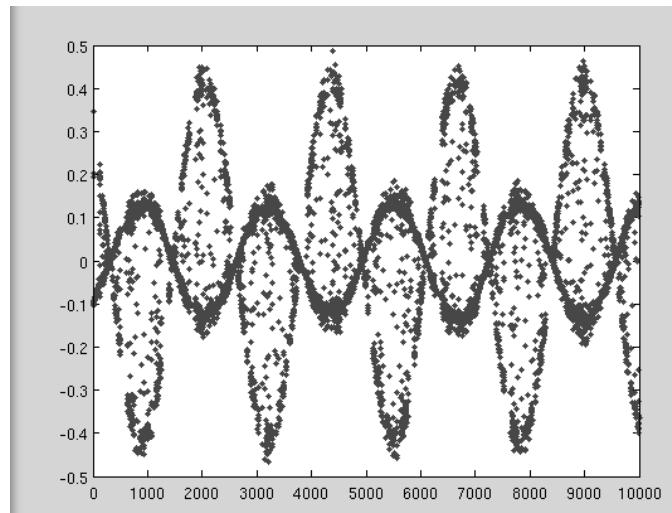


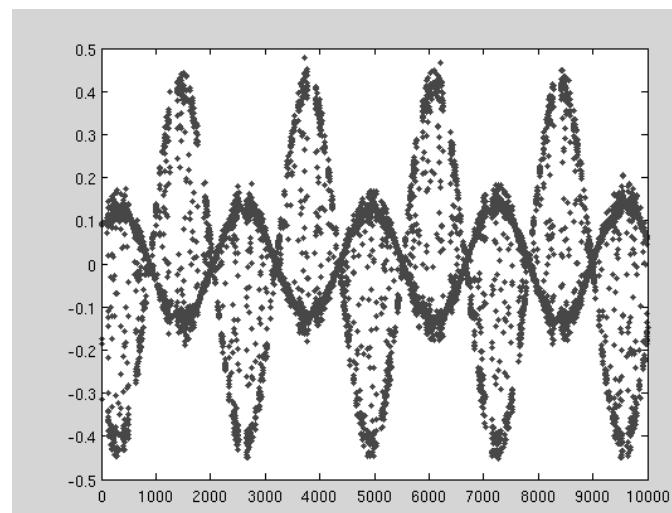
Figure 5.12 The structure of main subsystem of `observeiq.mdl`. Since the *SDRu Receiver* block produces the complex output, the *Complex to Real-Image* block converts the complex-valued signal to real and imaginary components.

Table 5.1 The input sources of `siggen.mdl`

<i>Set Index</i>	<i>Real</i>	<i>Imaginary</i>
1	0	0
2	0	1
3	1	0
4	0	Sinc
5	Sinc	0
6	Sinc	Sinc



(a)



(b)

Figure 5.13 Sample plots from the two scopes when both *I* and *Q* inputs are sine waves. (a) Output from *Real Scope*; (b) output from *Img Scope*.

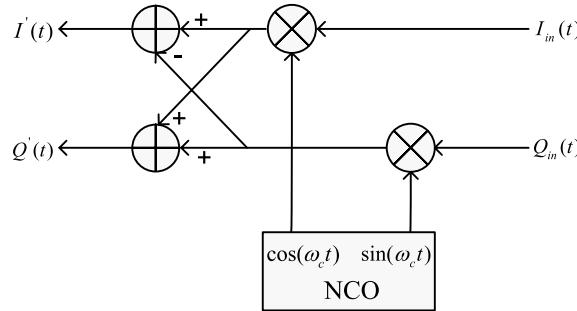


Figure 5.14 The structure of digital up converter on transmitter path, where a baseband I/Q complex signal $I_{in}(t)$ and $Q_{in}(t)$ is upconverted to the IF band by a carrier frequency of ω_c .

At the receiver path, the standard FPGA configuration includes digital down converters (DDC) implemented with a 4-stage cascaded integrator-comb (CIC), as shown in Figure 5.15. First, the DDC down-converts the received signal from the IF band to the base band. Next, it decimates the signal so that the data rate can be adapted by the Gigabit Ethernet and is reasonable for the processing capability of computers. The complex input signal (IF) is multiplied by the constant frequency (usually the IF) exponential signal. The resulting signal is also complex, and centered at 0. Then the signal is decimated with a factor M . The decimator can be treated as a low pass filter followed by a down sampler. In this structure, the coordinate rotation digital computer (CORDIC) [9], also known as the digit-by-digit method and Volder's algorithm, is a simple and efficient algorithm to calculate hyperbolic and trigonometric functions. It is commonly used when no hardware multiplier is available as the only operations it requires are addition, subtraction, bitshift and table lookup.

Given this insight on the structure of DUC and DDC, we can now perform the following mathematical derivations:

5.3 OPEN-ENDED DESIGN PROJECT: AUTOMATIC FREQUENCY OFFSET COMPENSATOR

5.3.1 Introduction

Starting from this chapter, there will be several open-ended design projects involving the implementation of an advanced digital transmission/reception system using SDR. These projects are related to what we have learned and done in each chapter, but require additional thinking and investigation. For each design project, a potential solution will be provided. However, there is no single solution for these projects, so please do not feel constrained to this solution. You are highly encouraged to propose your own solution and try it out. Each reader is expected to come up with his/her own innovative design.



The trigonometric identities are needed when expressing the I and Q data. Please refer to Appendix E.

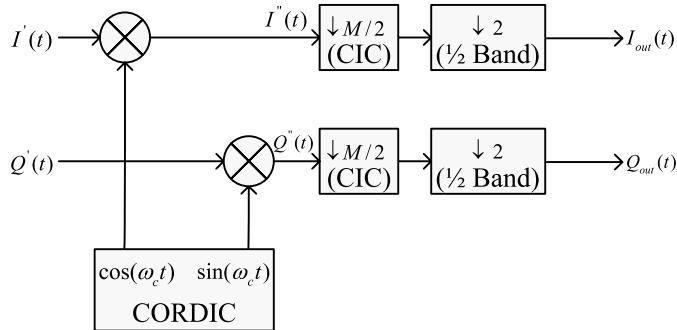


Figure 5.15 The structure of digital down converter on receiver path, where the input RF signal $I'(t)$ and $Q'(t)$ is first down-converted to the IF band by a carrier frequency of ω_c , and then decimated with a factor M .

1. In Figure 5.14, suppose the input I/Q complex signal is $I_{in}(t)$ and $Q_{in}(t)$, the carrier frequency is ω_c , what is the output of the DUC, denoted as $I(t)$ and $Q(t)$?
2. In Figure 5.15, suppose the input I/Q complex signal is $I(t)$ and $Q(t)$, which has been expressed in the previous step. Also assume that the carrier frequency has a little offset ω compared to the DUC, so it is $\omega_c + \omega$. What is the output after CORDIC and before CIC? Please express them as $I''(t)$ and $Q''(t)$.
3. In Figure 5.15, for simplicity, the rest part after CORDIC, including CIC and 1/2 Band, can be viewed as a low pass filter. $I''(t)$ and $Q''(t)$ will go through the low pass filters, where the double-frequency component like $2\omega_c$ will disappear. What is the output of the low pass filters, denoted as $I_{out}(t)$ and $Q_{out}(t)$?
4. Given $I_{out}(t)$ and $Q_{out}(t)$, can you accurately recover $I_{in}(t)$ and $Q_{in}(t)$? If yes, mathematically give the relationship between $I_{in}(t)$, $Q_{in}(t)$ and $I_{out}(t)$, $Q_{out}(t)$. If no, give the reasons.

Q

5.3.2 Objective

Although every communications system will need to compensate for frequency offsets introduced by nonideal RF crystals, it is difficult and often impractical to compensate for this manually. Full communications systems often include automatic frequency offset compensation that can correct this nonideality without end-user intervention. This feature is essential for devices that must communicate with other devices with unknown or changing frequency offsets. The objective of this project is to design and implement a software-defined radio (SDR) communication system capable of automatically calculating the frequency offset between two USRP platforms.

In Section 5.2.1, we have already found the frequency offset of two USRPs manually, namely, by comparing the FFT of the transmitted and received signals. In this project, let us discover this offset in an automatic way. In other words, in the receiver model, the only thing we need to do is to hit the “start simulation” button, which will result in this model providing us with the value of the frequency offset.

5.3.3 Theoretical Background

Before applying any method to USRP boards, it is highly recommended to test the method with Simulink-only model without the USRP transmitter and receiver blocks and see whether it works. This is used to examine the approach taken under ideal conditions before actually testing it in the lab setting.



Although we are required to find the frequency offset of two USRP boards, we are actually trying to find the frequency offset of a received signal compared to a transmitted signal.

In hardware implementation, the frequency offset is introduced by the real-world communication channel. While in the Simulink-only model, we can introduce the frequency offset by *Phase/Frequency Offset* block, as shown in Figure 5.16. Let us specify a frequency offset in this block, and see whether the Simulink model can give us the number we have specified.

The solution we provide here is based on the theory that raising the received signal to the power of M removes the modulation from the signal and creates a tone at $M - f$, where f is the frequency offset between the transmitted signal and the received signal. Specifically, if we take the square of a transmitted signal, the FFT of the received signal will be shifted double of the frequency offset. Assume the original transmitted signal is $x(t)$, and the original received signal is $y(t)$, whose FFT is $Y(f)$.

If the transmitted signal is squared, which is $x(t)^2$, then the FFT of the received signal becomes $Y(f - 2f)$. Before the square operation, the maximum magnitude of $Y(f)$ is located at 0 Hz, then after the the square operation, the maximum magnitude will move to $2f$ Hz. Therefore, if we can find the frequency which corresponds to the maximum magnitude of the new FFT, this frequency, when divided by 2, represents the frequency location of the offset. This is the estimated offset frequency. In Simulink, we can easily find the magnitude of FFT using the *Magnitude FFT* block.

However, unlike Section 5.2.1, we cannot read the frequency value directly from the FFT plot. Instead, we should employ an automatic way of finding it out.

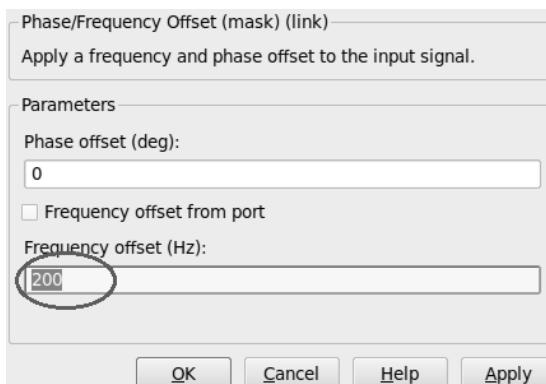


Figure 5.16 The frequency offset of the Simulink model can be introduced by filling out the **Phase offset** and **Frequency offset** parameters of the *Phase/Frequency Offset* block.

Therefore, next to *Magnitude FFT*, we need another important block, namely, *Maximum*. This block returns the value and index of the maximum elements of the input signal coming out from *Magnitude FFT* block. Indices are the locations of maximums, which should be equal to double of the offset, $2 \Delta f$, in this system.

Another key to the correct frequency representation is the conversion of the FFT indices found in the previous step into actual frequency units.

Suppose the FFT index found in the previous step is idx , the frame size used in this model is F_{size} the FFT length of *Magnitude FFT* block is N , then the actual frequency offset in Hz can be calculated by:



$$\Delta f = \frac{idx \times F_{size}}{N \times 2}$$

More generally, if we take the M -th order of the transmitted signal, the actual frequency offset in Hz can be calculated by:

$$\Delta f = \frac{idx \times F_{size}}{N \times M}$$

The techniques discussed above all happen on the receiver side. While on the transmitter side, we might need the following blocks in the model:

1. *Random Integer Generator*: Generate random uniformly distributed integers in the ranges of 0 and 1. These integers are served as the digital input to the communication system.
2. *Baseband Modulator*: Modulate the input signal. In order to transmit the digital input through the channel, modulation is required to convert the signal from digital domain to analog domain.
3. *Raised Cosine Transmit Filter*: Upsample and filter the input signal using a square root raised cosine FIR filter. As mentioned in Section 2.4, the purpose of using transmit filter is to eliminate the intersymbol interference.

But of course, we are not constrained to these blocks. Instead, we can use all the blocks available in Simulink. If necessary, we can also use MATLAB to implement our method.



1. When you are done with your design, compare your results here from what you have obtained in Section 5.2.1. Are they identical to each other? If not, which one do you think is more accurate? Why?
2. Using `siggen.mdl`, you can generate various frequency offsets over a wide frequency band. Record these offset values, as well as the results provided by your design, on which frequency band do they have a good match of each other? Can you explain the reason?

5.4 CHAPTER SUMMARY

This chapter serves as a starting point for the SDR experimentation. In this chapter, we first implement a bare bones communication system in Simulink, where we apply two error-correction techniques, namely, repetition coding and interleaving. Then, several experiments with USRP hardware make us familiar with this platform, which highlights the frequency offset and I/Q data of USRP. In the end, the open-ended design project helps to find out the frequency offset between two USRP boards.

5.5 PROBLEMS

1. *Repetition Coding* Suppose that a repetition code with repetition factor $R = 5$ is used in a binary pulse amplitude modulation (BPAM) transmission system, where a binary digit “0” is mapped to an amplitude value of “-1” while a binary digit “1” is mapped to an amplitude value of “+1”. Note that the bit period T_b is equal to the symbol period T_s . The BPAM transmission is sent across an AWGN channel, where the noise possesses a mean of $\mu = 0$ and a standard deviation of $\sigma = 0.1$. At the receiver, a limiter is employed to decode the BPAM transmission intercepted from the channel into the appropriate binary digits, where the decoding threshold received amplitude value is equal to 0.
 - (a) What is the probability that a single received symbol accidentally gets decoded incorrectly?
 - (b) What is the probability that repetition decoder yields an output that is incorrect?
 - (c) What would be the impact on the error performance of the repetition decoder when the noise standard deviation of the AWGN channel is increased to $\sigma = 1$?
2. *Binary Partial Response Signaling* A binary pulse amplitude modulation (BPAM) partial-response signaling (PRS) system shown in Figure 5.17 generates its pulse-shaped transmission output by exciting a linear time invariant (LTI) pulse-shaping filter $g(t)$ by an impulse train sequence consisting of amplitude values defined by:

$$B_n = I_n + I_{n-1} \quad (5.5)$$

at a rate $1/T$ symbols per second. The sequence $\{I_n\}$ consists of binary digits selected independently from the alphabet $\{-1, +1\}$ with equal probability. Hence, the filtered signal has the form:

$$\nu(t) = \sum_{n=-\infty}^{\infty} B_n g(t - nT), \quad T = \frac{1}{2W} \quad (5.6)$$

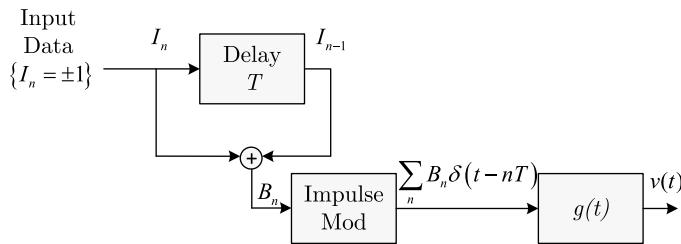


Figure 5.17 Schematic of a Binary PAM partial-response signaling system.

- Sketch the signal space diagram for $v(t)$ and determine the probability of occurrence of each symbol.
- Determine the autocorrelation of the sequence $\{B_n\}$.
- Solve for the power spectral density (PSD) of the sequence $\{B_n\}$.
- Given that $g(t) = u(t) - u(t - T)$, where $u(t)$ is a unit step function, solve for the PSD of the output signal $v(t)$.

HINT: The following Fourier Transform Pair may be useful:

$$\frac{t}{T} \div \quad Tsinc(Tf) \quad (5.7)$$

3. *Quadrature Partial Response Signaling* A quadrature partial-response signaling (QPRS) system is shown in Figure 5.18, where the in-phase and quadrature components of the transmitted signal each consist of the BPAM PRS implementation discussed in the previous problem. The transmission output of the QPRS system is equal to $s(t) = \text{Re}\{\nu(t)e^{j2\pi f_c t}\}$, where $\nu(t) = \nu_c(t) + j\nu_s(t)$, $B_n = I_n + I_{n-1}$, and $C_n = J_n + J_{n-1}$. Note that the sequences $\{B_n\}$ and $\{C_n\}$ are independent, and that the binary outputs of I_n and J_n are equiprobable. Assuming that $g(t) = u(t) - u(t - T)$, where $u(t)$ is a unit step function:

- Sketch the signal space diagram for the QPRS signal and determine the probability of occurrence of each symbol.
 - Determine the autocorrelation of $\nu_c(t)$, $\nu_s(t)$, and $\nu(t)$.
 - Solve for the power spectral density (PSD) of $\nu_c(t)$, $\nu_s(t)$, and $\nu(t)$.
4. *Block Interleaving* Refer to Section 5.1.2.1 for the definition of block interleaving, and answer the following questions:

- Show how a sequence of bits labeled b_1 through b_{20} can be block interleaved when $N = 5$ and $M = 4$. What is the resulting interleaved sequence?
- If the interleaved sequence experiences an error burst lasting five bit periods, show how the error burst is dispersed in the de-interleaved sequence.
- What is the amount of end-to-end delay caused by the block interleaving and de-interleaving processes?

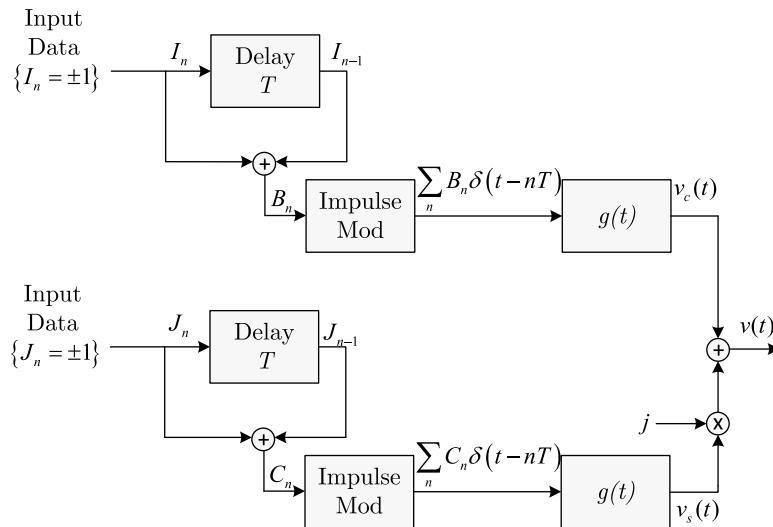


Figure 5.18 Schematic of a Quadrature partial-response signaling system.

5. *Convolutional Interleaving* Refer to Section 5.1.2.2 for the definition and block diagram of convolutional interleaving, and answer the following questions:

- Show how a sequence of bits labeled b_1 through b_{30} can be interleaved when $N = 4$. What is the resulting interleaved sequence?
- If the interleaved sequence experiences an error burst lasting five bit periods, show how the error burst is dispersed in the de-interleaved sequence.
- What is the amount of end-to-end delay caused by the interleaving and de-interleaving processes?

6. *Frequency Offset Compensation* In Section 5.2.1, you conduct several experiments on two USRPs with frequency offset, and then try to compensate the offset. As a theoretical background, in this problem, we will study the frequency offset between transmitter and receiver on frequency domain, and see how this causes the problem to modulation and demodulation.

The modulation and demodulation process is shown in Figure 5.19.

On transmitter, assume $x(t)$ is the baseband signal we are going to transmit. The carrier is a cosine wave with frequency f_c , so the modulated signal is $x(t)\cos(2\pi f_c t)$.

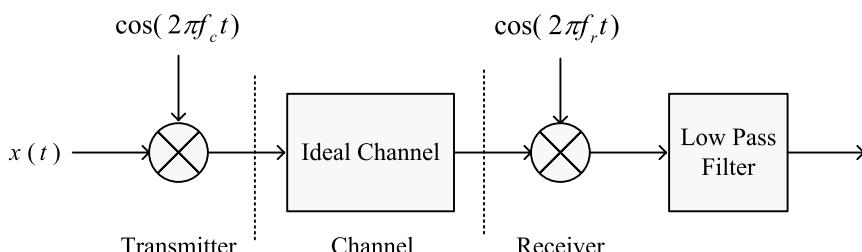


Figure 5.19 The modulation and demodulation process.

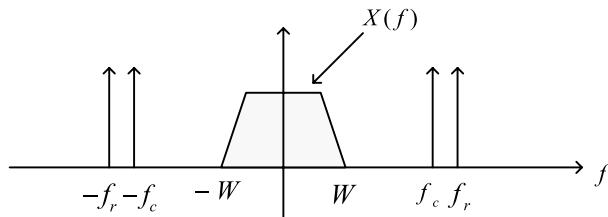


Figure 5.20 The spectrum of transmitted signal $x(t)$, and two carrier waves.

$\cos(2\pi f_c t)$. On frequency domain, the spectrum of $x(t)$ is $X(f)$, with bandwidth W , as shown in Figure 5.20. The spectrum of the carrier cosine wave is also shown in Figure 5.20, as two delta function located at f_c and $-f_c$.

Assume we have an ideal channel, so there is no distortion to the signal during transmission. On receiver, in order to demodulate the signal, we multiply the received signal by $\cos(2\pi f_r t)$, where f_r is receiver carrier frequency, which is slightly larger than f_c . The spectrum of this cosine wave is also shown in Figure 5.20, as two delta function located at f_r and $-f_r$.

In the end, we use a rectangular shape low pass filter with bandwidth W to cut high frequencies and reconstruct the original signal.

- (a) Sketch the spectrum of signal after modulation.
- (b) Sketch the spectrum of signal after demodulation.
- (c) Sketch the spectrum of signal after lowpass filter.
- (d) Compare the spectrum of signal after lowpass filter with the spectrum of original signal $X(f)$. Is this signal successfully recovered or not? Please explain.

References

- [1] *SDR Experimentation Using Simulink*, http://dot.ece.wpi.edu/?page_id=239.
- [2] Jacobsmeyer, J. M., *Introduction to Error-Control Coding*, www.pericle.com/papers/Error_Control_Tutorial.pdf.
- [3] Forney, G. D., “Burst-Correcting Codes for the Classic Bursty Channel,” in the *IEEE Transactions on Communications*. COM-19: 772–781, 1971.
- [4] Sklar, B., *Digital Communications Fundamentals and Applications*, Prentice Hall, 1988.
- [5] ASSA Radio Astronomy Group. *Radio Astronomy Projects*. <http://www.radio-assa.org.au/book/export/html/14>.
- [6] *USRP2 FAQ*. <http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ>.
- [7] IEEE 802.11 Working Group, *IEEE 802.11: The Working Group Setting the Standards for Wireless LANs*. <http://www.ieee802.org/11/>.
- [8] Kroupa, V. F., *Direct Digital Frequency Synthesizers*, Wiley-IEEE Press, 1998.
- [9] Vankka, J., *Digital Synthesizers and Transmitters for Software Radio*, Springer, 2005.

Receiver Structure and Waveform Synthesis of a Transmitter and a Receiver

In this chapter, we will introduce several techniques for designing and implementing two different receiver structures commonly used in the creation of a digital communication system, as well as assess their performance observed during over-the-air transmission. At the same time, we will show how to construct a series of orthonormal basis functions that can be combined in order to produce a wide range of signal waveforms that could be employed by a digital transmitter and receiver. In the experimental portion of this chapter, we will implement two different modulation structures and then observe their performance. Finally, to wrap up this chapter, we will develop a frame synchronization design for the open-ended design project.

6.1 SOFTWARE IMPLEMENTATION

Recall from Sections 4.7.1 and 4.7.2 the design of a maximum-likelihood receiver structure based on a matched filter realization, as well as a correlator-based approach. Using the theory introduced in Section 4.6, we will now proceed to build our own implementation for the matched filter and correlator-based receivers. Specifically, we will implement a two-step approach, where the first step creates the observation vector from the received signal waveforms, followed by the decision-making process in the second step. For the purpose of this experiment, we will use PAM transmission and an AWGN channel to validate the design presented in this chapter, as shown in Figure 6.1.

6.1.1 Observation Vector Construction

Given a waveform, we would like to use an observation vector for the receiver, so the first step of the implementation is to convert the received waveform $x(t)$ to a vector representation X , which is shown in Figure 6.2. Referring to Figure 6.2, this part of the experiment will construct an observation vector X , which will later be used as the input to a correlator-based detector. This section employs the orthonormal functions $\{f_m(t)\}$ specified in problem 1 of Section 6.5 to generate a bank of correlators operating on the received signal $x(t)$.

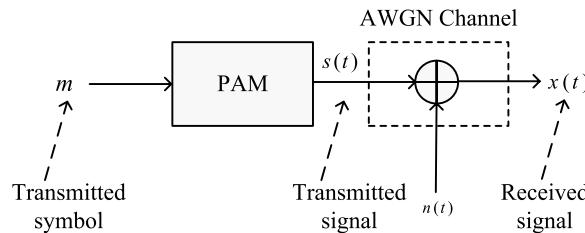


Figure 6.1 The received signal $x(t)$ is generated by a PAM transmission and an AWGN channel.



Given the presence of noise in the transmission channel, it is necessary to collect as much information about the received signal as possible in order to form an accurate decision. With more information of the received signals, the effect of noise is more likely to be dismissed.

Let us download and open the `correlator.m` file. The code has been *partially* implemented, and we will finish the rest of the code with the following steps and plot the results.

For the first part of the code, let us first start off this design by defining and implementing the four *equiprobable* transmitted signals specified in problem 1 of Section 6.5, $s_1(t), s_2(t), s_3(t), s_4(t)$, as shown in Figure 6.3. Each of them corresponds to one of the four transmitted symbols $m_i, i = 1, 2, 3, 4$. Since these four transmitted symbols are of equal probability, these four transmitted signals are also equiprobable. In this part, $\{s_i(t)\}$ are given in the form of signals in time domain. As specified by `correlator.m`, the duration of each signal is 3s, and the number of symbols is 100, defined by the *number* variable, so we should set the transmission time to 300s in order to ensure that 100 symbols are transmitted and received in each simulation loop.



Plot a randomly generated stream of these waveforms $s(t)$ in MATLAB. What do you observe? For example, a sample plot of $s(t)$ generated by 10 transmitted symbols m_i is shown in Figure 6.4.

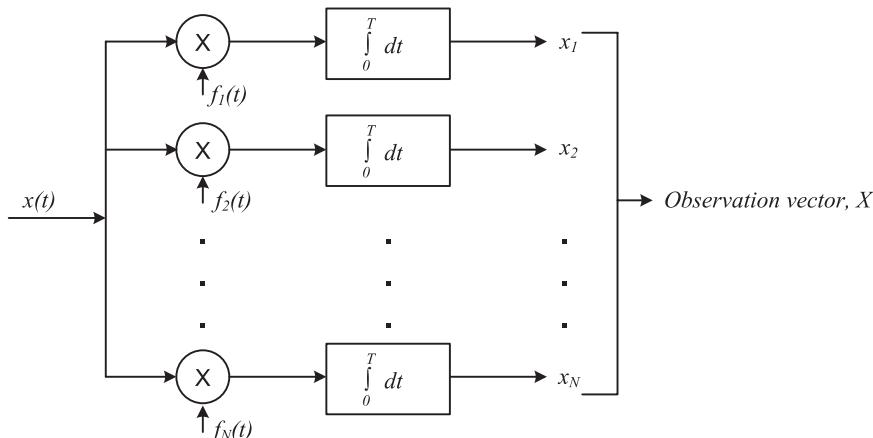


Figure 6.2 A bank of correlators operating on the received signal $x(t)$ to produce the observation vector \mathbf{X} that will be used for a correlator-based detector.

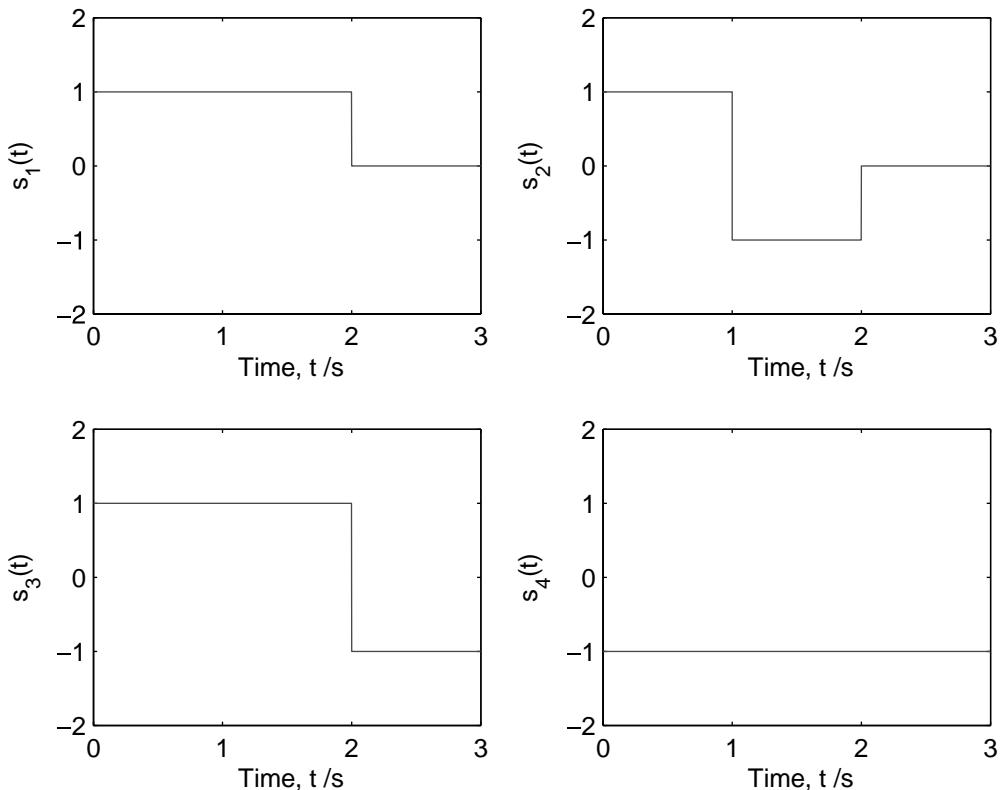


Figure 6.3 The waveform of four *equiprobable* transmitted signals, $s_1(t)$, $s_2(t)$, $s_3(t)$, and $s_4(t)$. Each of them lasts for 3s, and corresponds to one of the transmitted symbols.

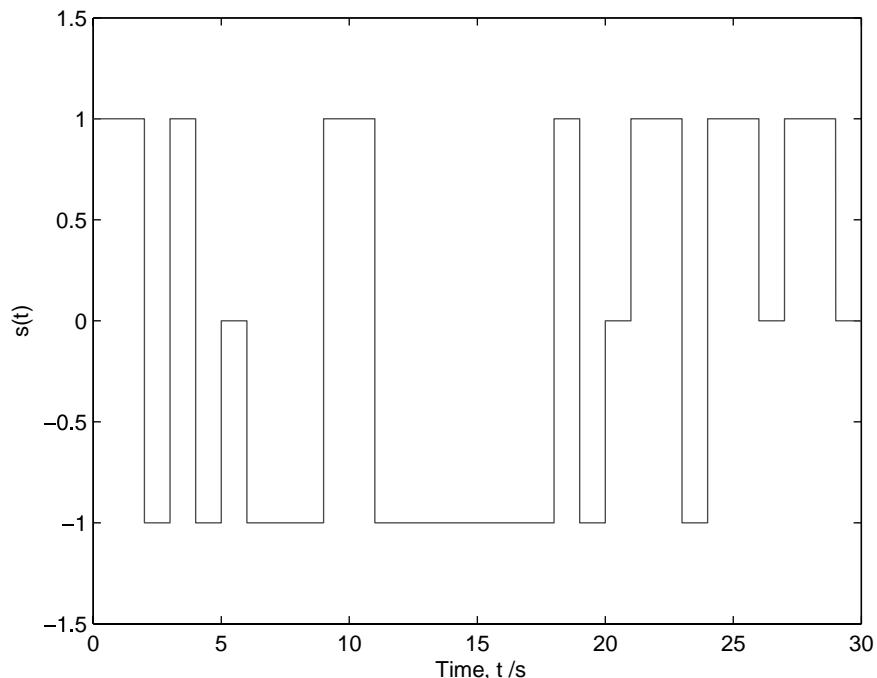


Figure 6.4 A sample plot of $s(t)$ generated by 10 transmitted symbols m_i .



1. Although the design presented here is for continuous time signals, we need to employ a discrete time approximation of $L = 1000$ samples per second in order for this to work in MATLAB. In other word, the sampling rate used is 1000 samples per second.
2. The set of signals $\{s_i(t)\}$ can be generated by concatenating scaled ones and zeros matrices together.
3. For every 3s, we will need to choose one of the four signals randomly. One approach for doing this task efficiently is to use *switch* and *case* in MATLAB.

Next, let us create an AWGN channel that will be used to add zero-mean white Gaussian noise of variance to the transmitted signal $s(t)$, which can be expressed as:

$$x(t) = s(t) + n(t) \quad (6.1)$$

where $s(t)$ is the transmitted signal, $x(t)$ is the received signal, and $n(t)$ is the white Gaussian noise that will be generated in this step in order to model the distortion introduced by the transmission channel.



Plot the time domain representations of the input signals $s(t)$ and output signals $x(t)$ for the channel for several different noise variances. Explain how the noise could potentially impair the successful decoding of the intercepted signal at the receiver.



We can use *randn* in MATLAB to define the Gaussian noise. For instance, to create 100 zero-mean Gaussian values with standard deviation 2, we can use the following command: `2 . * randn (100,1) ;`

Once the received signal waveforms have been implemented in MATLAB, let us proceed to define the orthonormal functions $\{f_m(t)\}$ derived in problem 1 of Section 6.5. This step is similar to the step that $\{s_i(t)\}$ are defined, just make sure that the vector for $\{s_i(t)\}$ and $\{f_m(t)\}$ has the same length.

In the end, let us implement an “integrate-and-dump” block in order to obtain the observation vector \mathbf{X} , defined as:

$$\mathbf{X} = (x_{1,1}, x_{1,2}, x_{1,3}), (x_{2,1}, x_{2,2}, x_{2,3}), \dots, (x_{100,1}, x_{100,2}, x_{100,3}) \quad (6.2)$$

where $\{x_{n,m}\}$ is generated by:

$$x_{n,m} = \int_0^T x_n(t)f_m(t)dt, \quad n = 1, 2, 3, \dots, 100, \quad m = 1, 2, 3 \quad (6.3)$$

This block integrates a series of values, uses the result, and restarts integrating, as shown in Figure 6.5. Basically, the integrate-and-dump block does a resampling operation on the filtered output.

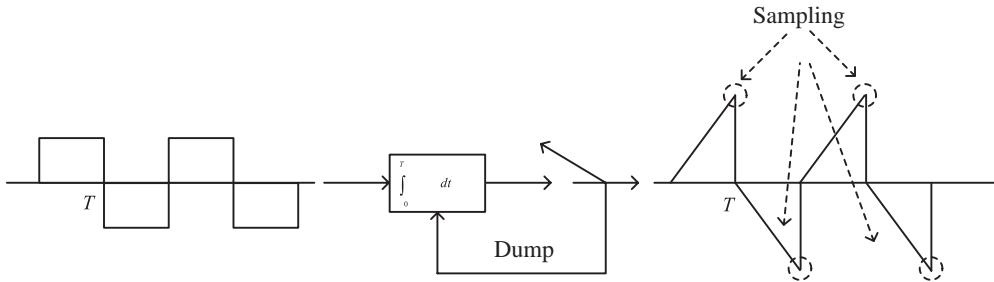


Figure 6.5 The operations done by an integrate-and-dump block. Notice how the integrator accumulates the input waveform across the time period T before it resets (i.e., dumps, in order to repeat the process again).



Since we are employing discrete time approximation for continuous time signals integration could be realized by *sum* in MATLAB.



When you have done all these steps test your implementation with a transmission of duration 300s, randomly consisting of one of the four equiprobable signals of duration $T = 3$ s each with zero-mean white Gaussian noise of variance 0.5 added. Assume perfect synchronization between the received signal and the communications system. Plot each of the elements of the observation vector \mathbf{X} using the *stem* command in MATLAB.

6.1.2 Maximum-Likelihood Decoder Implementation

As mentioned at the beginning of Section 6.1, we are implementing a two-step approach, where the first step creates the observation vector from the received signal and the second step makes the decision. The implementation of the correlator in Section 6.1.1 has created the observation vector \mathbf{X} , which is the transformation of the received signal into vector space. So now, in this section, we will proceed to the second step, which is finding the best match by comparing the observation vector to the signal space representation of the transmitted waveforms. This step is known as the maximum-likelihood (ML) decoder, as shown in Figure 6.6. This decoder will operate on the observation vector \mathbf{X} from Section 6.1.1 to produce an estimate m_n of the n th transmitted symbol m_n in a way that would minimize the average probability of symbol error.

According to Figure 6.6, the maximum-likelihood algorithm works as follows: Given the observation vector \mathbf{X} at the input to the decoder as well as the vector representations of the transmitted signals $\{s\}$, we first correlate \mathbf{X} with s_i across all i using a element-by-element product and an accumulator. Next, we normalize the correlation result by the corresponding signal energy E_i in order to facilitate a fair comparison. Finally, the resulting decision values for each of the branches are compared against each other, and the branch with the largest resulting value is selected.

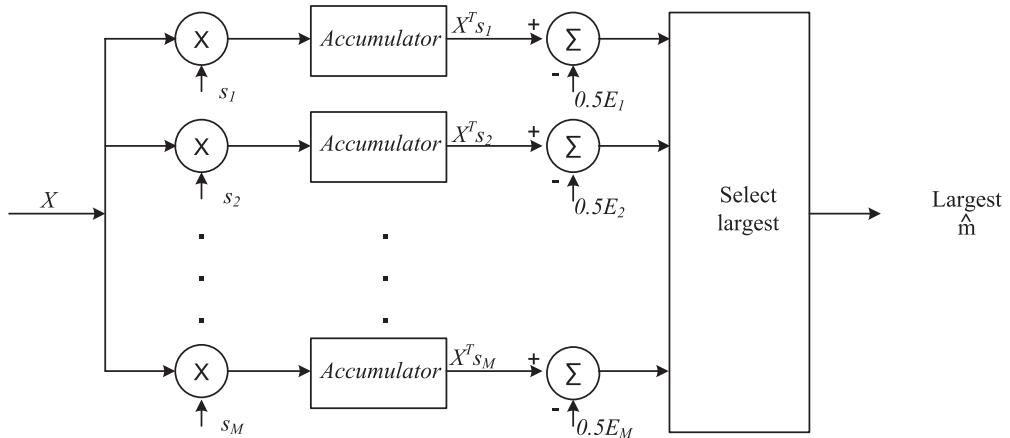


Figure 6.6 A maximum-likelihood (ML) decoder system operating on the observation vector \mathbf{X} from Section 6.1.1 to produce an estimate m of the transmitted symbol m_i to minimize the average probability of symbol error. Notice how the element-by-element product of the vectors X and s_i followed by an accumulator is equal to the dot product, $X^T s_i$.

In order to get started with this stage of the implementation, let us download and open the `decoder.m` file. Although this code is *incomplete*, it will serve as a starting point for the rest of this experiment.

For the first part of the code, let us define the four vector representations of the signals $\{s_i(t)\}$, which are denoted as sv in the code and have been derived in problem 1 of Section 6.5. Each signal vector sv_i should be a 1×3 vector.

As long as we have the sv_i vector, we can calculate the energy of each signal using the following expression:

$$E_i = \| sv_i \|^2 = \langle sv_i, sv_i \rangle = \sum_{j=1}^3 sv_{ij}^2, \quad i = 1, 2, 3, 4 \quad (6.4)$$

where $\langle sv_i, sv_i \rangle$ is the inner product of sv_i and sv_{ij} is the component of each signal vector.

Given the energy of each signal, let us proceed to implement the accumulator and subtraction as shown in Figure 6.6. For accumulator, it adds up all the element-by-element product between X and s_i .



Accumulator can be realized by sum in MATLAB.



Plot the output of the accumulator for each of the branches. Provide your observations and explain.

For subtraction, we normalize the accumulated result by the corresponding signal energy E_i calculated in (6.4). The subtraction is necessary due to the theory of the maximum-likelihood decoder, presented in Section 4.6.3. The decoder decision rule minimizes the probability of error, which is expressed in (8.11).

Let us rewrite the decision rule as follows:

$$\begin{aligned}
 s_k &= \arg \min_{s_i} \|x_j - s_i\| \\
 &= \arg \min_{s_i} (x_j - s_i)(x_j - s_i) \\
 &= \arg \min_{s_i} (x_j \otimes x_j - 2x_j \otimes s_i + s_i \otimes s_i)
 \end{aligned} \tag{6.5}$$

where x_j , x_j is common to all decision metrics for different s_i , so we can omit it, thus yielding:

$$\begin{aligned}
 \min_{s_i} (-2x_j \otimes s_i + s_i \otimes s_i) &= \max_{s_i} (2x_j \otimes s_i - s_i \otimes s_i) \\
 &= \max_{s_i} 2 \left(\int_0^T x_j(t)s_i(t)dt - 0.5 \int_0^T s_i^2(t)dt \right)
 \end{aligned} \tag{6.6}$$

Therefore, it is necessary to subtract half of the signal energy in each branch. Note that if all energy values are the same for each possible signal waveform, we can dispense with the energy normalization process since this will have no impact on the decision making. In the end, select the largest branch and decode it to produce the estimate \hat{m} .



We can use *max* in MATLAB to select the largest branch.



Remove the energy subtraction from each of the branches and repeat your experiment.

1. Do you get the same results as before?
5. What would be required for the results with and without energy subtraction to be equivalent?

There are several factors that can impact the performance of the entire system, such as limits of integration, energy subtraction, and SNR of the channel. Energy subtraction has been discussed before. As for the other two factors, if we integrate over a shorter duration than the symbol length, we are correlating the received symbol with a smaller portion of the transmitted symbol. If we increase the SNR value, it implies that the transmit power is also increased.

6.1.3 Correlator Realization of a Receiver in Simulink

In Section 6.1.1 and 6.1.2, we have implemented a matched filter-based receiver structure using MATLAB. In this part of the experiment, we will implement another type of the receiver structure, namely, the correlator realization, in Simulink. The receiver structure has been shown in Figure 4.17, which employs the same transmitted signal $s_i(t)$ as in Section 6.1.1. Consequently, we will be constructing a transmitter that is capable of randomly generating $s_1(t)$, $s_2(t)$, $s_3(t)$, and $s_4(t)$, devising an

additive white Gaussian noise (AWGN) channel, and a four-branch receiver using correlation in each branch to help determine which symbol was transmitted every $T = 1$ second. Let us assume perfect synchronization in this system.

Given these factors, try out the following changes to the system and compare them with the original results:

Q

1. In Section 6.1.1, do not integrate the entire period T ; integrate until $0.75T$. Plot both the estimate m and the actual transmitted symbol m_i and compare it with the original plot.
2. In Section 6.1.2, do not subtract the energy of $s_i(t)$ from each branch. Plot both the estimate m and the actual transmitted symbol m_i and compare it with the original plot.
5. Combine your implementation from Sections 6.1.1 and 6.1.2. Change the parameters of AWGN channel in Section 6.1.1 to make the signal-to-noise ratio (SNR) range from 10^{-5} to 10^{-1} . Compare the m in Section 6.1.2 with the input x in Section 6.1.1 to get the bit error rate. Plot a BER curve. NOTE: In order to get an accurate BER curve, you may need a large number of transmitted signals, especially when the SNR value is high.

In order to prepare for our implementation, let us download and open the `integrate_and_dump.mdl` file, which is shown in Figure 6.7. In this Simulink model, we can find three important components that will be used in our own implementation. Note that the *Integrate and Dump* block accumulates samples according to the signal possessing the most samples per unit time. In this case, the *Gaussian*

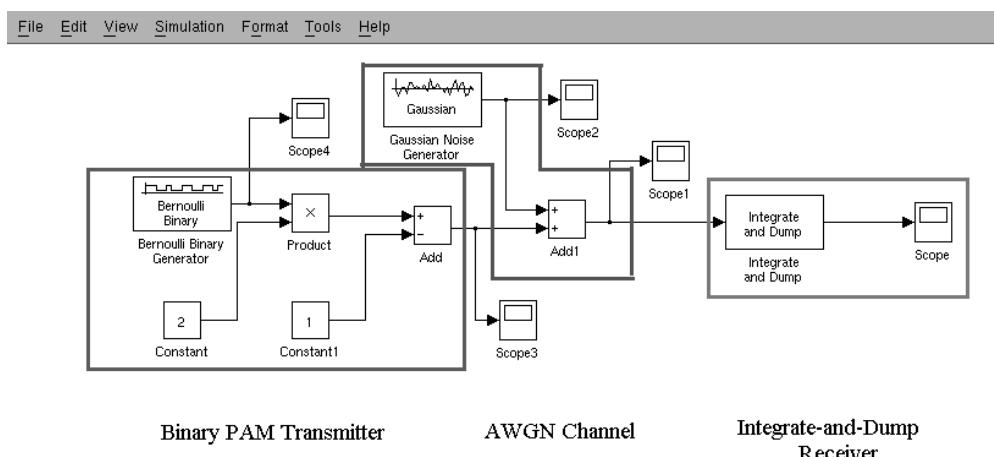


Figure 6.7 A simple Simulink model of a digital transceiver employing an integrate-and-dump block at the receiver.

Noise Generator is producing 100 samples per 1 second duration. We can run the experiment for different pulse durations T (the default is $T = 1$ second).



Explain the end-to-end operation of this basic Simulink model, from the binary PAM transmitter all the way across to the output of the integrate-and-dump block. Use time domain plots to help illustrate your explanation.

Now let us create a new Simulink model that will contain our design for the four-branch correlator-based receiver. First, implement a transmitter that can generate each of the four waveforms shown in Figure 6.17. The *Multipoint Switch* block can be used to determine which waveform is transmitted for each symbol period of time.



Plot the output of this transmitter to show the randomly generated sequence of waveforms $s_1(t)$, $s_2(t)$, $s_3(t)$, and $s_4(t)$.

Next, pass this signal stream through an AWGN channel in order to corrupt the signal with noise. Vary the amount of noise introduced to the signal stream and observe the shape of the received signal $r(t)$. Note that other channel effects such as fading can also corrupt the signal. However, we are only focusing on the AWGN in this exercise.

Given the corrupted intercepted signal $r(t)$, we will conduct the following steps to create the receiver: First, for each branch, we will repeat $s_i(t)$ indefinitely such that it aligns with the corresponding intercepted signal $r(t)$. One approach is to use the *repeating sequence* block in the basic Simulink sources blockset. Then, multiply the repeating $s_i(t)$ waveforms in each of the branches with the intercepted signal $r(t)$. Take the output of this multiplication and input it into the integrate-and-dump block.



How much noise must be added to make the signal stream unrecognizable?



What did you set as the number of samples to be integrated?

As discussed before, due to the unique symbol energies, for proper operation, the system must compensate for the differences in each branch. Therefore, take the output of the integrate-and-dump block and subtract from it the energy of the corresponding waveform $s_i(t)$. Since the receiver knows about all the waveforms ahead

of time, simply calculate manually the symbol energy and use the *constant* source block and the *addition* block to subtract off the energy from the integrated signal stream. The relevant signal energy is scaled and subtracted from each branch, thus producing a measure of the difference between the received signal and each of the waveforms for each sample instant.



What do you notice about the relative energy levels for each of the branches?

Finally, implement the decision-making block that selects the maximum value at each time instant $T = 1$ seconds. The ML decoder just chooses the symbol with the highest correlation as the estimate of the transmitted symbol, which is then converted to binary and compared to a binary version of the transmitted symbol. Note that there is a delay of one sample interval in the decoder output. In order to calculate the error rate, the binary input also needs to be delayed by one sample interval to align with the decoded symbol.



Plot the output of the decision-making process, indicating which of the four waveforms were selected. Why is there a delay in the decoded signal relative to the originally transmitted signal?

As long as we have created the transmitter and the receiver, let us calculate the bit error rate of this receiver design for a range of noise variance values. Plot the BER curve of this model down to 10^{-3} . A MATLAB script can be created to set various values for the SNR, which are used to control the Simulink model to determine the BER parameters. The BER block can be used to terminate the simulations for each iteration. The termination condition is set to terminate after either a certain number of errors are generated or a certain number of symbols have been transmitted.



Suppose you removed the energy subtraction for all the branches. Does the BER performance of the system change?

6.2 USRP HARDWARE IMPLEMENTATION

In the previous section, we studied how to design actual digital receivers in software based on the ML concept. However, there is significant value in evaluating these designs in actual hardware as well in order to provide insights that only real-world implementations can provide. Therefore, we will start off with the design of two different modulation schemes employed in a basic transmitter and receiver implementation. From this experiment, we will get our first experience of how the simulation world differs from actual hardware world.

6.2.1 Differential Binary Phase-Shift Keying

Differential phase shift keying is a noncoherent form of phase shift keying that avoids the need for a coherent reference signal at the receiver. Noncoherent receivers are relatively easy and cheap to build and hence are widely used in wireless communications.

6.2.1.1 Transmitter

The transmitter has a basic structure, which is made up of four blocks, as shown in Figure 6.8.

The purpose and special parameters of each block is as follows: *Signal From Workspace* block is the source of this model. In this block, we use a valid MATLAB expression (`[1 0]`) to specify the *Signal* parameter, which generates a repeated ‘10.’ We use this signal since it is easy to observe. In the *Sample time* parameter, there is a variable named *BitRate*. This variable is specified by callback functions, which will be introduced shortly. We set *Samples per frame* to be 179, because the input frame size of the *SDRu Transmitter* block is set as 358, and *Raised Cosine Transmit Filter* has an upsampling factor of 2.

In a DBPSK system, the input binary sequence is differentially modulated using a DBPSK modulator. In Simulink, this operation is conducted by the *DBPSK Modulator Baseband* block. The output of this block is a baseband representation of the modulated signal.

The *Raised Cosine Transmit Filter* block upsamples and filters the input signal using a square root raised cosine FIR filter. The icon of the block shows the impulse response of the filter. Similar to the *Signal From Workspace* block, the variable named *oversampling* is defined in callback functions.

The *SDRu Transmitter* block is the sink of this model, which can be obtained by typing *sdrilib* in the command window. Please note that the input frame size to this block does not have to be 358. It can be any number you would like.

6.2.1.2 Receiver

The receiver behaves in an inverse manner relative to the transmitter, but there are more things going on, so it is more complicated than the transmitter. The receiver model is made up of three parts, as shown in Figure 6.9.

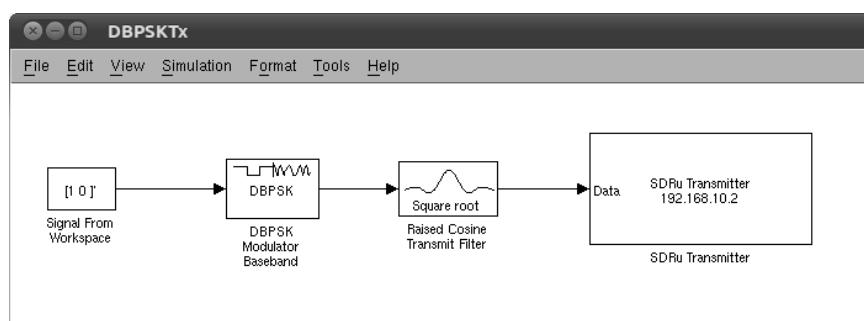


Figure 6.8 The structure of DBPSK transmitter, where a binary source is modulated using DBPSK and filtered using raised cosine pulse shaping filter.

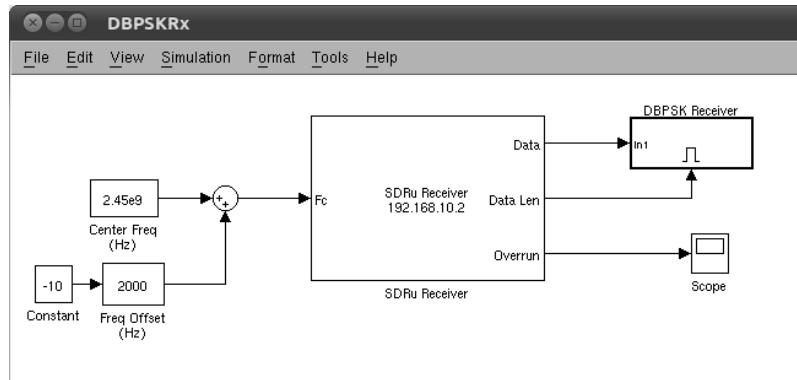


Figure 6.9 The structure of a DBPSK receiver, where the transmitted signal is received by the *SDRu Receiver* block and fed into the *DBPSK Receiver* subsystem. Frequency offset between two USRP2s should be compensated to ensure signal reception.

The first part is the frequency offset compensator. We should use the frequency offset found in Section 5.2.1 in this part. The second part is the *SDRu Receiver* block and the third part is an enabled subsystem, where all the main operations of this model reside, as shown in Figure 6.10.

The purpose and special parameters of each block inside the enabled subsystem is as follows: Since the output of the *SDRu Receiver* block is sample based, we use the *Frame Conversion* block to convert it to framebased. This block does not make any changes to the input signal other than the sampling mode.

In order to form a pulse shaping filter with the *Raised Cosine Transmit Filter*, the *Raised Cosine Receive Filter* block filters the input signal using a square root raised cosine FIR filter. The icon of the *Raised Cosine Receive Filter* block shows the impulse response of the filter.

One of the most significant advantages of a DPSK modulation scheme is that we do not need to worry about the *carrier recovery*, which estimates and compensates for frequency and phase differences between a received signal's carrier wave and the receiver's local oscillator for the purpose of coherent demodulation. However, we

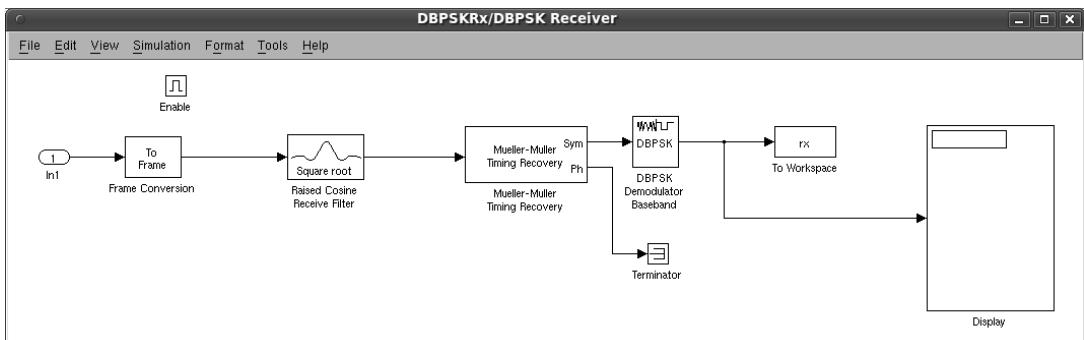


Figure 6.10 The structure of *DBPSK Receiver* subsystem, where a symbol is filtered using raised cosine pulse shaping filter and demodulated using DBPSK. The timing recovery is conducted by the *Mueller-Muller Timing Recovery* block.

still need to implement some form of *timing recovery*. The purpose of the timing recovery is to obtain symbol synchronization. Two quantities must be determined by the receiver to achieve symbol synchronization. The first is the sampling frequency, and the other is sampling phase. The *Mueller-Muller Timing Recovery* block recovers the symbol timing phase of the input signal using the Mueller-Muller method. This block implements a decision-directed, data-aided feedback method that requires prior recovery of the carrier phase.

Corresponding to the *DBPSK Modulator Baseband* on the transmitter side, the *DBPSK Demodulator Baseband* block demodulates a signal that was modulated using the differential binary phase shift keying method. The input is a baseband representation of the modulated signal.

In the end, *To Workspace* and *Display* blocks serve as the sink of this model. Using these two blocks, we can observe the received data in two ways. We can either observe a portion of data directly through *Display*, or save a larger number of data to workspace through *To Workspace*.

6.2.1.3 Using Callback Functions

Callback functions are a powerful way of customizing the Simulink model. A *callback* is a function that executes when you perform various actions on your model, such as clicking on a block or starting a simulation. We can use callbacks to execute a MATLAB script or other MATLAB commands. We can use block, port, or model parameters to specify callback functions.

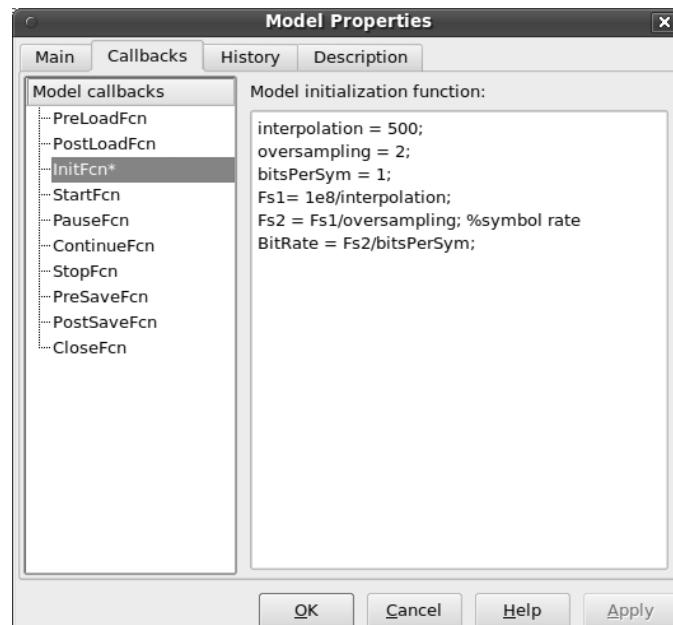


Figure 6.11 Create model callback functions using the Callbacks pane of the model's Model Properties dialog box. In our DBPSK example, several parameters are defined in InitFcn.

Common tasks you can achieve by using callback functions include



- Loading variables into the MATLAB workspace automatically when you open your Simulink model;
- Executing a MATLAB script by double-clicking on a block;
- Executing a series of commands before starting a simulation;
- Executing commands when a block diagram is closed.

You can create model callback functions interactively or programmatically. Use the Callbacks pane of the Model Properties dialog box to create model callbacks interactively, as shown in Figure 6.11. You can access it by right clicking on the blank space of your model. In Callbacks pane, the main callbacks you can define include the following: PreLoadFcn, PostLoadFcn, StartFcn, StopFcn, and CloseFcn.

To create a callback programmatically, use the `set_param` command to assign a MATLAB expression that implements the function of the model parameter corresponding to the callback. Read the online documentation [1] for more information about creating a callback programmatically.

6.2.2 Differential Quadrature Phase-Shift Keying

With the theory of DBPSK presented in the previous section, let us now construct a DQPSK system, which employs many of the same design features as DBPSK. First, let us create a DQPSK transmitter model called `DQPSKTx.mdl`, which should contain the *DQPSK Modulator Baseband* block and the *Raised Cosine Transmit Filter* block. Just as in the DBPSK implementation, we will need to select several operating parameters for this model, such as *Signal*, *Sample time*, and *Samples per frame*. Next, construct a DQPSK receiver named `DQPSKRx.mdl`. Use the *DQPSK Demodulator Baseband* block and the *Raised Cosine Receive Filter* block. It is vitally important that we compensate the frequency offset between the transmitter and the receiver. Then, run these two models and plot the data you have saved to workspace.



Compare the performance between DBPSK and DQPSK. Which one is better? Justify your answer.

6.2.3 Accelerate the Simulink Model that Uses USRP Blocks

In Section 6.2.1, when we plot the received data from DBPSK, we may find some constant zeros even if we have compensated the frequency offset. At the same time, by observing Overrun of *SDR_u Receiver* block, we can find that there are packets being dropped during the USRP2 transmission to the host. This problem is mainly due to the Simulink model not being fast enough to keep up with the processing

speed of the USRP2 board, such that the model cannot be executed in real time. Here is a collection of performance improvements we can make in our Simulink models to approach, if not achieve, real time:

1. In the **Simulation Configuration Parameters Data Import/Export dialog**, turn off all logging.
2. Make sure that the model is single rate. If the model requires resampling, then choose rational coefficients that will keep the model single rate.
3. Do not add any *Buffer* blocks to the model.
4. You should try to run with *Rapid Accelerator* instead of *Normal* mode. Be aware that some scopes do not plot data when run in *Rapid Accelerator* mode, but scopes inevitably slow down a model in any case. (See Point 6.)
5. Try to avoid feedback loops. Typically, such loops imply scalar processing, which will slow down the model considerably.
6. Perhaps the most obvious trick of all is not to use scopes unless absolutely necessary. To visualize your data, send it to a workspace variable and postprocess it.
7. If you are using *Accelerator* or *Rapid Accelerator*, set the **Simulation Configuration Parameters Optimization Compiler optimization level** to “Optimizations on (faster runs).”
8. If the model has lots of *Constant* blocks, it could help slightly if **Simulation Configuration Parameters Optimization Inline parameters** is checked on. This will cause the sample time of those *Constant* blocks to truly become inf such that Simulink will get the values once and only once during a run.
5. If the model generates code, the Solver setting should be *Fixed-step/discrete*. The tasking mode should be *Single Tasking*.

Applying several or all of these approaches, we should be able to improve the real-time processing ability of our Simulink models with USRP blocks. Besides, as of MATLAB 2012b, Simulink will have a so-called Performance Advisor, which can automatically improve a model’s performance.

6.3 OPEN-ENDED DESIGN PROJECT: FRAME SYNCHRONIZATION

In the previous experiments in this chapter, have sent digital information consisting of ones and zeros. However, in real-world situations, we usually need to send messages that are made of packets or frames of data. Therefore, knowing where each frame starts is a crucial step for performing any format of data reception. In this section, we will develop frame synchronization mechanism in an open-ended manner using the principle of correlation.

6.3.1 Frame Synchronization

Frame synchronization is the process in the telecommunications transmission system to align the digital channel (time slot) at the receiving end with the corresponding time slot at the transmission end as it occurs. For example, you transmit a packet that contains numerous frames. At the receiver side, if you want to know where a specific frame actually starts, then you need to implement frame synchronization.

Frame synchronization in packet-based protocols is often realized using a preamble in the transmitted signal. This preamble is a known signal chosen to be easily detected and is transmitted immediately before the user's data. When the preamble is detected, the receiver now knows where the user data begins and can ensure that it is handled correctly. Thus, frame synchronization involves the following steps: In the first step, the transmitter injects a fixed-length symbol pattern, called a marker, into the beginning of each frame to form a marker and frame pair, which is known as a *packet*. Packets are then converted from symbols into a waveform and transmitted through the channel. The receiver detects the arrival of packets by searching for the marker, removes the markers from the data stream, and recovers the transmitted messages. Marker detection is the most important step for frame synchronization. In this open-ended design project, a 13-bit Barker code is proposed to form preambles. This code is chosen to have an extremely peaky result when correlated with itself, enabling easy detection.

In this project, we will first introduce two Simulink models that implement frame synchronization. In these models, Barker code is employed as the marker. Then, we are going to build on the second model and incorporate the USRP2 into it. In the end, we will transmit “Hello world” from end to end using two USRP2s.

6.3.2 Barker Code

Barker codes are commonly used for frame synchronization in digital communication systems. A Barker code is a sequence of N values of +1 and -1:

$$a_j \text{ for } j = 1, 2, \dots, N \quad (6.7)$$

such that:

$$\left| \begin{array}{cc} N & \nu \\ a_j a_{j+\nu} \\ \hline j=1 & \end{array} \right| = 1 \quad (6.8)$$

for all $1 \leq \nu < N$.

Barker codes have a length of at most 13 bits and possess low correlation sidelobes. A correlation sidelobe is the correlation of a codeword with a time-shifted version of itself. An example of autocorrelation function of Barker-7 code is shown in Figure 6.12. It is obvious from this figure that the sidelobes are low.

6.3.3 Simulink Models

In this section, we will introduce two Simulink models that implement frame synchronization using Barker code as the marker. The first model illustrates the basic idea, and the second model employs the idea.

6.3.3.1 Basic Model

To get started, let us download and open the `mFindFrameSync.mdl` file, as shown in Figure 6.13. There are several key blocks in this model:

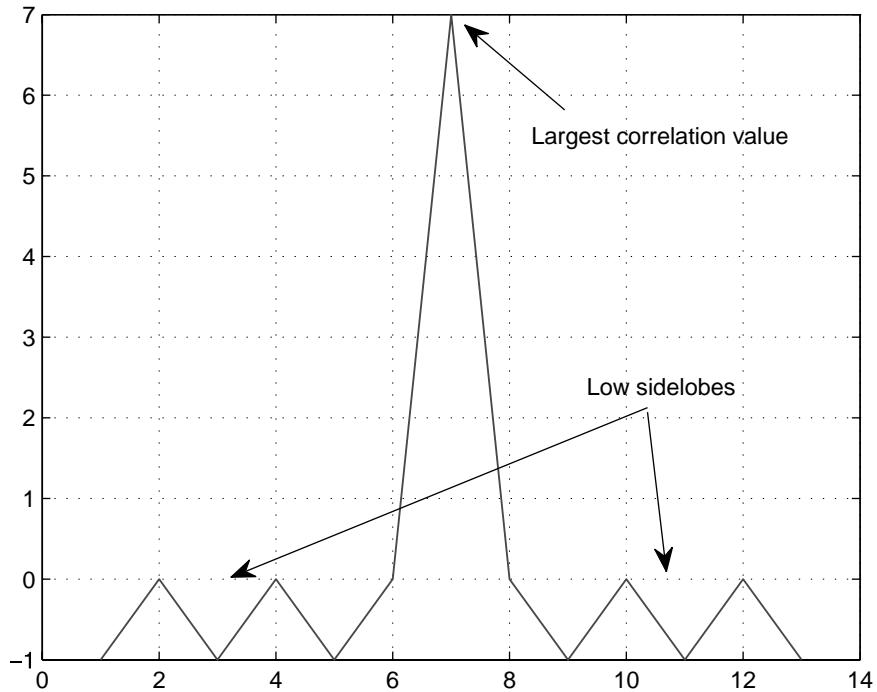


Figure 6.12 Autocorrelation function of Barker-7 code, which has a low sidelobes.

At the beginning, in the *Barker Code Generator* block, let us specify the length of the Barker code, and this block will give the corresponding codeword, which is either 1 or -1. However, the Bernoulli binary is either 1 or 0. In order to attach the Barker code to the frame, we need to change its format using the *Bipolar to Unipolar Converter* block. Then, the Barker code is attached to the beginning of each frame using *Matrix Concatenate* to form a marker and frame pair, which is known as a packet. In the middle, the *Delay* block is used to simulate a real channel, which incurs a delay of 5. In the end, the *Error Rate Calculation* block uses the delay value we have found to calculate the computation delay and verifies whether we have found it correctly. If the value is correct, the error rate should be 0.

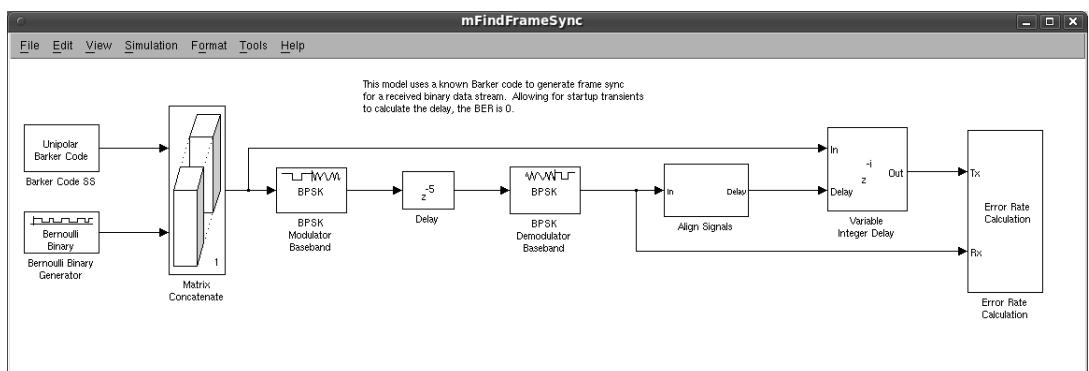
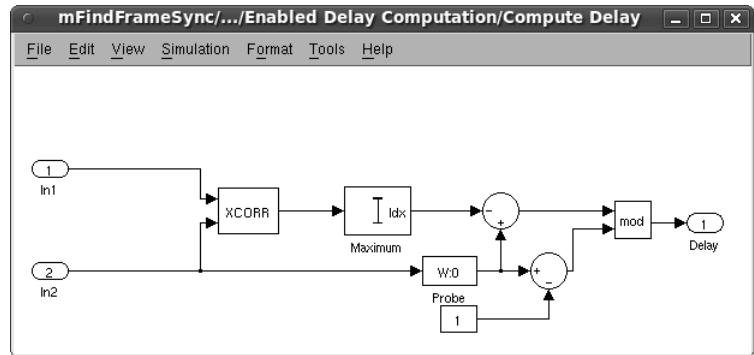
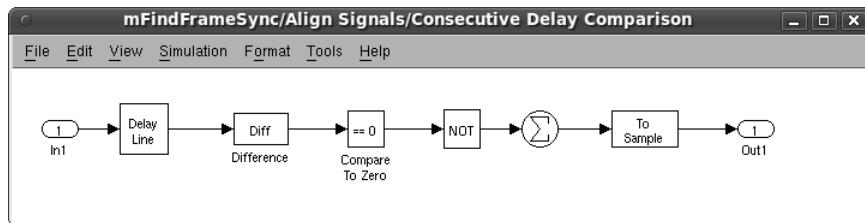


Figure 6.13 Simulink model that realizes frame synchronization using Barker code.



(a)



(b)

Figure 6.14 Two key subsystems in Simulink model `mFindFrameSync.mdl`. (a) Compute Delay subsystem, which calculates the delay of the channel by detecting the peak of the correlation of the received data and the Barker code. (b) Consecutive Delay Comparison subsystem, which checks whether the calculated delay remains the same for several iterations.

In addition, there are two key subsystems in this model: *Compute Delay* (*Align Signals Enabled Delay Computation Compute Delay*), as shown in Figure 6.14(a), calculates the delay of the channel by detecting the peak of the correlation of the received data and the Barker code. *Consecutive Delay Comparison* (*Align Signals Consecutive Delay Comparison*), as shown in Figure 6.14(b), checks whether the calculated delay remains the same for `numDelayCalcs` iterations. If it is, the *Enabled Delay Computation* subsystem is disabled, such that we don't need to calculate the delay again and again. This subsystem guarantees the result of this model and also improves the efficiency.

6.3.3.2 Application of the Basic Idea

According to the basic idea, we can construct a more advanced digital communication system. Let us download and open the `mFindFrameStart.mdl` file, as shown in Figure 6.15. Previously, what we have done is transmit binary bits. However, using this model, we can transmit a sentence, such as “Hello world.” Therefore, in addition to this Simulink model, we need another m-file to conduct the converter between characters and binary bits, which has been realized in the `charToBitsAndBack.m` file.

The variable “Delay” is a most important value in this model, as it provides the information concerning where the start of the frame is, such that the receiver could know where should be the starting point of decoding.

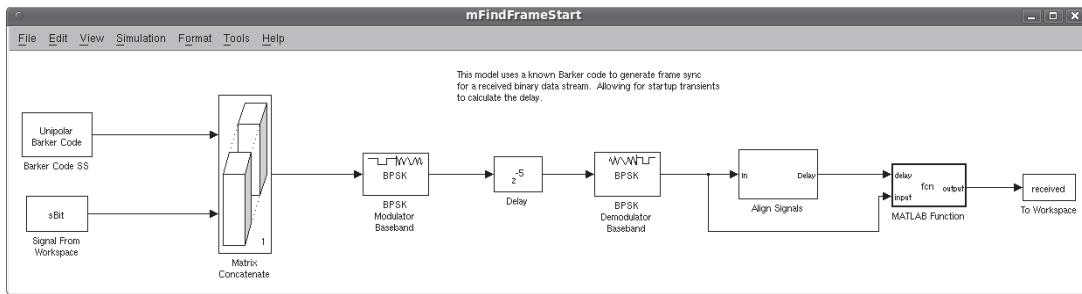


Figure 6.15 Simulink model that realizes frame transmission using Barker code.

Compared to the basic model, there are three different blocks: First, in the *Signal From Workspace* block, let us specify a source from the workspace in order to use it as our test message. For example, suppose we would like to transmit the test message “Hello world” to the awaiting receiver. Since “Hello world” is in ASCII character format, in order to be transmitted, we need to convert it to the format that the computer can process, which is the binary format. We run `charToBitsAndBack.m` and get the corresponding bit streams. We save this bit stream in a variable called “sBit” so that it can be used as the signal source. In this block, “samples per frame” should be the same as the length of “sBit.” Although the length of bit stream for “Hello world” is 77, 10 zeros are added to the end of this bit stream, so the length of “sBit” becomes 87.

Next, in *MATLAB Function* block, we use a MATLAB Function to pick up the useful information out of a frame. In this example, $\text{delay}+13+1$ is the first bit in the stream and $\text{delay}+13+77$ is the last bit. The number 13 corresponds to the length of the Barker code, and 77 is the length of bit stream for “Hello world.” Therefore, if we want to transmit some other sentences, or we want to use a Barker code of some other length, these values need to be modified.

At last, in *To Workspace* block, we save the useful bits to workspace, and these bits can be converted to ASCII characters using the second half of `charToBitsAndBack.m`. The converted results are illustrated in Figure 6.16, which shows that we have successfully received the transmitted messages.

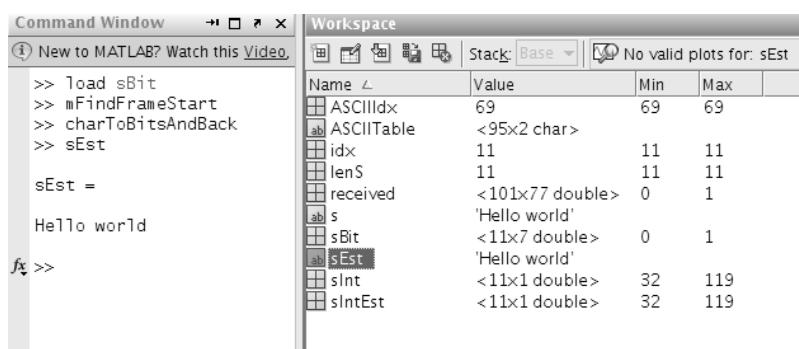


Figure 6.16 The converted results at the receiver side.

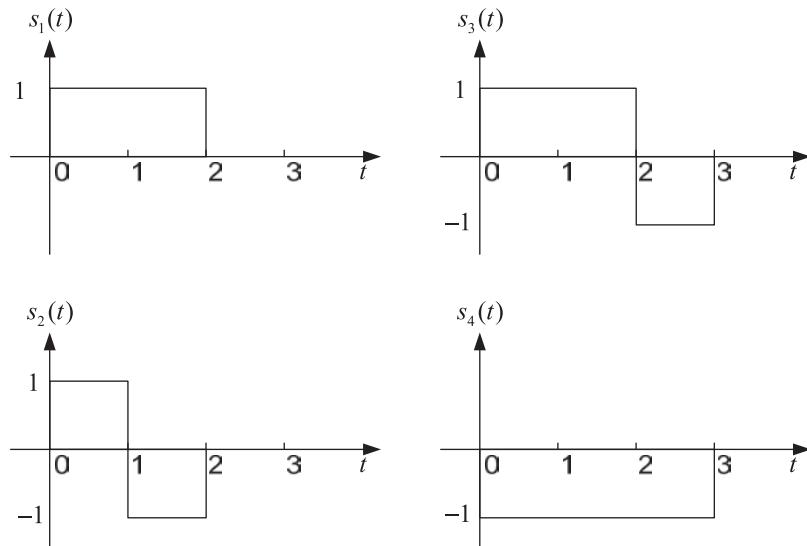


Figure 6.17 Four-signal waveforms for the Gram-Schmidt Orthogonalization procedure.

6.3.3.3 Incorporating USRPs

When incorporating USRPs into the second model, it is suggested to use DBPSK to ensure the best performance of bit transmission.

6.3.4 Hints for Implementation

According to the input frame size of the SDRu Transmitter block, we will need to buffer N frames of data, which makes it less “real time,” but that is OK. Note that there is always a tradeoff between buffer size and latency. It is always suggested to try working with the raw data offline in MATLAB before trying out real-time data retrieval or the Simulink model that do this automatically.

6.3.5 Hints for Debugging

When you have done the complete implementation, you can use the following steps for debugging. First, collect incoming data from the SDRu Receiver block and save it in an mat file. Then, correlate the Barker sequence with stored incoming data in MATLAB and find the mode of the largest correlation value every 179 samples. You should visualize it, not just expect it to be “13.” Next, take the mean of the mode, which can be off by ± 1 . In the end, feed incoming data back into your Simulink receiver model without USRP2 and fix the delay. Once verified, implement the system entirely in Simulink.

6.4 CHAPTER SUMMARY

This chapter covers two basic receiver structures commonly used in the creation of a digital communication system and assesses their performance observed during over-the-air transmission. It also shows how to construct a series of orthonormal basis

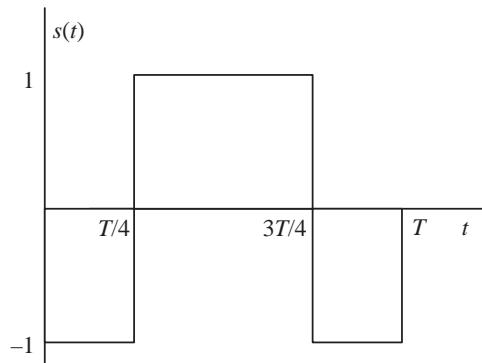


Figure 6.18 Pulse representing binary "1."

functions that can be combined to produce a wide range of signal waveforms. In the hardware experimental part of this chapter, two different modulation schemes are implemented using USRP hardware. Finally, a frame synchronization approach is designed for the open-ended design project, which enables the packet transmission.

6.5 PROBLEMS

1. [Gram-Schmidt Orthogonalization] Carry out the Gram-Schmidt orthogonalization procedure of the signals in Figure 6.16 in the order $s_3(t)$, $s_1(t)$, $s_4(t)$, $s_2(t)$ and thus obtain a set of orthonormal functions $\{f_n(t)\}$. Then, determine the vector representation of the signals $\{s_n(t)\}$ by using the orthonormal functions $\{f_m(t)\}$. Also, determine the signal energies.
2. [Matched Filter Realization] Consider the signal: $s(t) = \frac{A}{T}t\cos(\omega_c t)$ for $0 \leq t \leq T$
 - (a) Determine the impulse response of the matched filter for the signal.
 - (b) Determine the output of the matched filter at $t = T$.
 - (c) Suppose the signal $s(t)$ is passed through a correlator that correlates the input $s(t)$ with $s(t)$. Determine the value of the correlator output at $t = T$. Compare your result with that in part (b).
3. [Maximum Likelihood Detection] A certain digital baseband modulation scheme uses the pulse shown in Figure 6.18 to represent binary symbol "1" and the negative of this pulse to represent binary "0." Derive the formula

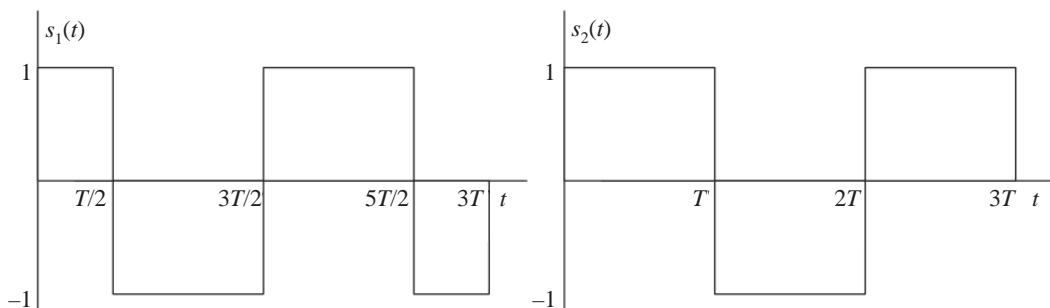


Figure 6.19 A pair of orthogonal signals.

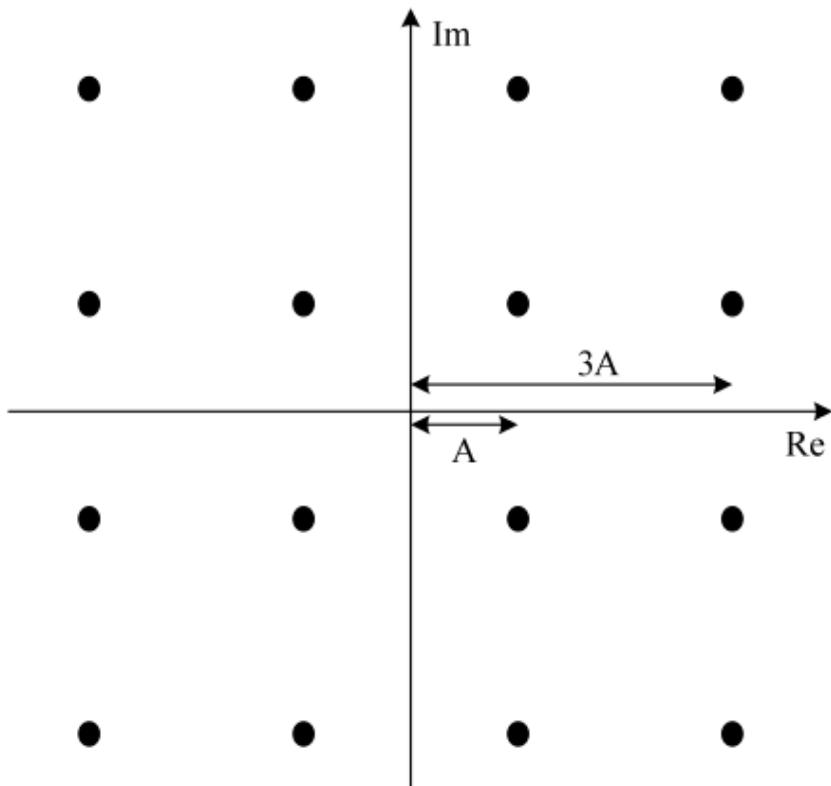


Figure 6.20 A 16-QAM constellation.

for the probability of error incurred by the maximum likelihood detection procedure applied to this form of signaling over an AWGN channel.

4. [Receiver Design] A pair of orthogonal signals $s_1(t)$ and $s_2(t)$ over the observation interval $0 < t < 3T$ are shown in Figure 6.19. The received signal is defined by

$$x(t) = s_k(t) + w(t), \quad 0 < t < 3T, \quad k = 1, 2, \quad (6.10)$$

where $w(t)$ is additive white Gaussian noise of zero mean and power spectral density $N_0/2$.

- (a) Design a receiver that decides in favor of signals $s_1(t)$ and $s_2(t)$, assuming that these two signals are equiprobable.
- (b) Calculate the average probability of symbol error incurred by this receiver for $E/N_0 = 4$, where E is the signal energy.

5. [Correlator Realization] A binary digital communication system employs the signals

$$\begin{aligned} s_0(t) &= 0, & 0 < t < T \\ s_1(t) &= A, & 0 < t < T \end{aligned} \quad (6.1)$$

for the transmission of information. The demodulator is implemented as a bank of correlators followed by samplers to sample the output of the correlators at $t + T$.

- (a) Determine the optimum detector for an AWGN channel and the optimum threshold, assuming that the signals are equiprobable.
- (b) Determine the probability of error as a function of the SNR. How does the above signaling scheme compare with antipodal signaling?
6. [Error Bound] The signal constellation for a communication system with 16 equiprobable symbols is shown in Figure 6.20. The channel is AWGN with noise power spectral density of $N_0/2$.
- (a) Using the union bound, find a bound in terms of A and N_0 on the error probability for this channel.
- (b) Determine the average SNR per bit for this channel.
- (c) Express the bound found in part (a) in terms of the average SNR per bit.
- (d) Compare the power efficiency of this system with a 16-level PAM system.

Reference

- [1] The MathWorks, *Using Callback Functions*, <http://www.mathworks.com/help/toolbox/simulink/ug/f4-122589.html>.

Multicarrier Modulation and Duplex Communications

In Section 4.2, we studied several single-carrier modulation schemes, where input binary bits are modulated by a carrier signal with the center frequency f_c . Instead of having one center frequency, multi-carrier modulation (MCM) multiplexes serial input data into several parallel streams and transmit them over independent subcarriers. These subcarriers can be individually modulated and manipulated, allowing for optimization with respect to the channel. This is especially true for a wireline communications system where the channel is well defined. This chapter will introduce fundamental principles of multicarrier modulation and show why MCM is an integral part of modern communications systems. Following this, we will look at the hardware implementation of a duplex communications system, which will provide us with the fundamentals of the media access control protocols. Note that for hardware implementation and open-ended design projects, we will focus on single carrier, due to inherent challenges of implementing multicarrier on USRP. This will be left as an exercise to the interest reader.

7.1 THEORETICAL PREPARATION

Why do digital communication systems use multicarrier modulation? What are the disadvantages of transmitting at a high data rate using a single carrier transmission? What happens if part of the channel is severely attenuated? MCM addresses these issues via a “divide-and-conquer” approach. More specifically, MCM transmits data in parallel frequency bands simultaneously rather than in a single large channel. Since dividing the transmission into small bands allows for individual treatment of these subcarriers, a multicarrier transmission scheme becomes robust to fast-fading channels and narrowband interference.

Note that there are several important distinctions between multicarrier transmission in a wired communications environment versus a wireless communications environment. Without loss of generality, we will be focusing in this book chapter on a wireless communications environment, but is important to note that certain assumptions made here are not necessarily optimal for a wireline implementation.

7.1.1 Single Carrier Transmission

Most modern communication systems, including all digital subscriber line (xDSL) modems [1–3], as well as most commercial communication standards, including WiFi [4, 5], WiMAX [6, 7] and LTE [8, 9], use digital signaling techniques to

transmit large quantities of information over a short period of time while being robust to transmission noise. The fundamental unit of information in these digital communication systems is the *bit*, which consists of only two values (“on” and “off,” “1” and “0,” and so). However, these systems usually map, or *modulate*, groups of bits into symbols prior to transmission. At the receiver, each symbol is compared to a known set of symbols and is *demodulated* into the group of bits corresponding to the closest matching symbol. In the next subsection, we will look at the modulation scheme that will be employed in this chapter.

7.1.1.1 Modulation and Demodulation of QAM Symbols

Quadrature amplitude modulation (QAM) is the technique of transmitting data on two quadrature carriers (i.e., they are 90° out of phase with each other, making them *orthogonal*). In what follows, we will consider a digital implementation of the modulation, so that the carrier signals are $\cos(\omega_k n)$ and $\sin(\omega_k n)$, with carrier frequency ω_k . The amplitude of each of these two carriers is specified by the sequence of input bits and is changed every $2N$ samples, which is called the *QAM symbol period*. The amplitudes can take on only a finite number of possible values. Figure 7.1 represents a discrete-time model of the QAM modulator, where the sequence of input bits $d[m]$ is used to determine the amplitudes of the two carriers. For each symbol period, D bits of input are taken from the input bit stream $d[m]$ and used to select one of the 2^D combinations of amplitudes for the two carriers. Calling these amplitudes $a[\ell]$ and $b[\ell]$, respectively, where ℓ represents the symbol time index, these amplitudes are kept constant for the duration of a symbol period by the upsampling and rectangular window filtering shown in Figure 7.1, yielding the piecewise constant signals $a'[n]$ and $b'[n]$. One can then write the expression of the modulated signal as

$$s[n] = a'[n]\cos(\omega_k n) + b'[n]\sin(\omega_k n) \quad (7.1)$$

where $\omega_k = 2\pi k/2N$ is the carrier frequency and $2N$ is the period of the symbol. Note that we have to limit the possible carrier frequencies to integer multiples of $2\pi/2N$ in order to keep orthogonality between the sinusoidal and cosinusoidal carriers in a digital implementation (see the following derivations).

In this chapter, we will be dealing with rectangular QAM signal constellations, such as those shown in Figures 7.2(a), 7.2(b), and 7.2(c) for 4-QAM, 16-QAM, and 64-QAM, respectively. This basically amounts to saying that $a[\ell], b[\ell] \in \{\pm(2k-1)E, 0\}$,

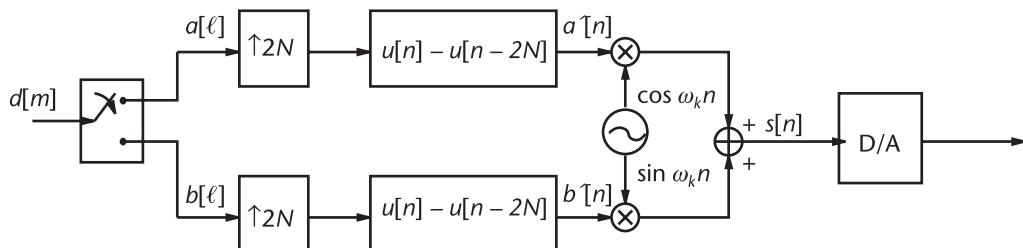


Figure 7.1 Rectangular QAM modulator (multirate model of a digital implementation), where the carrier signals are $\cos(\omega_k n)$ and $\sin(\omega_k n)$, and their amplitudes are determined by input bits $d[m]$.

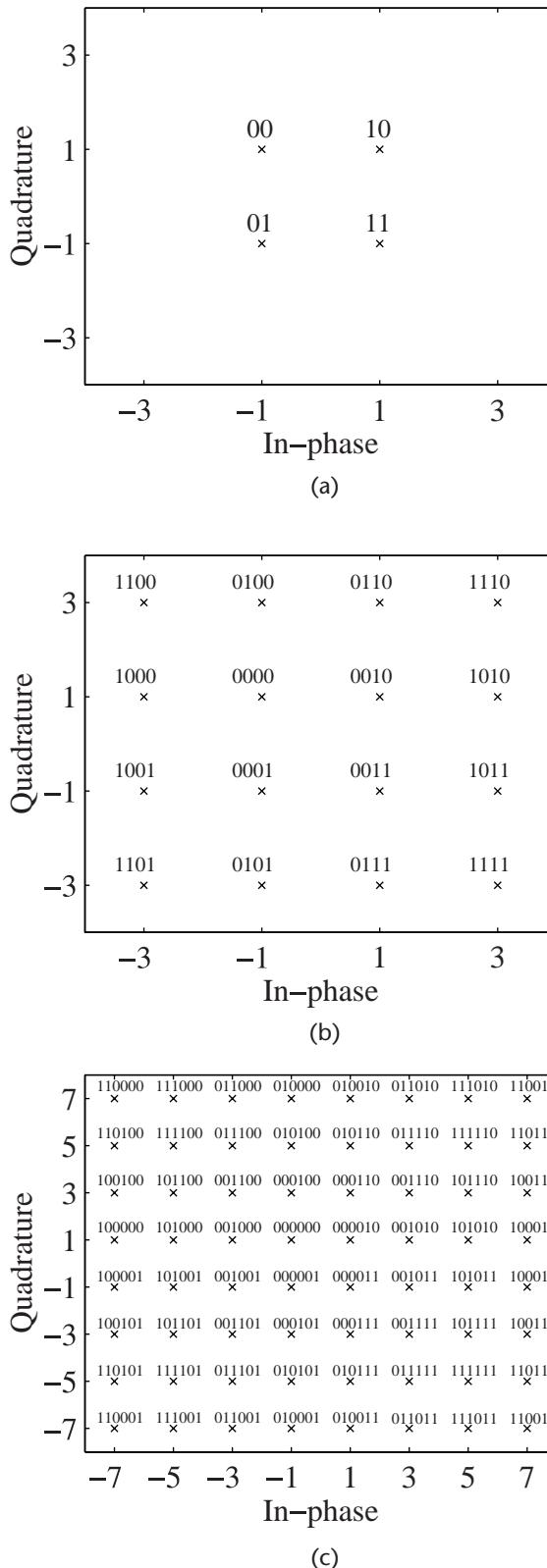


Figure 7.2 Three types of QAM signal constellations. The in-phase values indicate the amplitude of the carrier $\cos(\omega_k n)$, while the quadrature values indicate the amplitude of the carrier $\sin(\omega_k n)$. (a) 4-QAM signal constellation. (b) Rectangular 16-QAM signal constellation. (c) Rectangular 64-QAM signal constellation.

$k = 1, \dots, 2^{D/2-1}$, where E is some positive constant that scales the energy of the sent signals and $D/2$ is the number of bits used to represent the amplitude level of one of the carriers during a symbol.

One of the advantages of QAM signaling is the fact that demodulation is relatively simple to perform. From Figure 7.3, the received signal, $r[n]$, is split into two streams and each multiplied by carriers, $\cos(\omega_k n)$ and $\sin(\omega_k n)$, followed by a summation block (implemented here through filtering by a rectangular window followed by downsampling). This process produces estimates of the in-phase and quadrature amplitudes, $\hat{a}[\ell]$ and $\hat{b}[\ell]$, namely,

$$\begin{aligned}
\hat{a}[\ell] &= \sum_{n=2\ell N}^{2\ell N+2N-1} r[n] \cos(\omega_k n) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} a'[n] \cos(\omega_k n) \cos(\omega_k n) + \sum_{n=2\ell N}^{2\ell N+2N-1} b'[n] \sin(\omega_k n) \cos(\omega_k n) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} a'[n] \cos\left(\frac{2\pi kn}{2N}\right) \cos\left(\frac{2\pi kn}{2N}\right) + \sum_{n=2\ell N}^{2\ell N+2N-1} b'[n] \sin\left(\frac{2\pi kn}{2N}\right) \cos\left(\frac{2\pi kn}{2N}\right) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} \frac{a'[n]}{2}
\end{aligned} \tag{7.2}$$

and

$$\begin{aligned}
\hat{b}[\ell] &= \sum_{n=2\ell N}^{2\ell N+2N-1} r[n] \sin(\omega_k n) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} a[n] \cos(\omega_k n) \sin(\omega_k n) + \sum_{n=2\ell N}^{2\ell N+2N-1} b[n] \sin(\omega_k n) \sin(\omega_k n) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} a[n] \cos\left(\frac{2\pi kn}{2N}\right) \sin\left(\frac{2\pi kn}{2N}\right) + \sum_{n=2\ell N}^{2\ell N+2N-1} b[n] \sin\left(\frac{2\pi kn}{2N}\right) \sin\left(\frac{2\pi kn}{2N}\right) \\
&= \sum_{n=2\ell N}^{2\ell N+2N-1} \frac{b[n]}{2}
\end{aligned} \tag{7.3}$$

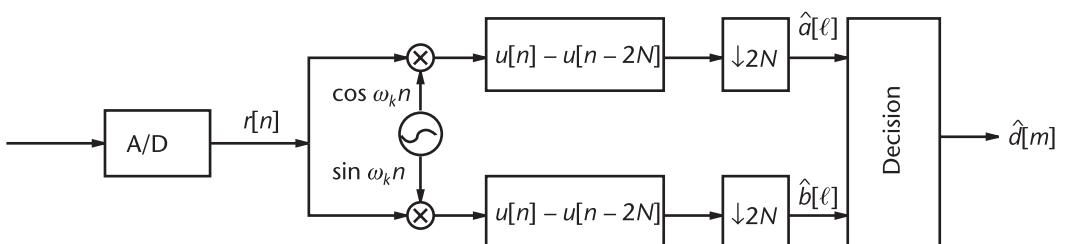


Figure 7.3 Rectangular QAM demodulator, where the received signal, $r[n]$, is split into two streams and each multiplied by carriers, followed by a summation block.

where, due to the orthogonality of the two carriers, the cross terms vanish, leaving the desired amplitude (after some trigonometric manipulation). The bits estimated from $\hat{a}[\ell]$ and $\hat{b}[\ell]$ bits are then multiplexed together, forming the reconstructed version of $d[m]$, $\hat{d}[m]$.

We have so far dealt with modulation and demodulation in an ideal setting. Thus, one should expect that $d[m] = \hat{d}[m]$ with probability of 1. However, in the next subsection we will examine a physical phenomenon that distorts the transmitted signal, resulting in transmission errors.

7.1.2 Multicarrier Transmission

As we have seen thus far, signals can be transmitted on a single carrier frequency. For instance, (7.1) modulates to the carrier frequency ω_k . However, the transmitted signal may only use up a small portion of the total available bandwidth. To increase bandwidth efficiency and throughput, it is possible to send additional QAM signals in other portions of the unused bandwidth simultaneously. An example of this is shown in Figure 7.4, where we have taken the QAM modulator of Figure 7.1 and put several of them in parallel, each with a different carrier frequency. The data for

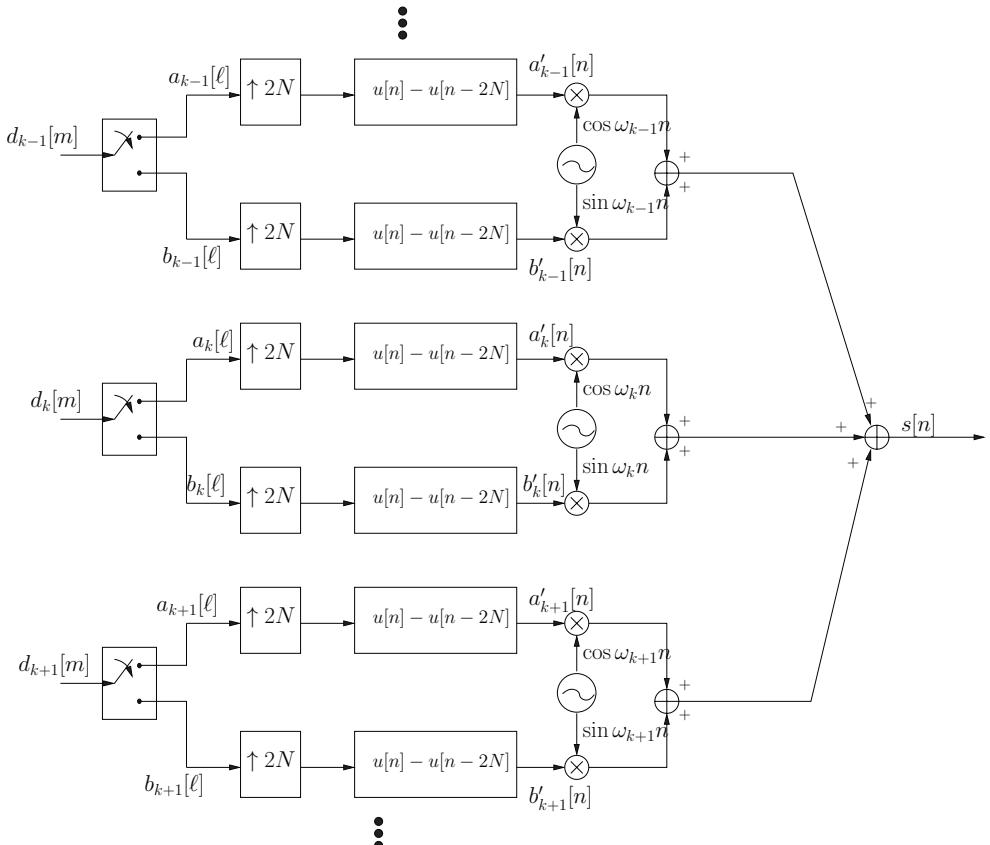


Figure 7.4 Transmitter of an orthogonally multiplexed QAM system, where several QAM modulators are put in parallel, each with a different carrier frequency.

each modulator came from portions of a high-speed bit stream. This is principle behind multicarrier modulation.

Orthogonal frequency division multiplexing (OFDM) is an efficient type of multicarrier modulation that employs the discrete Fourier transform (DFT) and inverse DFT (IDFT) to modulate and demodulate the data streams. Since the carriers used in Figure 7.4 are a sinusoidal function of $2\pi kn/2N$, it should come as no surprise that a $2N$ -point DFT or IDFT can carry out the same modulation, since it also contains summations of terms of the form $e^{\pm 2\pi kn/2N}$. The setup of an OFDM system is presented in Figure 7.5. A high-speed digital input, $d[m]$, is demultiplexed into N subcarriers using a commutator. The data on each subcarrier is then modulated into an M -QAM symbol, which maps a group of $\log_2(M)$ bits at a time. Unlike the representation of (7.1), for subcarrier k we will rearrange $a_k[\ell]$ and $b_k[\ell]$ into real and imaginary components such that the output of the “modulator” block is $p_k[\ell] = a_k[\ell] + jb_k[\ell]$. In order for the output of the IDFT block to be real, given N subcarriers we must use a $2N$ -point IDFT, where terminals $k = 0$ and $k = N$ are “don’t care” inputs. For the subcarriers $1 \leq k \leq N - 1$, the inputs are $p_k[\ell] = a_k[\ell] + jb_k[\ell]$, while for the subcarriers $N + 1 \leq k \leq 2N - 1$, the inputs are $p_k[\ell] = a_{2N-k}[\ell] + jb_{2N-k}[\ell]$.

The IDFT is then performed, yielding

$$s[2\ell N + n] = \frac{1}{2N} \sum_{k=0}^{2N-1} p_k[\ell] e^{j(2\pi nk/2N)} \quad (7.4)$$

where this time $2N$ consecutive samples of $s[n]$ constitute an OFDM symbol, which is a sum of N different QAM symbols.

This results in the data being modulated on several subchannels. This is achieved by multiplying each data stream by a $\sin(Nx)/\sin(x)$, several of which are shown in Figure 7.6.

The subcarriers are then multiplexed together using a commutator, forming the signal $s[n]$, and transmitted to the receiver. Once at the receiver, the signal is demultiplexed into $2N$ subcarriers of data, $\hat{s}[n]$, using a commutator and a $2N$ -point DFT, defined as

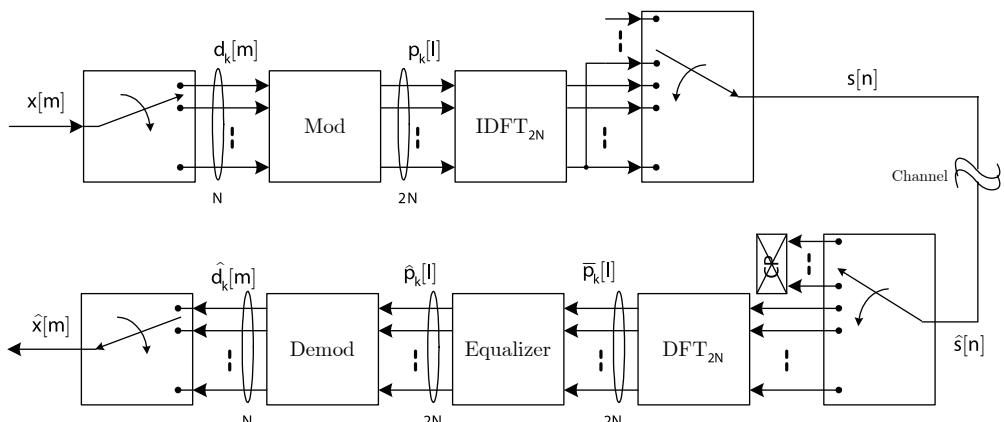


Figure 7.5 Overall schematic of an orthogonal frequency division multiplexing system, where the DFT and IDFT are employed to modulate and demodulate the data streams.

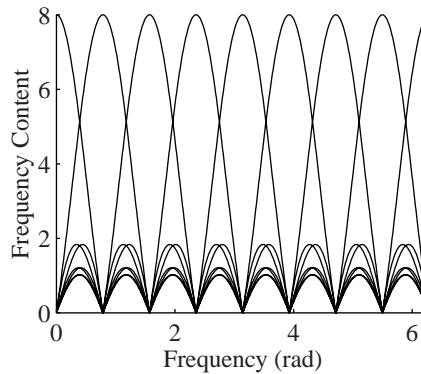


Figure 7.6 Characteristics of orthogonal frequency division multiplexing: frequency response of OFDM subcarriers.

$$\bar{p}_k[\ell] = \sum_{n=0}^{2N-1} \hat{s}[2\ell N + n] e^{-j(2\pi nk/2N)} \quad (7.5)$$

and applied to the inputs, yielding the estimates of $p_k[\ell]$, $p_k[\ell]$. The output of the equalizer, $p_k[\ell]$, then passes through a demodulator, and the result multiplexed together using a commutator, yielding the reconstructed high-speed bit stream, $\hat{d}[m]$.

7.1.3 Dispersive Channel Environment

Until now, we have considered only an OFDM system operating under ideal conditions (i.e., the transmitter is connected directly to the receiver without any introduced distortion). Now we will cover channel distortion due to dispersive propagation and how it is compensated for in OFDM system.

From our high school physics courses, we learned about wave propagation and how they combine constructively and destructively. The exact same principles hold in high-speed data transmission.

For example, in a wireless communication system, the transmitter emanates radiation in all directions (unless the antenna is “directional,” in which case, the energy is focused at a particular azimuth). In an open environment, like a barren farm field, the energy would continue to propagate until some of it reaches the receiver antenna. As for the rest of the energy, it continues on until it dissipates.

In an indoor environment, as depicted in Figure 7.7(c), the situation is different. The line-of-sight component (if it exists), p_1 , arrives at the receiver antenna first, just like in the open field case. However, the rest of the energy does not simply dissipate. Rather, the energy is reflected by the walls and other objects in the room. Some of these reflections, such as p_2 and p_3 , will make their way to the receiver antenna, although not with the same phase or amplitude. All these received components are functions of several parameters, including their overall distance between the transmitter and receiver antennas as well as the number of reflections. At the receiver, these components are just copies of the same transmitted signal, but with different arrival times, amplitudes, and phases. Therefore, one can view the channel as an impulse response that is being convolved with the transmitted signal. In the

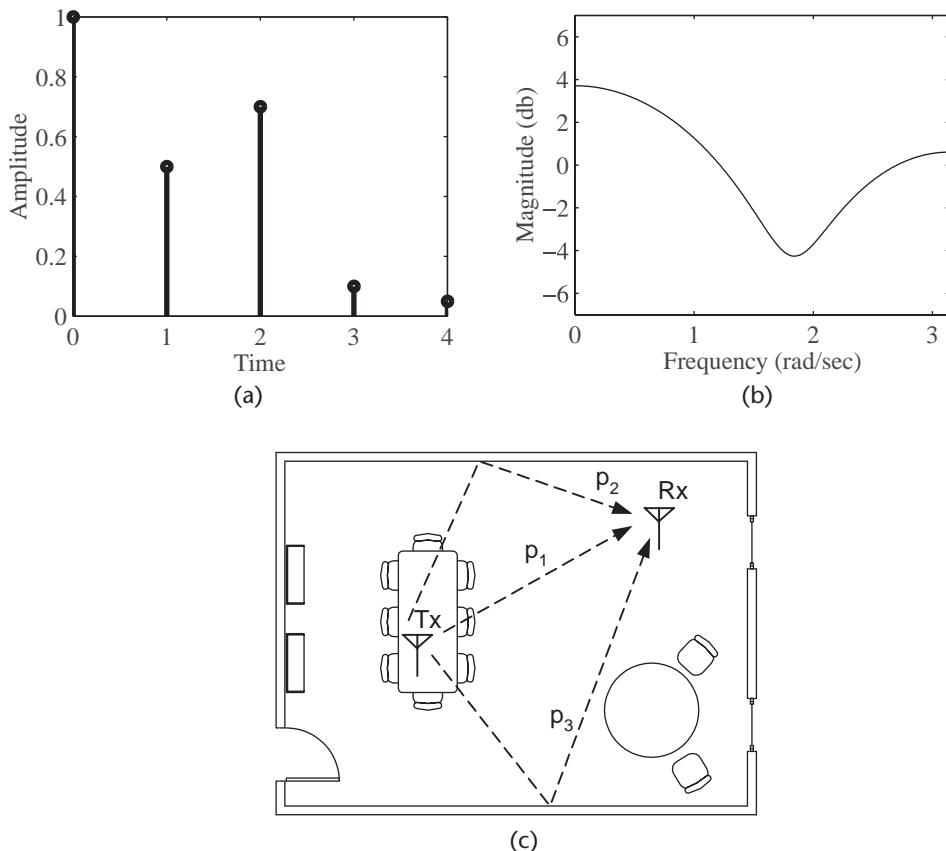


Figure 7.7 Example of a channel response due to dispersive propagation. Notice the three distinctive paths, p_1 , p_2 and p_3 , that start at the transmitter Tx and are intercepted at the receiver Rx . (a) Impulse response. (b) Frequency response. (c) The process by which dispersive propagation arises.

open field case, the *channel impulse response* (CIR) would be a delta, since no other copies would be received by the receiver antenna. On the other hand, an indoor environment would have several copies intercepted at the receiver antenna, and thus its CIR would be similar to the example in Figure 7.7(a). The corresponding frequency response of the example CIR is shown in Figure 7.7(b).

In an xDSL environment, the same principles can be applied to the wireline environment. The transmitted signal is sent across a network of telephone wires, with numerous junctions, bridging taps, and connections to other customer appliances (e.g., telephones, xDSL modems). If the impedances are not matched well in the network, reflections occur and will reach the devices connected to the network, including the desired receiver.

With the introduction of the CIR, new problems arise in our implementation that need to be addressed. In Section 7.1.4, we will look at how to undo the smearing effect the CIR has on the transmitted signal. In Section 7.1.5, we will look at one technique employed extensively in OFDM that inverts the CIR effects in the frequency domain.

7.1.4 OFDM with Cyclic Prefix

The CIR can be modeled as a finite impulse response filter that is convolved with a sampled version of the transmitted signal. As a result, the CIR smears past samples onto current samples, which are smeared onto future samples. The effect of this smearing causes distortion of the transmitted signal, increasing the aggregate BER of the system and resulting in a loss in performance.

Although equalizers can be designed to undo the effects of the channel, there is a tradeoff between complexity and distortion minimization that is associated with the choice of an equalizer. In particular, the distortion due to the smearing of a previous OFDM symbol onto a successive symbol is a difficult problem. One simple solution is to put a few “dummy” samples between the symbols in order to capture the intersymbol smearing effect. The most popular choice for these K dummy samples are the last K samples of the current OFDM symbol. The dummy samples in this case are known as a *cyclic prefix*, as shown in Figure 7.8(a).

Therefore, when the OFDM symbols with cyclic prefixes are passed through the channel, the smearing from the previous symbols are captured by the cyclic prefixes, as shown in Figure 7.8(b). As a result, the symbols only experience smearing of samples from within their own symbol.

At the receiver, the cyclic prefix is removed, as shown in Figure 7.8(c), and the OFDM symbols proceed with demodulation and equalization.

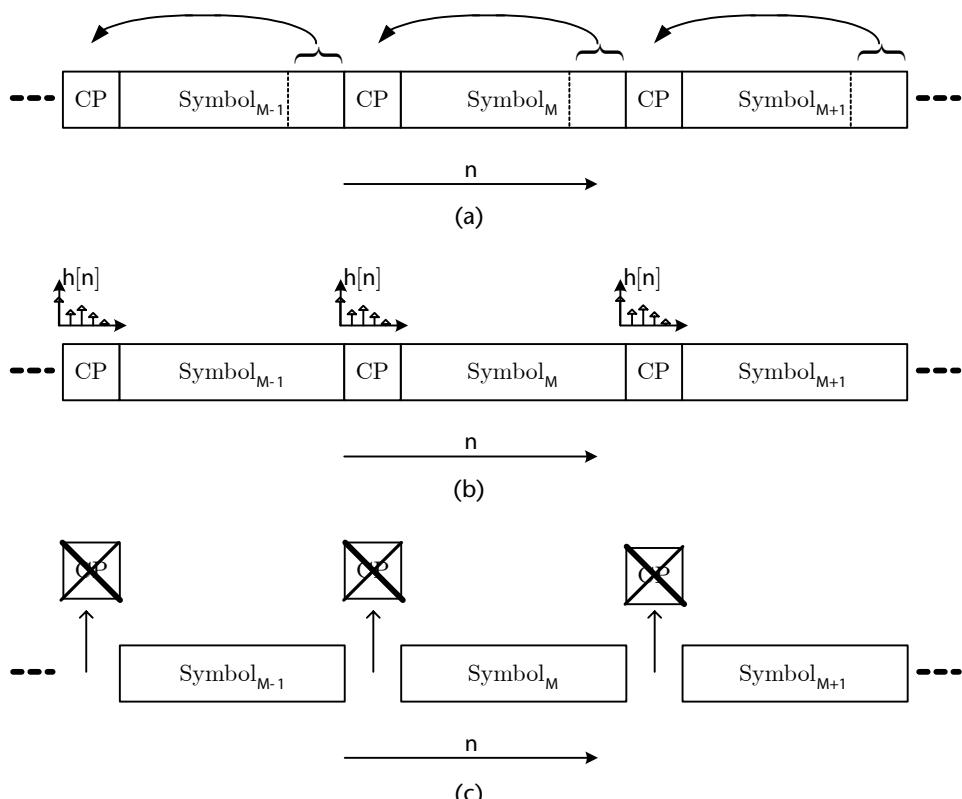


Figure 7.8 The process of adding, smearing capturing, and removal of a cyclic prefix. (a) Add cyclic prefix to an OFDM symbol. (b) Smear channel $h[n]$ from the previous symbol into the cyclic prefix. (c) Remove the cyclic prefix.

Despite the usefulness of the cyclic prefix, there are several disadvantages. First, the length of the cyclic prefix must be sufficient to capture the effects of the CIR. If not, the cyclic prefix fail to prevent distortion introduced from other symbols. The second disadvantage is the amount of overhead introduced by the cyclic prefix. By adding more samples to buffer the symbols, we must send more information across the channel to the receiver. This means to get the same throughput as a system without the cyclic prefix, we must transmit at a higher data rate.

7.1.5 Frequency Domain Equalization

Once the cyclic prefix has taken care of the intersymbol interference, the received signal has the DFT applied to it. However, smearing between samples still exists and must be compensated for. This is achieved by multiplying the subcarriers with the inverse of the channel frequency response.

The interset of the cyclic prefix resides in the fact that it actually transforms the linear convolution between the transmitted signal $s[n]$ and the channel impulse response $h[n]$ into a symbol-by-symbol circular convolution. To clearly see this, let us take a closer look at a given OFDM symbol with a cyclic prefix, the symbol starting at time $n = 0$. Denoting by $s[0], \dots, s[2N - 1]$ the $2N$ samples of output of the transmitter IDFT for the first OFDM symbol, the addition of the cyclic prefix gives rise to a new signal, namely,

$$\bar{s}[n] = \begin{cases} s[n + 2N - K] & 0 \leq n \leq K - 1 \\ s[n - K] & K \leq n \leq 2N - 1 \end{cases}$$

Denoting now by $\bar{r}[n]$ the result of the convolution of the signal $\bar{s}[n]$ with the length- L channel impulse response $h[n]$, one has that

$$\begin{aligned} \bar{r}[n] &= \sum_{k=0}^{L-1} h[k] \bar{s}[n - k] \\ &= \begin{cases} \sum_{k=0}^{n-K} h[k] s[n - K - k] + \sum_{k=n-K+1}^{L-1} s[n - k + 2N - K] & K \leq n \leq K + L - 1 \\ \sum_{k=0}^{L-1} h[k] s[n - K - k] & K + L \leq n \leq 2N - 1 \end{cases} \end{aligned}$$

From this equation, it is rapidly seen that, after removal of the cyclic prefix, the received sequence $r[n] = \bar{r}[n + K]$ is

$$r[n] = \sum_{k=0}^{2N-1} h[k] s[((n - k))_{2N}] = h[n] \circledast s[n] \quad (7.6)$$

Thus, the received samples, after removal of the cyclic prefix, are just made up of the circular convolution of the sent signal (i.e., $2N$ samples per symbol) with the channel impulse response $h[n]$. If one now looks at (7.6) in the frequency domain, it comes that

$$R[k] = H[k] \cdot S[k]$$

where capital letters represent $2N$ -point DFTs of the corresponding sequences. Referring back to Figure 7.5, the $2N$ -point DFT $R[k]$ of the received samples is already computed and is denoted by $\bar{p}_k[\ell]$.

Referring to Figures 7.6 and 7.7(b), if we consider the multiplication of the corresponding frequency samples together, we notice that each of the subcarriers experiences a different channel “gain” $H[k]$. Therefore, what must be done is to multiply each subcarrier with a gain that is an inverse to the channel frequency response acting on that subcarrier. This is the principle behind *per tone equalization*. By knowing what the channel frequency gains are at the different subcarriers, one can use them to reverse the distortion caused by the channel by dividing the subcarriers with them. For instance, if the system has 64 subcarriers centered at frequencies $\omega_k = 2\pi k/64$, $k = 0, \dots, 63$, then one would take the CIR $h[n]$ and take its 64-point FFT, resulting with the frequency response $H[k]$, $k = 0, \dots, 63$. Then, to reverse the effect of the channel on each subcarrier, simply take the inverse of the channel frequency response point corresponding to that subcarrier,

$$W[k] = \frac{1}{H[k]} \quad (7.7)$$

and multiply the subcarrier with it.

7.1.6 Bit and Power Allocation

Resource allocation is an important aspect of OFDM design, since it determines the bit error rate (BER) performance of the OFDM system. Given a fixed bit rate and a transmitted power constraint, the BER can be minimized by properly allocating the bit and power levels over the subcarriers. In this section, we will introduce the theory and technique concerning the bit and power allocation.

7.1.6.1 Bit Allocation

Most OFDM systems use the same signal constellation across all subcarriers, as shown in Figure 7.9(a), where the commutator allocates bit groupings of the same size to each subcarrier. However, their overall error probability is dominated by the subcarriers with the worst performance. To improve performance, adaptive bit allocation can be employed, where the signal constellation size distribution across the subcarriers varies according to the measured signal-to-noise ratio (SNR) values, as shown in Figure 7.9(b), where the commutator allocates bit groupings of different sizes. In extreme situations, some subcarriers can be “turned off” or *nulled* if the subcarrier SNR values are poor.

Assuming that both the transmitter and receiver possess knowledge of the subcarrier SNR levels, one is able to determine the subcarrier BER values. Since the subcarrier BER values are dependent on the choice of modulation used for a given SNR value, we can vary the modulation scheme used in each subcarrier in order to change the subcarrier BER value. Larger signal constellations (e.g., 64-QAM) require larger SNR values in order to reach the same BER values as smaller constellations (e.g., 4-QAM), which have smaller SNR values.

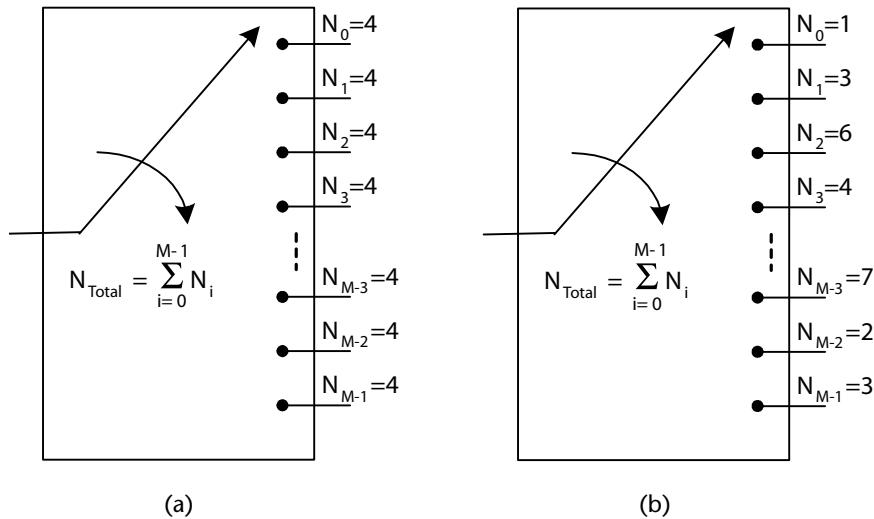


Figure 7.9 Comparison of variable and constant rate commutators with equivalent total rate.
 (a) Constant-rate commutator. (b) Variable-rate commutator.

We will simply allocate the number of bits b_i on subcarrier i according to

$$b_i = \log_2 \left(1 + \frac{\gamma_i}{\sigma^2} \right) \quad (7.8)$$

where γ_i is the SNR of subcarrier i (not in dB). Of course, (7.8) gives rise to noninteger numbers of bits. Round the resulting b_i s appropriately.

7.1.6.2 Power Distribution of the Transmitted Signal

Channel responses are not always linear. In particular, certain frequencies can be far more attenuated than others. It can be shown that the optimum power distribution for the subcarriers should be a constant K . Determining what this level should be is done through “water pouring,” or choosing the power at a frequency $P(f)$ such that all $P(f_1) = P(f_n)$.

$P(f)$ is given by

$$P(f) = \begin{cases} \lambda \frac{N(f)}{|H(F)|^2}, & f \in F \\ 0, & f \notin F \end{cases} \quad (7.9)$$

where $N(f)$ is the PDF of Gaussian noise, $H(f)$ is the transfer function representing a linear channel, and λ is the value for which:

$${}_F P(f)df = P \quad (7.10)$$

or the area under the curve in Figure 7.10. This water filling is done so that the probability of bit error is the same for all subcarriers.

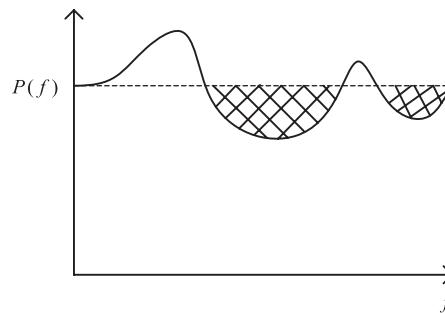


Figure 7.10 An illustration of the water filling principle. Notice how power is allocated to each subcarrier, such that the resulting power would be a constant K .

7.2 SOFTWARE IMPLEMENTATION

Using the theory that has been introduced in Section 7.1, we will now proceed to build our own implementation for one type of the MCM transmission system, namely, an OFDM system. Specifically, we will design and prototype an OFDM communication system using MATLAB, followed by its implementation in Simulink, both of which consist of a transmitter, a channel, and a receiver. With these implementations, we can observe the BER performance and the transmission spectrum of the OFDM system.

7.2.1 MATLAB Design of Multicarrier Transmission

In this section, we will design and simulate an OFDM communication system using MATLAB. When the system is completed, as shown in Figure 7.5, it should include the following main components:

- A *transmitter*, which includes commutator, QAM modulator, IFFT, and cyclic prefix add modules.
- A *channel*, which simulates the noise and the channel impulse response of an actual transmission environment.
- A *receiver*, which includes cyclic prefix remove, FFT, QAM demodulator, and commutator modules.

Since the complete OFDM system is a rather complicated structure, let us break down the whole system into several smaller pieces and start the implementation from these basic functions. For each function, after its implementation, we will evaluate it in order to make sure that it is functioning properly. This is a very useful and widely employed strategy for engineers when developing and testing a large scale, complicated system.

First of all, let us implement the rectangular QAM modulator as shown in Figure 7.1, given an arbitrary carrier frequency ω_k . The M -QAM modulator modulates every $b = \log_2 M$ binary bits into one of the M complex numbers. We can get a better idea of this modulation process by looking at the constellation plot, as shown in Figure 7.2. Corresponding to the QAM modulator, let us implement the rectangular QAM demodulator as shown in Figure 7.3, given an arbitrary

carrier frequency ω_k . The M-QAM demodulator converts the complex numbers to binary bits.

Q

1. Test the cascade of the QAM modulator and demodulator to see if the input is completely recovered at the output. Do you see any discrepancies?
2. Plot the frequency magnitude response of each single carrier transmitter after the channel and add them up in one plot. What does it look like?
3. Implement the constellations of 4-QAM, 16-QAM, and 64-QAM. Can you get the same results as shown in Figure 7.2?

When it comes to the channel, AWGN is generated to simulate the noise in the real transmission environment. Let us implement an AWGN generator that accepts as inputs the desired noise variance σ^2 , the noise mean m , and the number of random points R .

Q

1. Verify that AWGN generator works by creating a histogram of its output and compare against the Gaussian PDF as described by (7.11) for $\mu = 3$, $\sigma^2 = 2$.

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(x - \mu_n)^2}{2\sigma_n^2}} \quad (7.11)$$

where μ_n and σ_n^2 are the mean and variance.

2. Verify that rectangular QAM modulator/demodulator works by introducing the appropriate amount of noise and comparing the resulting BER with the BER in Figure 7.11 at 10^{-3} . What values of SNR did you employ?

Next, let us create several copies of rectangular QAM modulator at transmitter, as well as several copies of rectangular QAM demodulator at receiver. With the help of the commutators, we can implement Figure 7.4 and its corresponding receiver. At the transmitter, according to the OFDM algorithm, the high-speed digital input

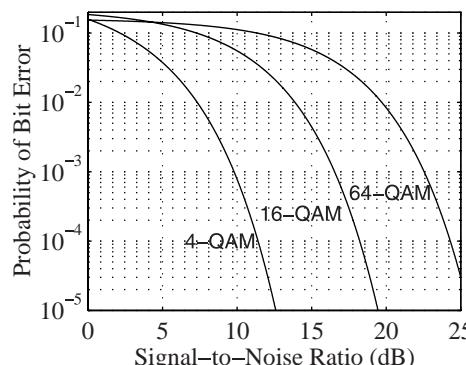


Figure 7.11 Probability of bit error curves for 4-QAM, rectangular 16-QAM, and 64-QAM.

$x[m]$ is demultiplexed into N subcarriers $d_k[m]$ using a commutator, in order to adjust the size of the signal to fit in the M -QAM modulator. While at the receiver, the commutator essentially performs parallel to serial conversion to regain the bit stream.



Verify that Figure 7.4 and its corresponding receiver work under ideal conditions. What carrier frequencies did you not use in the implementation.

Then, let us implement Figure 7.5 using IDFT and DFT blocks. In MATLAB, these two blocks can be realized using `ifft` and `fft` functions. Note that the size of the input/output should be a variable. At the transmitter, in order to get a real-valued time series $s[n]$, according to the IDFT properties, a $2N$ -point IDFT should be implemented. However, the output of the M -QAM modulator are Appoint complex numbers. Therefore, before going through the IDFT block, we need to rearrange and expand the N -point complex numbers to $2N$ -point as follows:

$$X[2N-k] = X[k], \quad \text{for } k = 1, 2, \dots, N-1 \quad (7.12)$$

where $X[k]$ are the $2N$ -point complex numbers and $X^*[k]$ are the $2N$ -point rearranged complex numbers. Note that $X[0]$ and $X[N]$ values can be assigned with any real numbers without affecting the IDFT output, which is also $2N$ -point. Since the FFT works well with sequences of length 2^n due to its structure and implementation, we zero pad the data in case we do not have this initial data format. At the receiver, the DFT block reverses the IDFT operation performed at the transmitter. Thus, the output of the DFT block are $2N$ -point complex numbers, N of which can be used to recover the output of M -QAM modulator on transmitting side.

Besides the AWGN, a channel filter should be employed here to simulate the CIR effect existing in a real-world communication channel, which can be modeled as a finite impulse response filter.

Implement the channel filter, given as an input the channel impulse response, and add the noise generator. Test your implementation with the following parameters, whose impulse response are shown in Figure 7.12:



$$\begin{aligned} b_1 &= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \\ b_2 &= [1 \quad 0.1 \quad 0.0001 \quad 0 \quad 0 \quad 0] \\ b_3 &= [1 \quad 0.3 \quad 0.4 \quad 0.32 \quad 0.2 \quad 0.1 \quad 0.05 \quad 0.06 \quad 0.02 \quad 0.009] \end{aligned} \quad (7.13)$$

with $\mu = 0$ and $\sigma^2 = 0.001$.

The last component we need to take into account is the cyclic prefix. Let us implement the cyclic prefix add and remove blocks of the OFDM system, as shown in Figure 7.8, with the length of the cyclic prefix an input variable. At the transmitter, cyclic prefix is added to eliminate the intercarrier interference (ICI) caused by

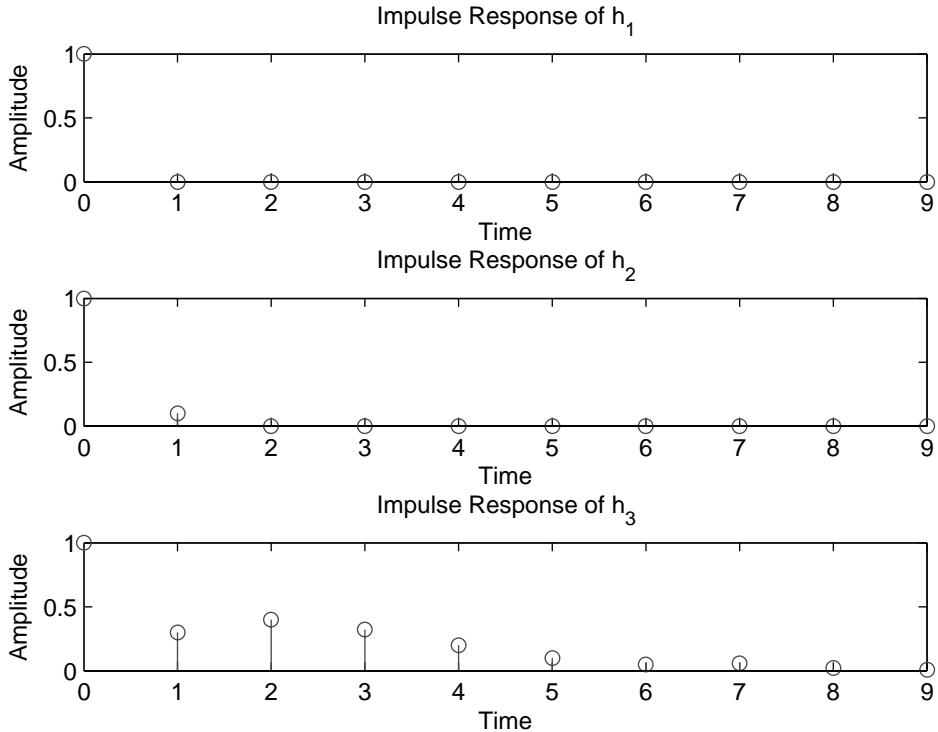


Figure 7.12 The impulse responses of three channels, which will result in three different ICI effects.

multipath propagation effects and the frequency-selective nature of the communications channel that is generally attributed to this real-world phenomenon. This selectivity can be represented by a channel impulse response, which will cause distortion of the transmitted signal and increase the aggregate BER of the system. For example, (7.13) represents three different channel impulse responses, which will result in three different ICI effects. At the receiver, cyclic prefix section is removed from the received data to recover the valid message. The cyclic prefix is no longer necessary here since it was added at the transmitter to avoid ICI and has thus served its purpose.

With the system completely implemented, we can now study how it performs in a variety of conditions. The performance measure commonly used in digital transmission systems is the BER. Given an SNR value and channel conditions, the BER is the fraction of bits that are errors to the total number of bits transmitted. The only question is “how many bits need to be transmitted in order to obtain a reliable BER?” The rule-of-thumb answer is that one continues transmitting bits until 100 errors have been reached [10].

7.2.2 Simulink Design of OFDM

In the previous section, we implemented an OFDM system in MATLAB and observed the BER performance. Since many of the MATLAB functions have their corresponding blocks in Simulink, we can try out the OFDM design in Simulink to see whether it simplifies our design and implementation. For this purpose, let us down-

load and open the `ofdm.mdl` file. The example OFDM transmitter and receiver has already been implemented and provided, as shown in Figure 7.13. We will finish the rest of the model with the following steps and experiment with the parameters.

Q

- With your implementation, plot the BER curves for the three channel realizations in (7.13) between BER values of 10^{-2} and 10^{-4} . In particular, simulate your system when it uses $N = 16$, 32, and 64 subcarriers and employs a cyclic prefix of length $N/4$. The high speed input $x[m]$ is a uniformly distributed. Plot the BER results.
- Employ both constant-rate commutators, which either use BPSK, 4-QAM, 16-QAM, and 64-QAM, and the variable-rate commutator. Plot the BER results. Which approach attains the best performance? Please justify.

In order to finish the model, let us start off this design by getting the AWGN block from Simulink Library Browser and connecting the transmitter and receiver via AWGN block. Then, connect *Spectrum Scope* after the AWGN channel and connect the transmitter and receiver to the inputs of *Error Rate Calculation* block. By doing this, we have completed this model. Now, let us play around this model by changing some parameters.

Q

- Change the SNR value of the AWGN channel and observe the effect in the frequency domain via *Spectrum Scope*.
- Open the *OFDM Transmitter* and *OFDM Receiver* subsystems, change the number of subcarriers, and observe the effect in the frequency domain via *Spectrum Scope*.

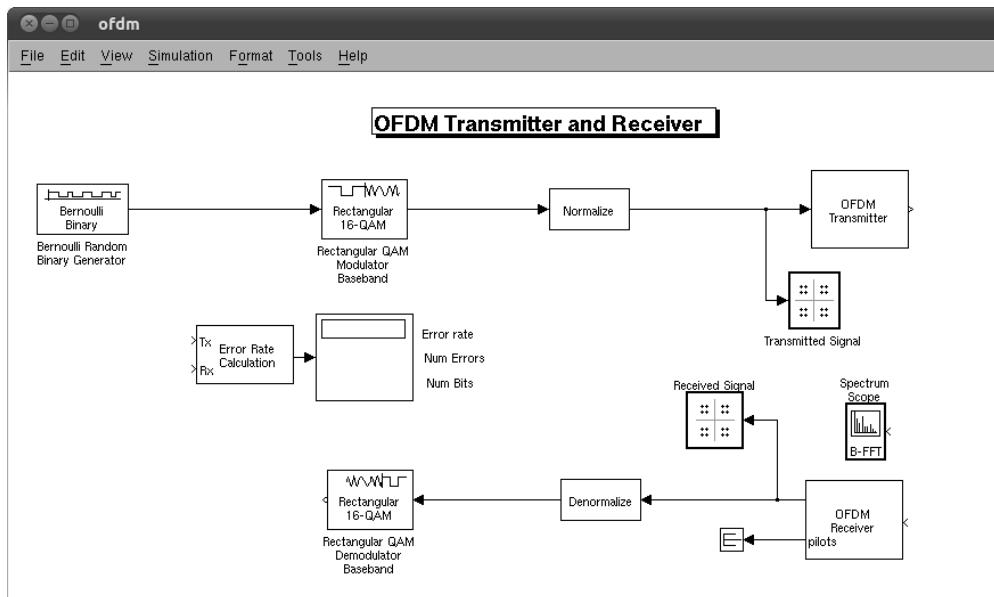


Figure 7.13 OFDM transmitter and receiver implementation in Simulink.

7.3 USRP HARDWARE IMPLEMENTATION

A *matched filter* is a theoretical framework that should not be mistaken for the name of a specific type of filter. It is a filtering process designed to take a received signal and minimize the effect of noise present in it. Hence, it maximizes the SNR of the filtered signal. It happens that an optimum filter does not exist for each signal shape transmitted, and it is a function only of the transmitted pulse shape. Due to its direct relationship to the transmitted pulse shape, it is called a matched filter. One commonly used matched filter design employed by many communication systems is the square root raised cosine filter.

In this section, we use two models to illustrate a typical setup in which a transmitter uses a square root raised cosine filter to perform pulse shaping and the corresponding receiver uses a square root raised cosine filter as a matched filter to the transmitted signal. At the receiver, we can plot an *eye diagram* from the filtered received signal, from which we can learn the effect of matched filter in a communication system.

7.3.1 Eye Diagram

In telecommunication, an *eye diagram*, also known as an *eye pattern*, is an oscilloscope display in which a digital data signal from a receiver is repetitively sampled and applied to the vertical input, while the data rate is used to trigger the horizontal sweep. It is so called because, for several types of coding, the pattern looks like a series of eyes between a pair of rails.

Several system performance measures can be derived by analyzing the display. If the signals are too long, too short, poorly synchronized with the system clock, too high, too low, too noisy, too slow to change, or have too much undershoot or overshoot, this can be observed from the eye diagram. For example, Figure 7.14 shows a typical eye pattern, where the x-axis represents timing and the y-axis represents amplitude. We can also obtain the information concerning time variation and amount of distortion from the eye diagram. In general, the most frequent usage of the eye pattern is for qualitatively assessing the extent of the ISI. As the eye closes, the ISI increases; as the eye opens, the ISI decreases.

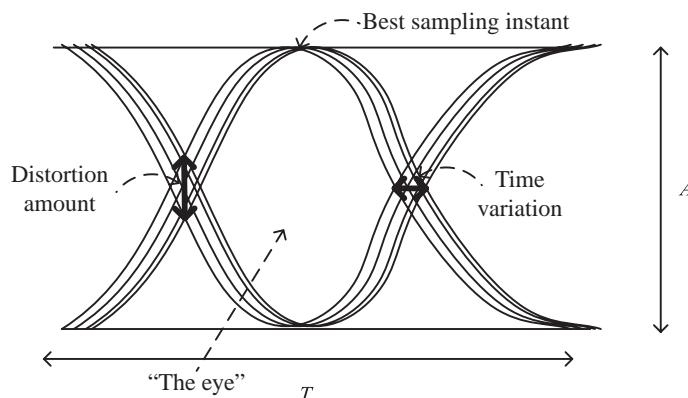


Figure 7.14 A typical eye pattern, where the x-axis represents timing and the y-axis represents amplitude.

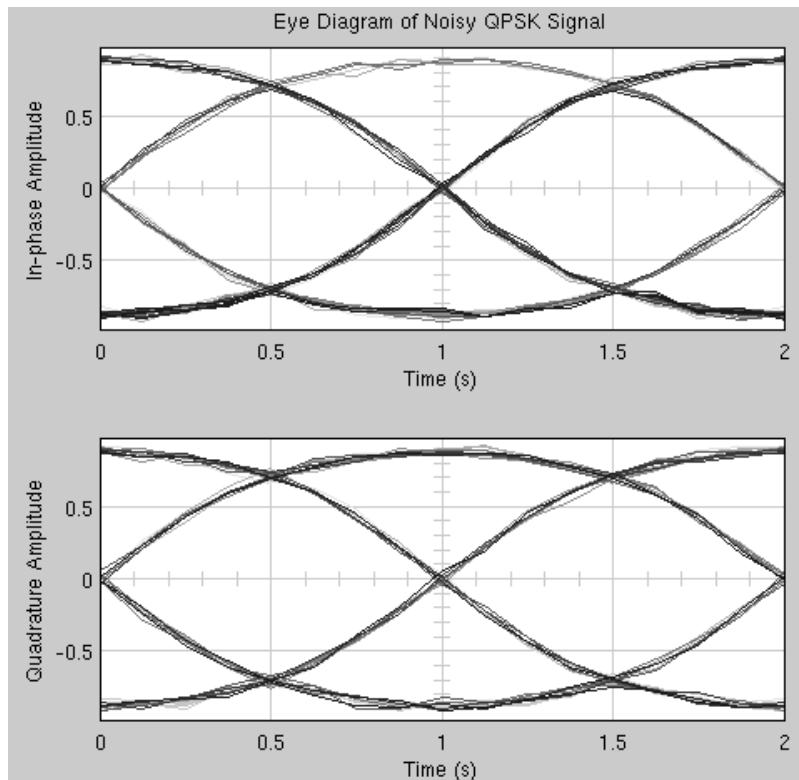


Figure 7.15 A typical eye pattern for the QPSK signal. The width of the opening indicates the time over which sampling for detection might be performed. The optimum sampling time corresponds to the maximum eye opening, yielding the greatest protection against noise. If there were no filtering in the system, then the system would look like a box rather than an eye.

7.3.1.1 Discrete-Time Eye Diagram Scope

The *Discrete-Time Eye Diagram Scope* block displays multiple traces of a modulated signal to produce an eye diagram. We can use this block to reveal the modulation characteristics of the signal, such as pulse shaping or channel distortions. This is a most important block that will be used in this section. For example, Figure 7.15 shows a typical eye pattern for the noisy QPSK signal. Generally speaking, an *open* eye pattern corresponds to minimal signal distortion. Distortion of the signal waveform due to intersymbol interference and noise appears as a *closure* of the eye pattern, which results in a significant probability that the received data might be incorrectly decoded.

7.3.2 Matched Filter Observation

In this section, we will use the DBPSK transmitter model on the transmitter side. On the receiver side, let us download and open the `rcf filter_receiver.mdl` file, as shown in Figure 7.16. The top level of this model looks similar to the DBPSK receiver model. However, its enabled subsystem contains the information that we are interested in.

For the first enabled subsystem, let us go into the *matched filter* subsystem, as shown in Figure 7.17, where there is a *Raised Cosine Receive Filter* and a *Discrete-Time Eye Diagram Scope*. The *Raised Cosine Receive Filter* is employed to be the matched

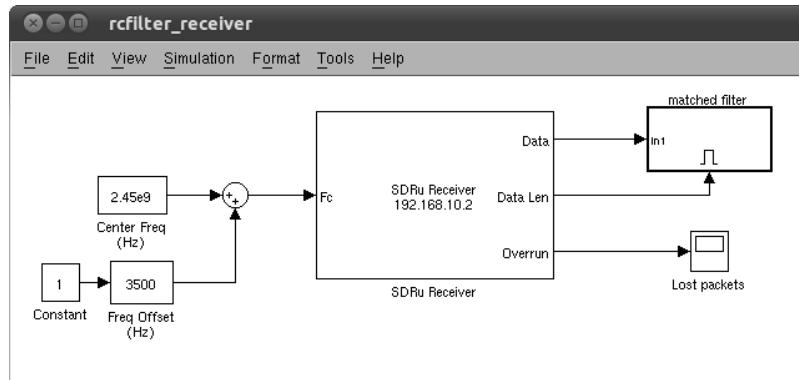


Figure 7.16 A model that illustrates a typical receiver setup, in which the receiver uses a square root raised cosine filter as a matched filter corresponding to the square root raised cosine filter on the transmitter.

filter for the *Raised Cosine Transmit Filter* on the transmitter. By double clicking the *Discrete-Time Eye Diagram Scope*, we can observe the eye diagram of our system.

Perform the following tasks and plot your observations.



1. Compare *Raised Cosine Receive Filter* and *Raised Cosine Transmit Filter* on the transmitter side, and see whether they match each other.
2. Change the parameters (group delay, rolloff factor, and so on) of the *Raised Cosine Receive Filter* to make it **match** the *Raised Cosine Transmit Filter*. How does the eye diagram appear? ^a
3. Change the parameters of the *Raised Cosine Receive Filter* to make it **mismatch** the *Raised Cosine Transmit Filter*. How does the eye diagram appear?
4. Delete the *Raised Cosine Receive Filter* block, as shown in Figure 7.18. How does the eye diagram appear?

^a To get better observation, you need to use “Autoscale” of eye diagram.

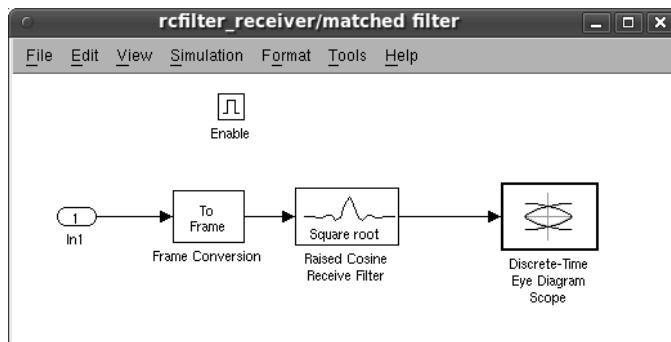


Figure 7.17 The subsystem of the receiver model, in which the receiver plots an *eye diagram* from the filtered received signal. Based on this diagram, we can learn the effect of matched filter in a communication system.

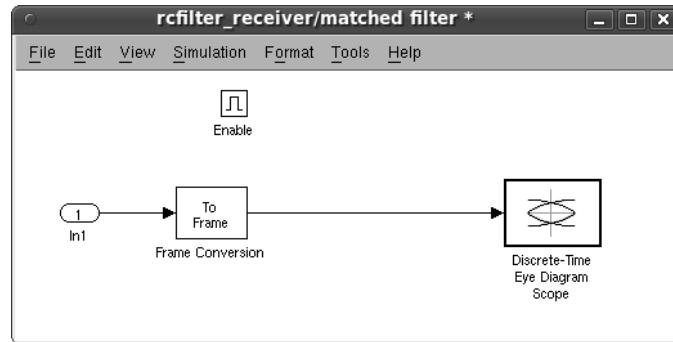


Figure 7.18 The subsystem of the receiver model without a *Raised Cosine Receive Filter*.

Please note that when the eye diagram has two widely opened “eyes,” it indicates the appropriate instants at which to sample the filtered signal before demodulating. It also indicates the absence of intersymbol interference at the sampling instants of the received waveform. A large SNR in the channel will produce a low-noise eye diagram. We can also construct a Simulink-only model using *AWGN Channel* block and then change the SNR parameter in the *AWGN Channel* block to see how the eyes in the diagram change.

7.4 OPEN-ENDED DESIGN PROJECT: DUPLEX COMMUNICATION

In this section, we will design a duplex single carrier communication system based on what we have accomplished thus far in this book. A duplex communication system is a system composed of two connected parties or devices that can communicate with one another in both directions. Duplex systems are often employed in many communications networks, either to allow for a communication “two-way street” between two connected parties or to provide a “reverse path” for the monitoring and remote adjustment of equipment in the field.

Systems that do not need the duplex capability use instead simplex communication. These include broadcast systems, where one station transmits and the others just “listen.” Several examples of communication systems employing simplex communications include television broadcasting and FM radio transmissions.

7.4.1 Duplex Communication

The models that we have been used so far in this book are operated in the simplex communication manner, such as the DBPSK transmitter and receiver models studied in the previous experiments. Since we are using the XCVR2450 transceiver daughterboard in our experiments described in this book, our USRP2 can only be either a transmitter or a receiver. However, we cannot employ both transmitter and receiver at the same time due to the design of this specific daughterboard. As described in [11], the XCVR2450 uses a common local oscillator (LO) for both receive and transmit, as shown in Figure 7.19. A change of one LO setting will be reflected in the other LO setting. Therefore, the XCVR2450 does not support full-duplex mode; attempting to operate in full-duplex will result in transmit-only operation.

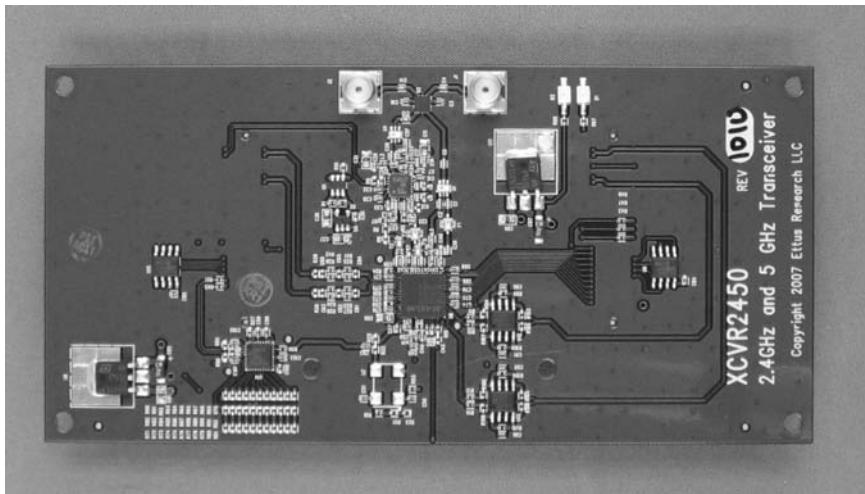


Figure 7.19 A photograph of XCVR2450 RF transceiver daughterboard (from [12]), where a common local oscillator is used for both receive and transmit.

Consequently, in this open-ended design project, we will implement a system that is capable of doing half-duplex communication.

7.4.2 Half-Duplex

A half-duplex (HDX) system provides communication in both directions, but only one direction at a time (not simultaneously). Typically, once a party begins receiving a signal, it must wait for the transmitter to stop transmitting before replying. An example of a half-duplex system is a two-party system such as a “walkie-talkie” style two-way radio, wherein one must use “over” or another previously designated command to indicate the end of transmission and ensure that only one party transmits at a time, because both parties transmit and receive on the same frequency. A good analogy for a half-duplex system would be a one-lane road with traffic controllers at each end. Traffic can flow in both directions, but only one direction at a time, regulated by the traffic controllers. There are several different ways to control the traffic in a half-duplex communication system, and we suggest you to use *time-division duplexing*.

7.4.3 Time-Division Duplexing

Time-division duplexing (TDD) refers to a transmission scheme that allows an asymmetric flow for uplink and downlink transmission that is more suited to data transmission. In a time-division duplex system, a common carrier is shared between the uplink and downlink, the resource being switched in time. Users are allocated one or more timeslots for uplink and downlink transmission, as shown in Figure 7.20.

For radio systems that aren’t moving quickly, an advantage of TDD is that the uplink and downlink radio paths are likely to be very similar. This means that techniques such as beamforming work well with TDD systems. Examples of time-division duplexing systems are as follows:

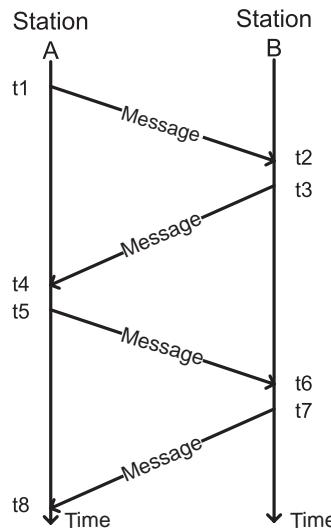


Figure 7.20 A half-duplex communication system using time-division duplexing. A common carrier is shared between station A and station B, the resource being switched in time.

- UMTS 3G supplementary air interfaces TD-CDMA for indoor mobile telecommunications [13];
- DECT wireless telephony [14];
- Half-duplex packet mode networks based on carrier sense multiple access [15]; for example, two-wire or hubbed Ethernet, wireless local area networks, and Bluetooth can be considered TDD systems;
- IEEE 802.16 WiMAX [16].

7.4.4 Useful Suggestions

Suppose we have a radio station A and a radio station B; let us perform the following tasks in time sequence: station A transmits “Hello world” to station B, station B transmits “Hello world” to station A, station A transmits “Hello world” to station B, station B transmits “Hello world” to station A, and so on, as shown in Figure 7.20.

At the beginning, let us try something simple to see whether it works. For example, we can transmit “1010”... instead of “Hello world,” such that frame synchronization is not necessary. We can test one cycle only, (i.e., station A transmits “Hello world” to station B and station B transmits “Hello world” to station A).

Then, in order to make it an automatic communication system, we need to write a MATLAB code for station A and another MATLAB code for station B, such that all we need to do is to click the “Run” button on two stations at the beginning of simulation. There are two useful MATLAB functions in this step. A MATLAB function called `sim` allows us to run our Simulink model from MATLAB code. Another MATLAB function called `pause` allows us to specify the interval between two actions in our MATLAB code. This function is necessary due to the different computation ability of PCs. We want both of them get ready before continuing to the next step. For example, a very simple piece of code is as follows:

```
sim('tx'); pause(10); sim('rx');
```

According to this code, the radio station first runs a Simulink model called “tx,” then it pauses for 10-s and in the end it runs a Simulink model called “rx.”

At last, in order to specify the simulation time of our model, we can go to **Simulation Configuration Parameters Solver** tab, and change Stop time.

7.4.5 Evaluation and Expected Outcomes

With individual models and MATLAB scripts for both stations completed, we can proceed to the testing process. Let us first start the two scripts at the same time, which makes station A initiate transmission and station B begin listening. After a certain period of time, station B will evaluate the received message and send a message back if it has received the correct message. In this way, the two stations will relay the message back and forth to each other, demonstrating a TDD half-duplex communication system. The most important task in this process is to find out the pause time for each station between the transmitter mode and the receiver mode. An example of the status output at station B is displayed in Figure 7.21. According to the figure, we can see that station B received the correct message for four times before it received a wrong message. When the wrong message is received, the script stops executing, and the communication process ends.

```
Command Window
(i) New to MATLAB? Watch this Video, see Demos, or read Getting Started.
listeing...
message =
Hello world
transmitting...
listeing...
message =
Help xpsme
wrong message, exit!
fx >>
```

Figure 7.21 An example of the status output at station B, where it shows the switch between the transmitter mode and the receiver mode as well as the received messages.

7.5 CHAPTER SUMMARY

This chapter focus on the principle and implementation of multicarrier modulation. In this chapter, the theory of multicarrier modulation is first introduced. Then, we implement the multicarrier transmission in both MATLAB and Simulink. In the end, the open-ended design project realizes a duplex communication system.

7.6 PROBLEMS

1. [Orthogonal Carriers] In Section 7.1.1.1, prove that the carriers $\cos(\omega_k n)$ and $\sin(\omega_k n)$ are orthogonal over the symbol period.¹ Are there any values of k that are exceptions? Why?
2. [Carrier Frequency] Prove that (7.2) and (7.3) are true. What are the exceptions for the carrier frequencies for perfect recovery under ideal conditions? Why is k constrained to be below N ?
3. [QAM Demodulator] Draw a schematic for an orthogonally multiplexed QAM demodulator as in Figure 7.4. Show that the subcarriers of the orthogonally multiplexed QAM system are orthogonal.
4. [2N-Point IDFT] In Section 7.1.2, why are $k = 0$ and $k = N$ “don’t care” inputs?
5. [OFDM Configuration] Show that the OFDM configuration and the orthogonally multiplexed QAM system are identical. Which of the two implementations is faster if the OFDM system employs FFTs? Prove it by comparing the number of operations.
6. [Cyclic Prefix Remove] Explain (7.6) in detail, and prove that $r[n] = h[n] \underbrace{s[n]}_{(2N)}$.
7. [Bit Allocation] Based on Figure 7.7(b), explain clearly how different subcarriers experience different SNR values and thus different BERs. Explain how bit allocation can be used advantageously.
8. [Subcarrier SNR] Find the expression for the subcarrier SNR, given the channel impulse response, the subcarrier transmit power levels, and the noise variance σ^2 .
9. [Multicarrier Transceiver] Suppose we are given a generic multicarrier transceiver as shown in Figure 7.22, where we have selected $N = 4$ for this problem. The frequency (magnitude) response of the synthesis filter $g^{(0)}(n)$ is shown in Figure 7.23(a), and the frequency responses of the synthesis filters $g^{(1)}(n)$, $g^{(2)}(n)$ and $g^{(3)}(n)$ are simply frequency shifted versions of $g^{(0)}(n)$ centered at $\pi/2$, π , and $3\pi/2$. Furthermore, the corresponding analysis filters possess the same frequency (magnitude) responses. Finally, we have arbitrarily defined the signal spectra for $d^{(0)}(n)$, $d^{(1)}(n)$, $d^{(2)}(n)$, and $d^{(3)}(n)$ in terms of the spectral shapes shown in Figures 7.23(b)–7.23(e).
 - (a) Initially assuming an ideal transmission channel between the transmitter and the receiver, sketch across the frequency range $[0, 2\pi]$ the signal
1. Two real sequences $b[n]$ and $g[n]$ are said to be orthogonal over a period 0 to P if $\sum_{n=0}^{P-1} b[n]g[n] = 0$.

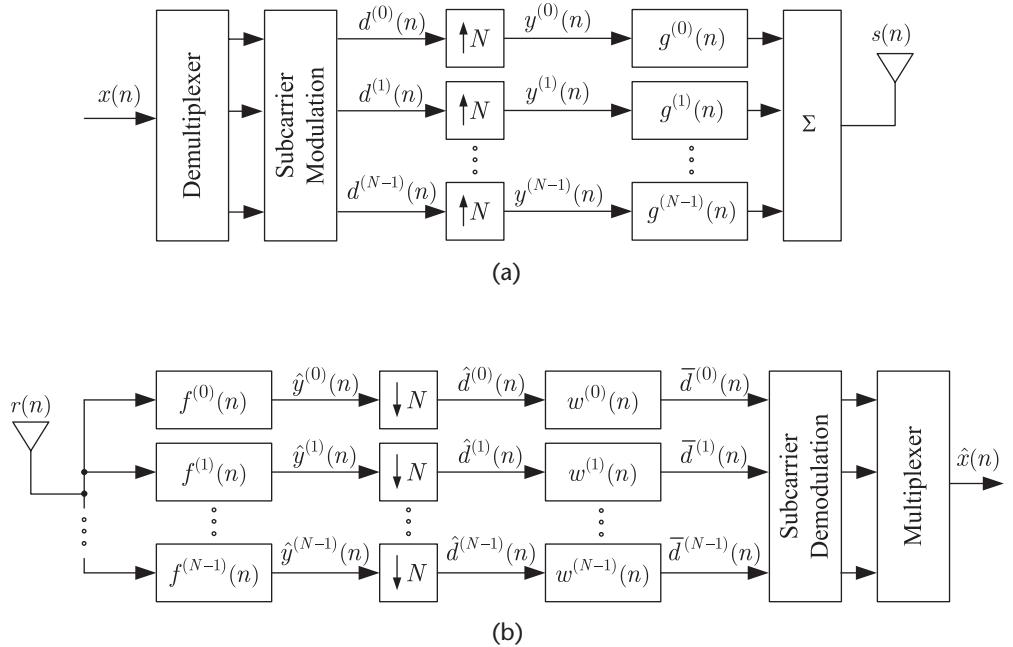


Figure 7.22 Schematics of generic multicarrier modulation employing synthesis and analysis filter-banks. (a) Generic multicarrier transmitter. (b) Generic multicarrier receiver.

spectra for the following: (a) upsampled signal $y^{(0)}(n)$, (b) output of the synthesis filter $g^{(0)}(n)$, (c) composite signal $s(n)$, (d) output of the analysis filter $f^{(1)}(n)$, (e) downsampled signal $\hat{d}^{(1)}(n)$.

NOTE: Please make sure to properly indicate values of the x -axis.

- (b) Suppose now that a frequency selective channel exists between the transmitter and receiver, as shown in Figure 7.23(f). Sketch the resulting signal spectra at the input to the multicarrier receiver, namely, $r(n)$. If each of the subcarrier equalizers $w^{(0)}(n)$, $w^{(1)}(n)$, $w^{(2)}(n)$, and $w^{(3)}(n)$ consist of only one coefficient, what should be its values and why?
HINT: Refer to Figure 7.23(f) and the frequency response values at the center of each subcarrier.
- (c) Suppose that the average frequency attenuation per subcarrier is equal to:

$$\mathbf{C} = \{C_i\}_{i=1}^8 = \{0.5, 0.25, 0.05, 0.75\}$$

the noise spectral density is equal to $N_0 = 10^{-6}$, the transmit power level for each subcarrier is $P_i = 1 \times 10^{-3}$, and the SNR gap is equal to $= 20$. We understand that the number of bits that can be allocated to subcarrier i is equal to:

$$b_i = \log_2 1 + \frac{\gamma_i}{N_0} \div, \quad i = 1, \dots, N \quad (7.14)$$

where the subcarrier signal-to-noise ratio (SNR) is equal to:

$$\gamma_i = \frac{P_i |C_i|^2}{N_0} \quad (7.15)$$

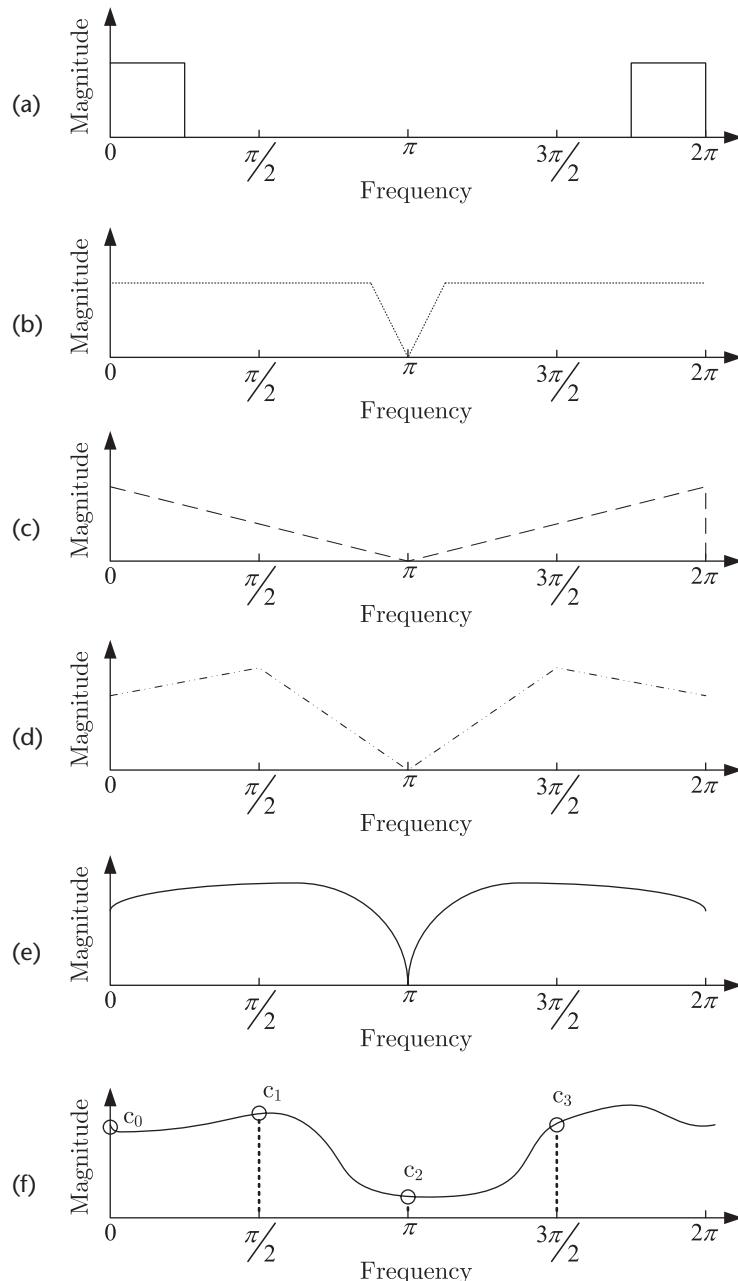


Figure 7.23 Sketches of the synthesis filter $g^{(0)}(n)$, all four signal spectra, and the channel response. (a) Frequency response for synthesis filter $g^{(0)}(n)$. (b) Signal spectrum for $d^{(0)}(n)$. (c) Signal spectrum for $d^{(2)}(n)$. (d) Signal spectrum for $d^{(2)}(n)$. (e) Signal spectrum for $d^{(3)}(n)$. (f) Frequency response of the wireless channel.

What is the bit allocation across all these subcarriers rounded to the nearest integer? What is the average quantization error between the optimal (real) b_i values and the rounded (integer) values across all the subcarriers?

- (d) Finally, suppose now that one of two possible modulation schemes can be employed per subcarrier, where modulation scheme 1 (i.e., MS1)

transmits two bits per symbol epoch while modulation scheme 2 (i.e., MS2) transmits four bits per symbol epoch. Using the information provided in part (c), the probability of bit error for each subcarrier when employing MSI is $P_{e,MS1} = \{7 \times 10^{-7}, 2 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-8}\}$ while for MS2 it is $P_{e,MS2} = \{9 \times 10^{-6}, 2 \times 10^{-5}, 1 \times 10^{-3}, 1 \times 10^{-6}\}$. In order to maximize the throughput given an average probability of bit error constraint of 1×10^{-5} , what modulation scheme should be used for each subcarrier? What is the maximum throughput? Please justify your answer.

HINT: When computing the average probability of bit error, remember to weigh each subcarrier quality by the number of bits being supported and divide by the total bits to be transmitted.

10. [Bit and Power Allocation]: Suppose we have an orthogonal frequency-division multiplexing (OFDM) communication system consisting of $N = 8$ subcarriers that is transmitting over a frequency selective fading channel possessing an average frequency attenuation per subcarrier equal to:

$$\mathbf{C} = \{C_i\}_{i=1}^8 = \{0.5, 1.0, 0.1, 0.25, 0.05, 0.75, 0.6, 0.2\}$$

and a noise spectral density of $N_0 = 10^{-8}$. The number of bits that can be allocated to subcarrier i is equal to:

$$b_i = \log_2 1 + \frac{\gamma_i}{N_0}, \quad i = 1, \dots, N \quad (7.16)$$

where the subcarrier signal-to-noise ratio (SNR) is equal to:

$$\gamma_i = \frac{P_i |C_i|^2}{N_0} \quad (7.17)$$

and P_i is the transmit power for subcarrier i .

- (a) Suppose that $P_i = 1 \times 10^{-5}$ for all subcarriers, and the SNR gap is equal to $= 20$. What is the bit allocation across all these subcarriers rounded to the nearest integer? What is the average quantization error between the optimal (real) b_i values and the rounded (integer) values across all the subcarriers?
- (b) Suppose $b_i = 3$ bits across all subcarriers given an SNR gap equal to $= 20$. What should be the approximate subcarrier power values P_i ?

References

- [1] Cioffi, J. M., “Asymmetric Digital Subscriber Lines,” C34 in *Communications Handbook*, CRC Press in Cooperation with IEEE Press, 1997.
- [2] Golden, P., H. Dedieu, and K. Jacobsen, *Fundamentals Of DSL Technology*, Auerbach Publications, 2004.
- [3] Golden, P., H. Dedieu, and K. Jacobsen, *Implementation And Applications of DSL Technology*, Auerbach Publications, 2007.
- [4] Hanzo, L. L., Y. Akhtman, L. Wang, and M. Jiang, *MIMO-OFDM for LTE, WiFi and*

- WiMAX: *Coherent versus Noncoherent and Cooperative Turbo Transceivers*, Wiley-IEEE Press, 2010.
- [5] Lee, B. G., and S. Choi. *Broadband Wireless Access and Local Networks: Mobile WiMax and WiFi*, Artech House, 2008.
 - [6] Pareek, D., *WiMAX: Taking Wireless to the Max*, Auerbach Publications, 2006.
 - [7] Nuaymi, L., *WiMAX: Technology for Broadband Wireless Access*, Wiley, 2007.
 - [8] Sesia, S., I. Toufik, and M. Baker, *Lte - The Umts Long Term Evolution: From Theory to Practice*, 2nd edition, Wiley, 2011.
 - [9] Cox, C., *An Introduction to LTE: LTE, LTE-Advanced, SAE and 4G Mobile Communications*, 2nd edition, Wiley, 2012.
 - [10] Jeruchim, M. C., P. Balaban, and K. S. Shanmugan. *Simulation of Communication Systems: Modeling, Methodology and Techniques*, 2nd edition, Springer, 2000.
 - [11] http://files.ettus.com/uhd_docs/manual/html/dbboards.html#xcvr-2450.
 - [12] <http://www.olifantasia.com/drupal2/en/node/12>.
 - [13] Muratore, F., *UMTS Mobile Communications for the Future*, Wiley, 2001.
 - [14] Phillips, J. A., and G. M. Namee. *Personal Wireless Communication with DECT and PWT*, Artech House, 1998.
 - [15] Spurgeon, C., *Ethernet: The Definitive Guide*, O'Reilly Media, 2000.
 - [16] Radha G. S. V., K. Rao, and G. Radhamani, *WiMAX: A Wireless Technology Revolution*, Auerbach Publications, 2007.

Spectrum Sensing Techniques

With advanced digital communication systems such as *cognitive radio* [1, 2] being used in a growing number of wireless applications, the topic of spectrum sensing has become increasingly important. This chapter will introduce the concept, the fundamental principles, and the practical applications of spectrum sensing. In the experimental part of this chapter, two popular spectral detectors will be implemented and their performance will be observed by applying them to several Simulink generated signals. This is followed by a USRP hardware implementation of a wideband spectrum sensing technique. Finally, the open-ended design project will focus on a commonly used medium access control (MAC) scheme referred to as *carrier sense multiple access with collision avoidance* (CSMA/CA), which heavily relies on spectrum sensing.

8.1 THEORETICAL PREPARATION

In recent years, a large portion of the assigned spectrum has been observed to be sparsely and sporadically utilized during several spectrum measurement campaigns [3, 4], as illustrated in Figure 8.1. In particular, spectrum occupancy by licensed transmissions are often concentrated across specific frequency ranges while a significant amount of the spectrum remains either underutilized or completely unoccupied. Therefore, *dynamic spectrum access* [6, 7] has been proposed for increasing spectrum efficiency via the real-time adjustment of radio resources using a combination of local spectrum sensing, probing, and autonomous establishment of local wireless connectivity among cognitive radio (CR) nodes and networks. During this process, *spectrum sensing* is employed for the purpose of identifying unoccupied licensed spectrum (i.e., spectral “white spaces”). Once these white spaces have been identified, secondary users (SU) opportunistically utilize these spectral white spaces by wirelessly operating across them while simultaneously not causing harmful interference to the primary users (PU).¹ Currently, there exists several techniques for spectrum sensing. This chapter will emphasize two of them, namely, the *energy detection* and *cyclostationary feature detection*.

8.1.1 Power Spectral Density

Although the concept of power spectral density has been introduced in Section 3.3.4, we will briefly revisit it in this section since it is a crucial concept in spectrum sensing. To analyze a signal in the frequency domain, the power spectral density (PSD), $S_{XX}(f)$,

1. Primary users are licensed users who are assigned certain channels, and secondary users are unlicensed users who are allowed to use the channels assigned to a primary user only when they do not cause any harmful interference to the primary user [8].

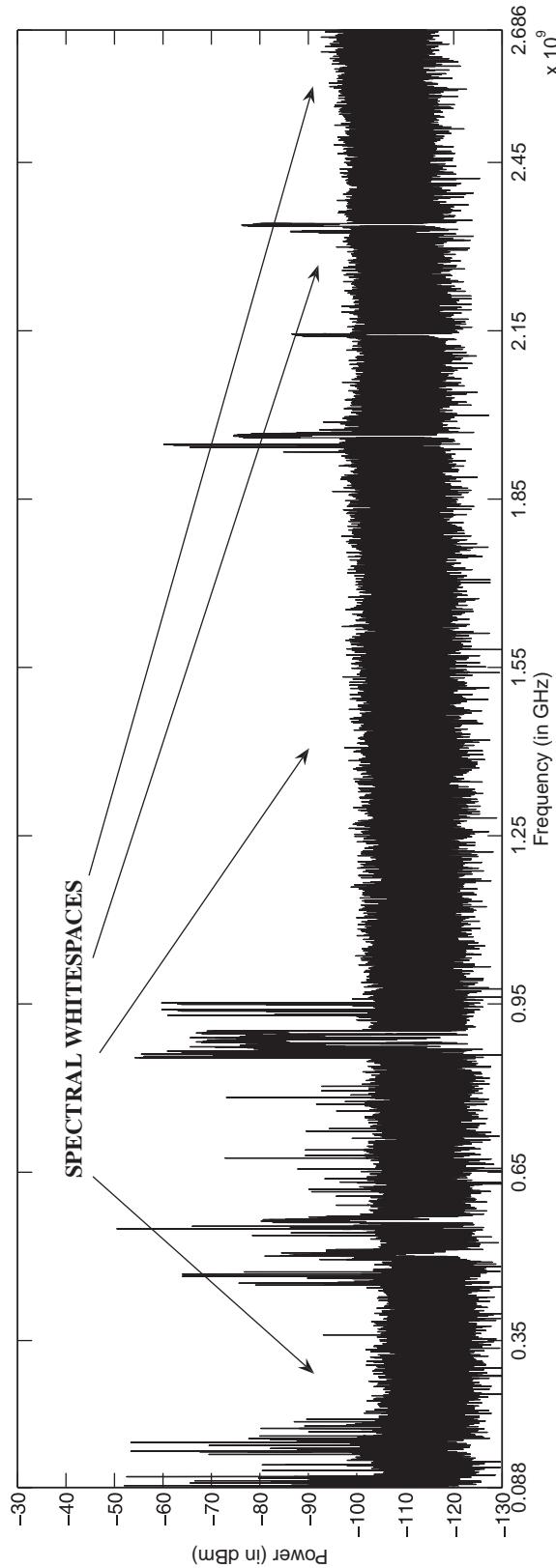


Figure 8.1 A power spectral density snapshot of wireless spectrum ranging from 88 MHz to 2686 MHz measured on July 11, 2008, in Worcester, MA, at coordinates $42^{\circ}16'N, 71^{\circ}48'14''W$ (from [5]).

is often used to characterize the signal, which is obtained by taking the Fourier transform of the autocorrelation $R_{XX}(\tau)$ of the WSS random process $X(t)$. The PSD and the autocorrelation of a function are mathematically related by the *Einstein-Wiener-Khinchin* (EWK) relations [9], namely:

$$S_{XX}(f) = \int_{-\infty}^{\infty} R_{XX}(\tau) e^{-j2\pi f\tau} d\tau \quad (8.1)$$

$$R_{XX}(f) = S_{XX}(\tau) e^{+j2\pi f\tau} d\tau \quad (8.2)$$

A very powerful consequence of the EWK relations is its usefulness when attempting to determine the autocorrelation function or PSD of a WSS random process that is the output of a linear time-invariant (LTI) system whose input is also a WSS random process. Specifically, suppose we denote $H(f)$ as the frequency response of an LTI system $h(t)$. We can then relate the power spectral density of input and output random processes with the following equation:

$$S_{YY}(f) = |H(f)|^2 S_{XX}(f) \quad (8.3)$$

where $S_{XX}(f)$ is the PSD of input random process and $S_{YY}(f)$ is the PSD of output random process. As we will see later, the experiments in Section 8.2 of this chapter are all based on this very useful relationship, as illustrated in Figure 8.2.

8.1.2 Practical Issues of Collecting Spectral Data

Although spectrum sensing is a rather intuitive process, its implementation possesses several engineering tradeoffs that can potentially impact the performance of the overall operation the results obtained. One of the key considerations when designing a spectrum sensing system is how to collect and store spectrum measurement data via a spectrum sweep. A spectrum analyzer can be employed to provide a nearly instantaneous snapshot of radio frequency bandwidth via the sampling of intercepted signals at a specified rate. For example, Figure 8.3(a) shows that Agilent CSA-N1996A spectrum analyzer is used to take spectrum measurements, and Figure 8.3(b) shows the PSD of a pulse-shaped QPSK signal collected by the spectrum analyzer.

There are several practical issues when parameterizing a spectrum sweep, such as the *sweep time* and *sweep resolution* (i.e., the speed and detail at which the spectrum sweeps are obtained). Higher resolution sweeps result in longer sweep time, but provide a more accurate measurement of the spectrum. The sweeps used

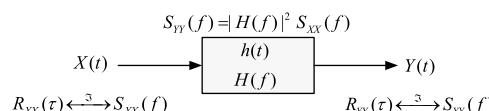
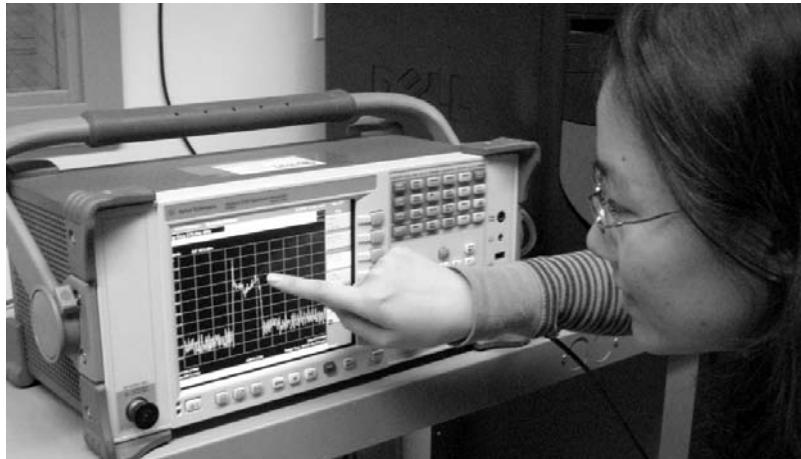
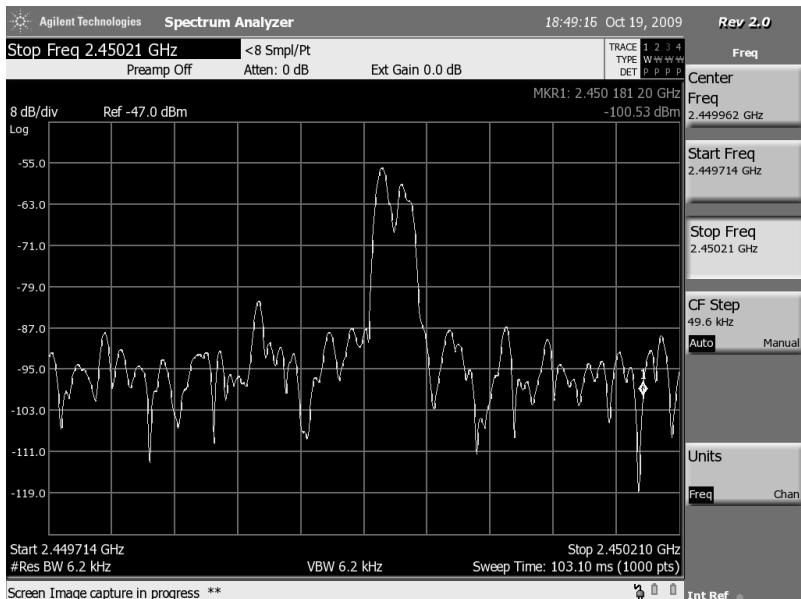


Figure 8.2 An example of how the an LTI system $h(t)$ can transform the PSD between the WSS random process input $X(t)$ and the WSS random process output $Y(t)$.



(a)



(b)

Figure 8.3 A spectrum analyzer is employed to provide a snapshot of radio frequency bandwidth. (a) Using Agilent CSA-N1996A spectrum analyzer to take spectrum measurements (from [10]). (b) Power spectral density of a pulse shaped QPSK signal, collected by an Agilent Technologies spectrum analyzer [11].

later in this chapter are the average of thousands of spectrum sweeps across a single bandwidth. Note that for the same measurement equipment, the speed at which spectrum measurements can be obtained varies from seconds to hours and days depending on the choice of several sweep parameters. However, the selection of these sweep parameters is heavily dependent on what signals are being observed, what sort of characteristics are being sought after in the spectrum measurements, and how the spectrum measurement information will be post-processed afterward.

Table 8.1 Five Parameters Used to Define Spectrum Measurement Processes

Parameter	Related Equation	Main Impacted Aspect
Sweep resolution (δ)	$\delta = \frac{B_1}{N_{FFT}}$	Accuracy
Window size (W)	$N_{step} = \frac{B}{W}$	Time
Dwell time (t_{dwell})	$t_{dwell} = \frac{t_{sweep}}{N_{sweep}}$	Time
Averaging sweeps	$\lim_{N \rightarrow \infty} \{E[n(t)]\} = 0$	Accuracy
Sweep time (t_{sweep})	$t_{sweep} = t_{dwell} \cdot N_{sweep}$	Time

As a result, Table 8.1 summarizes several parameters used to define spectrum measurement processes and the aspects of measurement they mainly affect. Then in the following subsection, we will understand how the choice of these parameters can directly impact the outcome of the measurements obtained.

8.1.2.1 Sweep Resolution

In order to provide a snapshot of the radio frequency bandwidth, a spectrum analyzer samples the intercepted signals at a specified rate. Considering a unit time frame, each sample in this frame displays a degree of freedom. Given a larger number of samples in this frame, more information of the intercepted signals can be represented. Thus, for an FFT-based spectrum analyzer, given the bandwidth of one single sweep, the sweep resolution is decided by the number of FFT points, which can be expressed as:

$$\delta = \frac{B_1}{N_{FFT}} \quad (8.4)$$

where B_1 is the bandwidth of one single sweep and N_{FFT} is the number of FFT points. Based on (8.4), we observe that a larger number of FFT points will result in a higher sweep resolution. For example, in Figure 8.4, there are two FFT plots of the same sine wave. Figure 8.4(a) is generated using a 32-point FFT, and Figure 8.4(b) is a 512-point FFT. Comparing these two plots, we find that 512-point FFT provides a higher resolution as well as a more accurate description of the spectrum than the 32-point FFT.

However, in order to handle the additional FFT samples, a spectrum analyzer that possesses a stronger computational processing ability is required, as well as a longer sensing time. Thus, there is an upper bound on the sweep resolution due to the limits of computational processing ability. On the other hand, a smaller FFT size may incur time-domain aliasing due to the undersampling in the frequency domain. Therefore, the input spectrum data is usually zero-padded to be consistent with the larger FFT size that provides the required resolution.

8.1.2.2 Window Size

The bandwidth of the spectrum we would like to analyze is usually larger than the bandwidth that one single sweep can support, so the concept of “window” needs to be introduced here. For each single sweep, the spectrum analyzer sweeps the size

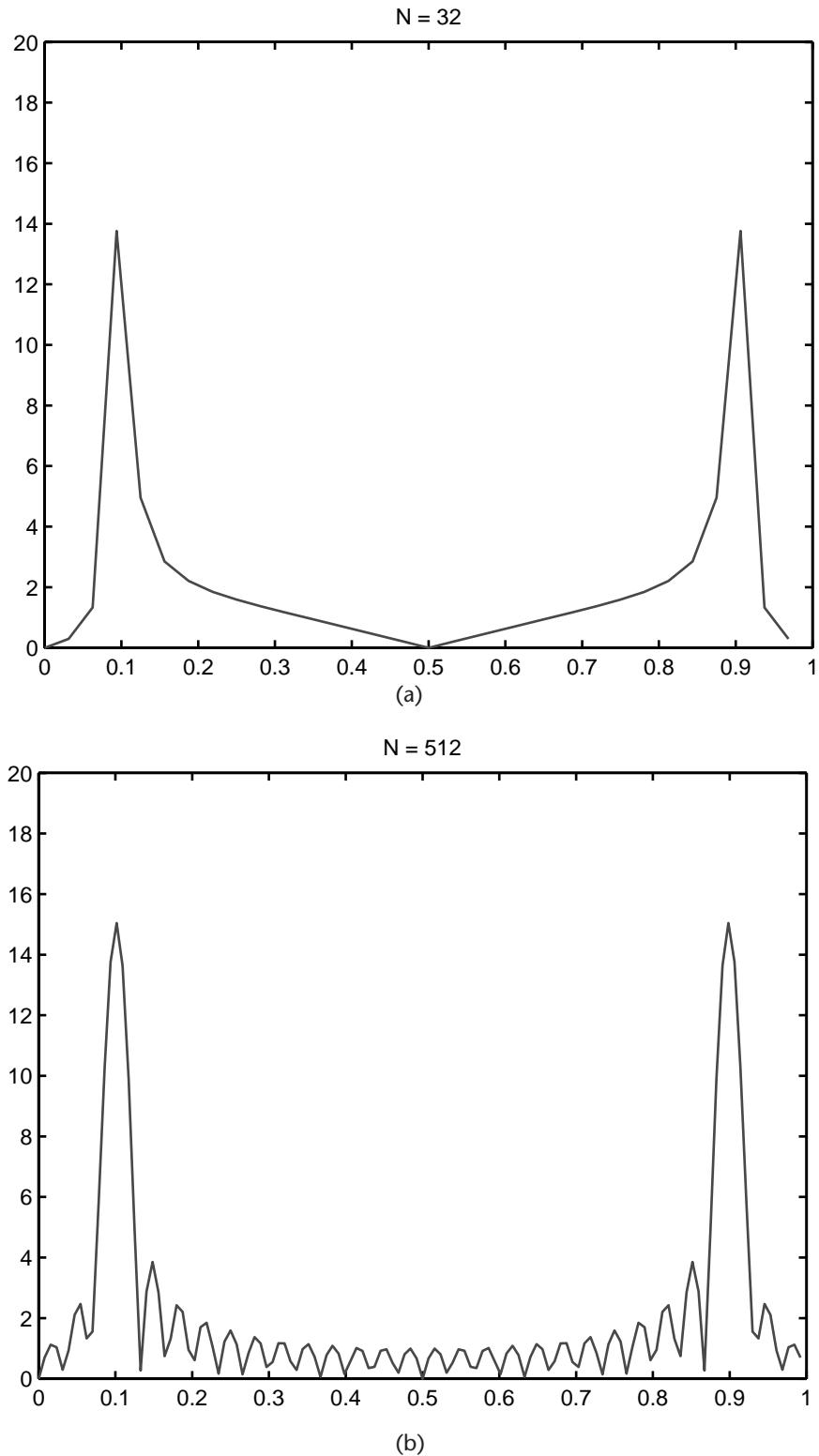


Figure 8.4 Two FFT plots of the same sine wave. (a) A 32-point FFT plot. (b) A 512-point FFT plot.

of a window, and then some post-processing approaches are employed to integrate these FFT results from the windows.

The bandwidth of the spectrum we would like to analyze can be calculated by:

$$B = f_{\text{stop}} - f_{\text{start}} \quad (8.5)$$

where f_{stop} and f_{start} are two input parameters to the spectrum analyzer, which specify the stop frequency and the start frequency. The total number of the sweeps to cover the whole bandwidth is referred to as frequency steps, N_{step} , defined as:

$$N_{\text{step}} = \frac{B}{W} \quad (8.6)$$

where B is the total bandwidth and W is the window size. For example, in Figure 8.5, the whole bandwidth is four times the window size, so the number of frequency steps is four.

Equation (8.6) implies that the larger the window size, the fewer frequency steps are required. However, according to (8.4), for a fixed FFT size the sweep resolution would become lower. In addition, the larger window size implies a higher sampling frequency that the processing computer must be able to keep up with. Thus, although a larger window size can effectively reduce the number of sweeps, it may also incur some other issues that need to be taken into account.

8.1.2.3 Dwell Time

Given the window size, the dwell time is the amount of time that the spectrum analyzer spends in each “window” or subband. This is one of the specifications provided by most commercial spectrum analyzers, since it is more likely to generate an accurate spectrum sensing result if enough time is spent in each subband. The dwell time can be adjusted by changing the number of samples per FFT point input parameter to the spectrum. If this input value is set as N , then N samples will be collected in order to calculate the FFT for each frequency point. It usually holds that the larger the dwell time, the larger the overall sweep time will be. These two factors can approximately be related by:

$$t_{\text{dwell}} = \frac{t_{\text{sweep}}}{N_{\text{sweep}}} \quad (8.7)$$

where t_{dwell} is the dwell time, t_{sweep} is the overall sweep time, and N_{sweep} is the total number of the sweeps.

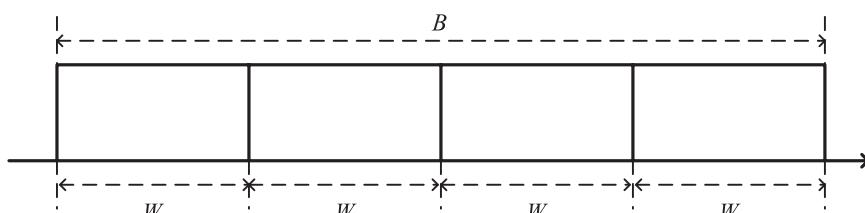


Figure 8.5 A simple example showing the relation between the total bandwidth B and the window size W .

8.1.2.4 Effect of Averaging Sweeps

When analyzing electro-magnetic spectrum, we usually sweep the spectrum several times and then take the average of these values. This is due to the fact that the averaging process can smooth out the spectrum. Since we usually assume the noise $n(t)$ to be an additive white Gaussian process with zero mean, as the number of samples approaches infinity, we will have:

$$\lim_{N \rightarrow \infty} \{E[n(t)]\} \rightarrow 0 \quad (8.8)$$

where $E[\cdot]$ is the expectation operator and N is the number of sweeps. For example, Figure 8.6(a) shows a single sine wave whose amplitude is 1 with the added white Gaussian noise. Figure 8.6(b) shows the result by accumulating 1000 sine waves in Figure 8.6(a) and taking their average. Comparing these two plots, we find that the effect of additive white Gaussian noise can be greatly eliminated by the averaging process.

In real-world applications, as the number of sweeps used in the averaging process increases, the noise level will be better captured. This can be used to estimate the noise statistics and would be very useful when determining the energy threshold for the SDR implementation of an energy detector.

8.1.2.5 Sweep Time

Given that the dwell time is the time spent in each subband, we define the sweep time as the time spent in the entire bandwidth of interest. It is determined by several factors, including the dwell time, window size, and the number of sweeps. For the first factor, increasing the dwell time usually increases the sweep time. For the second factor, increasing the window size decreases the number of frequency steps according to (8.6), and hence decreases the sweep time. For a fixed window size, although increasing the FFT size will result in a finer frequency resolution, the number of the frequency steps remains the same. Therefore, without accounting for the computational processing ability, the sweep time would be the same. However, in a real-world situation, with a larger FFT size, the resulting higher frequency resolution increases the computational processing time and hence the total sweep time. For the last factor, it is obvious that a larger number of the sweeps will result in a longer sweep time.

In conclusion, when collecting spectral data, we should consider all of the issues introduced here, in order to generate an accurate result, while at the same time be time efficient.

8.1.3 Hypothesis Testing

As mentioned at the beginning of Section 8.1, in dynamic spectrum access networks, spectrum sensing is employed for the purpose of identifying unoccupied licensed spectrum, which is equivalent to detecting the frequency locations of the primary signals. Therefore, spectrum sensing can be interpreted as a signal detection problem. Most signal detection problems can be formulated in the framework

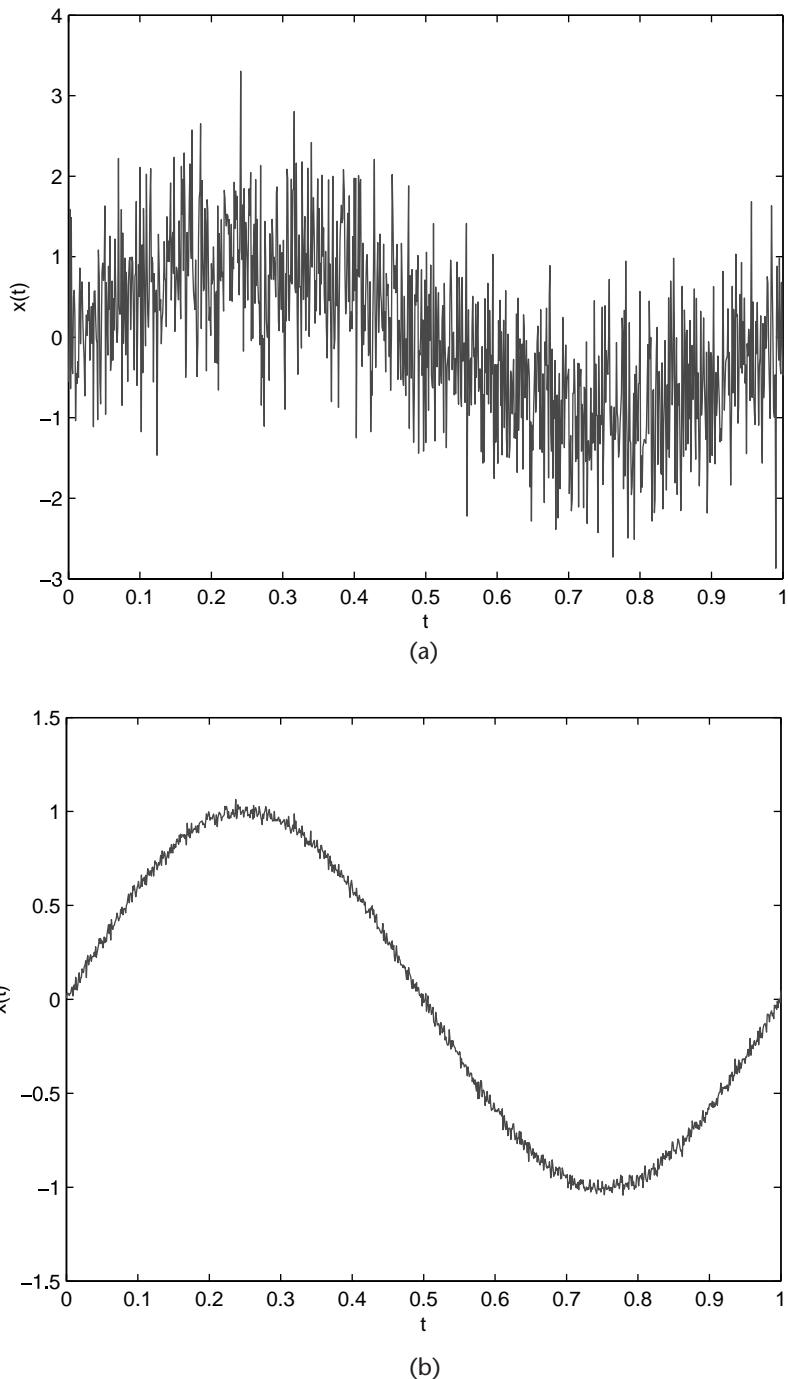


Figure 8.6 The sine wave with and without the averaging process. (a) A sine wave whose amplitude is 1 with the added white Gaussian noise. (b) Accumulating 1000 sine waves and taking their average.

of an M -ary hypothesis test, where we have an observation (possibly a vector or function) upon which we wish to decide among M possible statistical situations describing the observations [12]. According to this criterion, the spectrum sensing performs a binary hypothesis testing in order to decide whether or not there

are primary signals in a particular channel. The two hypotheses are denoted as follows:

$$\begin{aligned}\mathcal{H}_0 &: \text{no primary signals} \\ \mathcal{H}_1 &: \text{primary signals exist}\end{aligned}\quad (8.9)$$

where \mathcal{H}_0 is usually referred to as null hypothesis and \mathcal{H}_1 is usually called alternative hypothesis.

For a null hypothesis, since there are no primary signals present, the received signal is just the noise in the RF environment. On the other hand, for the alternative hypothesis, the received signal would be the superposition of the noise and the primary signals. Thus, the two hypotheses in (8.9) can be represented as:

$$\begin{aligned}\mathcal{H}_0 &: x[k] = n[k] \\ \mathcal{H}_1 &: x[k] = s[k] + n[k]\end{aligned}\quad (8.10)$$

for $k = 1, \dots, N$, where N is the number of received signals, $x[k]$ is the received signal, $n[k]$ is the noise in the RF environment, and $s[k]$ is the primary signal. Consequently, the spectrum sensing can be considered a detection problem, as based on the observation x , we need to decide among two possible statistical situations describing the observation, which can be expressed as:

$$\delta(x) = \begin{cases} 1 & x \in \Gamma_1 \\ 0 & x \in \Gamma_1^c \end{cases}\quad (8.11)$$

When the observation x falls inside the region Γ_1 , we will choose \mathcal{H}_1 . However, if the observation falls outside the region Γ_1 , we will choose \mathcal{H}_0 . Therefore, (8.11) is known as *decision rule*, which is a function that maps an observation to an appropriate hypothesis [12]. In the context of spectrum sensing, different spectral detectors and classifiers are actually the implementations of different decision rules. In Section 8.1.4, two decision rules will be introduced.

Regardless of the precise signal model or detector used, sensing errors are inevitable due to additive noise, limited observations, and the inherent randomness of the observed data [13]. In testing \mathcal{H}_0 versus \mathcal{H}_1 in (8.9), there are two types of errors that can be made, namely, \mathcal{H}_0 can be falsely rejected or \mathcal{H}_1 can be falsely rejected [12]. In the first hypothesis, there are actually no primary signals in the channel, but the testing detects an occupied channel, so this type of error is called a *false alarm* or *type I error*. In the second hypothesis, there actually exist primary signals in the channel, but the testing detects only a vacant channel. Thus, we refer to this type of error as a *missed detection* or *type II error*. Consequently, a false alarm may lead to a potentially wasted opportunity for the SU to transmit, while a missed detection could potentially lead to a collision with the PU [13].

Given these two types of errors, the performance of a detector can be characterized by two parameters, namely, the *probability of false alarm* (P_F), and the *probability of missed detection* (P_M) [14], which correspond to type I and type II errors, respectively, and thus can be defined as:

$$P_F = P\{\text{Decide } \mathcal{H}_1 | \mathcal{H}_0\} \quad (8.12)$$

and

$$P_M = P\{\text{Decide } \mathcal{H}_0 | \mathcal{H}_1\} \quad (8.13)$$

Note that based on P_M , another frequently used parameter is the *probability of detection*, which can be derived as follows:

$$P_D = 1 - P_M = P\{\text{Decide } \mathcal{H}_1 | \mathcal{H}_1\} \quad (8.14)$$

which characterizes the detector's ability to identify the primary signals in the channel, so P_D is usually referred to as the power of the detector.

As for detectors, we would like their probability of false alarm as low as possible, and, at the same time, their probability of detection as high as possible. However, in real-world situation, this is not achievable, because these two parameters are constraining each other. To show their relationship, a plot called *receiver operating characteristic* (ROC) is usually employed [15], as shown in Figure 8.7, where its x-axis is the probability of false alarm and its y-axis is the probability of detection. From this plot, we observe that as P_D increases, the P_F is also increasing. An optimal point that reaches the highest P_D and the lowest P_F does not exist. Therefore, the detection problem is also a tradeoff, which depends on how the type I and type II errors should be balanced.

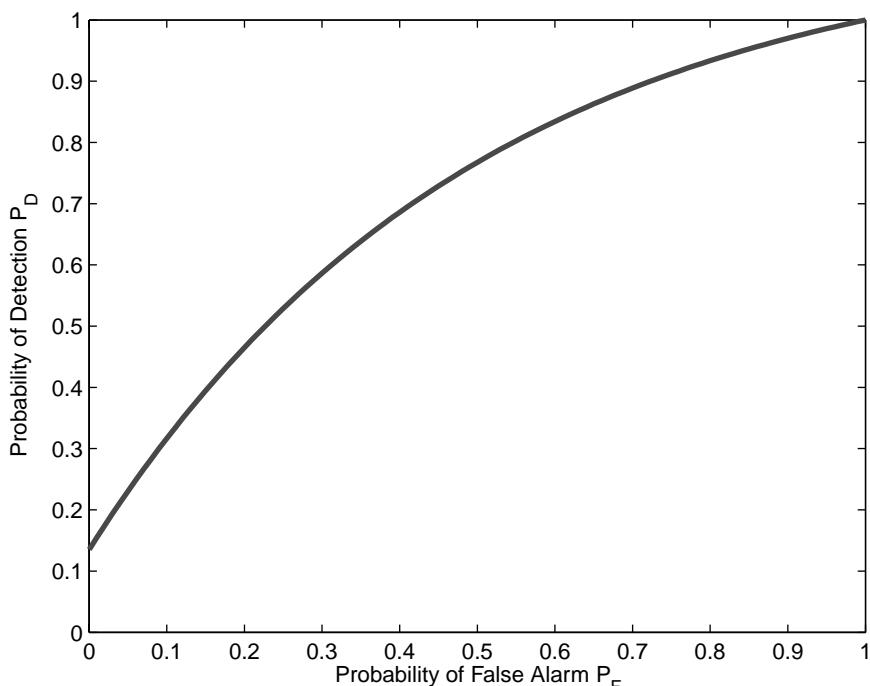


Figure 8.7 A typical receiver operating characteristic (ROC), where the x-axis is the probability of false alarm (P_F) and the y-axis is the probability of detection (P_D).

8.1.4 Spectral Detectors and Classifiers

The spectral detectors and classifiers are implementations of the different decision rules. For example, the energy detector in Section 8.1.4.1 assumes that the received signal will have more energy when the channel is occupied than when it is vacant; thus, the two decision regions in (8.11) are divided by an energy threshold. The cyclostationary feature detector in Section 8.1.4.2 is not only a detector, but also a classifier that recognizes the cyclic features of different modulation schemes in advance, so the decision regions are divided according to the cyclic features of the modulation schemes.

8.1.4.1 Energy Detector

Energy detection uses the energy spectra of the received signal in order to identify the frequency locations of the transmitted signal. This detection approach relies on only the energy present in the channel, and no phase information is required. Since the energy of a signal $x(t)$ is defined as:

$$E = \int |x(t)|^2 dt \quad (8.15)$$

a decision statistic for energy detector can be:

$$D = \sum_{k=1}^N (x[k])^2 \quad (8.16)$$

where $x[k]$ is the received signal and N is the total number of the received signals. The underlying assumption is that with the presence of a signal in the channel, there would be significantly more energy than if there was no signal present. Therefore, energy detection involves the application of a *threshold* T in the frequency domain, which is used to decide whether a transmission is present at a specific frequency, as shown in Figure 8.8. With any portion of the frequency band where the energy exceeds the threshold is considered to be occupied by a transmission, with the decision rule is as follows:

$$\delta(D) = \begin{cases} 1 & D > T \\ 0 & D < T \end{cases} \quad (8.17)$$

where D is the decision statistic calculated by (8.16) and T is the predefined energy threshold. Therefore, in Figure 8.8, any portion of the frequency band where the energy exceeds -100 dBm is considered an occupied channel.

Since different transmitters employ different signal power levels and transmission ranges, one of the major concerns of energy detection is the selection of an appropriate threshold. A threshold that works for one transmission may not be appropriate for another. Figure 8.9 shows two typical detection errors caused by inappropriate energy detection threshold. In Figure 8.9(a), the threshold is too low, so some noise is considered primary signals, resulting in a type I error,

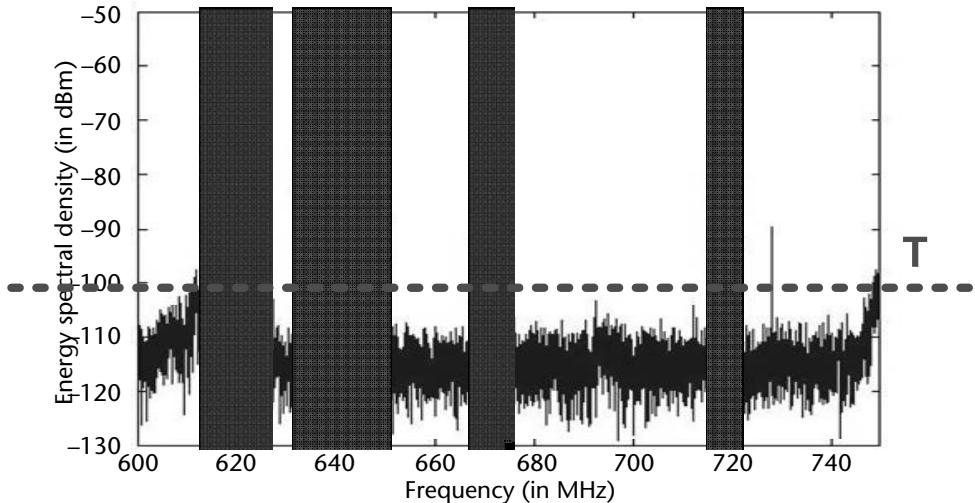


Figure 8.8 Energy detection threshold level, denoted as T . Any portion of the frequency band where the energy exceeds the threshold is considered occupied by a transmission.

false alarm. In Figure 8.9(b), the threshold is too high, so some primary signals are ignored, incurring the type II error, missed detection.

8.1.4.2 Cyclostationary Feature Detector

Modulation recognition and signal classification have been subject's of considerable research for more than over two decades. Classification schemes can generally be separated into one of two broad categories: likelihood-based (LB) approaches and feature-based (FB) approaches [16]. Cyclostationary feature detection is an FB technique based on the fact that communications signals are not accurately described as a stationary process, but rather more appropriately modeled as a cyclostationary process [17].

A cyclostationary signal possesses statistics that vary periodically with time. By definition, a signal $x(t)$ is wide-sense cyclostationary if its mean and autocorrelation are periodic [18], namely:

$$M_x(t) = M_x(t + T_0) \quad (8.18)$$

and

$$R_x(t, \tau) = R_x(t + T_0, \tau) \quad (8.19)$$

where $M_x(t)$ is the mean value of the signal $x(t)$ and $R(t, \tau)$ is the autocorrelation function of the signal $x(t)$. These periodicities occur for signals possessing well-defined characteristics due to several processes such as sampling, scanning, modulating, multiplexing, and coding, which can be exploited to determine the modulation scheme of the unknown signal [17].

The periodic nature of the signal allows it to be expressed as a Fourier series [18, 19]:

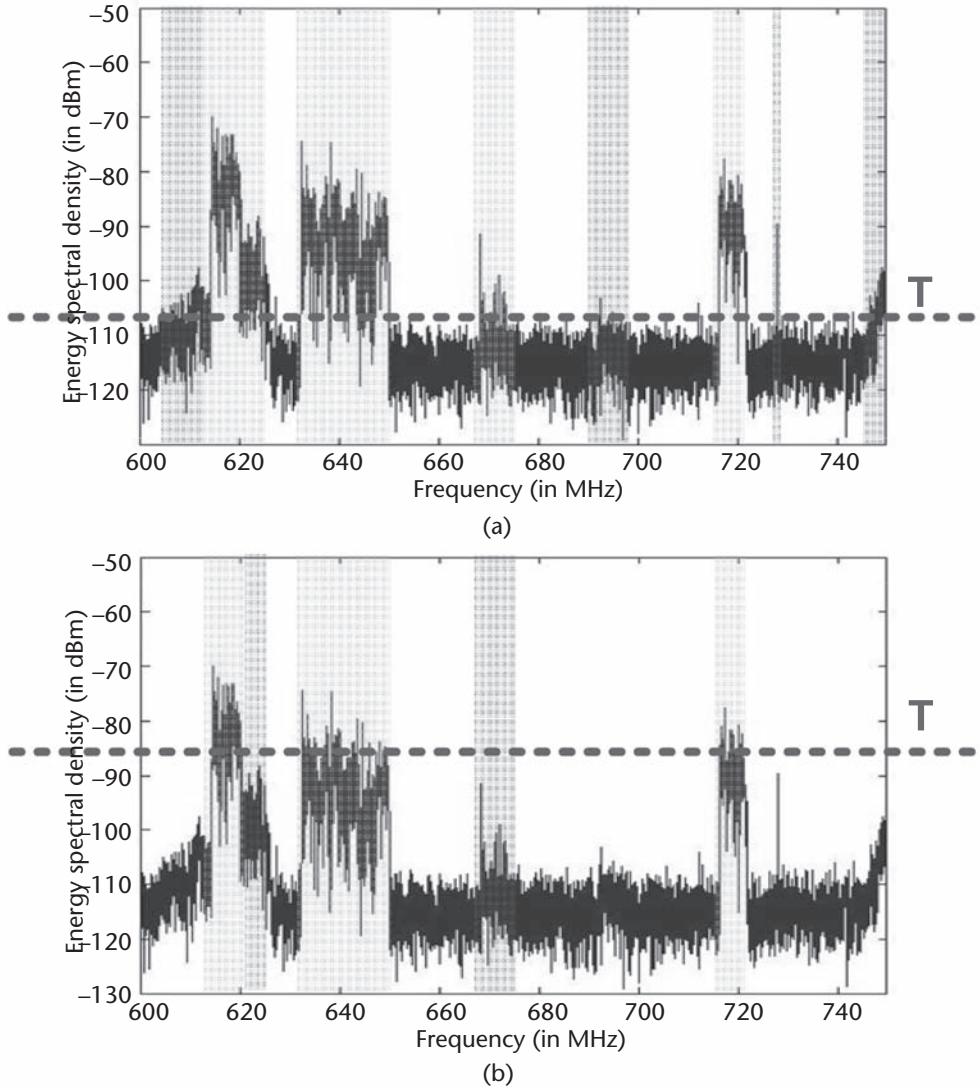


Figure 8.9 Inappropriate energy detection threshold levels. (a) Low energy detection threshold level yields type I error, *false alarm*. (b) High energy detection threshold level yields type II error, *missed detection*.

$$R_x(t, \tau) = E[x(t + \frac{\tau}{2})x^*(t - \frac{\tau}{2})] = \sum_{\{\alpha\}} R_x^\alpha(\tau) e^{j2\pi\alpha t} \quad (8.20)$$

where $E\{\cdot\}$ is the expectation operator, $\{\alpha\}$ is the set of Fourier components, and $R_x^\alpha(\tau)$ is the cyclic autocorrelation function (CAF) given by:

$$R_x^\alpha(\tau) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} R_x(t, \tau) e^{-j2\pi\alpha t} dt \quad (8.21)$$

Alternatively, in the case when $R_x(t, \tau)$ is periodic in t with period T_0 , (8.21) can be expressed as:

$$R_x^\alpha(\tau) = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} R_x(t, \tau) e^{-j2\pi\alpha t} dt \quad (8.22)$$

Consequently, the Fourier transform of the CAF, which is referred to as the spectral correlation function (SCF), is given by:

$$S_x^\alpha(f) = \int_{-\infty}^{\infty} R_x^\alpha(\tau) e^{-j2\pi f \tau} d\tau \quad (8.23)$$

which can be shown to be equivalent, assuming cycloergodicity, to the following expression [18]:

$$S_X^\alpha(f) = \lim_{T \rightarrow \infty} \lim_{t \rightarrow -\infty} \frac{1}{t} \int_{t/2}^{t+T/2} X_T(t, f + \frac{\alpha}{2}) \frac{dX_T(t, f)}{dt} dt \quad (8.24)$$

where $X_T(t, f)$ is the time varying Fourier transform defined as:

$$X_T(t, f) = \int_{t-T/2}^{t+T/2} x(u) e^{j2\pi f u} du \quad (8.25)$$

A significant advantage of the SCF is its lack of sensitivity to additive white noise, since modulated information is a cyclostationary process while noise is not. In other words, the spectral components of white noise are uncorrelated, so it does not contribute to the resulting SCF for any value of $\alpha \neq 0$. This property is especially useful when the noise power exceeds the signal power, which would make the signal undetectable when using a simple energy detector. As a result, cyclic detectors can successfully operate even in low SNR environments.

To derive a normalized version of the SCF, the spectral coherence function (SOF) is given by:

$$C_X^\alpha(f) = \frac{S_X^\alpha(f)}{[S_X^0(f + \alpha/2) \times S_X^0(f - \alpha/2)]^{1/2}} \quad (8.26)$$

Therefore, the three important functions derived in cyclostationary feature detector can be summarized as in Table 8.2.

The magnitude of the SOF varies from 0 to 1 and represents the strength of the second order periodicity within the signal. The SOF contains the spectral features of interest. These features are nonzero frequency components of the signal at various cyclic frequencies. All modulation schemes contain a range of spectral components at different cyclic frequencies, thus distinguishing them from other modulation schemes, (i.e., the spectral components form a spectral fingerprint for the specific modulation scheme). The SOFs of two typical modulation schemes, QPSK and 4PAM, are shown in Figure 8.10(a) and Figure 8.10(b). Notice how the SOF for each modulation scheme generates a highly distinct spectral image. These distinctions allow signals to be classified from cyclic analysis. In general, the plot of SOF is a three-dimensional figure, where the x-axis represents the cyclic frequency

Table 8.2 Three Important Functions Derived in Cyclostationary Feature Detector

Function Name	Equation	Characteristic
Cyclic autocorrelation function (CAF)	$R_x^\alpha(\tau) = \lim_T \frac{1}{T} \int_{-T/2}^{T/2} R_x(t, \tau) e^{-j2\pi\alpha t} dt$	Fourier series of autocorrelation
Spectral correlation function (SCF)	$S_x^\alpha(f) = R_x^\alpha(\tau) e^{j2\pi f \tau} d\tau$	Fourier transform of the CAF
Spectral coherence function (SOF)	$C_X^\alpha(f) = \frac{S_X^\alpha(f)}{\left[S_X^0(f + \alpha/2) \times S_X^0(f - \alpha/2) \right]^{1/2}}$	Normalized version of the SCF

α , the y-axis represents the spectral frequency f , and the z-axis represents the corresponding magnitude of the SOF for each (α, f) pair. Note that when α does not equal zero, the SOF values are approximately zero.

Cyclostationary feature detectors can be implemented via FFTs. Knowledge of the noise variance is not required to set the detection threshold. Hence, the detector does not suffer from the “SNR wall” problem of the energy detector. However, the performance of the detector degrades in the presence of timing and frequency jitters (which smear out the spectral lines) and RF nonlinearities (which induce spurious peaks). Representative papers that consider the approach are [20–22].

8.2 SOFTWARE IMPLEMENTATION

In Section 8.1.4, two forms of spectral detectors and classifiers were presented. In this section, we will implement these detectors using software and apply them to some Simulink-generated signals in order to make detection. From these software implementations, we will have a better understanding of the characteristics of these two detectors, as well as their specific applications.

8.2.1 Constructing Energy Detector

In the first portion of the experiment, we will generate several signals and construct the first type of spectral detector, namely, the energy detector, according to the theory presented in Section 8.1.4.1. Following this, we will explore one important aspect of this detector, namely, the selection of an appropriate threshold. Note that this portion of the chapter is entirely performed in Simulink.

8.2.1.1 Signal Generation

To get started, the first step of this experiment is to generate several signals belonging to different families of modulation schemes. Later, we will apply the spectral detectors to these signals and observe how certain modulation schemes can vary spectrally while others possess similar characteristics.

First, let us download and open the `datagen.mdl` file, as shown in Figure 8.11. In this Simulink model, the *Random Integer Generator* blocks generate M -ary integers depending on the modulation scheme in the given branch.

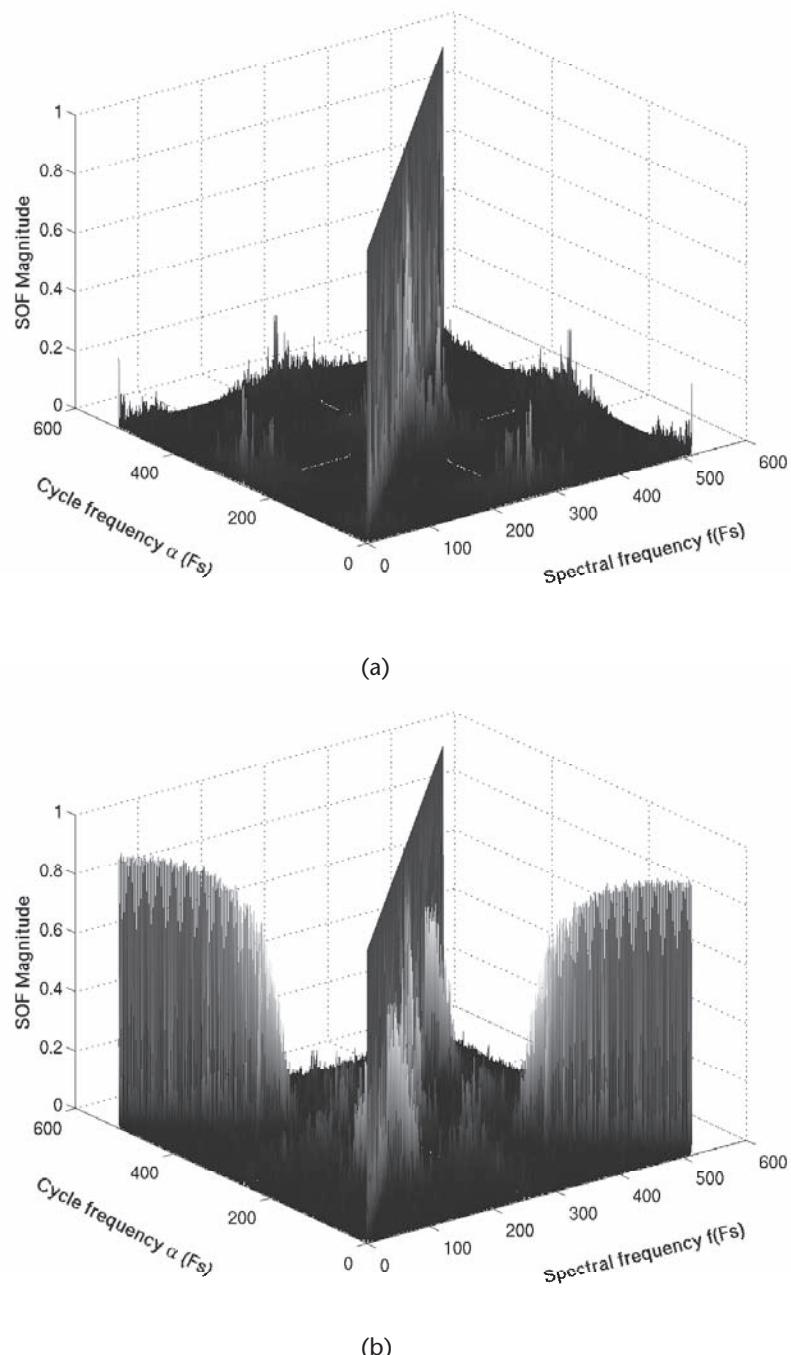


Figure 8.10 Distinctive cyclic features of different modulation schemes. (a) SOF of QPSK signal in an AWGN channel at 10 dB SNR. (b) SOF of 4PAM signal in an AWGN channel at 10 dB SNR.

Then, the modulated signals are pulse shaped for over-the-air transmission. In the end, vectors of each transmission are saved to the MATLAB workspace by *Signal To Workspace* blocks.

Once the model has been downloaded and opened in Simulink, let us run the model once and check the MATLAB workspace in order to see whether the signals have been saved. If yes, let us attach the *Spectrum Scope* block to the AWGN Channel block to observe the output of each channel. Please note that in order to get a satisfactory frequency resolution, the FFT length of the *Spectrum Scope* block should be set properly.

8.2.1.2 Energy Detector Construction

Given these signals in the workspace, we will now proceed to construct a simple energy detector in Simulink that will analyze the signals in the workspace and determine whether or not a signal is present based on the decision statistic and a threshold. However, the decision statistic defined in (8.16) is quite inflexible, particularly in the case of narrowband signals and sinewaves. An alternative approach could be devised by using a periodogram to estimate the spectrum via squared magnitude of the FFT, as depicted in Figure 8.12 [27]. Correspondingly, the following steps will assist in producing the decision statistic of the intercepted signal:



Vary the SNR value of the AWGN Channel block and observe the output spectrum of each channel. At what point do the signals become unobservable due to noise?

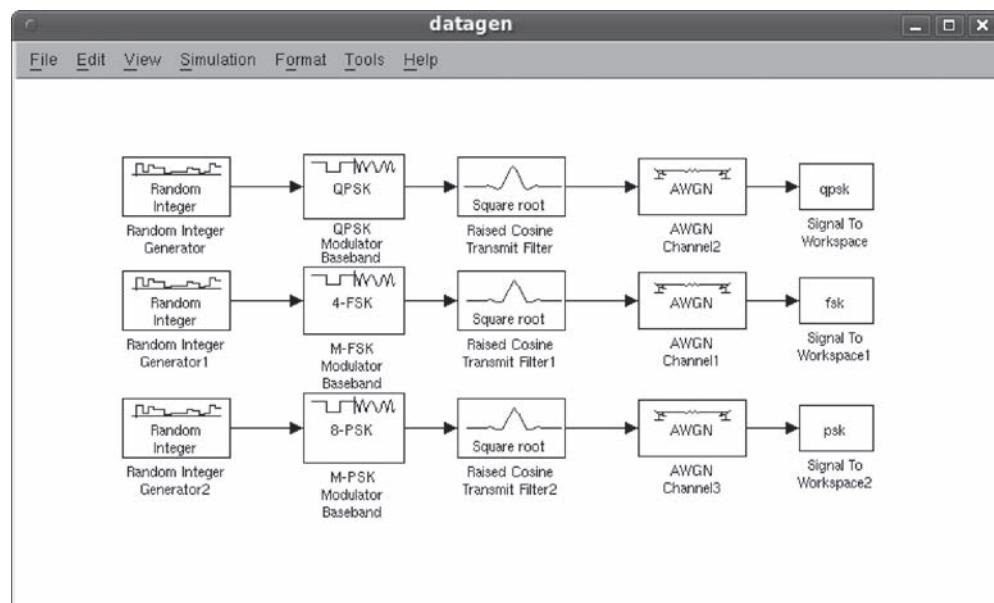


Figure 8.11 Signal generation model, where signals of three different modulation schemes are saved to the MATLAB workspace.

1. *Prefiltering* of the intercepted signal extracts the frequency band of interest.
2. *Analog-to-digital conversion* (ADC) transforms the filtered intercepted signal into discrete time samples.
3. *Fast Fourier transform* (FFT) provides the frequency representation of the signal.
4. *Square-law device* yields the square of the magnitude of the frequency response from the FFT output.
5. *Average N samples* of the square of the FFT magnitude can be done by a moving average filter of length N , which windows and time-averages the last N samples of input. The window size N should be adjusted depending upon the performance of the system.²

Please note that in real-world applications, the first two blocks are usually implemented in the hardware, which depends on the software-defined radio equipment itself and is not under our control. Therefore, for the implementation in this section, only the last three blocks need to be implemented, and the signals from Section 8.2.1.1 will serve as the discrete-time signals.

8.2.1.3 Energy Threshold Selection and Hypothesis Testing

To perform the actual energy detection, we need to apply a threshold to the decision statistic obtained in Section 8.2.1.2 in order to decide whether a transmission is present at a specific frequency. In other words, any portion of the frequency band where the energy exceeds the threshold is considered occupied by a transmission, since energy detection is a binary decision-making process consisting of two hypotheses: \mathcal{H}_0 (idle) or \mathcal{H}_1 (occupied). However, one of the primary challenges of energy detection is the selection of an appropriate threshold. This is due to the fact that the threshold may work for one transmission but may not be sufficient for another, since the transmitters might be employing different signal power levels or the transmission ranges may vary altogether.

Let us now conduct a series of trials across a range of SNR values to determine the threshold. For each SNR value, let us conduct 10 trials, where each trial picks a

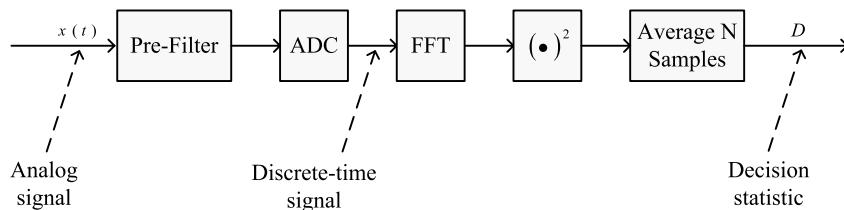


Figure 8.12 Schematic of an energy detector implementation employing prefiltering and a square-law device to produce the decision statistic of the intercepted signals.

2. You can perform a linear averaging, which averages the N samples in a nonweighted manner. Or, you can perform an exponential averaging, which averages the N samples in a weighted manner, and the most recent samples are given more weighting than the older ones.

different threshold. Using this threshold, we compute the probability of false alarm (the number of times we incorrectly say that there is a signal present) and the probability of missed detection (the number of times we did not find the signal). After 10 trials, we select the threshold that performs the best concerning both metrics and set it as the threshold value that will be employed in the energy detection for this specific SNR.



Which modulation scheme is most often detected? Which modulation scheme is most often not detected?

8.2.2 Observing Cyclostationary Detector

In the second portion of the experiment, we will apply the second type of spectral detector, namely, the cyclostationary detector, according to the theory presented in Section 8.1.4.2. Since the theory of the cyclostationary detector is relatively complicated, we will employ some ready-made M-files already containing this implementation of a cyclostationary detector and observe the results.³ Furthermore, we will continue to use the Simulink model in Figure 8.11 to generate signals and examine their spectra. First, let us open the `datagen.md1` file and reset the parameters of each channel to the following values: $\text{SNR} = 100 \text{ dB}$, $\beta = 0.5$. Run the `datagen.md1` model and save the received signals in your workspace, as this will be used later.

All modulation schemes contain a range of spectral components at different cyclic frequencies, thus distinguishing them from other modulation schemes. Recall that the spectral coherence function (SOF) defined in (8.26) is a measure of the correlation between cyclic frequencies, and cyclostationary detection relies on the fact that the SOF for each modulation scheme generates a highly distinct spectral image. These distinctions allow signals to be classified based on a cyclic analysis. Therefore, in this subsection, we will focus on observing the spectral coherence functions generated by different modulation schemes. First, let us download and open the `cyclic.m` file, which generates the SOF, and use the following lines of code to plot the SOF:

```
[ SCF Cx ] = cyclic( qpsk );
Cxplot = surf( abs( Cx ) );
set( Cxplot, 'edgecolor', 'interp' );
```

Please note that in this code, `Cx` is the SOF generated from the received QPSK signal, and `SCF` is the corresponding spectral correlation function. The SOF is plotted by the `surf` function, and the specifications of the plot is listed by the `set` function.

3. The interested reader can check out [16–19] for more information about cyclostationary detectors.



1. Generate an AWGN vector and plot its SOF. Does this plot support the conclusion that cyclostationary detectors are robust to Gaussian noise? Why?
2. Why might cyclostationary detectors be susceptible to frequency selective fading?
3. Adjust the roll-off factors of the pulse shaping filters and re-examine the coherence functions. What effect does the excess bandwidth have on the effectiveness of the detector?

8.3 USRP HARDWARE EXPERIMENTATION

So far we have used *Spectrum Scope* frequently in our Simulink experiments, as it displays the mean-squared spectrum or power spectral density of the input signal. However, the bandwidth that we can observe using this scope is limited. For example, in Section 5.2.1, *Spectrum Scope* is used to observe the FFT of the received signal, as shown in Figure 8.13, where the observed frequency range is from -100 kHz to 100 kHz. In other words, the bandwidth is 200 kHz.

If we want to observe the FFT over a wider frequency range, for example, 4.9 GHz to 5.9 GHz, which is one of the frequency ranges that the XCVR2450 daughterboard supports, can we still use the *Spectrum Scope*? The answer is yes. However, we need to divide the wideband channel into several narrowband subchannels in order to use the *Spectrum Scope* on each subchannel, and then apply a “sliding window” approach to integrate the FFT results. In this section, we will implement a wideband spectrum sensing technique using two types of sliding window. This can be accomplished by the following steps.

To get started with this implementation, let us take the Simulink design of energy detection from Section 8.2.1.2 and incorporate into it the *SDRu Receiver* block. By doing this, we can use USRP2 to collect the spectrum data.

Now, let us proceed to develop a sliding window approach upon the energy detection implementation from the previous step such that it can cover a wider frequency range, for instance, 200 MHz. Two types of windows will be employed here, namely,

- A *rectangular window*, as shown in Figure 8.14(a), which starts at f_1 and ends at f_2 sharply.
- A *tapered window*, such as a Hamming window [23], as shown in Figure 8.14(b), which also starts at f_1 and ends at f_2 , but with gradual transitions on both ends.

The purpose here is to sweep individual segments of a much wider frequency range using these two types of windows and then post-process them into a single frequency sweep result, as shown in Figure 8.15. When performing spectral analysis on finite length of data, a key function of these windows is to minimize the discontinuities of truncated spectrum, thus reducing spectral leakage [24], since the

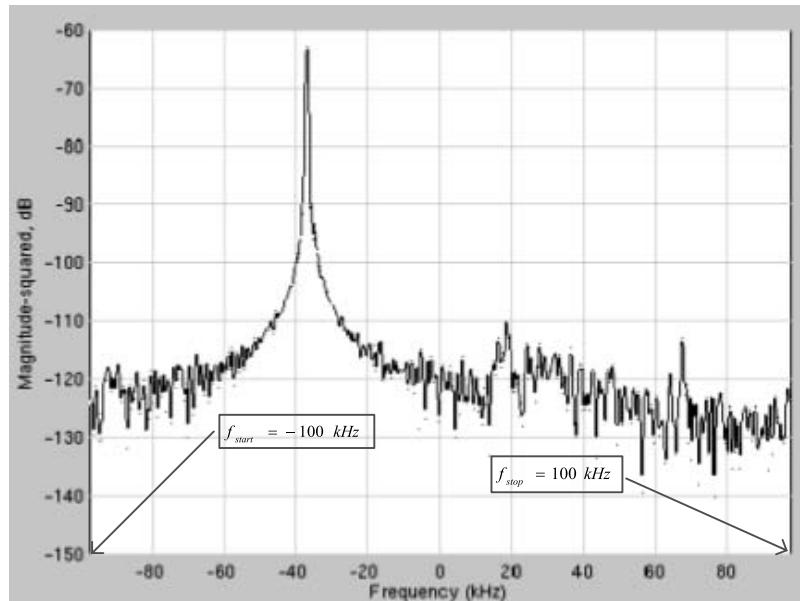


Figure 8.13 Using *Spectrum Scope* to observe the FFT of the received signal, where the observed frequency range is from -100 kHz to 100 kHz and the bandwidth is 200 kHz .

amount of spectral leakage depends on the amplitude of the discontinuity. As the discontinuity becomes larger, spectral leakage increases, and vice versa. Comparing Figure 8.15(a) and Figure 8.15(b), we can find that the tapered windows can effectively reduce the amplitude of the discontinuities at the boundaries of each period, and the resulting spectrum gradually tapers to zero at the ends. Minimizing the discontinuities along the transition edges is one of the advantages of tapered windows over the rectangular windows.

As long as these two types of windows have been implemented, let us apply them for 5.1 GHz to 5.3 GHz frequency range and repeat the spectrum sensing for 25 times, respectively. In the end, average the results for both window types.



What do you observe?

Then, let us transmit a sinusoidal signal in that frequency range and perform the test again.



Can you see the sinusoidal signal on your results?

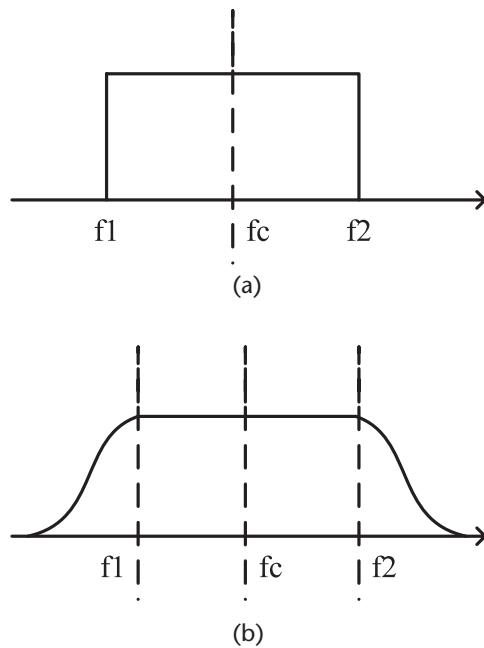


Figure 8.14 Two types of windows, which will be used to divide the wideband channel into narrow-band subchannels. (a) Rectangular window. (b) Hamming window.

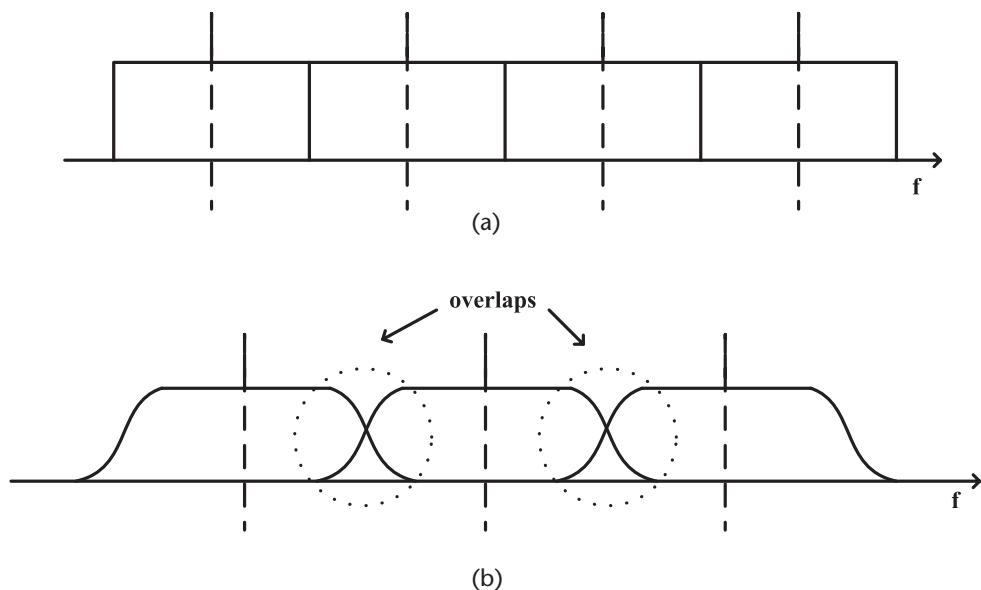


Figure 8.15 A single frequency sweep result from two types of windows. (a) A single frequency sweep result using a rectangular window. (b) A single frequency sweep result using a Hamming window. Note there are overlaps in transition areas.

Let us summarize our observations by answering the following questions:

Q

1. How long does it take to sweep 200 MHz of spectrum?
2. Is there a relationship between sweep time, FFT size, and frequency resolution?
3. What do you notice about the quality of your spectrum sweep when you average more sweeps?
4. What is the impact of the individual window size when sweeping a large frequency band?
5. Is it possible to obtain the frequency response of a wideband spectral range using a single FFT and no windowing? If yes, is there a tradeoff with frequency resolution? If no, please explain the reason.

8.4 OPEN-ENDED DESIGN PROJECT: CSMA/CA

In the previous experiments found within this book, we were dealing with a wireless network consisting of only two nodes. However, in many real-world situations, we encounter multiple nodes communicating with each other in a distributed fashion, in which these nodes share a common channel or medium as well as employ a medium access control (MAC) protocol that enables everyone access to this medium. In this section, we will design and implement an SDR communication system capable of achieving medium access control using a simple random backoff process for collision avoidance and a spectrum sensing procedure that has been designed in the earlier parts of this chapter.

8.4.1 Carrier Sense Multiple Access

Carrier sense multiple access (CSMA) is a probabilistic media access control protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus, or a band of the electromagnetic spectrum [25]. CSMA is composed of two logically defined components:

- *Carrier sense* describes the fact that a transmitter uses feedback from a receiver that detects a carrier wave before trying to send. That is, it tries to detect the presence of an encoded signal from another station before attempting to transmit. If a carrier is sensed, the station waits for the transmission in progress to finish before initiating its own transmission.
- *Multiple access* describes the fact that multiple stations send and receive on the medium. Transmissions by one node are generally received by all other stations using the medium.

8.4.2 Collision Avoidance

Carrier sense multiple access with collision avoidance (CSMA/CA) is a variant of CSMA. Collision avoidance is used to improve CSMA performance by not allowing wireless transmission of a node if another node is transmitting, thus reducing the probability of collision due to the use of a random binary exponential backoff time.

In this protocol, a carrier sensing scheme is used, namely, a node wishing to transmit data has to first listen to the channel for a predetermined amount of time to determine whether or not another node is transmitting on the channel within the wireless range. If the channel is sensed “idle,” then the node is permitted to begin the transmission process. If the channel is sensed as “busy,” the node defers its transmission for a random period of time, namely the backoff time. A simple example of CSMA/CA is shown in Figure 8.16, where node 1 wants to transmit, but the channel is sensed as “busy,” so node 1 defers its transmission for five slots.

8.4.3 Implementation Approach

Although medium access control is usually employed in multiple node communication systems, as a starting point, we will implement a two-node (radio 1 and radio 2) communication system in this section. These two radios are both transceivers, sharing a common communication channel and using CSMA/CA protocol. Since they are transceivers, they can switch between *transmit* and *receive* modes. We can define the duration of each mode. Please note since this system is also a half-duplex communication system, we can build it upon the system obtained in Section 7.4. However, unlike Section 7.4, where two MATLAB scripts are used to control the switch between different modes, in this section, CSMA/CA is employed to switch the modes. Therefore, our system should perform the following three stages, as shown in Figure 8.17:

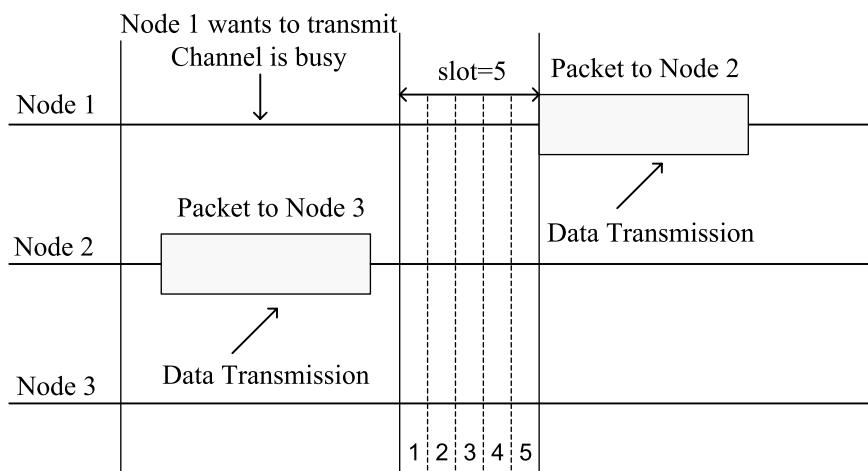


Figure 8.16 An example of a 3-node communication system employing CSMA/CA protocol (based on [26]), where node 1 employs a backoff time of five time slots.

- Stage 1:
 - Radio 1 is in *transmit* mode. It switches between transmitting a random number of “Hello world” frames and listening for a codeword from radio 2.
 - Radio 2 does the spectrum sensing using energy detection. If the channel is sensed as “busy,” radio 2 enters a random backoff time. If the channel is sensed “idle,” radio 2 begins to transmit a codeword to radio 1 and the system enters stage 2.
- Stage 2:
 - Radio 2 is in the *transmit* mode. It repeats sending a random number of “change” to radio 1.
 - Radio 1 is in the *receive* mode, and it does the spectrum sensing using energy detection. If the channel is sensed as “busy,” radio 1 keeps receiving the incoming message. If the channel is sensed “idle,” radio 1 begins to decode the message it has received, and the system enters stage 3.
- Stage 3:
 - If radio 1 can get at least one “Change” in its decoded message, radio 1 begins to broadcast “Goodbye,” and their communication ends.
 - If radio 1 does not get any “Change” in its decoded message, the whole system returns to the origin and starts from stage 1.

8.4.4 Useful Suggestions

Since energy detection is used to determine whether the channel is idle or busy, we should select a reasonable threshold for our detector. Besides the two radios, we need to implement an exponential random number generator, which is used to generate the backoff time for CSMA/CA protocol while the system is running. The number of “Hello world” frames and the number of “change” codewords can also be exponential random variables. However, they don’t need to

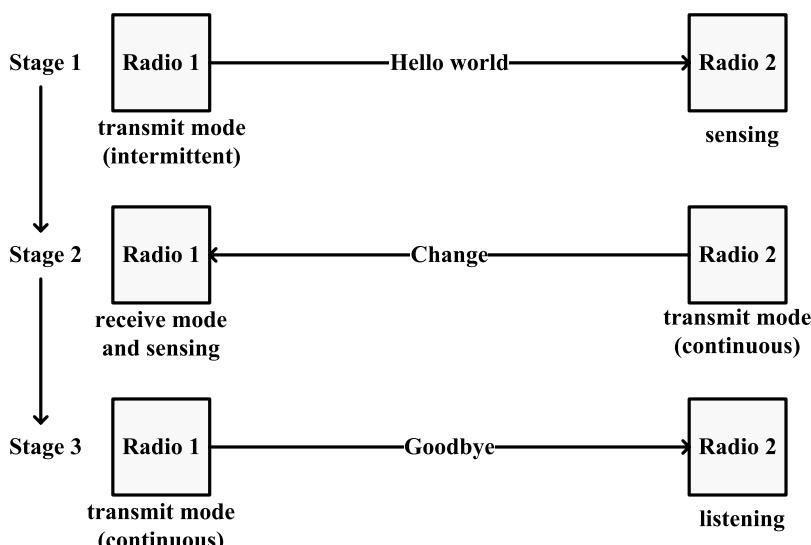


Figure 8.17 Three stages in CSMA/CA protocol implementation.

```
Command Window
① New to MATLAB? Watch this Video, see Demos, or read Getting Started.
```

```
received_message =  
  
Goodbye  
    ←———— Enter Stage 1, sensing  
Sensing...  
Free spectrum...  
Sensing...  
Free spectrum...  
Sensing...  
Occupied spectrum...  
  
received_message =  
  
Hello world  
    ←———— Enter Stage 2, transmitting  
Sensing...  
Free spectrum...  
Transmitting...  
Sensing...  
Free spectrum...  
Transmitting...  
Sensing...  
Free spectrum...  
Transmitting...  
Sensing...  
Occupied spectrum...  
    ←———— Enter Stage 3, listening  
received_message =  
  
Goodbye  
  
fx >>
```

Figure 8.18 An example of a successful implementation at radio 2, which demonstrates the three stages introduced in Section 8.4.3.

be generated while the system is running. Instead, they can be defined before the system is running.

8.4.5 Evaluation and Expected Outcomes

With individual models for both radios completed, we can proceed to the testing process. Let us start the two models at the same time, which makes radio 1 initiate transmission and radio 2 begin sensing. As soon as radio 2 detects an occupied channel, it will evaluate the received message and send a “Change” message back if it has received the correct message “Hello world.” Since radio 1 is in sensing mode right now, once it detects an occupied channel, it will decode the received message and send a “Goodbye” message back if a “Change” message is received. In this way, the two radios will relay the message back and forth to each other, demonstrating a half-duplex communication system. The most important task in this process is to find out the proper sensing time, transmission time, and back-off interval for both radios.

An example of a successful implementation at radio 2 is displayed in Figure 8.18, where the three stages introduced in Section 8.4.3 are clearly demonstrated. According to the figure, we can see that once “Goodbye” is received, radio 2 enters

stage 1 and keeps sensing. When “Hello world” is received, radio 2 continues to stage 2 and begins transmitting. After a predetermined transmission time, radio 2 proceeds to stage 3, in the listening mode.

8.5 CHAPTER SUMMARY

This chapter focuses on the principle and implementation of two spectrum sensing techniques, namely, the energy detection and the cyclostationary feature detection. In this chapter, the theory of spectrum sensing and the practical issues of collecting spectral data are first introduced, followed by the implementation of the energy detector and the observation of the cyclostationary feature detector in MATLAB. Using USRP, a wideband spectrum sensor is implemented in the hardware experiment. In the end, the open-ended design project realizes the CSMA/CA protocol, which heavily relies on the spectrum sensing techniques and enables the multiple access schemes.

8.6 PROBLEMS

1. [Power Spectral Density] Calculate and plot the power spectral density of a 100 MHz sinusoidal tone. How would you expect this to look on a spectrum analyzer?
2. [Power Spectral Density] The power spectral density of a narrowband noise signal, $n(t)$, is shown in Figure 8.19. The carrier frequency is 10 Hz.
 - (a) Find the power spectral densities of the in-phase and quadrature components of $n(t)$.
 - (b) Find their cross-spectral densities.
3. [Power Spectral Density] The input to a time-invariant linear system with impulse response $h(t)$ is a white Gaussian noise process $X(t)$ with two-sided spectral density $\frac{N_0}{2}$. The output is the random process $Y(t)$. The filter’s response is given in terms of $p_T(t)$, the rectangular pulse of duration T , by $h(t) = \frac{\sin(2\pi t)}{T} p_T(t)$.
 - (a) Find the cross-correlation function $R_{XY}(\tau)$, $-T < \tau < T$.
 - (b) Find $R_{YY}(0)$.

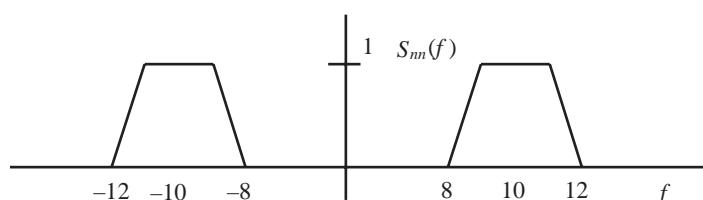


Figure 8.19 Power spectral density of a narrowband noise signal, $n(t)$.

4. [Hypothesis Testing] Given a random variable $X \sim N(\mu, \sigma^2)$, where μ and σ^2 are unknown. x_1, x_2, \dots, x_n are n samples of X . Let us define $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, $Q = \sum_{i=1}^n (x_i - \bar{x})^2$. Suppose we have the following hypothesis testing:

$$\begin{aligned}\mathcal{H}_0 : \mu &= \mu_0 \\ \mathcal{H}_1 : \mu &\neq \mu_0\end{aligned}\quad (8.27)$$

- (a) What should be the decision statistic D ?
(b) What is the region to reject \mathcal{H}_0 ?

5. [Hypothesis Testing] Given a random variable $X \sim N(\mu, \sigma^2)$, where μ is unknown. x_1, x_2, \dots, x_n are n samples of X , whose variance is S^2 . Suppose we have the following hypothesis testing:

$$\begin{aligned}\mathcal{H}_0 : \sigma^2 &= 16 \\ \mathcal{H}_1 : \sigma^2 &< 16\end{aligned}\quad (8.28)$$

- (a) What should be the decision statistic D ?
(b) What is the region to reject \mathcal{H}_0 ?

6. [Hypothesis Testing] Given two random variables $X \sim N(\mu_1, \sigma_1^2)$ and $Y \sim N(\mu_2, \sigma_2^2)$, where σ_1^2 and σ_2^2 are unknown. x_1, x_2, \dots, x_{n_1} are n_1 samples of X , y_1, y_2, \dots, y_{n_2} are n_2 samples of Y . All these samples are independent. Suppose we have the following hypothesis testing:

$$\begin{aligned}\mathcal{H}_0 : \mu_1 &= \mu_2 \\ \mathcal{H}_1 : \mu_1 &\neq \mu_2\end{aligned}\quad (8.29)$$

- (a) What should be the decision statistic D ?
(b) What is the distribution of D ?

7. [Hypothesis Testing] A factory manufactures one type of 68 mm nail. In real production, its length satisfies the Gaussian distribution of $N(\mu, 3.6^2)$. Suppose we have the following hypothesis testing:

$$\begin{aligned}\mathcal{H}_0 : \mu &= 68 \\ \mathcal{H}_1 : \mu &\neq 68\end{aligned}\quad (8.30)$$

Assume \bar{x} is the mean of n samples and the testing will be conducted according to the following criteria: When $|\bar{x} - 68| > 1$, reject \mathcal{H}_0 ; otherwise, accept \mathcal{H}_0 .

- (a) When $n = 36$, what is the probability of type I error?
(b) When $n = 64$, what is the probability of type I error?
(c) When $n = 64$ and \mathcal{H}_0 does not hold (assume $\mu = 70$), what is the probability of type II error?

8. [Hypothesis Testing] Consider the problem of detecting between two real Gaussian random variables with means μ_i and variance σ_i^2 , $i = 0, 1$. Show that the P_{FA} is given by

$$P(Z > \tau | \mathcal{H}_0) = Q \left(\frac{\tau - \mu_0}{\sigma_0} \right) \quad (8.31)$$

where $Q(\tau) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\tau} e^{-t^2/2} dt$ is the standard Gaussian tail function.

9. [Cyclostationary Process] If $x(t)$ is wide-sense cyclostationary, its mean and autocorrelation are periodic. Express $R_x(t - \frac{\tau}{2}, t + \frac{\tau}{2})$ as a Fourier series. If $x(t)$ is cyclostationary, what does that imply about frequency components at m/T_0 , where m is an integer?
10. [Cyclostationary Process] Given the cyclic autocorrelation function in (8.21) and using the Einstein-Wiener-Khintchine (EWK) relation, derive the spectral correlation function of a wide sense cyclostationary process.

References

- [1] Mitola III, J., and G. Q. Maguire, "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, Vol. 6, No. 4, Aug. 1999, pp. 13–18.
- [2] Haykin, S., "Cognitive Radio: Brain-Empowered Wireless Communications," *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 2, Feb. 2005, pp. 201–220.
- [3] McHenry, M. A., P. A. Tenhula, D. McCloskey, D. A. Roberson and C. S. Hood, "Chicago Spectrum Occupancy Measurements Analysis and a Long-Term Studies Proposal," *Proceedings of Workshop on Technology and Policy for Accessing Spectrum*, Boston, MA, 2006.
- [4] Pagadarai, S., and A. M. Wyglinski, "A Quantitative Assessment of Wireless Spectrum Measurements for Dynamic Spectrum Access," *Proceedings of the IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, Hannover, Germany, 2009.
- [5] Pagadarai, S., R. Rajbanshi, and A. M. Wyglinski, "Agile Transmission Techniques," *Cognitive Radio Communications and Networks: Principles and Practice*, Elsevier, 2009.
- [6] Akyildiz, I. F., W. Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A Survey," *Computer Networks*, Vol. 50, No. 13, Sept 2006, pp. 2127–2159.
- [7] Zhao, Q., and B. M. Sadler, "A Survey of Dynamic Spectrum Access," *IEEE Signal Processing*, Vol. 24, No. 3, May 2007, pp. 79–89.
- [8] Peng Ning, Y. L., and H. Dai, "Authenticating Primary Users' Signals in Cognitive Radio Networks via Integrated Cryptographic and Wireless Link Signatures," *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2010.
- [9] Hanssen, A., and L. Scharf, "A Theory of Polyspectra for Nonstationary Stochastic Processes," *IEEE Transactions on Signal Processing*, Vol. 51, No. 5, May 2003, pp. 1243–1252.
- [10] WPI Wireless Innovation Lab, *Welcome to SQUIRRELWeb*, <http://www.spectrum.wpi.edu/>.
- [11] Leferman, M. J., "Rapid Prototyping Interface for Software Defined Radio Experimentation," Master of Thesis Worcester Polytechnic Institute, Worcester, MA, 2010.
- [12] Poor, H. V., *An Introduction to Signal Detection and Estimation*, Springer, 2010.
- [13] Zhao, Q., and A. Swami, "Spectrum Sensing and Identification," *Cognitive Radio Communications and Networks: Principles and Practice*, Elsevier, 2009.
- [14] Kay, S. M., "Statistical Decision Theory I," *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*, Prentice Hall, 1998.
- [15] Shanmugan, K. S., and A. M. Breipohl, "Signal Detection," *Random Signals: Detection, Estimation and Data Analysis*, Wiley, 1988.
- [16] Like, E. C., *Non-Cooperative Modulation Recognition Via Exploitation of Cyclic Statistics*, Master's Thesis, 2007.
- [17] Like, E. C., V. D. Chakravarthy, P. Ratazzi, and Z. Wu, "Signal Classification in Fading Channels Using Cyclic Spectral Analysis," *EURASIP Journal on Wireless Communications and Networking*, 2009.

- [18] Gardner, W. A., W. A. Brown, and C.-K. Chen, “Spectral Correlation of Modulated Signals—Part II: Digital Modulation,” *IEEE Transactions on Communications*, Vol. 35, No. 6, 1987, pp. 595–601.
- [19] Gardner, W. A., *Cyclostationarity in Communications and Signal Processing*, IEEE Press, Piscataway, NJ, 1993.
- [20] Cabric, D., S. M. Mishra, and R. W. Brodersen, “Implementation Issues in Spectrum Sensing for Cognitive Radios,” *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, 2004.
- [21] Kim, K., I. A. Akbar, K. K. Bae, J.-S. Um, and C. M. Spooner, “Cyclostationary Approaches to Signal Detection and Classification in Cognitive Radio,” *Proceedings of IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Dublin, Ireland, 2007.
- [22] Sutton, P. D., K. E. Nolan, and L. E. Doyle, “Cyclostationary Signatures in Practical Cognitive Radio Applications,” *IEEE Journal on Selected Areas in Communications*, 2008.
- [23] Hamming, R. W., “Windows,” *Digital Filters*, 3rd edition, Dover Publications, 1997.
- [24] Harris, F. J., “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,” *Proceedings of IEEE*, Vol. 66, No. 1, Jan. 1978, pp. 51–83.
- [25] Tanenbaum, A. S., “The Medium Access Control Sublayer,” *Computer Networks*, 4th edition, Prentice Hall, 2002.
- [26] http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.mac.html.
- [27] Cabric D., A. Tkachenko, and R. W. Brodersen, “Experimental Study of Spectrum Sensing Based on Energy Detection and Network Cooperation,” *Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum*, Boston, MA, 2006.

Applications of Software-Defined Radio

Given the latest advances in SDR technology with respect to computational horsepower, energy efficiency, and form factor compactness, communication system designers are devising innovative wireless transmission implementations capable of enabling a wide range of wireless data transmission applications deemed either impossible or prohibitively costly only several years ago. Until this chapter, the focus of this book was primarily on providing a solid theoretical foundation with respect to digital communications and related areas, as well as synthesizing those concepts in a series of practical, hands-on exercises that help provide a deeper understanding about how digital transceivers actually operate in the real world.

In this chapter, we provide a context for the theory and practice covered thus far in this book by exploring three state-of-the-art applications where SDR technology has made significant, almost revolutionary, contributions. Specifically, we first start in Section 9.1 with an exploration of wireless systems capable of performing autonomous decision making, referred to as cognitive radio. Then, we focus in Section 9.2 on the use of SDR technology in enabling next-generation wireless communications in vehicular networks. Finally, we study how SDR technology can be adapted for satellite communication systems in Section 9.3 before concluding this chapter.

9.1 COGNITIVE RADIO AND INTELLIGENT WIRELESS ADAPTATION

Cognitive radio [1] is a communication systems paradigm that focuses on employing highly agile, environmentally aware, intelligent wireless platforms in order to autonomously choose and fine-tune device operating parameters based on the prevailing radio and network environmental conditions. In many situations, SDR technology is primarily responsible for making cognitive radio a reality due to its operational agility resulting from a digital baseband that is entirely implemented in software and/or programmable logic.

In order to enable a substantial level of intelligent decision making in these cognitive radio systems, *machine learning* techniques have been proposed to either partially or entirely automate the (re)configuration process (see [2–6] and references therein). However, the solutions produced by these techniques often depend on explicit inputs by the human operators, which require some knowledge of the wireless device and the target networking behavior (e.g., high data rates, high error robustness, and low network latency [7]). Moreover, the target behavior specified by the human operators may not match the actual target behavior.

Most modern wireless networking devices support multiple applications, such as streaming multimedia content via the Internet, voice-over-IP telephony, online

distance learning, and electronic file downloading/sharing. Since these applications possess different device configurations in order to achieve the desired user experience, many wireless networking devices have some degree of flexibility when choosing a configuration. However, the configuration selection process is still mostly conducted by a human operator or via an exhaustive search approach when the number of parameters representing the configuration is small. Nevertheless, for scenarios possessing a large device configuration solution space, researchers are studying the use of machine learning techniques in adaptation algorithms for wireless networking devices [4–6].

One major issue is the accuracy of decisions made by the adaptation algorithm, which is based on the quality and quantity of inputs to the device. Thus, with more information available to the algorithm, it becomes capable of making more informed decisions that achieve the desired user experience more precisely. Referring to Figure 9.1, three types of parameters employed by the adaptation algorithm are:

1. *Device configurations*: Essentially the “knobs” of the wireless networking device, these are a collection of parameters that can be altered to change the current operating state of the device. Note that several potential configurations may not be possible to implement and are thus disallowed by the adaptation algorithm.
2. *Environmental parameters*: Represented as the “dials” in Figure 9.1, these parameters represent the information about the current status of the device, as well as its sensed wireless environment using external sensors.
3. *Target networking experience*: This is usually represented by quantitative “metric” that approximately describes the average human user’s experience when operating the wireless networking device. The goal of the adaptation algorithm is to achieve the best-possible value for a given metric.

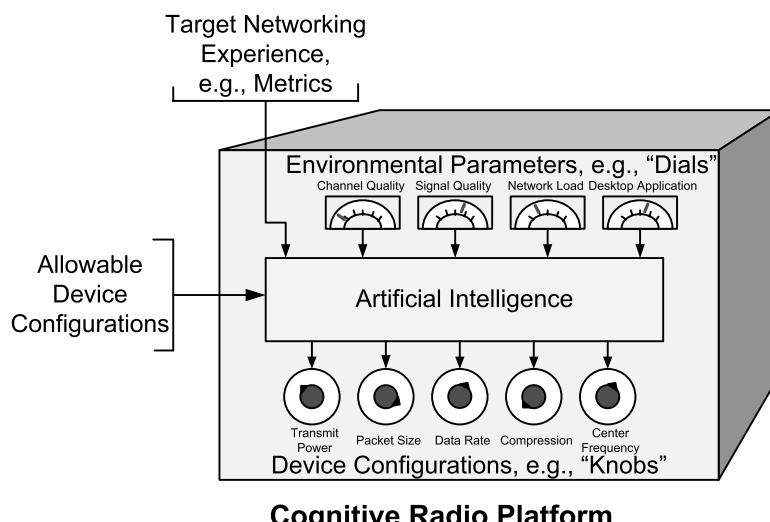


Figure 9.1 Wireless networking device employing “knobs” and “dials” in order to perform adaptation.

Note that a combination of a device configuration and a set of environmental parameters may yield a high value for a specific metric given one wireless networking application, while resulting in a low value for another different application. The following subsection describes these types of parameters in more detail.

9.1.1 Wireless Device Parameters

The definition of an optimal decision is a combination of device configuration and environmental parameters that achieve the target networking experience better than any other combination available. Defining a proper list of parameters constituting a device configuration to be employed in an adaptation algorithm is of prime importance. A well-constructed list consists of common wireless parameters that each possess a large impact on the target networking behavior. Table 9.1 shows a list of nine transmission parameters commonly used in wireless networking devices.

Environmental parameters inform the system of the surrounding environment characteristics. These characteristics include internal information about the device operating state and external information representing the wireless channel environment. Both types of information can be used to assist the cognitive radio in making decisions. These variables along with the target networking experience are used as inputs to the algorithm. Table 9.2 shows a list of six environmental parameters that can affect the operational state of a cognitive radio device.

The purpose of a machine learning-based adaptation algorithm is to autonomously improve the wireless networking experience for the human operator. However, improving this experience can mean several different things. Much research is focused on improving the accommodation of many wireless users within the same network. Other important aspects include providing error-free communications, high data rates, limiting interference between users, and even the actual power consumption of the wireless networking device, which is extremely important in mobile applications. As shown in Table 9.3, we have defined five common target networking experiences that guide the adaptation algorithm to a specific optimal solution for the cognitive radio system.

The target experiences presented in Table 9.2 represent the means for guiding the state of the adaptation process for the wireless networking device. The

Table 9.1 Several Common Wireless Networking Device Configuration Parameters (from [1, 5])

Parameter	Description
Transmit power	Raw transmission power
Modulation type	Type of modulation
Modulation index	Number of symbols for a given modulation scheme
Carrier frequency	Center frequency of the carrier
Bandwidth	Bandwidth of transmission signal
Channel coding rate	Specific rate of coding
Frame size	Size of transmission frame
Time-division duplexing	Percentage of transmit time
Symbol rate	Number of symbols per second

Table 9.2 Several Common Wireless Networking Environmental Parameters (from [1, 5])

<i>Parameter</i>	<i>Description</i>
Signal power	Signal power as seen by the receiver
Noise power	Noise power density for a given channel
Delay spread	Variance of the path delays and their amplitudes for a channel
Battery life	Estimated energy left in batteries
Power consumption	Power consumption of current configuration
Spectrum information	Spectrum occupancy information

Table 9.3 Several Common Wireless Networking Target Experiences (from [1, 5])

<i>Objective</i>	<i>Description</i>
Minimize bit error rate	Improve overall BER of the transmission environment
Maximize data throughput	Increase overall data throughput transmitted by radio
Minimize power consumption	Decrease amount of power consumed by system
Minimize interference	Reduce the radio interference contributions
Maximize spectral efficiency	Maximize the efficient use of the frequency spectrum

algorithm makes use of these experiences through relationships that describe how modifying the device parameters achieve these objectives. To facilitate the decision-making process, each target networking experience must be represented by a mathematical relationship that relates a device configuration and environmental parameters to a scalar value that describes how well this set achieves the specific goal [1, 5]. These functions will provide a way for the adaptation algorithm to rank combinations of configurations and environmental parameters, ultimately leading to a final decision.

Note that it is possible to specify several target networking experiences simultaneously, with the final score being represented by a numerical value. In this case, the individual scores of the target experiences are weighted according to their importance in the specific application and summed together, forming the final overall score [5].

9.2 VEHICULAR COMMUNICATION NETWORKS

All forms of wireless communications require access to radio frequency (RF) spectrum. However, given the finite amount of RF spectrum and the rapidly growing number of wireless applications and end users, many sectors are experiencing the effects of spectrum scarcity due to the inefficient utilization of this natural resource. One solution that is capable of accommodating this growing demand for RF spectrum while providing greater wireless access is *dynamic spectrum access* (DSA). As opposed to the conventional spectrum allocation paradigm, which focuses on static licensed spectrum assignments where the license holders possess exclusive transmission rights, the DSA paradigm enables unlicensed devices to temporarily “borrow” unused licensed spectrum while ensuring the rights of the incumbent license holders are respected. One technology capable of achieving

DSA is cognitive radio, which often employs autonomous and flexible communications techniques implemented via an SDR programmable wireless communications platform.¹

There has been much research and development over the past several decades in order to facilitate ubiquitous, reliable, and efficient wireless data connectivity between individual vehicles as well as with roadside information infrastructure [8]. The objective of these research efforts into vehicular networking is to further enhance the overall driving experience while on the road [9–12]. In particular, wireless connectivity can be used to enable a greater level of vehicular safety such that the driver's situational awareness is enhanced [13]. Given the potential benefits of vehicular networking, it is expected that commercial vehicle manufacturers will introduce this technology in the near future [14–16].

Despite the advantages of employing wireless systems in vehicular applications, there is increasing concern that the rapid growth of this sector coupled with significant bandwidth requirements of the vehicular wireless transmissions will result in a *spectrum scarcity* issue. One remedy for this issue is to employ SDR technology that is specifically designed to perform DSA networking across several vehicles as well as with associated roadside infrastructure, which is referred to as *vehicular dynamic spectrum access* (VDSA) [15, 17, 18]. Given the potential of this solution, the feasibility of VDSA has been extensively studied for several candidate frequency bands [17], networking protocols enabling VDSA communications have been assessed [14–16], the theoretical capacity of VDSA networks have been derived [18], and experimental prototype VDSA platforms and test-beds have been constructed [19, 20].

9.2.1 VDSA Overview

VDSA operates by establishing a wireless data transmission link between any two vehicular communication devices across unoccupied portions of licensed spectrum. Thus, a vehicular wireless device must be capable of identifying all the *unoccupied* frequency bands via spectrum sensing prior to any data transmission. An example of VDSA is shown in Figure 9.2 where “car A” communicates with “car B” by forming a secondary (unlicensed) user (SU) transmission link L_{AB} across some unoccupied spectrum that is located between two primary (licensed) user (PU) signals. Suppose that all vehicular communications is decentralized in this scenario, and that all information is shared between vehicles using multihop relaying. Consequently, wireless data transmission is subsequently performed between “car B” and “car C” via the transmission link L_{BC} , which is also located within a region of unoccupied spectrum, in order to relay the information to all vehicles within the geographical vicinity.

- Given the growing number of wireless interfaces being employed on a single vehicle, it is also anticipated that automobile manufacturers will eventually integrate all the communication systems on a vehicle into a single SDR platform. Such an approach would facilitate remote updating of wireless standards and protocols on vehicles without the need for an expensive upgrade recall, as well as simplify the communications infrastructure on the vehicle itself.

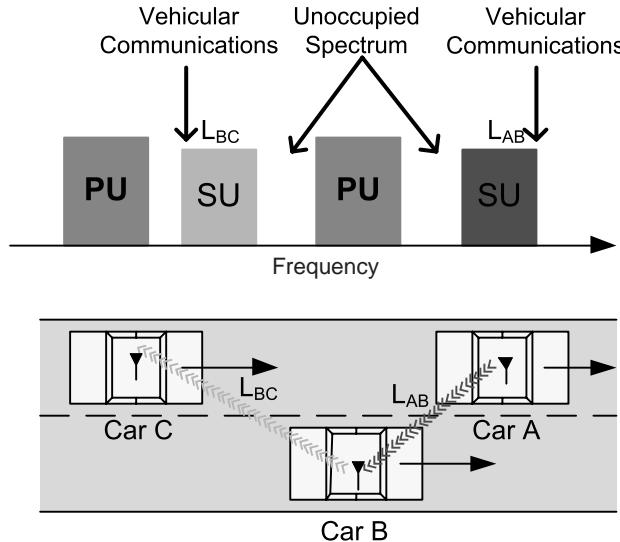


Figure 9.2 An illustration of the *vehicular dynamic spectrum access* (VDSA) concept (from [20]). In this scenario, the vehicular secondary (unlicensed) users (SUs) temporarily access unoccupied spectrum from primary (licensed) users (PUs) for performing wireless data transmission while on the road.

9.2.2 Transmitter Design

An architecture for a VDSA transmitter is shown in Figure 9.3, where the primary concern when transmitting information within a VDSA networking architecture is the potential for interference with existing primary and other secondary user signals located within the spectral and geographical vicinity. As a result, several factors must be considered when designing the VDSA transmitter, namely [20]:

- *Finding primary user – unoccupied frequencies:* The identification of unoccupied frequency bands for performing VDSA will be conducted using a combination of spectrum database searching and active spectrum sensing. The former is expected to provide a rough assessment of all spectrum opportunities within a geographical vicinity, while the latter fine tunes the spectrum situational awareness of the VDSA network by indicating the instantaneous availability of a specific frequency band.
- *Avoiding secondary users:* The mantra of any DSA paradigm and their variants is that the secondary users can never interfere with any of the incumbent primary (licensed) transmissions co-existing within a specific geographical and frequency vicinity. Since there might not be any centralized coordination with respect to the spectrum access between primary and secondary users, the latter

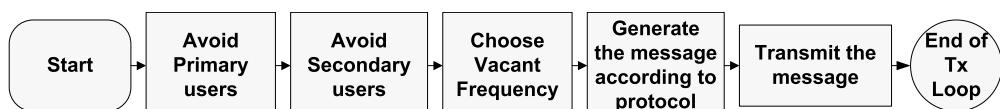


Figure 9.3 An illustration of a VDSA transmitter design showing how it searches for and avoids both PUs and SUs, as well as determine the most optimal location for the next SU carrier frequency, prior to secondary transmission [20].

may need to employ some sort of contention-based physical layer medium access control (MAC) in order to check out the availability of a frequency band to avoid interference with the primary user or other secondary users.

- *Message generation and protocol:* Assuming that the VDSA network is operating in a decentralized mode, it is necessary that the header information of each frame shared within the network contains information needed to facilitate the formation and continuation of any VDSA transmission links. Examples of various information to be included in the header are the unique identification number of the source vehicular wireless platform, time stamp information, number of hops the message can be relayed before it is considered out of date, and identification number of the destination vehicle.

9.2.3 Receiver Design

The receiver design also heavily relies on the occupancy behavior of the wireless spectrum. For instance, referring to Figure 9.3, the power spectral density (PSD) is derived from the spectrum sensing measurements and all peaks exceeding three standard deviations of the mean PSD are identified, (i.e., three-sigma outliers). The center frequency for each of the three-sigma outlier peaks is located, and the intercepted message is then demodulated and decoded. Consequently, the authors in [20] proposed two important steps that a VDSA receiver should possess when determining how to access the wireless spectrum in a secondary manner:

- *Spectrum sensing and statistical analysis:* Using energy detection, which is a simple form of discriminating between noise and signals across a frequency band via the use of an energy detection threshold [16], the transmitter and receiver of a VDSA wireless link attempt to locate each other across a frequency band in order to lock onto each other to perform data transmission. Given that energy detection is a relatively simple spectrum sensing technique, the value of the energy detection threshold used to differentiate between a signal and spurious noise was selected such that only *three-sigma outliers* [21] of the energy detector were identified as potential signals.
- *Center frequency detection:* Once the frequency locations of the VDSA transmissions have been identified, the surrounding noise is filtered out and the process of fine tuning the center frequency commences. This can be accomplished by integrating the possible signal PSD over signal bandwidth, which removes the impact of the noise present in the channel, and then finding the frequency value that partitions the overall signal energy by half assuming a symmetric PSD. Once the center frequency of the potential signal has been determined, the receiver attempts to demodulate and decode the signal at that center frequency.

9.2.4 VDSA Test-Bed Implementation

Experimentation of an actual VDSA network employing vehicles operating under highway conditions was conducted at Worcester Polytechnic Institute (WPI) on March 12, 2011 at 12:50 PM. The experiment consists of five vehicles, as shown in Figure 9.4,

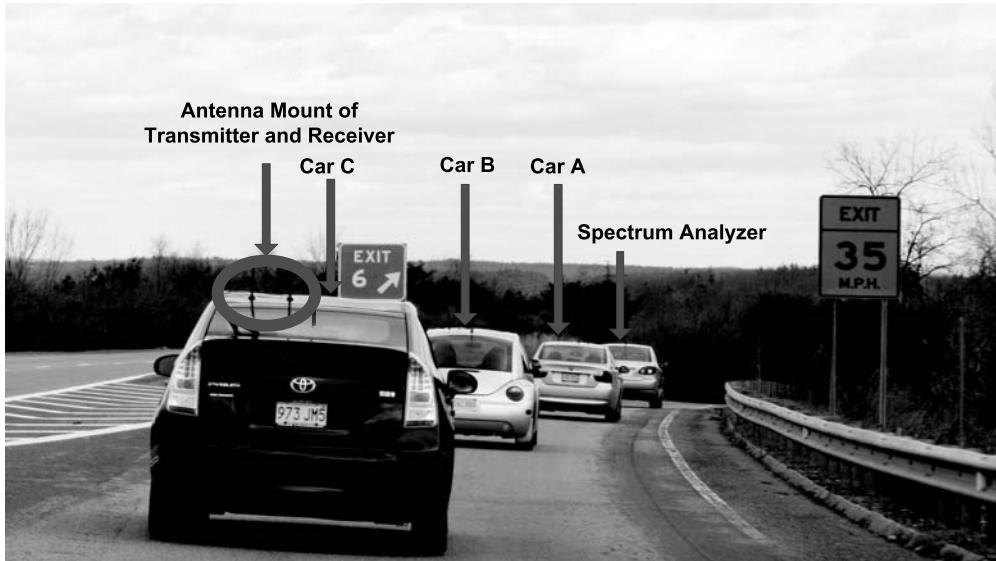


Figure 9.4 A photograph of the actual VDSA networking experiment conducted on March 12, 2011 at 12:50 PM [20]. Picture taken from car D. The figure shows the VDSA network implemented on the highway I-190 north of Worcester, MA. As indicated on the figure, the leading car is performing spectrum sensing and the rest of the cars form the proposed VDSA networking architecture.

where the leading car was equipped with a spectrum analyzer in order to record all the VDSA transmissions occurring during the experiment, while each of the remaining cars was equipped with a USRP2 SDR platform with its antenna anchored to the exterior of each vehicle. The experiment was initiated in Worcester, MA along Interstate I-190 north to Leominster, MA followed by heading east along Route 2 until the intersection with Interstate I-495, where the proposed VDSA network proceeded south until the intersection between Interstate I-495 and Interstate I-290, where the vehicles finally headed back west on Interstate I-290 to Worcester, MA.

During the experiment, car 1 of the VDSA network was continuously broadcasting hazard messages, while the remaining three cars constantly performed spectrum sensing over a 100-MHz-wide frequency band from 2.4 GHz to 2.5 GHz and performed multihop relaying in the event that a message was successfully intercepted, as shown in Figure 9.5. The receiver would lock on to the frequency and start to decode the message as soon as transmission was detected using the three-sigma rule. If the message was successfully decoded and recorded by the USRP2, it switched to transmit mode and broadcasted the message to the rest of the cars. In the event that no message was decoded, or the decoded message did not correspond to the desired network protocol, the USRP2 ignored the transmission and started another iteration of spectrum sensing. Each radio was connected to a laptop that would record and time stamp every received message for performance analysis.

9.3 SATELLITE COMMUNICATIONS

In the past, communication systems often required specialized hardware in order to achieve some level of functionality. Thus, to keep production costs low, these

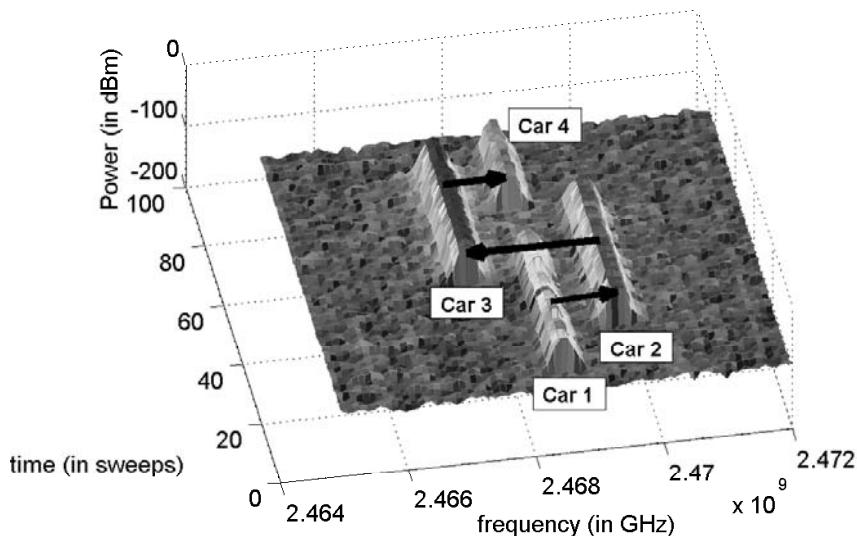


Figure 9.5 Three-dimensional plot showing a sample of the experimental results obtained from the VDSA test-bed involving the secondary user transmission in the neighborhood of licensed user signals. Notice the multihop transmission of “car 1” “car 2” “car 3” “car 4,” where the transmission frequencies were dynamically chosen based on availability.

systems contained only the hardware necessary to perform the tasks that they were designed for, which made these systems difficult to modify and upgrade. However, with the advent of various digital processing and computing technologies, communication systems have evolved to the point where many of the encoders, modulators, filters, and decoders used by these communication systems can be implemented either partially or entirely using software and/or programmable logic [22]. Those communications systems that implement all of their baseband functionality in software are referred to as SDR [23].

While some SDR platforms employ general-purpose processors or even digital signal processors (DSPs), many use a technology called field-programmable gate arrays (FPGA). An FPGA consists of reconfigurable logic elements and a switch matrix to route signals between them. These devices can be configured to support simple logic operations, such as addition, or more complex systems, such as digital filters. In addition, some FPGAs can support dynamic reconfiguration, where a system is capable of swapping components as needed, without any reprogramming. Consequently, the firmware on an FPGA can be updated remotely in order to install new blocks and remove unused blocks. Although relatively difficult to use with respect to the design and prototype of a communication system, one of the primary advantages of an FPGA is that it is very computationally powerful, making it a suitable choice for any SDR implementation involving complex operations. Commercially available examples of FPGA-based SDR platforms include Lyrtech’s small form factor (SFF) SDR development platform and the Wireless Open-Access Research Platform (WARP) [24, 25]. Although the USRP family of SDR platforms available from Ettus Research LLC possess an FPGA, these have been mostly used to perform sampling and filtering operations on incoming and outgoing data streams.

One wireless application where FPGA-based SDR platforms are beginning to emerge is in the area of satellite communications, where SDR systems are employed on satellites in order to enable advanced communications with terrestrial ground stations, as well as with other satellites in orbit around the Earth [22, 26, 27]. In particular, one type of satellite platform that is beginning to witness a substantial increase in the utilization of SDR technology is CubeSat systems. A *CubeSat* is a small form-factor satellite conforming to the CubeSat specifications published by California Polytechnic State University [28]. A 1U form factor CubeSat possesses a length, width, and height of 10 cm on each side and has a total weight that cannot exceed 1.33 kilograms. In addition to this 1U configuration, the CubeSat specifications allow for 2U ($10 \times 10 \times 20$ cm) and 3U ($10 \times 10 \times 30$ cm) configurations.

One of the main design constraints for implementing a satellite system is the physical size of the platform itself. Given the significant costs involved in deploying a satellite system in orbit, each module that forms part of the satellite system must use the smallest amount of space possible. Consequently, off-the-shelf solutions for SDR systems that can be employed on satellite systems simply do not exist, and there currently exists significant research and development efforts by government laboratories, industry, and academia in order to create a family of SDR platforms capable of being used on satellite systems. This design requirement is further compounded by the relatively small form factor associated with CubeSat systems, which also include a suite of sensors and other instruments necessary for the functionality of the satellite. An example of an FPGA-based SDR platform for a CubeSat is shown in Figure 9.6.

As a result, there exists several desired characteristics for a CubeSat FPGA-based SDR platform that is sought after by the community, namely [30]:

1. *Reconfigurability*. The system is easily reconfigured to support any number of encoding, modulation, and other signal processing schemes.
2. *Small size*. The system fits the dimensions for a 1U CubeSat ($10 \times 10 \times 10$ cm) and meets all other CubeSat standard requirements [28].
3. *Plug and play*. The system uses the Space Plug-and-Play Avionics (SPA) protocol for plug and play capabilities.
4. *Open source*. The system uses open source hardware and software so that others can adapt it to fit their project goals and so that they can create derivative systems as needed.

One such CubeSat FPGA-based SDR platform that attempts to meet these characteristics was presented in [30–32], with the hardware platform shown in Figure 9.6 and the SDR firmware architecture shown in Figure 9.7. The actual SDR communications hardware is based on the redesign of a USRP1 SDR platform by Ettus Research LLC, which publishes their open source SDR hardware designs and schematics online. At the center of the SDR architecture is the MicroBlaze softcore microprocessor, which acts as a traffic mediator, guiding data between the USB2 port to the various other devices in the system. The SPI and I2C controllers provide the users with a means of sending configuration information to the USRP and to the RF daughtercards. One of the RS232 controllers connects to an serial port on the board, while the other connects to the AT90, which controls the

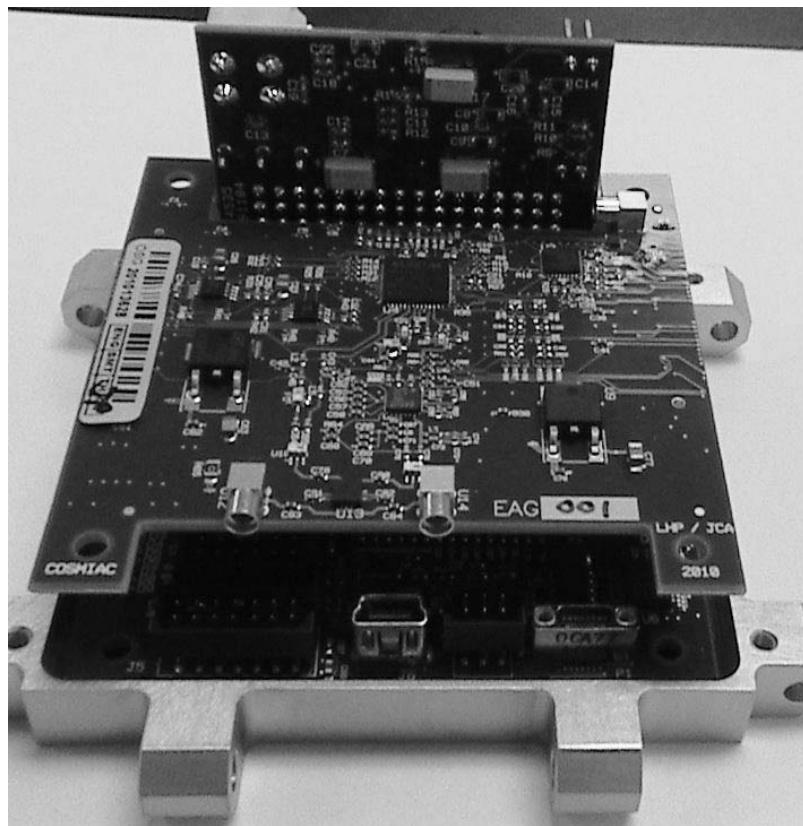


Figure 9.6 A photograph of the COSMIAC CubeSat SDR system. The bottom board is the FPGA board while the one on top is the RF daughter card [29].

ASIM for the device. The other components simply control the peripherals that give them their names.

Several advantages of this implementation of a CubeSat FPGA-based SDR platform include the following [30]:

- It employs well-tested components from Xilinx including the DDR2 memory controller, the SPI and I2C controllers, and the MicroBlaze microprocessor.
- The Embedded Development Kit (EDK) software from Xilinx provides users with a relatively simple way to design MicroBlaze-based systems for custom hardware.
- Using a MicroBlaze processor allows one to write much of the control logic in the C programming language.
- This system should be compatible with GNU Radio with few modifications since the MicroBlaze and other components can emulate the FX2 USB2 controller with proper configuration.

As for disadvantages, this system presented in [30] employs components that only work with the Xilinx toolset, so the system loses its platform independence. Furthermore, the USB2 controller that is at the core of the USRP1 implementation is currently priced at \$14,000, which can be prohibitively expensive.

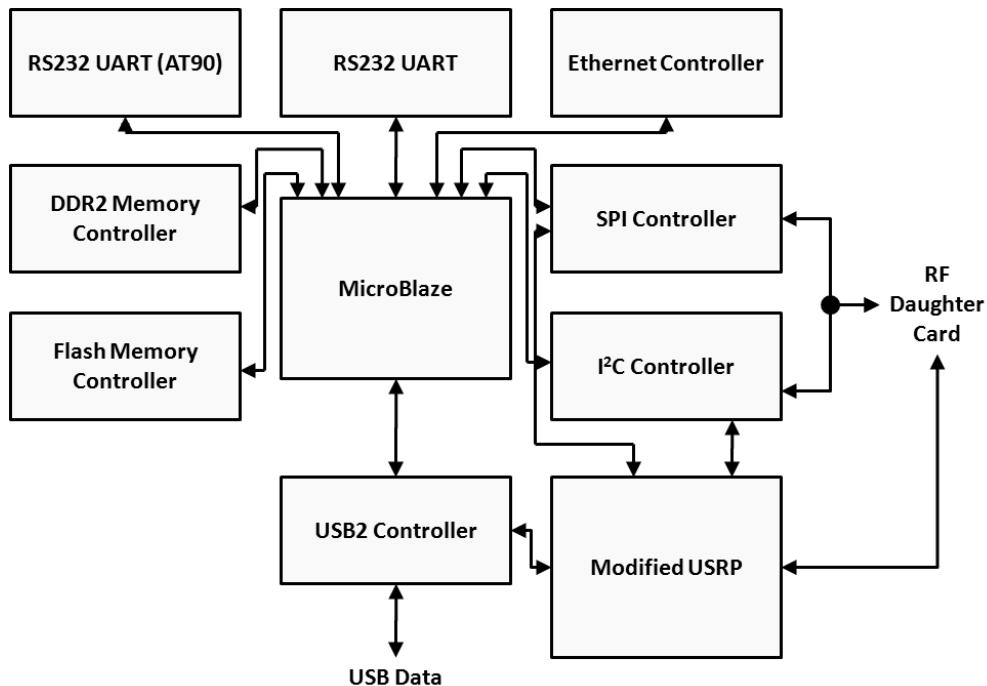


Figure 9.7 Block diagram of the proposed CubeSat SDR firmware. The MicroBlaze processor acts as the central control for all of the various communications blocks [30].

9.4 CHAPTER SUMMARY

In this chapter, we briefly explored three applications where SDR technology is employed in order to facilitate wireless data transmission in challenging environments. Whether it is intelligently and adaptively seeking out transmission opportunities across time, frequency, and space; enabling the transfer of information between vehicles while driving along a stretch of highway; or facilitating the exchange of data between a satellite in orbit and a ground station back on Earth, SDR is increasingly becoming the core technology that is enabling these and many other applications in today's society.

References

- [1] Wyglinski, A. M., M. Nekovee, and T. Hou, *Cognitive Radio Communications and Networks: Principles and Practice*, Academic Press, 2009, <http://www.elsevierdirect.com/ISBN/9780123747150/> Cognitive-Radi-Communications-and-Networks.
- [2] Barker, B., A. Agah, and A. M. Wyglinski, "Mission-Oriented Communications Properties for Software Defined Radio Configuration," *Cognitive Radio Networks*, CRC Press, 2008.
- [3] Newman, T., A. M. Wyglinski, J. B. Evans, "Cognitive Radio Implementation for Efficient Wireless Communication," *Encyclopedia of Wireless and Mobile Communications*, CRC Press, 2007.
- [4] Newman, T., R. Rajbanshi, A. M. Wyglinski, J. B. Evans, and G. J. Minden, "Population Adaptation for Genetic Algorithm-Based Cognitive Radios," *ACM/Springer Mobile Networks and Applications*, 2008.

- [5] Newman, T. R., B. A. Barker, A. M. Wyglinski, A. Agah, and J. B. Evans et al., “Cognitive Engine Implementation for Wireless Multicarrier Transceivers,” *Wiley Journal on Wireless Communications and Mobile Computing*, Vol. 7, No. 9, Nov. 2007, pp. 1129–1142.
- [6] Rieser, C. J., “Biologically Inspired Cognitive Radio Engine Model Utilizing Distributed Genetic Algorithms for Secure and Robust Wireless Communications and Networking” PhD Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2005.
- [7] Troxel, G. D., E. Blossom, S. Boswell, M. Brodsky, A. Caro, I. Castineyra et al., “Adaptive Dynamic Radio Open-Source Intelligent Team (ADROIT): Cognitively-Controlled Collaboration Among SDR Nodes,” *Proceedings of the IEEE Communications Society Conference on Sensors, Mesh and Ad Hoc Communications and Networks (SECON)—Workshop on Networking Technologies for Software-Defined Radio Networks*, Reston, VA, Aug. 2006.
- [8] Intelligent Transportation Society of America, *IntelliDrive Task Force*, <http://intellidrive.taskforce.itsa.wikispaces.net/>.
- [9] Willke, T. L., P. Tientrakool, and N. F. Maxemchuk, “A Survey of Inter-Vehicle Communication Protocols and Their Applications,” *IEEE Communications Surveys and Tutorials*, Vol. 11, 2009.
- [10] Khaled, Y., M. Tsukada, J. Santa, and T. Ernst, “On the Design of Efficient Vehicular Applications,” *Proceedings of the IEEE Vehicular Technology Conference*, Dresden, Germany, Apr. 2009.
- [11] Papadimitratos, P., A. de La Fortelle, K. Evenssen, and R. Bringnol, “Vehicular Communication Systems: Enabling Technologies, Applications, and Future Outlook on Intelligent Transportation,” *IEEE Communications Magazine*, Vol. 47, 2009, pp. 84–95.
- [12] Chennikara-Varghese, J., W. Chen, T. Hikita, and R. Onishi, “Local Peer Groups and Vehicle-to-Infrastructure Communications,” *IEEE Global Communication Conference—Workshop*, Nov. 2007.
- [13] Chen, W., and S. Cai, “Ad Hoc Peer-to-Peer Network Architecture for Vehicle Safety Communications,” *IEEE Commun. Magazine*, Apr. 2005.
- [14] Tsukamoto, K., S. Matsuoka, O. Altintas, M. Tsuru, and Y. Oie, “Distributed Channel Coordination in Cognitive Wireless Vehicle-to-Vehicle Communications,” *Proceedings of the Wireless Access in Vehicular Environments (WAVE) Conference*, Dearborn, MI, Dec. 2008.
- [15] Tsukamoto, K., Y. Omori, O. Altintas, M. Tsuru, and Y. Oie, “On Spatially-Aware Channel Selection in Dynamic Spectrum Access Multi-Hop Inter-Vehicle Communications,” *Proceedings of the IEEE Vehicular Technology Conference—Fall*, Anchorage, AK, Sep. 2009.
- [16] Lacatus, C., R. Vuyyuru, O. Altintas, D. Borota, and Ivan Seskar, “Evaluation of Energy-Based Spectrum Sensing Algorithm for Vehicular Networks,” *Proceedings of the SDR Forum Technical Conference*, Washington DC, Dec. 2009.
- [17] Pagadarai, S., A. M. Wyglinski, and R. Vuyyuru, “Characterization of Vacant UHF TV Channels for Vehicular Dynamic Spectrum Access,” *Proc. IEEE Vehicul. Networking Conf.*, Oct. 2009.
- [18] Chen, S., A. M. Wyglinski, R. Vuyyuru, and O. Altintas, “Feasibility Analysis of Vehicular Dynamic Spectrum Access via Queueing Theory Model,” *Proc. IEEE Vehicul. Networking Conf.*, Dec. 2010.
- [19] Altintas, O., M. Nishibori, R. Vuyyuru, Y. Fujii, and K. Nishida et al., “Implementation and Evaluation of Distributed Control and Data Channel Coordination Algorithms for V2V Dynamic Spectrum Access,” *Proceedings of the SDR Forum Technical Conference*, Washington DC, Nov. 2010.
- [20] Rzayev, T., Y. Shi, A. Vafeiadis, S. Pagadarai, and A. M. Wyglinski, “Implementation of a Vehicular Networking Architecture Supporting Dynamic Spectrum Access,” *Proceedings of the IEEE Vehicular Networking Conference*, Amsterdam, The Netherlands, Nov. 2011.
- [21] Pukelsheim, F., “The Three Sigma Rule,” *The American Statistician*, Vol. 48, No. 2, May 1994, pp. 88–91.

- [22] Wireless Innovation Forum, "The Three Sigma Rule," *What is Software Defined Radio*, <http://data.memberclicks.com/site/sdf/SoftwareDefinedRadio.pdf>.
- [23] Youngblood, G., "A Software-Defined Radio for the Masses, Part 1," *QEX*, Jul/Aug: 13–21, 2002, <http://www.arrl.org/files/file/Technology/tis/info/pdf/020708qex013.pdf>.
- [24] Mouser—Lyrtech SFF SDR Development Platform, April 2011, <http://www.mouser.com/Lyrtech/>.
- [25] Wireless Open-Access Research Platform, April 2011, <http://warp.rice.edu/index.php>.
- [26] Vulcan Wireless SDR Products, April 2011, <http://www.vulcanwireless.com/products/>.
- [27] Lynaugh, K., and M. Davis, *Software Defined Radios for Small Satellites*, Nov. 2010, <http://www.cosmiac.org/ReSpace2010/>.
- [28] California Polytechnic State University, *CubeSat Design Specification Rev. 12*, 2009, http://www.cubesat.org/images/developers/cds_rev12.pdf.
- [29] COSMIAC CubeSat SDR Photograph, April 2011, <http://www.cosmiac.org/images/CCRBwSDR.JPG>.
- [30] Olivieri, S., J. Aarestad, L. H. Pollard, A. M. Wyglinski, C. Kief et al., "Modular FPGA-Based Software Defined Radio for CubeSats," *Proceedings on the IEEE International Conference on Communications*, Ottawa, Canada, June 2012.
- [31] Olivieri, S., A. M. Wyglinski, L. H. Pollard, J. Aarestad and C. Kief, "Responsive Satellite Communications via FPGA-Based Software-Defined Radio for SPA-U Compatible Platforms," Presented at the ReSpace/MAPLD 2010 Conference in Albuquerque, NM, <http://www.cosmiac.org/ReSpace2010/>. Nov. 2010.
- [32] Olivieri, S. J., "Modular FPGA-Based Software Defined Radio for CubeSats," Master's Thesis, Worcester Polytechnic Institute, Worcester, MA, May 2011.

Getting Started with MATLAB and Simulink

You will be using MATLAB and Simulink for the experiments, as well as for the open-ended design projects in this book. This appendix serves as a brief refresher of MATLAB, since you should have used it before. However, if you don't have extensive experience with Simulink, then this appendix shows you how to get started with the tool. Please note the MATLAB portion of this appendix is mainly based on the MATLAB Documentation presented in [1] and the Simulink portion is based on the Simulink Basics Tutorial presented in [2], but here we extract the most important and fundamental concepts such that you can quickly get started by reading this appendix. For more information about these two products, you are encouraged to refer to [1] and [2].

A.1 MATLAB INTRODUCTION

MATLAB is widely used in all areas of applied mathematics, in education and research at universities, and in industry. MATLAB stands for MATrix LABoratory and the software is built up around vectors and matrices. Consequently, this makes the software particularly useful for solving problems in linear algebra, but also for solving algebraic and differential equations as well as numerical integration. MATLAB possesses a collection of graphical tools capable of producing advanced GUI and data plots in both 2D and 3D. MATLAB also has several toolboxes useful for performing communications signal processing, image processing, optimization, and other specialized operations.

A.2 EDIT AND RUN A PROGRAM IN MATLAB

When writing programs, you will need to do this in a separate window, called the *editor*. To open the editor, go to the “File” menu and choose either the “New...Script” (if you want to create a new program) or “Open” (to open an existing document) option. In the editor, you can now type in your code in much the same way that you would use a text editor or a word processor. There are menus for editing the text that are also similar to any word processor. While typing your code in the editor, no commands will be performed. Consequently, in order to run a program, you will need to do the following:

1. Save your code as *<filename>.m*, where *<filename>* is anything you wish to name the file.
2. Go to the command window. If necessary, change directories to the directory containing your file. This can be accomplished using the `cd` command common to UNIX and DOS. Or, alternatively, you can select the button labeled “...” in the Current Directory window, and change directories graphically. You can view your current directory by typing the `pwd` command.

3. In order to run the program, type the name of the file containing your program at the prompt. When typing the filename in the command window, do not include “.m”. By pressing enter, MATLAB will run your program and perform all the commands given in your file.

In case your code has errors, MATLAB will complain when you try to run the program in the command window. When this happens, try to interpret the error message and make the necessary changes to your code in the editor. The error that is reported by MATLAB is hyperlinked to the line in the file that caused the problem. Using your mouse, you can jump directly to the line in your program that has caused the error. After you have made the changes, make sure you save your file before trying to run the program again in the command window.

A.3 USEFUL MATLAB TOOLS

This section introduces general techniques for finding errors, as well as using automatic code analysis functions in order to detect possible areas for improvement within the MATLAB code. In particular, the MATLAB debugger features located within the Editor, as well as equivalent Command Window debugging functions, will be covered.

Debugging is the process by which you isolate and fix problems with your code. Debugging helps to correct two kinds of errors:

- Syntax errors: For example, misspelling a function name or omitting a parenthesis.
- Run-time errors: These errors are usually algorithmic in nature. For example, you might modify the wrong variable or code a calculation incorrectly. Run-time errors are usually apparent when an M-file produces unexpected results. Run-time errors are difficult to track down because the function’s local workspace is lost when the error forces a return to the MATLAB base workspace.

A.3.1 Code Analysis and M-Lint Messages

MATLAB can check your code for problems and recommend modifications to maximize the performance and maintainability through messages, sometimes referred to as M-Lint messages. The term *lint* is the name given to a similar tool used with other programming languages such as C. In MATLAB, the M-Lint tool displays a message for each line of an M-file it determines possesses the potential to be improved. For example, a common M-Lint message is that a variable is defined but never used in the M-file.

You can check for coding problems using three different ways, all of which report the same messages:

- Continuously check code in the Editor while you work: View M-Lint messages and modify your file based on the messages. The messages update automatically and continuously so you can see if your changes addressed the

issues noted in the messages. Some messages offer extended information, automatic code correction, or both.

- Run a report for an existing MATLAB code file: From a file in the Editor, select Tools > Code Analyzer > Show Code Analyzer Report.
- Run a report for all files in a folder: In the Current Folder browser, click the Actions button, then select Reports > Code Analyzer Report.

For each message, review the message and the associated code in order to make changes to the code itself based on the message via the following process:

- Click the line number to open the M-file in the Editor/Debugger at that line.
- Review the M-Lint message in the report and change the code in the M-file, based on the message.
- Note that in some cases, you should not make any changes based on the M-Lint messages because the M-Lint messages do not apply to that specific situation. M-Lint does not provide perfect information about every situation.
- Save the M-file. Consider saving the file to a different name if you made significant changes that might introduce errors. Then you can refer to the original file as you resolve problems with the updated file.
- If you are not sure what a message means or what to change in the code as a result, use the Help browser to look for related topics.

You can also get M-Lint messages using the `mlint` function. For more information about this function, you can type `help mlint` in the Command Window. Read the online documentation [3] for more information about this tool.

A.3.2 Debugger

The MATLAB Editor, graphical debugger, and MATLAB debugging functions are useful for correcting run-time problems. They enable access to function workspaces and examine or change the values they contain. You can also set and clear *breakpoints*, which are indicators that temporarily halt execution in a file. While stopped at a breakpoint, you can change the workspace contexts, view the function call stack, and execute the lines in a file one by one.

There are two important techniques in debugging: one is the *breakpoint* while the other is the *step*. Setting *breakpoints* to pause the execution of a function enables you to examine values where you think the problem might be located. While debugging, you can also *step* through an M-file, pausing at points where you want to examine values. There are three basic types of breakpoints that you can set in the M-files, namely:

- A standard breakpoint, which stops at a specified line in an M-file.
- A conditional breakpoint, which stops at a specified line in an M-file only under specified conditions.
- An error breakpoint that stops in any M-file when it produces the specified type of warning, error, or NaN or infinite value.

You cannot set breakpoints while MATLAB is busy (e.g., running an M-file) unless that M-file is paused at a breakpoint. While the program is paused, you can view the value of any variable currently in the workspace, thus allowing you to examine

values when you want to see whether a line of code has produced the expected result or not. If the result is as expected, continue running or step to the next line. If the result is not as expected, then that line, or a previous line, contains an error.

While debugging, you can change the value of a variable in the current workspace to see if the new value produces expected results. While the program is paused, assign a new value to the variable in the Command Window, Workspace browser, or Array Editor. Then continue running or stepping through the program. If the new value does not produce the expected results, the program has a different or another problem.

Besides using the Editor, which is a graphical user interface, you can also debug MATLAB files by using debugging functions from the Command Window, or you can use both methods interchangeably. Read the online documentation [4] for more information about this tool.

A.3.3 Profiler

Profiling is a way to measure the amount of time a program spends on performing various functions. Using the MATLAB Profiler, you can identify which functions in your code consume the most time. You can then determine why you are calling them and look for ways to minimize their use. It is often helpful to decide whether the number of times a particular function is called is reasonable. Since programs often have several layers, your code may not explicitly call the most time-consuming functions. Rather, functions within your code might be calling other time-consuming functions that can be several layers down into the code. In this case, it is important to determine which of your functions are responsible for such calls.

Profiling helps to uncover performance problems that you can solve by:

- Avoiding unnecessary computation.
- Changing your algorithm to avoid costly functions.
- Avoiding recomputation by storing results for future use.

When you reach the point where most of the time is spent on calls to a small number of built-in functions, you have probably optimized the code as much as you can expect. You can use any of the following methods to open the Profiler:

- Select Desktop → Profiler from the MATLAB desktop.
- Select Tools → Open Profiler from the menu in the MATLAB Editor/Debugger.
- Select one or more statements in the Command History window, right-click to view the context menu, and choose Profile Code.
- Enter the following function in the Command Window: `profile viewer`.

To profile an M-file or a line of code, follow these steps after you open the Profiler, as shown in Figure A.1:

1. In the *Run this code* field in the Profiler, type the statement you want to run.
2. Click *Start Profiling* (or press *Enter* after typing the statement).
3. When profiling is complete, the *Profile Summary* report appears in the Profiler window. Read the online documentation [5] for more information about this tool.

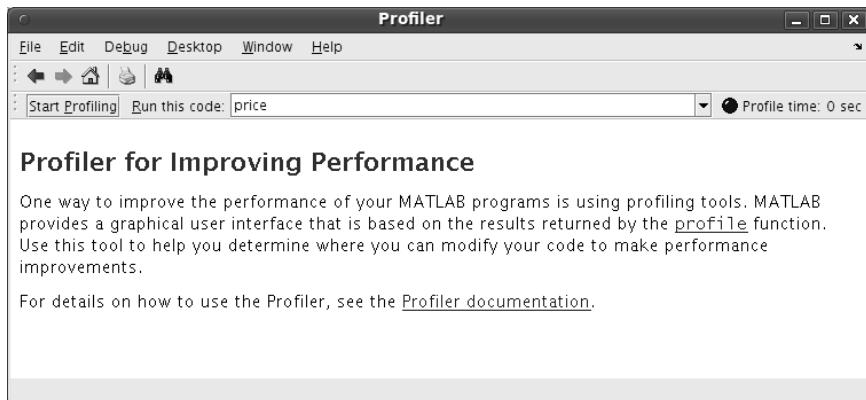


Figure A.1 The profiler window.

A.4 SIMULINK INTRODUCTION

Simulink (Simulation and Link) is an extension of MATLAB created by MathWorks Inc. It works with MATLAB to offer modeling, simulation, and analysis of dynamical systems within a graphical user interface (GUI) environment. The construction of a model is simplified with click-and-drag mouse operations. Simulink includes a comprehensive block library of toolboxes for both linear and nonlinear analyses. Models are hierarchical, which allow using both top-down and bottom-up approaches. As Simulink is an integral part of MATLAB, it is easy to switch back and forth during the analysis process and thus, the user may take full advantage of features offered in both environments. This section presents the basic features of Simulink and is focused on Communications System Toolbox.

A.5 GETTING STARTED IN SIMULINK

A.5.1 Start a Simulink Session

To start a Simulink session, you need to bring up the MATLAB program first. Then, from MATLAB command window, enter:

```
>> simulink
```

Alternatively, you may click on the Simulink icon located on the toolbar, as shown in Figure A.2.

The library browser window of Simulink, as shown in Figure A.3, will pop up presenting the libraries for model construction. To see the content of the toolbox, click on the “+” sign at the beginning of each toolbox. For example, Figure A.3 shows the content of Communications System Toolbox.

A.5.2 Start a Simulink Model

To start a model, click on the *new file* icon as shown in Figure A.3. Alternatively, you may use the keyboard shortcut CTRL+N. A new window will appear on the screen. You will be constructing and simulating your model in this window. A screenshot of a typical working model window is shown in Figure A.4.

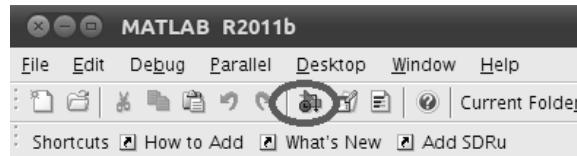


Figure A.2 Start a Simulink session by clicking on the Simulink icon, which is circled.

A.5.3 Simulink Model Settings

In this book, most of the simulations and implementations are designed for communication systems, so a proper Simulink setting for this purpose is very helpful.

`commstartup` changes the default Simulink model settings to values more appropriate for the simulation of communication systems. The changes apply to new models that you create later in the MATLAB session, but not to previously created models. To install the communications-related model settings each time you start MATLAB, invoke `commstartup` from your `startup.m` file. To be more specific, the settings in `commstartup` cause models to:

- Use the variable-step discrete solver in single-tasking mode.
- Use starting and ending times of 0 and Inf, respectively.

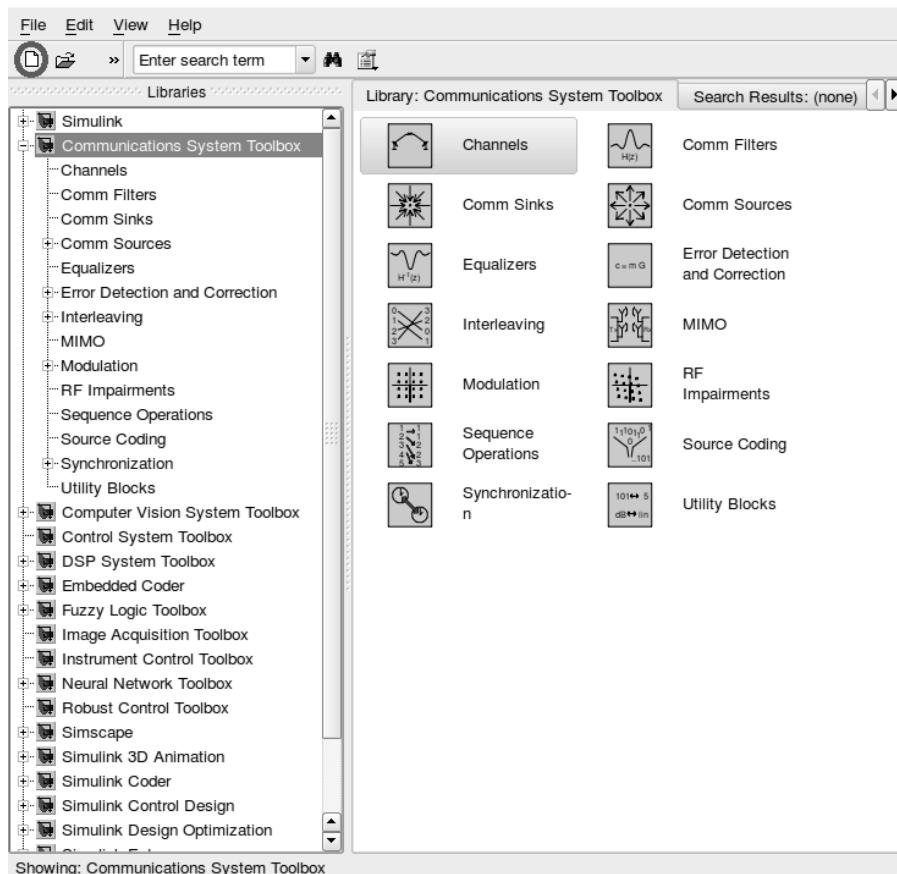


Figure A.3 Simulink library browser. To create a new model, click on the *new file* icon, which is circled.



Figure A.4 A screenshot of a typical Simulink model window.

- Avoid producing a warning or error message for inherited sample times in source blocks.
- Set the Simulink Boolean logic signals parameter to Off.
- Avoid saving output or time information to the workspace.
- Produce an error upon detecting an algebraic loop.
- Inline parameters if you use the Model Reference feature of Simulink.

If your communications model does not work well with these default settings, you can change each of the individual settings as the model requires.

A.6 BUILD A SIMULINK MODEL

To demonstrate how a system is represented using Simulink, we will build the block diagram for a simple model consisting of a random number input and two scope displays, which is shown in Figure A.5.

This model consists of four blocks: *Random Number*, *Abs*, and two *Scopes*. The *Random Number* is a source block from which a random signal is generated. This signal is transferred through a line in the direction indicated by the arrow to

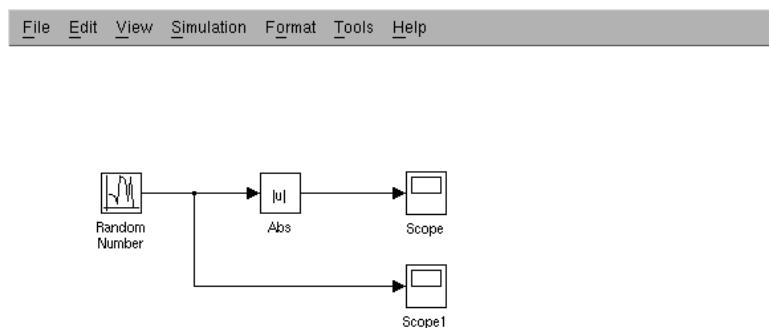


Figure A.5 A simple Simulink model consisting of four blocks.

the *Abs* Math Block. The *Abs* block modifies its input signal (calculates its absolute value) and outputs a new signal through a line to the *Scope* block. The *Scope* is a sink block used to display a signal like an oscilloscope.

Building a Simulink model is usually accomplished through the following steps:

1. Get the necessary blocks from the Library Browser and place them in the model window.
2. Modify the parameters of the blocks according to the system that we are modeling.
3. Connect all the blocks with lines to complete the Simulink model.

Each of these steps will be explained in detail using our example system. Once a system is built, simulations can be run to analyze its behavior.

A.6.1 Obtain the Blocks

Each of the blocks we will use in our example model will be gathered from the Simulink Library Browser. To place the *Random Number* block into the model window, follow these steps:

1. Click on the “+” in front of “Simulink”, since “Sources” is a subfolder beneath the “Simulink” folder.
2. Scroll down until you see the “Sources” folder. Click on “Sources”, the right window will display the various source blocks available for us to use and *Random Number* is one of them, as shown in Figure A.6.
3. To insert the *Random Number* block into your model window, click on it in the Library Browser and drag the block into your window.

The same method can be used to place the *Abs* and *Scope* blocks in the model window. The *Abs* block can be found in the “Math Operations” subfolder and the *Scope* block is located in the “Sink” subfolder. Arrange the four blocks in the model window by selecting and dragging an individual block to a new location so that they look similar to Figure A.7.

A.6.2 Set the Parameters

Simulink allows us to modify the default settings of the blocks in our model so that they can accurately reflect the characteristics of the system we are analyzing. For example, we can modify the *Random Number* block by double-clicking on it. Doing so will cause the following window to appear, as shown in Figure A.8. Note that Simulink gives a brief explanation of the block’s function in the top portion of this window. In this case, it shows “Output a normally (Gaussian) distributed random signal. Output is repeatable for a given seed.”

This window allows us to adjust the mean, variance, and seed of the random input. The sample time value indicates the time interval between successive readings of the signal.

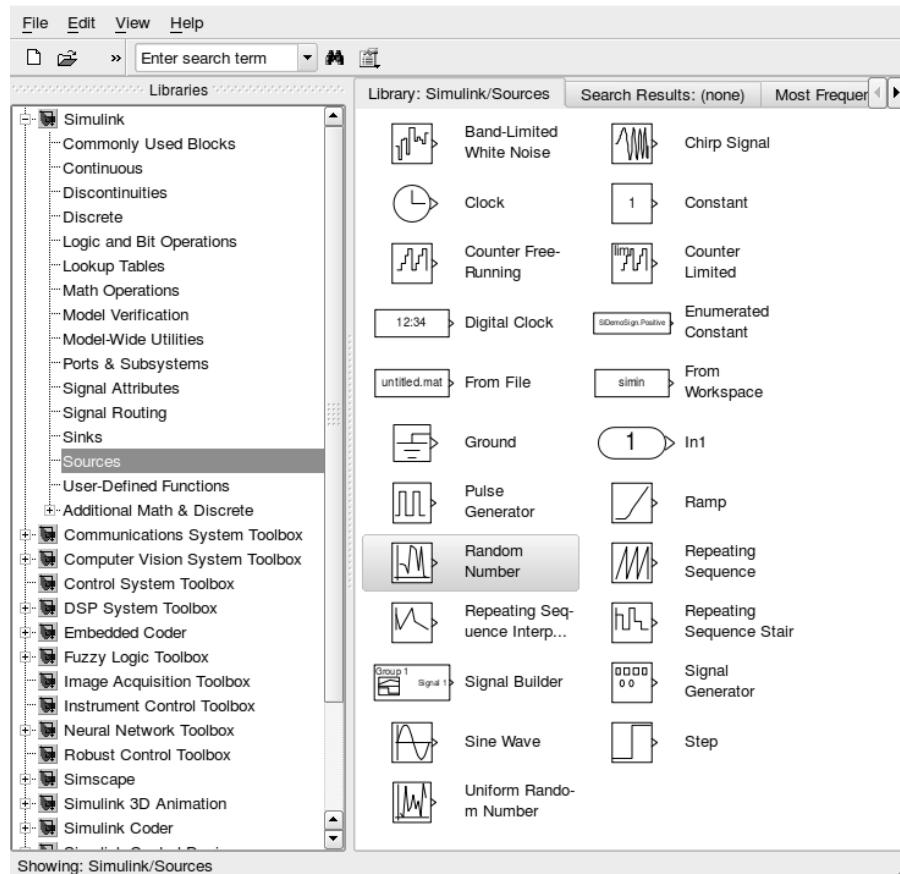


Figure A.6 Obtain *Random Number* block from Simulink Library Browser.

Let us assume that our system's random input has the following Gaussian distribution: Mean = 0, Variance = 1, Seed = 0. Enter these values into the appropriate fields and leave the "Sample time" set to 0.1. Then click "OK" to accept them and exit the window.

The *Abs* block simply returns the absolute value for the input and the *Scope* block plots its input signal as a function of time, so there are no system parameters

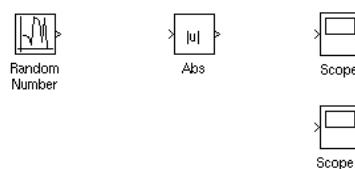


Figure A.7 Obtain all the necessary blocks for the model and arrange them in the model window.

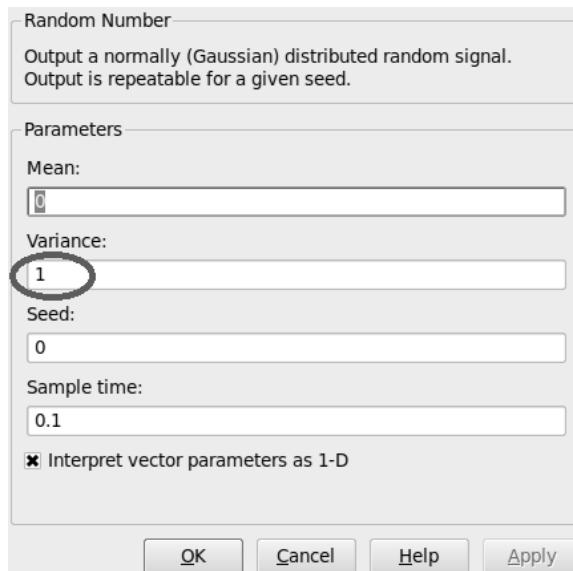


Figure A.8 By double-clicking on the *Random Number* block, we can open the block and set the parameters. The “Variance” parameter is highlighted because its value will be changed in Section A.7.

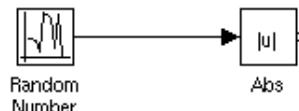
that we can change for these two blocks. We will look at the *Scope* block in more detail after we have run our simulation.

A.6.3 Connect the Blocks

In order to accurately reflect the system we are modeling, the Simulink blocks must be properly connected. In our example system, the signal output by the *Random Number* block is transmitted to the *Abs* block. The *Abs* block outputs its absolute value to the *Scope* block, which graphs the signal as a function of time. Thus, we need to draw lines from the output of the *Random Number* block to the input of the *Abs* block, and from the output of the *Abs* block to the input of the *Scope* block.

Lines are drawn by dragging the mouse from where a signal starts (output terminal of a block) to where it ends (input terminal of another block). When drawing lines, it is important to make sure that the signal reaches each of its intended terminals. Simulink will turn the mouse pointer into a crosshair when it is close enough to an output terminal to begin drawing a line, and the pointer will change into a double crosshair when it is close enough to snap to an input terminal. A signal is properly connected if its arrowhead is filled in, as shown in Figure A.9(a). If the arrowhead is open, as shown in Figure A.9(b), it means that the signal is not connected to both blocks. To fix an open signal, you can treat the open arrowhead as an output terminal and continue drawing the line to an input terminal in the same manner as explained before.

When drawing lines, you do not need to worry about the path you follow. The lines will route themselves automatically. Once blocks are connected, they can be repositioned for a neater appearance. This is done by clicking on and dragging each block to its desired location (signals will stay properly connected and will reroute



(a)



(b)

Figure A.9 Two blocks can be connected by drawing a line between them. (a) Two blocks are properly connected, where an arrowhead is filled in. (b) Two blocks are not connected, where the arrowhead is open.

themselves). After drawing in the lines and repositioning the blocks, the example system model should look like Figure A.5.

In some models, it will be necessary to branch a signal so that it is transmitted to two or more different input terminals. For example, in this Simulink model, the output of the *Random Number* block is transmitted to two different input terminals, one is *Abs* block, the other is *Scope 1* block. This can be done by first placing the mouse cursor at the location where the signal is to branch. Then, using either the CTRL key in conjunction with the left mouse button or just the right mouse button, drag the new line to its intended destination. The routing of lines and the location of branches can be changed by dragging them to their desired new position. To delete an incorrectly drawn line, simply click on it to select it, and hit the DELETE key.

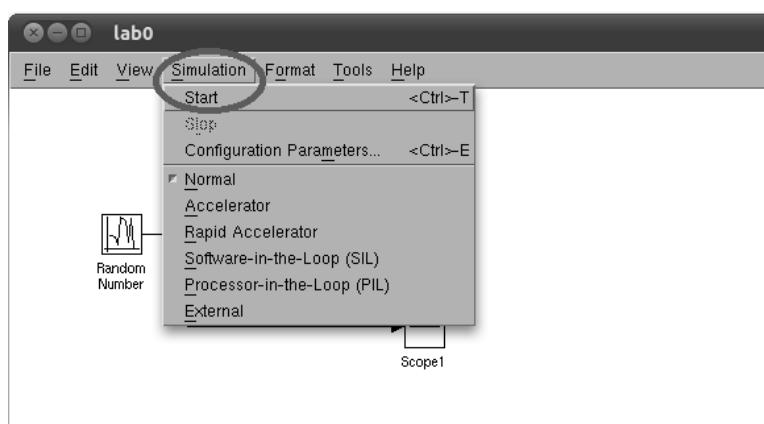
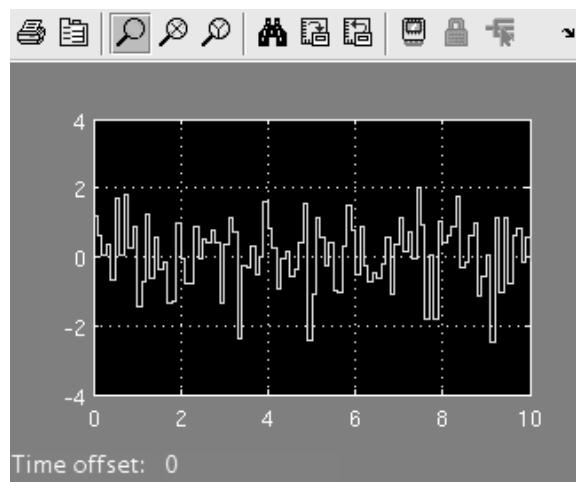


Figure A.10 Start running a simulation by clicking on the Start option of the Simulation menu, which is circled.

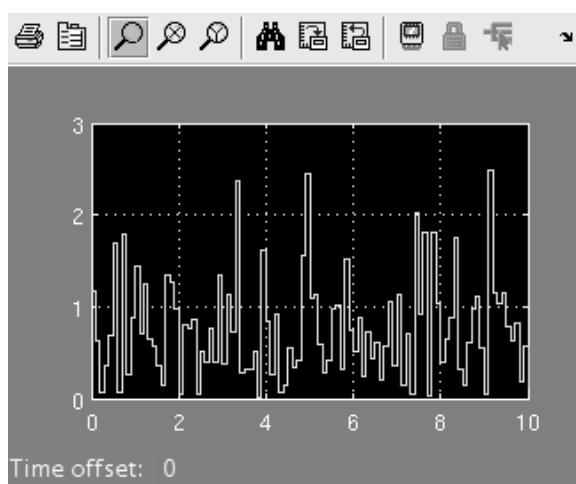
A.7 RUN SIMULINK SIMULATIONS

Now that our model has been constructed, we are ready to simulate the system. To do this, go to the “Simulation” menu and click on “Start”, as shown in Figure A.10, or just click on the “Start/Pause Simulation” button in the model window toolbar (looks like the “Play” button on a VCR) if you are using Windows operating system. Since our example is a relatively simple model, its simulation runs almost instantaneously. With more complicated systems, however, you will be able to see the progress of the simulation by observing its running time in the lower box of the model window.

Double click the *Scope* and the *Scope 1* block to view the random signal and its absolute value for the simulation as a function of time. Once the Scope window appears, click the “Autoscale” button in its toolbar (looks like a pair of binoculars) to scale the graph



(a)



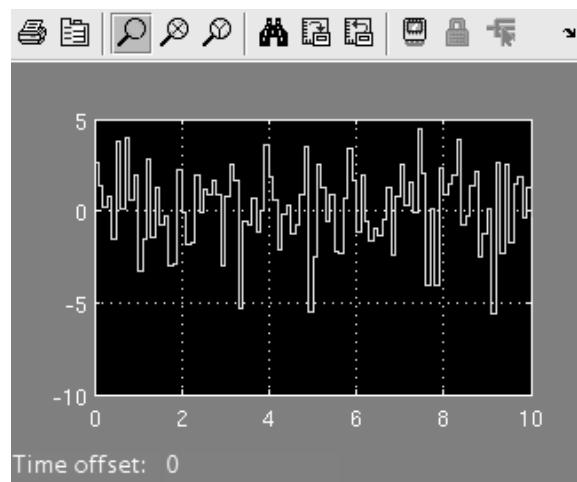
(b)

Figure A.11 The scope displays of the Simulink model. (a) The random signal displayed by *Scope 1*. (b) The absolute value of the random signal displayed by *Scope*.

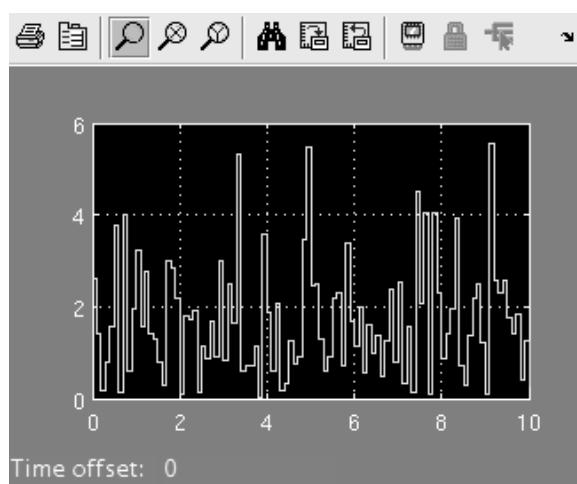
to better fit the window. Having done this, you should be able to see the results shown in Figure A.11. It is very obvious that corresponding to each point in Figure A.11(a), Figure A.11(b) returns its absolute value, which satisfies the theory of this model.

As mentioned in Section A.6.2, we can easily change the parameters of the block by double clicking the block. How would this change affect the output observed by the scopes? Let us change the “Variance” of the *Random Number* block to be 2 and keep the other parameters as they were. Then, re-run the simulation and view the scopes. Please note the scope graphs will not change unless the simulation is re-run, even though the “Variance” has been modified. The new scope graphs should now look like Figure A.12.

Comparing Figure A.12 with Figure A.11, we notice that the only difference between this output and the one from our original system is the amplitude of the



(a)



(b)

Figure A.12 The scope displays of the Simulink model with variance = 2. (a) The random signal displayed by Scope 1. (b) The absolute value of the random signal displayed by Scope.

random signal. The shapes are identical due to the same seed value. However, since the variance is doubled, the new amplitude is about $\sqrt{2}$ times as the original amplitude.

References

- [1] The MathWorks, *MATLAB Documentation*, <http://www.mathworks.com/help/techdoc/>.
- [2] University of Michigan, *Simulink Basics Tutorial*, http://www.engin.umich.edu/group/ctm/working/mac/simulink_basics/.
- [3] The MathWorks, *Avoid Mistakes While Editing Code*, http://www.mathworks.com/help/techdoc/matlab_env/brqxeeu-151.html.
- [4] The MathWorks, *Debugging Process and Features*, http://www.mathworks.com/help/techdoc/matlab_env/brqxeeu-175.html.
- [5] The MathWorks, *Profiling for Improving Performance*, http://www.mathworks.com/help/techdoc/matlab_env/f9-17018.html.

Universal Hardware Driver (UHD)

The UHD is the “Universal Software Radio Peripheral” hardware driver. It works on all major platforms (Linux, Windows, and Mac), and can be built with GCC, Clang, and MSVC compilers [1].

The goal of the UHD is to provide a host driver and API for current and future Ettus Research products. Users will be able to use the UHD driver standalone or with 3rd party applications, such as Simulink. However, there are a few things you need to do before you can actually use your ethernet-based USRP hardware with the MATLAB R2011b¹ software via UHD interface, which will be introduced in the following sections. In this appendix, we assume that the employed operating system is Ubuntu [2].

B.1 SETTING UP YOUR HARDWARE

Connect the ethernet-based USRP board to your computer using a gigabit Ethernet cable. For optimal results, attach the USRP directly to the computer, on a private network (i.e., one without any other computers, routers, or switches). Connecting the USRP board directly to your computer requires the following:

- A version of USRP firmware which supports UHD. It will be introduced in Section B.2.
- A gigabit Ethernet card, which supports flow control and has a static IP address. For more information, follow these guidelines below.

B.2 INSTALLING UHD-BASED USRP I/O BLOCKS

With the latest UHD interface support and Communications System Toolbox, you can use ethernet-based USRP with MATLAB and Simulink. Starting MATLAB 2012b, USRP support package can be downloaded and installed using “target-installer” command in MATLAB. Before MATLAB 2012b, go to [3], download the support package. You need to fill out a form and submit it before you can save the package in your local computer. After you get the package, install it following the steps 1–4 in README.txt.

1. MATLAB and Simulink supports USRP radios since version R2011b.

B.3 BURNING THE FIRMWARE TO AN SD CARD

MATLAB R2011b support for ethernet-based USRP devices has been tested using UHD binary download release 003.002.003. For versions other than R2012b, the user should check the MathWorks documentation for the support package to find out which version of UHD is supported. You can download the necessary firmware images from [4], and burn them to an SD card following the instructions below. Please note you need to be a root user in order to perform the following steps on Ubuntu operating system.

1. Insert the SD memory card into the USB burner.
2. From the terminal, change to the directory containing the USRP SD Card Burner GUI using ‘cd’ command.
3. Launch the USRP SD Card Burner GUI by typing ‘sudo python usrp2_card_burner_gui.py’ at the terminal if you are using this card with a USRP2 radio. usrp2_card_burner_gui.py can be downloaded from [5]. Note: Do not save directly from the link on this page. Go into the link, and choose “Download” to save.
4. Select the drive for the memory card device. For example, if the memory card device is on the sdd drive, select /dev/sdd.
5. Click the “Firmware Image” button and navigate to the firmware image file.
6. Click the “FPGA Image” button and navigate to the image file.
7. Click the “Burn SD Card” button.
8. If a prompt indicating that burning the firmware appears, confirm that it states the verification information passed and click “OK.”
9. Insert the SD memory card into the USRP radio and power on.

B.4 CONFIGURE THE ETHERNET CARD

In Ubuntu operating system, in order to configure the Ethernet card for your USRP hardware, you need to perform the following tasks:

1. Select System → Preferences → Network Connections
2. Select the network card to which the USRP will be attached. If you are not sure which network card the USRP is attached to, you can use ifconfig command to find it out.
3. Click the “Edit” button
4. Go to the “IPv4 Settings” tab
5. Select “Manual” for “Method”, and click the “Add” button
6. Use the following addresses:
 - Address: 192.168.10.1
 - Netmask: 255.255.255.0
 - Leave Gateway blank

Since USRP radio has an IP address of 192.168.10.2, you need to select a number 192.168.10.X where X is anything but 2.

B.5 MODIFY THE IPTABLES

Iptables is a firewall, installed by default on all official Ubuntu distributions (Ubuntu, Kubuntu, Xubuntu). When you install Ubuntu, iptables is there. However, in order to use the SDRu Transmitter and Receiver blocks, you need to modify the Iptables to allow all traffic. You can access the Iptables on your computer by typing the following commands on the terminal:

- sudo bash
- cd /root/scripts/
- emacs iptables.sh

Now, you should have the Iptables open in front of you. You only need to modify two lines concerning the default firewall policies:

- IPTABLES_COM -P INPUT ACCEPT
- IPTABLES_COM -P FORWARD ACCEPT

When this modification is done, you should save the Iptables and execute it by typing the following command on the terminal:

- ./iptables.sh

B.6 EACH TIME YOU USE

In future MATLAB sessions, you must run `setupsdru` to use Communications with USRP. `setupsdru` is under the `commusrp` directory.

You can automate this step by including the commands prompted by the “Add SDRu” shortcut to your `startup.m` file. If you prefer to run it manually you can use the shortcut labelled “Add SDRu”, that has been provided on the MATLAB shortcut bar. The installer will run `setupsdru` when you first install Communications with USRP.

B.7 PROBLEMS WITH UNICODE

A user may get the following error when he runs `findsdru` (after running `setupsdru`):

*??? The conversion from a local code page string to unicode changes the number of characters. This is not supported. Error in => `usrp_uhd_mapi`

This error occurs due to incompatible localization between the USRP hardware and your host computer. It can be fixed on Linux by going to a command shell and executing:

- `export LANG=C`, or
- `export LANG=en_US.ISO8859-1`

These commands reset the locale to be compatible with the `usrp_uhd_mapi` code.

References

- [1] <http://ettus-apps.sourcerepo.com/redmine/ettus/projects/uhd/wiki/>
- [2] <http://www.ubuntu.com/>
- [3] <http://www.mathworks.com/programs/usrp/>
- [4] http://files.ettus.com/binaries/uhd_stable/releases/uhd_003.002.003-release/images-only/
- [5] <https://ettus-apps.sourcerepo.com/redmine/ettus/projects/uhd/repository/revisions/master/show/host/utils>

Data Flow on USRP

This appendix introduces how data flows on the USRP board. After reading this appendix, you should have a better insight on the USRP board. To make things clearer, the data flow is introduced in two parts, receive (RX) path and transmit (TX) path, as shown in Figure C.1.

C.1 RECEIVE PATH

At the RX path, we have four ADCs, and four DDCs. Generally speaking, on each daughterboard, the two analog input signals are sent to two separate ADCs. The digitized samples are then sent to the FPGA for processing. Upon entering the FPGA, the digitized signals are routed by a multiplexer (MUX) to the appropriate digital down-converter (DDC). There are four DDCs. Each of them has two inputs (I and Q). The MUX is like a router or a circuit switcher. It determines which ADC (or constant zero) is connected to each DDC input. We can control the MUX using `usrp.set_mux()` method in Python. This allows for having multiple channels selected out of the same ADC sample stream.

The standard FPGA configuration includes digital down-converters implemented with four stages cascaded integrator-comb (CIC) filters. CIC filters are very-high-performance filters using only adds and delays. For spectral shaping and out-of-band signals rejection, there are also 31 tap halfband filters cascaded with the CIC filters to form complete DDC stage. The standard FPGA configuration implements two complete digital down-converters. Also there is an image with four DDCs but without halfband filters. This allows one, two, or four separate RX channels. It's possible to specify the firmware and fpga files that are to be used by loading the corresponding rbf files. By default, an rbf is loaded from `/usr/local/share/usrp/rev{2,4}`. The one used unless you specify the `fpga_filename` constructor argument when instantiating a usrp source or sink is `std_2rxhb_2tx.rbf`. Table C.1 lists the current three different rbf files and their usage.

Now let's take a closer look at the digital down-converter. What does it do? First, it down-converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. Figure C.2 shows the block diagram of the DDC. The complex input signal (IF) is multiplied by the constant frequency (usually also IF) exponential signal. The resulting signal is also complex and entered at 0. Then we decimate the signal with a factor M .

Note that when there are multiple channels (up to four), the channels are interleaved. For example, with four channels, the sequence sent over the USB would be "I0 Q0 I1 Q1 I2 Q2 I3 Q3 I0 Q0 I1 Q1," and so on. The USRP can operate in full-duplex

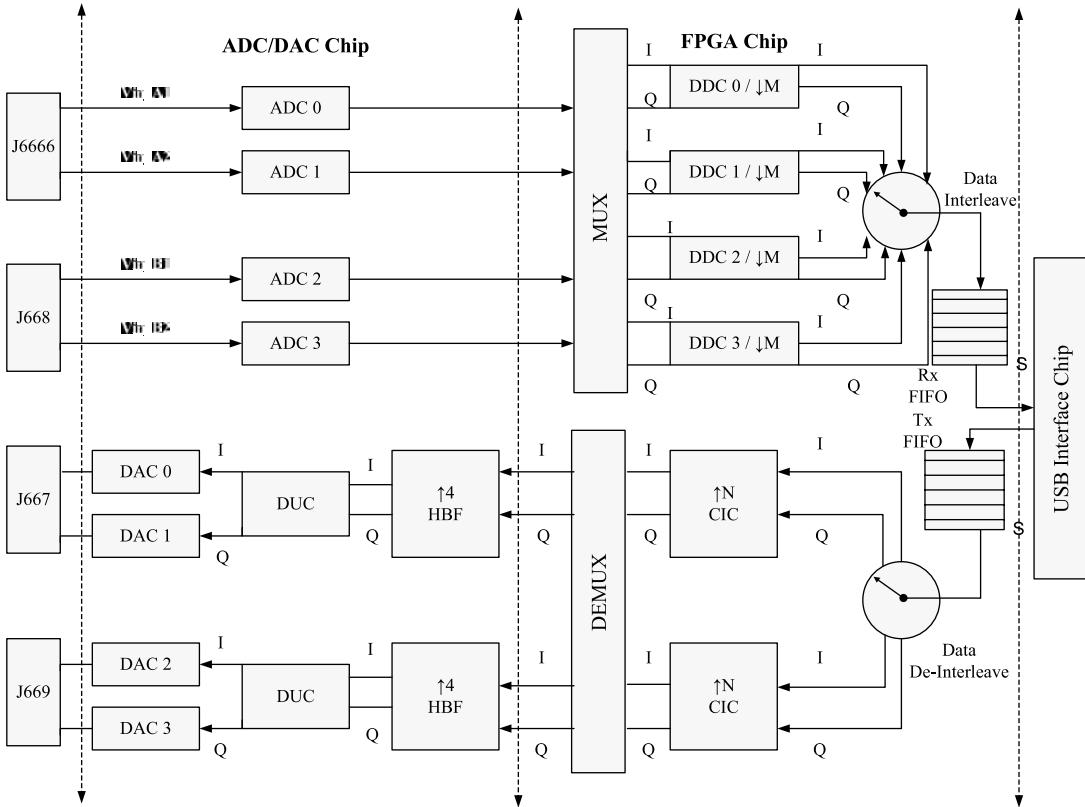


Figure C.1 USRP receive and transmit paths [1].

mode. When in this mode, the transmit and receive sides are completely independent of one another. The only consideration is that the combined data rate over the bus must be 32 MBps or less. Finally, the I/Q complex signal enters the computer via the USB.

To illustrate the effect of half-band filter in Rx path, two different situations are discussed. Given the bandwidth of each Rx channel is f , if we set the center frequency at 0Hz, then, at frequency domain, it is $[-f/2, f/2]$.

C.1.1 Situation 1

Suppose there are four users (u_1, u_2, u_3 and u_4), and each of them uses one of the Rx channels, as shown in Figure C.3. In frequency domain, the bandwidth they are occupying are all from $f/2$ to $f/2$. In this situation, the FPGA rbf file should be `std_4rx_01x.rbf`, which means the FPGA firmware contains four Rx paths without halfbands and zero tx paths.

Table C.1 Three rbf Files and Their Usage

rbf File Name	FPGA Description
<code>multi_2rxhb_2tx.rbf</code>	There are two or more USRPs
<code>std_2rxhb_2tx.rbf</code>	Contains two Rx paths with halfband filters and two tx paths
<code>std_4rx_0tx.rbf</code>	Contains four Rx paths without halfbands and zero tx paths

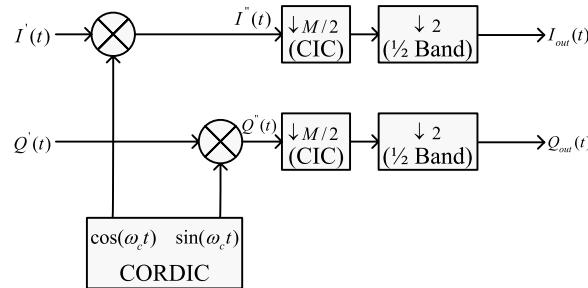


Figure C.2 The structure of digital down converter on receiver path, where the input RF signal $I'(t)$ and $Q'(t)$ is first down-converted to the IF band by a carrier frequency of ω_c , and then decimated with a factor M .

In analog-to-digital converter (ADC), continuous signals are converted to discrete digital numbers. A common and useful way to do this is through the use of a periodic impulse train signal [2], $p(t)$, multiplied by the continuous time signal $x(t)$. This is known as impulse train sampling, as introduced in Section 2.3.2.

Then, the low-pass filter takes out the signal around center frequency and suppresses the rest replicas. If the decimation value of DDC is $M/2$, after entering the

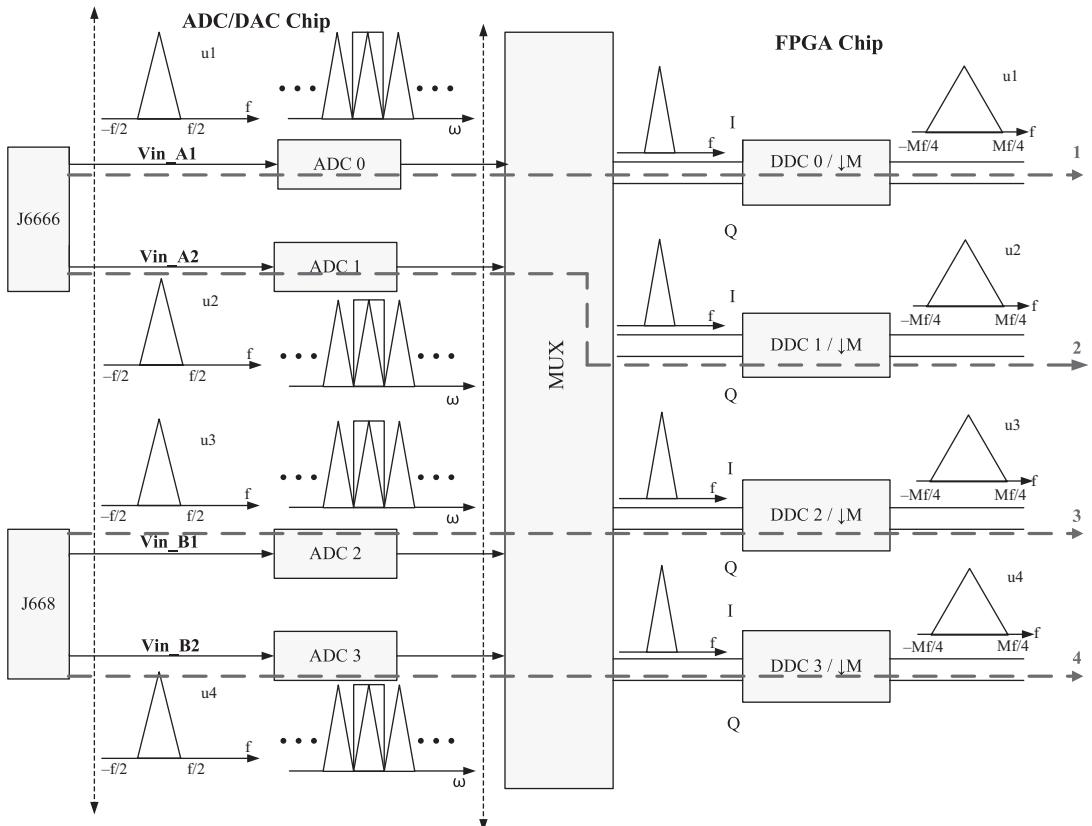


Figure C.3 4 Rx channels (labeled 1, 2, 3, and 4), starting from the inputs A1, A2, B1, and B2, passing through one of the ADCs and one of the DDCs, then reaching the USB in the end.

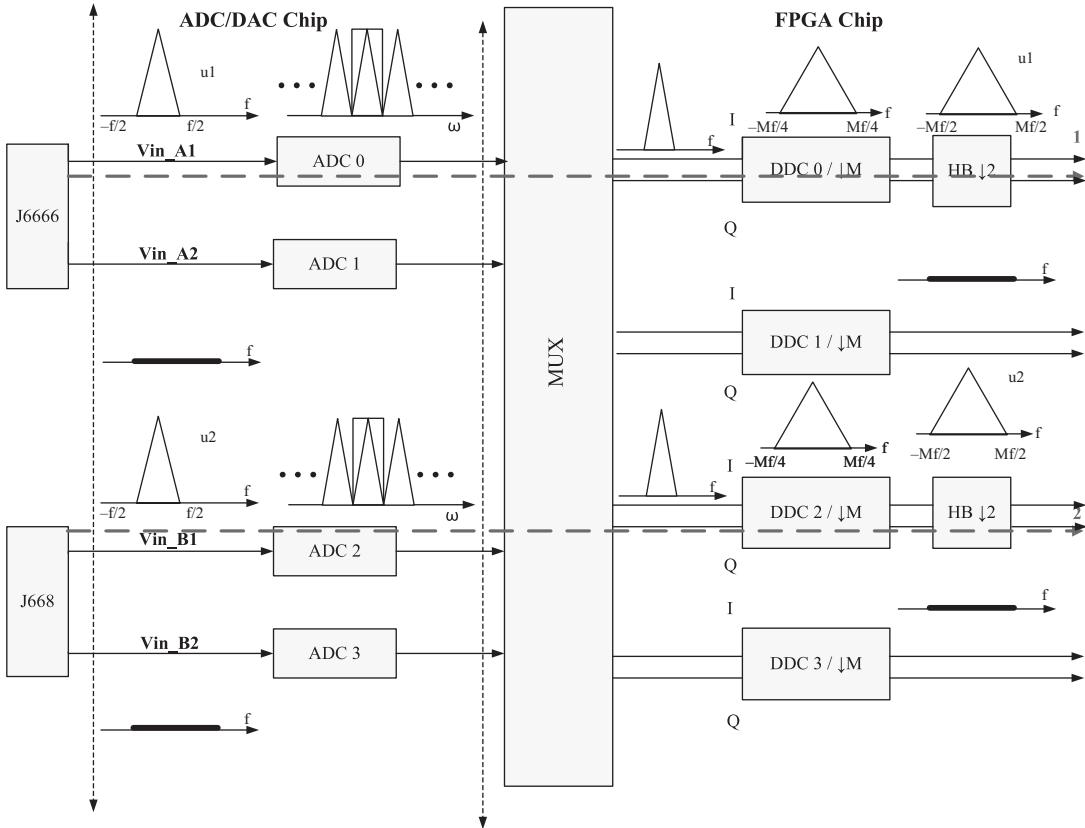


Figure C.4 Two Rx channels (labeled 1 and 2), starting from the inputs A1 and B1, passing through one of the ADCs and one of the DDCs with half-band filter, then reaching the USB in the end.

DDC, the bandwidth of each signal will be expanded to $[-M f/4, M f/4]$. The sum of all the bandwidth is $2Mf$.

C.1.2 Situation 2

Suppose there are two users (u_1 and u_2), and each of them uses one of the Rx channels, as shown in Figure C.4. In frequency domain, the bandwidth they are occupying are both from $-f/2$ to $f/2$. In this situation, the FPGA rbf file should be `std_2rxhb_2tx.rbf`, which means FPGA firmware contains two Rx paths with half-bands and two tx paths. If the decimation value of DDC is $M/2$, after entering the DDC, the bandwidth of each signal will be expanded to $[-M f/4, M f/4]$. Then, after entering the half-band filter, the bandwidth of each signal will be further expanded to $[-M f/2, M f/2]$. The sum of all the bandwidth is still $2Mf$.

C.2 TRANSMIT PATH

At the TX path, the story is pretty much the same, except that it happens in reverse. We need to send a baseband I/Q complex signal to the USRP board. The digital up-

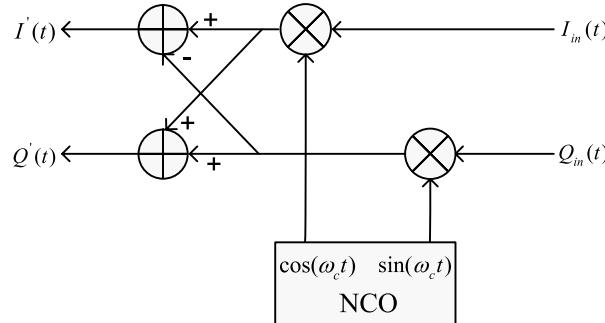


Figure C.5 The structure of digital up-converter on transmitter path, where a baseband I/Q complex signal $I_{in}(t)$ and $Q_{in}(t)$ is upconverted to the IF band by a carrier frequency of ω_c . $I'(t)$ and $Q'(t)$ will be sent to the DAC.

converter (DUC) will interpolate the signal, up-convert it to the IF band, and finally send it through the DAC, as shown in Figure C.5.

More specifically, first, interleaved data sent from the host computer is pushed into the transmit FIFO on the USRP. Each complex sample is 32 bits long (16 bits for in-phase, and 16 bits for quadrature). This data is de-interleaved and sent to the input of an interpolation stage. Assuming, the user has specified an interpolation factor of L , this interpolation stage will interpolate the input data by a factor of $L/4$ using CIC filters.

The output of the interpolation stage is sent to the demultiplexer (DEMUX). The DEMUX is less complicated than the receiver MUX. Here, the in-phase and quadrature output of each CIC interpolator is sent to in-phase and quadrature inputs of one of the DAC chips on the motherboard. The user specifies which DAC chip receives the output of each CIC interpolator [3].

Inside of the AD9862 CODEC chip, the complex-valued signal is interpolated by a factor of four using half-band filter interpolators. This completes the requested factor of L interpolation. After the half-band interpolators, the complex-valued signal is sent to a DUC. Note, at this point the signal is not necessarily modulated to a carrier frequency. The daughterboard might further up-convert the signal.

The in-phase and quadrature output of the DUC are sent as 14-bit samples to individual digital-to-analog converters, which convert them to analog signals at a rate of 128 mega-samples per second. These analog signals are then sent from the AD9862 to either daughterboard interface J667 or J669, which represent slots TxA and TxB, respectively.

Different from the DDCs, the DUCs on the transmit side are actually contained in the AD9862 CODEC chips, not in the FPGA. The only transmit signal processing blocks in the FPGA are the interpolators. The interpolator outputs can be routed to any of the 4 CODEC inputs [1].

As we have mentioned before, the multiple RX channels (1, 2, or 4) must all be the same data rate (i.e., same decimation ratio). The same applies to the 1, 2, or 4 TX channels, which each must be at the same data rate (which may be different from the RX rate) [4].

References

- [1] Patton, Lee K., *A Gnn Radio Based Software-Defined Radar*, Master's Thesis, 2007.
- [2] Proakis, John G., and Dimitris G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications*, 3rd ed., Upper Saddle River, NJ, 1996.
- [3] Shen, Dawei, *Gnn Radio Tutorials*, <http://www.nd.edu/~dshen/GNU/>, [Online]. 2005.
- [4] Abbas Hamza, Firas, *The USRP under 1.5X Magnifying Lens!*, gnuradio.org/redmine/attachments/129/USRP_Documentation.pdf, 2008.

Quick Reference Sheet

D.1 LINUX

D.1.1 Helpful Commands

sudo - Run commands with administrative privileges.

Note: If you use “sudo” to run a single command as root, you might see error messages. To avoid these error messages, simply use “sudo -i” or “sudo su” to log in to run multiple commands as root. Or, use “sudo -H” to force sudo to use root’s home directory instead.

ifconfig - Configure network interfaces.

- Use the “-a” option to list all interfaces.
- Use “up”/“down” to enable/disable interfaces.

D.1.2 Modify the Iptables

Iptables is a firewall, installed by default on all official Ubuntu distributions (Ubuntu, Kubuntu, Xubuntu). When you install Ubuntu, iptables is there. However, in order to use the SDRu Transmitter and Receiver Blocks, you need to modify the Iptables to allow all traffic.

You can access the Iptables on your computer by typing the following commands on the terminal:

- sudo bash
- cd /root/scripts/
- emacs iptables.sh

Now, you should have the Iptables open in front of you. You only need to modify two lines concerning the default firewall policies:

- IPTABLES _COM -P INPUT ACCEPT
- IPTABLES -COM -P FORWARD ACCEPT

When this modification is done, you should save the Iptables and execute it by typing the following command on the terminal:

- ./iptables.sh

D.2 MATLAB

D.2.1 How to Start MATLAB

- Windows: Choose the submenu “Programs” from the “Start” menu. From the “Programs” menu, open the “MATLAB” submenu. From the “MATLAB” submenu, choose “MATLAB.”
- Linux: At the prompt, type `matlab`.
- Mac: Double-click on the icon for MATLAB.

You can quit MATLAB by typing `exit` in the command window.

D.2.2 The MATLAB Environment

The MATLAB environment consists of menus, buttons, and a writing area similar to an ordinary word processor.

D.2.2.1 Command Window

The writing area that you will see when you start MATLAB, is called the *command window*. In this window, you give the commands to MATLAB. Once you have typed the command you wish MATLAB to perform, press `<enter>`. If you want to interrupt a command that MATLAB is running, type `<ctrl> + <c>`. The command window is also useful if you just want to use MATLAB as a scientific calculator or as a graphing tool.

D.2.2.2 Command History Window

The commands you type in the command window are stored by MATLAB and can be viewed in the *command history window*. To repeat a command you have already used, you can simply double-click on the command in the history window, or use the `<up arrow>` at the command prompt to iterate through the commands you have used until you reach the command you desire to repeat.

D.2.3 Obtaining Help

To obtain help on any of the MATLAB commands, you simply need to type `help <command>` at the command prompt. For example, to obtain help on the “`setsupsdru`” function, we type at the command prompt: `help setsupsdru`.

You may also get help about commands using the “Help Desk,” which can be accessed by selecting the MATLAB Help option under the Help menu.

D.2.4 Variables in MATLAB

You can define your own variables in MATLAB. Suppose you need to use the value of $2\cos(3)$ repeatedly. Instead of typing $2*\cos(3)$ over and over again, you can denote this variable as x by typing the following: `x=2*cos(3)`.

If you do not want to have the result of a calculation printed out to the command window, you can put a semi-colon at the end of the command; MATLAB still performs the command in the “background.”

- To know what variables have been declared, type `whos`. Alternatively, you can view the values by opening the *workspace window*. This is done by selecting the **Workspace** option from the **View** menu.
- To erase all variables from the MATLAB memory, type `clear`.
- To erase a specific variable, say `x`, type `clear x`.
- To clear two specific variables, say `x` and `y`, type `clear x y`, that is separate the different variables with a space. Variables can also be cleared by selecting them in the Workspace window and selecting the delete option.

D.2.5 Vectors and Matrices in MATLAB

D.2.5.1 Vectors

Create a vector by putting the elements within [] brackets. For example, `x=[1 2 3 4 5 6 7 8 9]`. We can also create this vector by typing `x=1:9`.

The vector `(1 1.1 1.2 1.3 1.4 1.5)` can be created by typing `x=[1 1.1 1.2 1.3 1.4 1.5]` or by typing `x=1:0.1:1.5`. The default step size is 1, as in the previous example. So if the actual step size is not 1, you need to specify it in your definition, as in this example.

D.2.5.2 Matrices

Matrices can be created according to the following example. The matrix `A` is created by typing `A=[1 2; 3 4; 5 6]`, (i.e., rows are separated with semi-colons).

D.2.5.3 Additional Information

- In MATLAB, the first index of a vector or matrix starts at 1, not 0 as is common with other programming languages.
- If the matrices (or vectors, which are special cases of matrices) are of the same dimensions, then matrix addition, matrix subtraction, and scalar multiplication works just like we are used to.
- If you want to apply an operation such as squaring each element in a matrix, you have to use a dot (.) before the operation you wish to apply. The dot allows you to do operations elementwise. All built-in functions such as `sin`, `cos`, `exp`, and so on automatically act elementwise on a matrix.

D.3 USRP2 HARDWARE

D.3.1 XCVR2450 Daughtercard

- The XCVR2450 daughtercard supports two frequency bands: 2.4 to 2.5 GHz and 4.9 to 5.9 GHz.

- There are two cables inside the USRP2 that connect the RF1 and RF2 SMA connectors on the front of the USRP2 case to connectors J2 and J1 on the daughtercard.
- The XCVR2450 always uses J1 for transmit and J2 for receive. This behavior appears to hard-coded into the FPGA firmware and is not software-controllable at present time.

D.3.2 Sampling

D.3.2.1 Decimation

The sampling frequency of the USRP2 is:

$$f_s = \frac{100}{D} \text{ MHz} \quad (\text{D.1})$$

where D is the *decimation*, whose value can be any multiple of 4 from 4 through 512. (Nonmultiples of 4 yield different types of cascaded integrator-comb (CIC) filters, but this probably isn't what you want.)

The decimation parameter affects the bandwidth of the received signal, since it affects the sampling rate. Decimation essentially involves lowpass-filtering and then downsampling by the decimation factor. Recall that downsampling by a factor of X is equivalent to sampling the original signal at a factor of f_s/X instead of f_s . Therefore, larger decimation factors result in smaller sampling rates and thus smaller bandwidths. (After all, the bandwidth depends on the Nyquist frequency, or $f_s/2$.)

The frequency resolution of the FFT is determined by the sampling frequency (f_s) and the number of points (N) in the FFT:

$$\text{frequency resolution} = \frac{f_s}{N} \quad (\text{D.2})$$

For example, an FFT with 1024 points and a sampling frequency of 128 kHz has a resolution of

$$\text{frequency resolution} = \frac{128 \text{ kHz}}{1024 \text{ points}} = 125 \text{ Hz} \quad (\text{D.3})$$

In other words, each bin of the FFT is 125 Hz wide. (Spectrum analyzers commonly refer to this value as *resolution bandwidth*.)

Note there is the tradeoff between resolution in time and resolution in frequency. FFTs with more points have finer-grain frequency resolution, but require more time to collect the samples (since there are more points). FFTs with fewer points have coarser resolution in frequency. Also note that larger FFTs are more computationally expensive, which is important to keep in mind because you will be running the FFTs in real time.

Larger decimation values result in a smaller sampling rate; thus, if you keep your FFT at 1024 points, larger decimation values will yield finer-grain frequency resolution because $(\text{BW})/(N \text{ points})$ will be a smaller resolution bandwidth value.

The punchline: You should use a high decimation value (e.g., 400 or 512) and a sufficient number of FFT points (e.g., 1024) when taking fine measurements, such as your oscillator frequency offset error measurements. This will afford the best possible measurement precision.

Finally, here are a few additional useful links:

- [http://en.wikipedia.org/wiki/Decimation_\(signal_processing\)](http://en.wikipedia.org/wiki/Decimation_(signal_processing))
- <http://zone.ni.com/devzone/cda/tut/p/id/3983>

D.3.2.2 Interpolation

The interpolation parameter is used for transmitting in DAC whereas the decimation parameter is used for receiving in ADC. Interpolation has the same range of values (any multiple of four from 4 to 512) as decimation.

According to [1]: “The FPGA talks to the DAC at 100 MS/s just like it talks to the ADC at 100 MS/s. The interpolation from 100 MS/s to 400 MS/s happens inside the DAC chip itself. Unless you are doing something fancy, you can think of the DAC as operating at 100 MS/s.”

D.3.3 Clocking

The clock rate of the Spartan-3 FPGA in the USRP2 is 100 MHz.

D.3.4 DDC and DUC

The USRP2 FPGA uses a digital down-converter (DDC) and a digital up-converter (DUC) to convert between baseband and intermediate frequency (IF).

D.3.4.1 Digital Down-Converter

Our standard FPGA configuration includes digital down-converters implemented with cascaded integrator-comb (CIC) filters. CIC filters are very-high-performance filters using only adds and delays. The FPGA implements four digital down converters, which allows one, two, or four separate RX channels. At the RX path, we have four ADCs and four DDCs. Each DDC has two inputs, I and Q. Each of the four ADCs can be routed to either of I or the Q input of any of the four DDCs. This allows for having multiple channels selected out of the same ADC sample stream.

The DDC basically does two things. First, it down-converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers’ computing capability.

The DDCs in USRP2 are very similar to the USRP1 with a four-stage CIC but with two HBF filters. The CIC can decimate in the range [1...128]. The high rate HBF has seven taps and it decimates by two. The low-rate HBF has 31 taps and also decimates by two.

D.3.4.2 Digital Up-Converter

The digital up-converter interpolates the signal, up-converts, it to the IF band, and finally sends it through the DAC. There is a four-stage CIC and two HBF filters each working in interpolation mode.

D.3.4.3 More Information

For more information on these converters, see the following pages:

- <http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ>
- <http://gnuradio.org/redmine/wiki/gnuradio/UsrpFAQDDC>

D.4 DIFFERENTIAL PHASE-SHIFT KEYING (DPSK)

Differential phase shift keying (DPSK) is a form of phase modulation that conveys data by changing the phase of the carrier wave. For example, a differentially encoded BPSK (DBPSK) system is capable of transmitting a binary “1” by adding 180° to the current phase, while a binary “0” can be transmitted by adding 0° to the current phase.

Since a physical channel is present between the transmitter and receiver within a communication system, this channel will often introduce an unknown phase shift to the PSK signal. Under these circumstances, the differential schemes can yield a better error rate than the ordinary schemes, which rely on precise phase information. Referring to both BPSK and QPSK modulation schemes, there is phase ambiguity whenever the constellation is rotated by some amount due to the presence of phase distortion within the communications channel through which the signal passes. This problem can be resolved by using the data to change rather than set the phase.

Reference

- [1] <http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ>.

Trigonometric Identities

$$\exp(\pm j\theta) = \cos(\theta) \pm j \sin(\theta)$$

$$\cos(\theta) = \frac{1}{2}[\exp(j\theta) + \exp(-j\theta)]$$

$$\sin(\theta) = \frac{1}{2j}[\exp(j\theta) - \exp(-j\theta)]$$

$$\sin^2(\theta) + \cos^2(\theta) = 1$$

$$\cos^2(\theta) - \sin^2(\theta) = \cos(2\theta)$$

$$\cos^2(\theta) = \frac{1}{2}[1 + \cos(2\theta)]$$

$$\sin^2(\theta) = \frac{1}{2}[1 - \cos(2\theta)]$$

$$2\sin(\theta)\cos(\theta) = \sin(2\theta)$$

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)$$

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$$

$$\tan(\alpha \pm \beta) = \frac{\tan(\alpha) \pm \tan(\beta)}{1 \mp \tan(\alpha)\tan(\beta)}$$

$$\sin(\alpha)\sin(\beta) = \frac{1}{2}[\cos(\alpha - \beta) - \cos(\alpha + \beta)]$$

$$\cos(\alpha)\cos(\beta) = \frac{1}{2}[\cos(\alpha - \beta) + \cos(\alpha + \beta)]$$

$$\sin(\alpha)\cos(\beta) = \frac{1}{2}[\sin(\alpha - \beta) + \sin(\alpha + \beta)]$$

About the Authors

Di Pu was born in Suzhou, China. She received her B.S. degree (with distinction) from Nanjing University of Science and Technology (NJUST), Nanjing, China, in 2007 and M.S. degree from Worcester Polytechnic Institute (WPI), Worcester, MA, in 2009. Since March 2008, she has been a member of the Wireless Innovation Laboratory, where she is conducting research into cognitive radio system implementations. Di Pu is a recipient of the 2007 Institute Fellowship for pursuing Ph.D. studies at WPI in electrical engineering.

Dr. Alexander M. Wyglinski is an associate professor of electrical and computer engineering at Worcester Polytechnic Institute (WPI) and director of the Wireless Innovation Laboratory (WI Lab). He received his Ph.D. degree from McGill University in 2005, his M.S.(Eng.) degree from Queens University at Kingston in 2000, and his B.Eng. degree from McGill University in 1999, all in electrical engineering. Prior to joining WPI, Dr. Wyglinski was at the Information and Telecommunication Technology Center (ITTC) of the University of Kansas from 2005 to 2007 as an assistant research professor.

Dr. Wyglinski is very actively involved in the wireless communications research community, especially in the fields of cognitive radio systems and dynamic spectrum access networks. He currently serves on the editorial boards of both *IEEE Communications* magazine and *IEEE Communications Surveys and Tutorials*; he was a tutorial co-chair for the 2008 *IEEE Symposia on New Frontiers in Dynamic Spectrum Access Networks* (IEEE DySPAN 2008); he is currently the student travel grants chair for the 2010 *IEEE Symposia on New Frontiers in Dynamic Spectrum Access Networks* (IEEE DySPAN 2010); he is the Communications Techniques and Technologies track co-chair of the 2009 *IEEE Military Communications Conference*; he was the Wireless Mobile Communications track chair of the 2008 *IEEE Military Communications Conference*; he has been a guest co-editor for the *IEEE Communications* magazine with respect to two feature topics on cognitive radio (May 2007, April 2008); he was a technical program committee co-chair of the *Second International Conference on Cognitive Radio Oriented Wireless Networks and Communications* (CrownCom 2007); he was a track chair for both the 64th and 66th *IEEE Vehicular Technology Conference* (VTC); and he is currently a technical program committee member on numerous IEEE and other international conferences in wireless communications and networks.

Dr. Wyglinski is a senior member of the IEEE, IEEE Communications Society, IEEE Signal Processing Society, IEEE Vehicular Technology Society, IEEE Women in Engineering, Eta Kappa Nu, and Sigma Xi. Dr. Wyglinski's current research interests are in the areas of wireless communications, wireless networks, cognitive radios, software-defined radios, transceiver optimization algorithms, dynamic spectrum access networks, spectrum sensing techniques, machine learning techniques for communication systems, signal processing techniques for digital communications, hybrid fiber-wireless networking, and both multihop and ad hoc networks.

Index

A

Accelerate the Simulink model, 166–167
Analog-to-digital conversion, 23
Aperiodic, 16
Applications of software-defined radio, 239
ASCII, 171
Autocorrelation function, 75
Autocovariance function, 76
Averaging sweeps, 214
AWGN channel, 117–118

B

Barker code, 168–169
Bayes’ rule, 58
Bit allocation, 187–188
Bit error rate (BER), 105
Bivariate normal, 71–72
Block interleaving, 135

C

Callback functions, 165
Carrier sense multiple access (CSMA), 230
Cascaded integrator-comb (CIC) filter, 44–46
Causal, 18
Central limit theorem (CLT), 71
Channel encoding, 92–93
Channel filter, 191
Cognitive radio, 207, 239
Collecting spectral data, 209–214
Collision avoidance (CA), 231
Commutator, 182
Conditional expectation, 62–64
Conditional probability, 57–58
Continuous-time, 15
Convolutional interleaving, 135
Correlator realization, 122–124, 159–162
Cumulative distribution function (CDF), 69

Cyclic autocorrelation function (CAF), 220
Cyclic prefix, 185–186
Cyclostationary feature detection, 219–222
Cyclostationary process, 219
Cyclostationary random process, 77

D

Decimation, 27–28
Decision region, 119
Decision rule, 116
Decision statistic, 218
Detection theory, 113
Differential phase shift keying (DPSK), 163–165
Differential quadrature phase-shift keying (DQPSK), 166
Digital down converter (DDC), 146
Digital modulation, 94
Digital transceiver, 1
Digital transmission, 89–91
Digital up converter (DUC), 145
Dirac comb, 25
Discrete-time, 15
Discrete Fourier transform (DFT), 182
Dispersive channel environment, 183–184
Duplex communication, 197
Dwell time, 213
Dynamic spectrum access (DSA), 207

E

Einstein-Wiener-Khinchin (EWK) relations, 79
Energy detection, 218–219
Energy subtraction, 158–159
Energy threshold, 218
Error bounding, 107–108
Euclidean distance, 97
Expected value/mean, 62
Eye diagram, 32

F

- False alarm/Type I error, 216
- FFT plot, 142
- Field programmable gate array (FPGA), 3
- Filtering, 39
- Finite impulse response (FIR), 44
- Fourier transform, 18–23
- Frame synchronization, 167–168
- Frequency domain equalization, 186–187
- Frequency offset, 138
- Frequency offset compensation, 138–140
- Function, 56–57

G

- Gaussian process, 77–78
- Gram-Schmidt orthogonalization, 110–112

H

- Half-duplex (HDX), 198
- Hypothesis testing, 105, 214–217

I

- IEEE 802.11, 140
- In-phase and quadrature, 98
- Independence, 59
- Infinite impulse response (IIR), 44
- Integrate-and-dump, 156
- Intercarrier interference (ICI), 191–192
- Interleaving, 134
- Interpolation, 29–30
- Intersymbol interference (ISI), 32
- Inverse discrete Fourier transform (IDFT), 182

K

- Kansas University Agile Radio (KUAR), 10

L

- Laplace transform, 39
- Linear time-invariant (LTI), 17
- Low-pass, high-pass, band-pass and band-stop, 40

M

- Machine learning, 239
- Mask parameter, 131–132

Matched filter, 194

- Matched filter realization, 120–122
- Maximum a posteriori (MAP) detector, 117
- Maximum likelihood decoder, 157–159
- Maximum likelihood (ML) detector, 117
- Medium access control (MAC), 230
- Memoryless, 18
- Missed detection/Type II error, 216
- Multicarrier modulation (MCM), 177
- Multicarrier transmission, 181–183

N

- Nyquist pulse, 34
- Nyquist pulse shaping theory, 32
- Nyquist sampling theorem, 26–27

O

- Observation vector, 153–154
- Orthogonal frequency division multiplexing (OFDM), 182–183
- Orthonormal basis functions, 108
- Oscillator, 138–139

P

- Partition, 54–56
- Parts per million (ppm), 139
- Peak frequency variation, 139
- Per tone equalization, 187
- Periodic, 16
- Phase shift keying (PSK), 99–104
- Poles and zeros, 41
- Power allocation, 188
- Power efficiency, 94–95
- Power of the detector, 217
- Power spectral density (PSD), 78
- Probability density function (PDF), 66
- Probability mass function (PMF), 61
- Probability of false alarm, 216–217
- Probability of missed detection, 216–217
- Pulse amplitude modulation (PAM), 95–97
- Pulse shaping, 30

Q

- Quadrature amplitude modulation (QAM), 98–99

R

- Raised cosine pulse, 37–38
- Random process, 72
- Random variable, 59–60
- Receiver design, 245
- Receiver operating characteristic (ROC), 217
- Receiver structure, 153
- Rectangular QAM, 178
- Rectangular window, 229
- Region of convergence (ROC), 41
- Relative frequency, 53
- Repetition coding, 132–133

S

- Sample space, 54
- Sampling, 23
- Sampling function, 24
- Satellite communication, 247
- SDRu Receiver block, 141
- SDRu Transmitter block, 141
- Set theory, 53–54
- Shannon’s channel coding theorem, 93–94
- Signal, 15
- Signal constellation, 179
- Signal waveform, 108
- Simplex communication, 197
- Sinc pulse, 34–37
- Single carrier transmission, 177–181
- Sliding window approach, 227
- Software-defined radio (SDR), 1
- Source encoding, 91
- SpeakEasy, 1–2
- Spectral coherence function (SOF), 221
- Spectral correlation function (SCF), 221
- Spectrum analyzer, 210
- Spectrum sensing, 207
- Square root raised cosine filter, 194–196
- Stationary process, 76
- Strict-sense stationary (SSS), 76–77
- Substitution law, 64
- Sweep resolution, 211
- Sweep time, 214
- System, 16

T

- Tapered window, 229
- Test-bed implementation, 245–246
- Time division duplexing (TDD), 198
- Total probability, 58
- Transmitter design, 244–245

U

- Uniform sampling, 24
- Universal Software Radio Peripheral (USRP), 8–9
- University Wireless Open Access Research Platform (WARP), 10

V

- Vector representation, 108–109
 - Vehicular communication network, 242
 - Venn diagram, 54
-
- Water filling principle, 189
 - Waveform synthesis, 153
 - Wide-sense stationary (WSS), 77
 - WiFi network, 141
 - Window size, 211
 - Wireless local area network (WLAN), 140–141

X

- XCVR2450 daughtercard, 139

Z

- Z transform, 39

