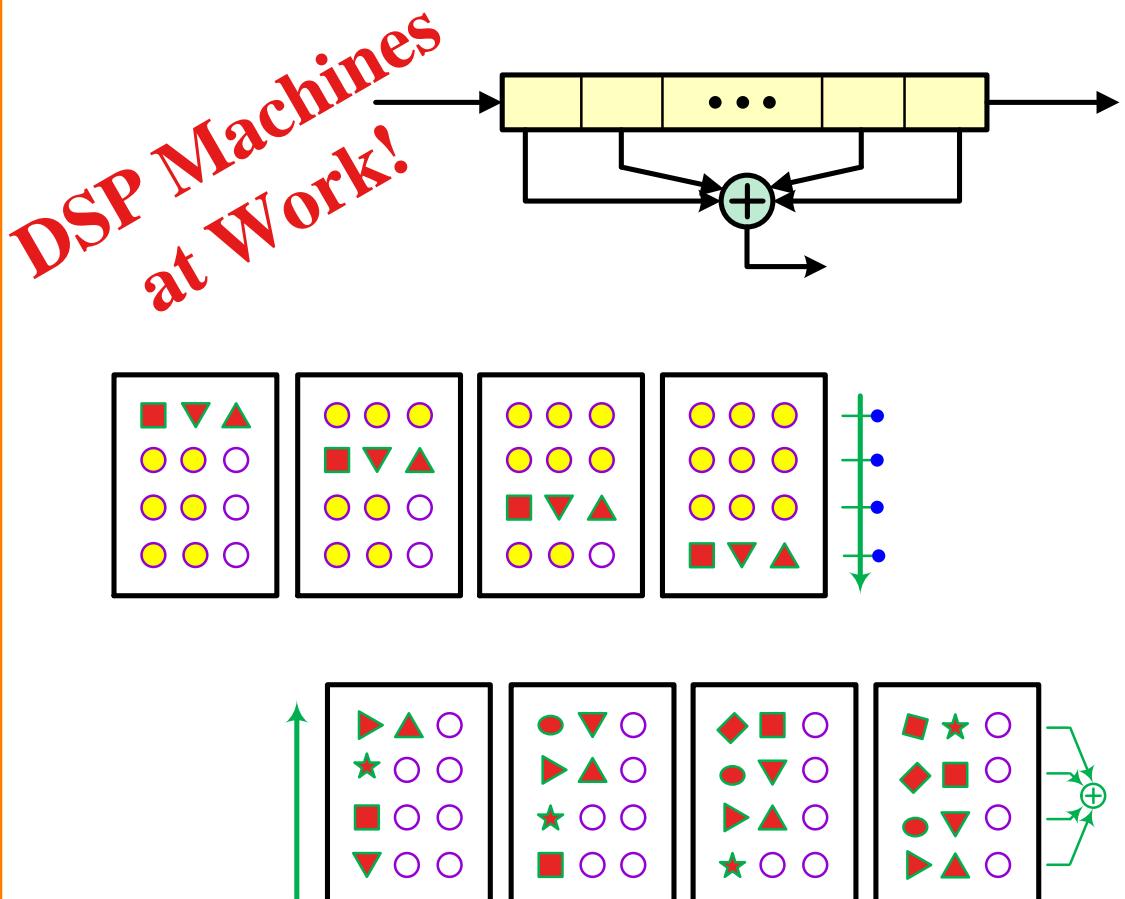


Wireless Communications from the Ground Up

An SDR Perspective



Qasim Chaudhari

Wireless Communications

From the Ground Up

An SDR Perspective

Qasim Chaudhari

Copyright © 2018 by Qasim Chaudhari

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher. While we believe that the information given in this work is correct, all parties must rely upon their own skill and judgement when making use of them. We do not assume any liability to anyone for any loss or damage caused by any error or omission in the work, whether such error or omission is the result of negligence or any other cause. Any and all such liability is disclaimed.

This book contains copyrighted material of Microchip Technology Incorporated replicated with permission. All rights reserved. No further replications may be made without Microchip Technology Inc.'s prior written consent.

For permission requests, contact info@wirelesspi.com.

Qasim Chaudhari
Melbourne, Australia
wirelesspi.com

In Loving Memory of My Mother

Rukhsana Jabeen

Contents

Preface	ix
1 Introduction to Signals	1
1.1 Basics of Signals	5
1.2 Transforming a Signal	11
1.3 Dealing with Complex Numbers	18
1.4 The Concept of Frequency	25
1.5 Sampling a Continuous-Time Signal	36
1.6 Correlation	40
1.7 Discrete Frequency	44
1.8 The Discrete Fourier Transform (DFT)	53
1.9 Effect of Time Shift in Frequency Domain	61
1.10 DFT Examples	67
1.10.1 A Rectangular Signal	67
1.10.2 A General Sinusoid	75
1.10.3 The Unit Impulse	78
1.10.4 The Sampling Sequence	79
1.11 A Digital Signal	82
1.12 The Small Picture	82
2 Introduction to Systems	83
2.1 Linear Systems	86
2.2 Time-Invariant Systems	88
2.3 System Characterization	89
2.4 Convolution	91
2.4.1 Regular Convolution	91
2.4.2 Complex Sinusoids and the Grand Scheme of Things	101
2.4.3 Circular Convolution	103
2.5 Finite Impulse Response (FIR) Filters	110
2.6 Some Useful FIR Filters	118
2.7 Sample Rate Conversion	123
2.7.1 Downsampling	124
2.7.2 Upsampling	131
2.8 Additive White Gaussian Noise (AWGN)	137
2.9 Appendix	141

3	Digital Communication with Linear Modulations	145
3.1	A Simple Communication System	147
3.2	Packing More Bits in One Symbol	149
3.3	Modulation: From Numbers to Signals	151
3.4	Demodulation: From Signals Back to Numbers	153
3.4.1	Matched Filter in Time Domain	156
3.4.2	Matched Filter in Frequency Domain	158
3.5	Pulse Amplitude Modulation (PAM)	163
3.6	Pulse Shaping Filter	171
3.7	Quadrature Amplitude Modulation (QAM)	201
3.8	Stethoscopes for a Communication System	217
3.9	Modulation Bandwidths	224
3.10	Computing Error Rates	225
3.11	Spectral Efficiency	233
4	Phase Locked Loop	237
4.1	What is Synchronization?	239
4.2	Basic Components of a PLL	243
4.3	Phase Error Detector	246
4.4	Proportional + Integrator (PI) Loop Filter	248
4.5	Numerically Controlled Oscillator (NCO)	249
4.6	Designing a PLL	252
4.7	Comments on Locking and Acquisition	265
5	Carrier Phase Synchronization	269
5.1	Effect of Carrier Phase Mismatch	272
5.2	The Fundamental Problem of Synchronization	278
5.3	The Big Picture	281
5.4	Enter the Correlation	283
5.5	Data-Aided/Decision-Directed Techniques	287
5.5.1	Feedforward: Conjugate Product Estimator	287
5.5.2	General Approach to Feedforward Synchronization	294
5.5.3	Feedback: Phase Difference Phase Error Detector	295
5.5.4	General Approach to Feedback Synchronization	306
5.5.5	Feedback: Cross Product Phase Error Detector	307
5.5.6	A Comparison of Phase Error Detectors	322
5.6	Non-Data-Aided Techniques	323
5.6.1	Feedforward: M-th Power Estimator	324
5.6.2	Feedback: M-th Power Phase Error Detector	326
5.6.3	The Costas Loop: A Classic Solution	328
5.7	Detection Aids	333
5.7.1	Carrier Lock Detectors	334
5.7.2	Resolving Phase Ambiguity	335
5.7.3	Cycle Slips, Hangup and the Role of AGC	339
5.8	The Small Picture	340
5.9	Appendix	340

6 Carrier Frequency Synchronization	347
6.1 Effect of Carrier Frequency Mismatch	349
6.2 The Big Picture	355
6.3 Timing-Aided Techniques	357
6.3.1 Enter the Correlation	357
6.3.2 Feedforward: Brute Force Estimator	360
6.3.3 Feedforward: The DFT Estimator	363
6.3.4 Feedforward: Multiple Correlations Estimators	367
6.3.5 A Frequency Locked Loop (FLL)	371
6.3.6 Feedback: Phase Based Frequency Error Detector	372
6.4 Non-Timing-Aided Techniques	379
6.4.1 Feedforward: Delay and Multiply Technique	380
6.4.2 Feedback: Derivative Frequency Error Detector	383
6.4.3 Feedback: Band Edge FLL	398
6.5 The Small Picture	411
6.6 Appendix	412
7 Timing or Clock Synchronization	417
7.1 Effect of Symbol Timing Mismatch	420
7.2 The Big Picture	430
7.3 Enter the Correlation	433
7.3.1 Feedforward: Brute Force Estimator	435
7.4 Why Squaring is Fundamental to Timing Synchronization	437
7.4.1 The Sinusoid Perspective	437
7.4.2 Role of Excess Bandwidth	448
7.4.3 Clock Recovery in Analog Modems	450
7.4.4 Feedforward: Digital Filter and Square Estimator	452
7.5 A Timing Locked Loop (TLL)	455
7.6 Symbol Centric Timing Error Detectors	458
7.6.1 Feedback: Derivative Timing Error Detector	459
7.6.2 Feedback: Early-Late Timing Error Detector	471
7.6.3 Why Symbol Centric?	477
7.7 Feedback: Zero Crossing (Gardner) Timing Error Detector	479
7.8 Interpolation	490
7.8.1 Piecewise Polynomial Interpolation	491
7.8.2 Farrow Structure	498
7.9 Interpolator Control	501
7.9.1 Modulo-1 Counter	502
7.9.2 Computing Basepoint Indices and Fractional Intervals	505
7.10 Sampling Clock Offset	506
7.11 Feedback: Mueller and Muller (M&M) Timing Error Detector	509
7.12 Feedback: Band Edge Timing Error Detector	516
7.13 Polyphase Clock Synchronization	528
7.14 The Small Picture	548
7.15 Appendix	550

8 Wireless Channel and Equalization	557
8.1 A Wireless Channel	560
8.1.1 Multipath Distortion Assuming No Carrier Wave	561
8.1.2 What Happens When a Carrier Wave Enters the Picture	565
8.1.3 Frequency Flat and Frequency Selective Fading	579
8.1.4 Doppler Shift: A Deceptive Villain	584
8.1.5 What Happens When Motion Enters the Picture	592
8.1.6 Time Flat (Slow) and Time Selective (Fast) Fading	596
8.1.7 From Channel Paths to Channel Taps	600
8.1.8 Dealing with a Time-Varying Channel	607
8.1.9 Effect of Channel on Rx Constellation	609
8.2 Channel Estimation	611
8.3 Equalization	613
8.3.1 The Big Picture	614
8.3.2 Brute Force Equalization: ML Sequence Estimation	618
8.3.3 Linear Equalizers	621
8.3.4 Least Mean Square (LMS) Equalizer	633
8.3.5 Decision Feedback Equalization	643
8.3.6 Blind Equalization: Constant Modulus Algorithm (CMA)	649
8.3.7 From Time to Frequency Domain Equalization	652
8.3.8 Single-Carrier Frequency Domain Equalization	654
9 Orthogonal Frequency Division Multiplexing (OFDM)	671
9.1 The Big Picture	674
9.2 How OFDM Works	677
9.2.1 From the Printer Port	677
9.2.2 To a Sliced Bread	694
9.3 An OFDM Transceiver	702
9.4 Timing Synchronization	710
9.5 Carrier Frequency Synchronization	730
9.6 Sampling Clock Synchronization	737
9.7 Channel Estimation	743
9.8 The Small Picture	749
10 The Chapter with No Title	751
10.1 Tx Architecture	755
10.1.1 Polyphase Filterbank Implementation	756
10.2 Rx Architectures	768
10.2.1 Tuned Radio Frequency (TRF) Rx	769
10.2.2 Heterodyne Rx	769
10.2.3 Zero-IF (Direct Conversion) Rx	776
10.2.4 Low-IF and Digital-IF Rx	781
10.2.5 Polyphase Filterbank Implementation	785
10.3 The Origins of Magic	807
10.4 The Small Picture	810
One Page Summary for Rx Algorithms	811
Bibliography	811

Preface

"As to methods there may be a million and then some, but principles are few. The man who grasps principles can successfully select his own methods. The man who tries methods, ignoring principles, is sure to have trouble."

Emerson

The Purpose

There are three different angles from which this book contributes to the understanding of wireless communication systems from the perspective of a Software Defined Radio (SDR).

1. *Visualization:* In my opinion, any language, including that of mathematics, is an unnatural mode of communication. For example, I can write the words darwaza, porte, puerta, umnyango, ovi and only certain people will understand what I mean. However, if I show you an image of a door, almost every single person on the planet will immediately get the concept. A figure imprints a massive amount of parallel information in our brains that is much easier to process and recall later. Since a human mind handles images very well, I try to visualize equations through beautiful figures which you will encounter throughout the text with logical and intuitive explanations.



2. *Mathematics:* If you are not a pure wireless communications academic, you would have found that the mainstream textbooks on this topic are filled with heavy mathematical details which makes this field an exclusive membership club for those who can understand several types of frequency variables and their corresponding Fourier transforms, probability and random processes and detection and estimation theories. While this is true for becoming a master, the Software Defined Radio (SDR) revolution and subsequent projects like GNU Radio have made it possible for anyone to sit down and construct their own unique radio by writing code. Many even do not need to know most of the above mentioned topics. All they need to understand is why an algorithm does what it does so that they know how to write its code, or modify it in an even better way. For this purpose, I have only relied on *school level mathematics* to explain all the concepts. You will not find any e or j of complex numbers here, nor will you encounter any integrals, probability theory and detection or estimation theory. The only things to know are a sine, cosine and a summation as

well as a derivative (which I have occasionally used). I have also reduced the font size for some unavoidable derivations or moved them to their respective appendices where possible. The reader can skip the content with a smaller font if desired.

3. *Why*: The best books written on implementing digital communication systems using Digital Signal Processing (DSP) algorithms are by fred harris [1] and Michael Rice [2]. As often happens with the grandmasters, they walk on a trail without exactly clarifying it for others. After reading their books, I started to feel that fred harris has mainly focused on '*how*' of communication systems in an unprecedented detail while Michael Rice has mainly covered '*what*' of communication systems in his simple and beautiful style. In this process, there were many '*why*' generated in my mind for which I had to find satisfactory answers. This book is a collection of those simple answers.

4. An extra little bonus is a one page summary of the crux of Rx algorithms, clarifying the role of particular parameters in the signal waveform. Most of the algorithm design can be understood by just grasping the concepts on this one page.

A common theme in this text is that some concepts seem easier in time domain and some others are simpler in frequency domain, while their mathematical derivations reinforce the idea. It is fun to grasp a concept covering all three sides. Finally, I have focused only on some topics of a digital receiver and left other material out because it is better to explain a few concepts well than to cover everything. If you are looking for a comprehensive resource, see the wonderful book by Proakis [3].

Some Acknowledgements

I want to acknowledge my family first. Writing a long technical book could have been a boring task but my daughters provided me great company while reading their Lands of Stories, Harry Potters and Ella + Olivia beside me. If I have gone into extra details of some concepts here, it might be due to my little son who keeps asking me questions with a double why. Finally, no words of gratitude are enough for my wife who held everything together during all these years.

Coming to the technical side, my understanding of digital and wireless communication systems is mainly influenced by the guys at Renzym, namely Azam Mehboob, Yasir Javed and Wasif Latif. They always inspired me to investigate how things work in a world beyond research papers as they were turning ideas into products in the areas of wireless communications and embedded systems. Having come this far, Renzym is a big success in my eyes.

On the teaching side, I developed this style of explaining concepts by giving presentations in our SDR meetup known as Cyberspectrum Melbourne. The attendees have always been a diverse set of SDR enthusiasts who wanted to know how they could make something work rather than the details of academic derivations. I cannot recall more than a handful of my presentations being interesting (80/20 rule) but each time, one embarrassingly boring talk after another, these people would come to me to explicitly tell me how good the presentation was and how much they enjoyed it. Without any exaggeration, the few concepts I have explained well in this book are all

due to them. In particular, my special thanks go to Pamela O'shea who is the Force behind Cyberspectrum Melbourne as well as Graham Cottew, Russell Muetzelfeldt, Chris Chippett, Chris Moore and Ashley Brighthope.

There are two persons from whom I have asked questions on email. One is fred harris who has been generous with his time and explanations while I was decoding his writings. You can learn more about DSP by reading two pages of his book than complete texts by most others. I think this probably comes down to the excellent mental models he has developed which make his work look more like that of an artist than of a scientist. The second is Rick Lyons who on occasions had to type mathematical equations spanning many pages to answer my questions when I began this project (where can one find people like him?).

This book would not have been possible without my friend Iqbal Shahid. When I was stuck into the initial inertia, he asked me to focus on writing only and helped me by drawing the figures for the first few sections. Affan Syed and Momin Uppal are two other special friends who have consistently showed me interesting angles out of ordinary concepts. I also want to thank Erchin Serpedin, Kerry Hinton, Brian Krongold, Wayne Rowe and Bill Moran for their support at my former workplaces.

Almost everything in this book comes from many engineers and researchers who have investigated and disseminated their ideas and they deserve the credits for the good part. Since this book is written, read and checked by a single person, there is a chance of technical errors and typos that are entirely my responsibility. If you find any such mistake, thank you for kindly informing me at info@wirelesspi.com.

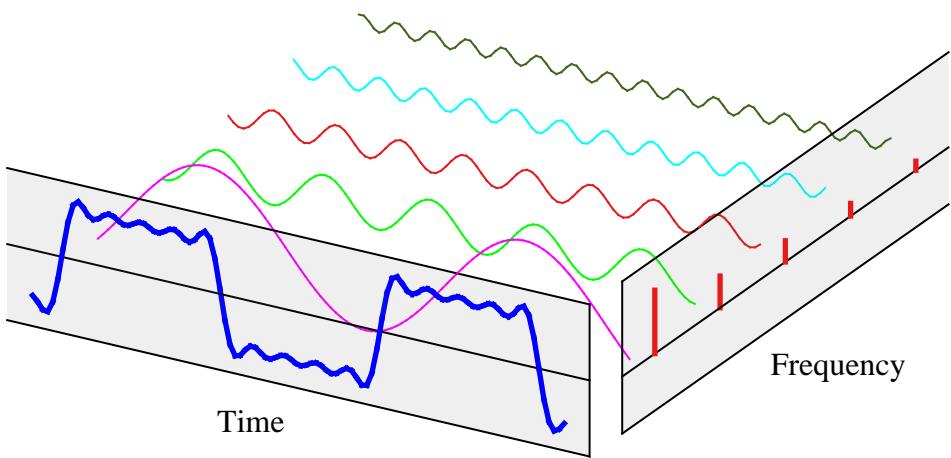
Finally, I personally feel indebted to Larry Page and Sergey Brin. The streamlined manner in which Google has connected our minds to on demand reservoirs of knowledge is truly remarkable.

When I started this book, I thought that it would be finished in *three months* (I read about Daniel Kahneman's planning fallacy later). After all, I knew 'exactly' what I was going to write and then allocated an extra two weeks for unforeseeable events. I like sunrise and had put my morning walks on hold to work on this manuscript. Time has come to start them again.

Qasim Chaudhari
Melbourne, Australia
Nov 2018

Chapter 1

Introduction to Signals



“Any sufficiently advanced technology is indistinguishable from magic.”

Arthur C. Clarke

One of the most memorable phrases in television history is the command from Captain Kirk to his chief engineer Scotty in the science fiction series Star Trek:

“Beam me up, Scotty!”

It refers to the futuristic teleportation machine, known as a *Transporter*, which disassembles a person or object into an energy pattern, then beam it to a target site where it is reassembled into the original matter.



Figure 1.1: The Star Trek crew

From Mechanics to Energy Beams

Imagine that such a device is invented at a point in the future. Until then, the knowledge of a person relating to transportation through automobiles is based on an internal combustion engine, pistons, crankshaft and so on. Suddenly we find ourselves in a completely different territory because designing and building the technology for transporting a person from location A to location B is now based around mass-energy equivalence in physics, according to which mass and energy are the same physical entity and can be converted into each other. Not only do we need to understand

$$E = mc^2$$

from Albert Einstein’s theory of special relativity but also come up with strategies to reproduce as closely as possible the said person or object from the incoming beam. This is a process that requires excellent mental models of the underlying mass → energy conversion phenomena, *not the mechanics of the engine*.

From Electronics to DSP

In not as dramatic but still a similar manner, wireless communications has profoundly transformed as a science in the previous years due to the digital revolution.

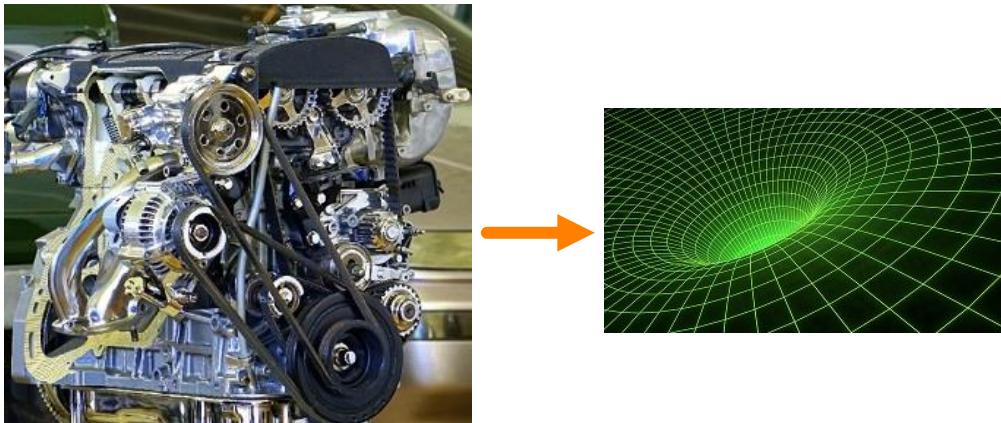


Figure 1.2: From mechanics of the engine to energy beams

- On the transmitter side, we have become able to send into air a sequence of numbers stored in a processor memory.
- On the receiver side, we have become able to acquire the momentary value of the incoming analog signal and convert it into its digital representation which is nothing but a sequence of numbers, known as a discrete-time sequence.

After the great advancement in computational power, most of the radio circuits performing analog signal processing (e.g., by using resistors, capacitors, inductors, op amps – basically physics and devices) have been replaced by powerful digital processors that can perform the necessary number crunching (basically algorithms run by computer programs) to unravel the original information at a better price-performance ratio. In a true software radio, this is a process that requires excellent mental models of the underlying discrete-time sequences, *not the electronics of the circuits*.

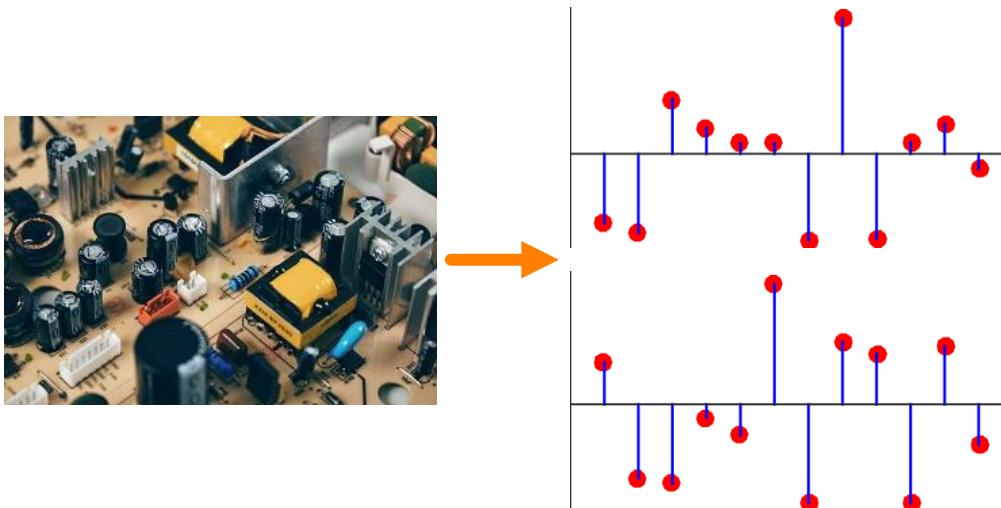


Figure 1.3: From electronics of the circuits to discrete-time sequences

Shown on the right side of Figure 1.3 is a discrete-time sequence that consists of a few random bits 0s and 1s mapped to signal levels $-1s$ and $+1s$, respectively. Why does it look more complicated than a stream of $-1s$ and $+1s$? Because in the real world, this signal has been distorted by the following impairments before arriving at the Rx processing machine.

- Carrier phase offset: 17°
- Timing phase offset: 31 % of a bit time
- Carrier frequency offset: 11% of the bit rate
- Multipath channel: A direct path and two reflected paths with relative amplitudes of 0.6 and 0.2 arriving after relative delays of 0.76 and 1.03, respectively

In this text, we will cover in detail the nature and solution for each and every imperfection described above. The I and Q parts shown here are the two components of a complex signal. I cover complex numbers in Section 1.3 as a starting point to build our Digital Signal Processing (DSP) toolkit. Also, I like to view the same IQ signal in a 3D figure shown in Figure 1.4 which offers an advantage over the separate two 2D figures, as we find out later.

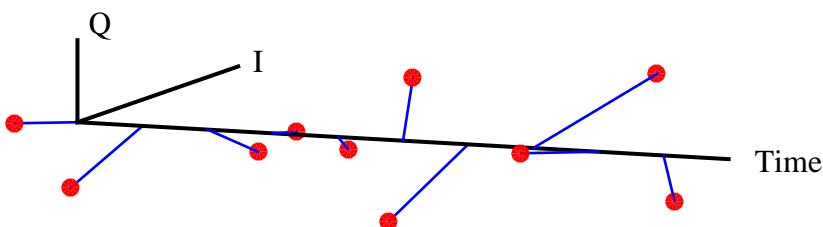


Figure 1.4: A 3D view of the discrete-time sequence of Figure 1.3

By just looking at Figure 1.4, it is hard to believe that after so much distortion, extracting the original stream of bits 0s and 1s from this complex sequence is even possible. Nevertheless, this amazing (ad)venture can be accomplished through the implementation of DSP algorithms, which is all a Software Defined Radio (SDR) is about. With this understanding in place, we start with an introduction to signals.

1.1 Basics of Signals

Throughout the history, humans have always been fascinated by the invisible, a prime example of which is starting fire out of thin air hundreds of thousands of years ago. An electromagnetic wave carrying energy and momentum as a result of an accelerating charged particle in one material and imparting them to another material far away is another example of this fascination. As a result, today we are living in an ocean of radio waves carrying wireless transmissions. The understanding of wireless communications through the electromagnetic waves begins with the concept of a signal.

A signal is any *measurable quantity* that varies with time or some other independent variable. If you record the words you speak in a computer memory and plot

against time, you get a speech signal. As an example, Figure 1.5 shows a signal for the words “Hello Wireless” recorded in a computer.

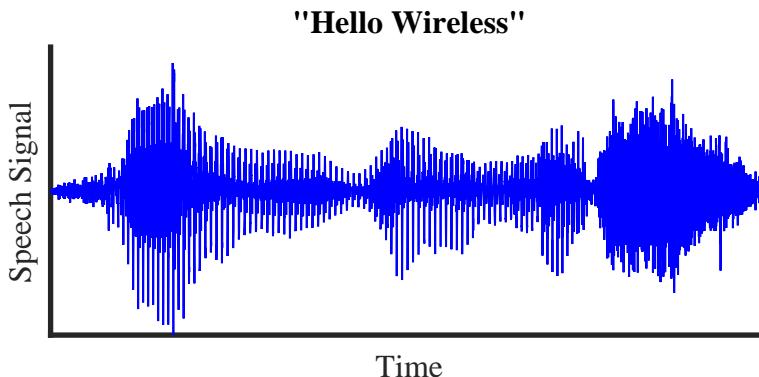


Figure 1.5: A speech signal recorded in a computer

Some familiar signals from everyday life are temperature or pressure readings, GPS signals that determine your vehicle’s position, biological signals like EEG and ECG that are electrical activity recordings of the brain and heart, respectively, and stock prices that reflect the expectations of corporate earnings.

Mathematically speaking, a signal is a function of an independent variable. For example, a function $s(t) = 3t^2$ can be considered as a signal varying quadratically with time. In this text, we will focus on signals comprising of one independent variable and one dependent variable. The independent variable on x-axis is usually time or frequency, while the value of the signal at each time instant or frequency bin is its amplitude.

Classification of continuous-time and discrete-time signals deals with the type of independent variable. If the signal amplitude is defined for every possible value of time, the signal is called a *continuous-time signal*. However, if the signal takes values at specific instances of time but not anywhere else, it is called a *discrete-time signal*. Basically, a discrete-time signal is just a sequence of numbers.

Example 1.1

Consider a football (soccer) player participating in a 20-match tournament. Suppose that his running speed is recorded at each instant of time in the 90-minute duration of a particular match and plotted against time. The result shown in Figure 1.6a is clearly a continuous-time signal.

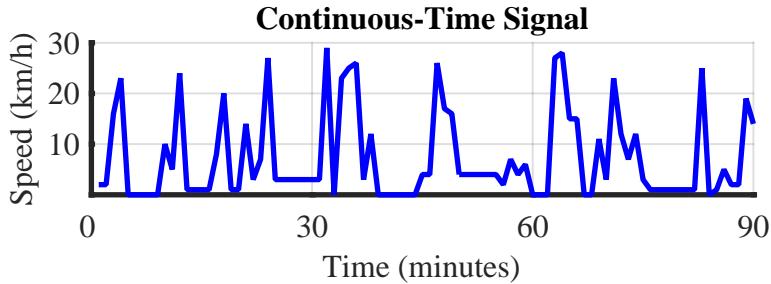
On the other hand, Figure 1.6b shows the number of goals he scored during those 20 matches, which is defined only for each match and not in between. Hence, it is a discrete-time signal.

Discrete-time signals usually arise in two ways:

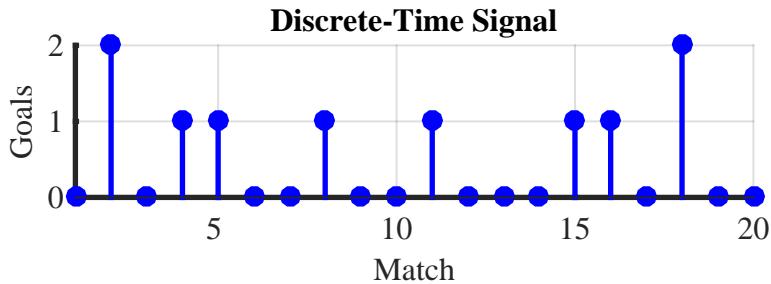
Inherently discrete-time: By recording the number of events over finite time periods.

For example, number of trees cut every year in a city for housing and development projects.

Conversion from continuous-time to discrete-time: By acquiring the values of a continuous-time signal at fixed time instants. This process is called sampling



(a) Player speed during a single match



(b) Player 1's goals in the tournament

Figure 1.6: Continuous-time and discrete-time signals

and is discussed in detail in Section 1.5. For example, the actual temperature outside varies continuously throughout the day, but the weather stations log the data after specific intervals, say every 30 minutes.

Mathematically, we represent a continuous-time signal as $s(t)$ and a discrete-time signal as $s[n]$, where t is a real number while n is an integer. For example, a discrete-time signal $s[n] = 3n^2$ can be plotted by finding $s[n]$ for various values of n . Each member $s[n]$ of a discrete-time signal is called a *sample*.

$$\begin{aligned}
 n = -5 &\rightarrow s[n] = 75 \\
 n = -4 &\rightarrow s[n] = 48 \\
 n = -3 &\rightarrow s[n] = 27 \\
 n = -2 &\rightarrow s[n] = 12 \\
 n = -1 &\rightarrow s[n] = 3 \\
 n = 0 &\rightarrow s[n] = 0 \\
 n = +1 &\rightarrow s[n] = 3 \\
 n = +2 &\rightarrow s[n] = 12 \\
 n = +3 &\rightarrow s[n] = 27 \\
 n = +4 &\rightarrow s[n] = 48 \\
 n = +5 &\rightarrow s[n] = 75
 \end{aligned}$$

Plotting each value of $s[n]$ against every n is then straightforward and has been drawn in Figure 1.7.

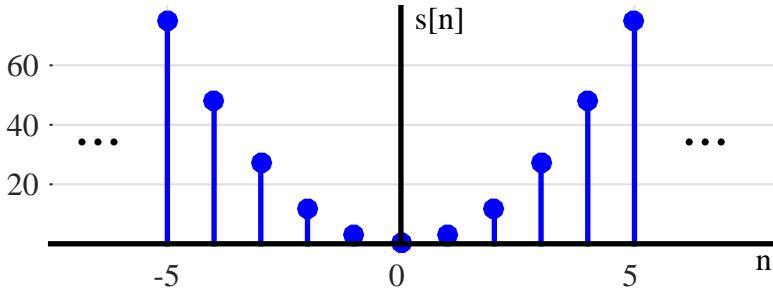


Figure 1.7: A discrete-time quadratic signal $s[n] = 3n^2$

Another way of representing a discrete-time signal is in the form of a sequence with an underline indicating the time origin ($n = 0$), such as

$$s[n] = \{\dots, 75, 48, 27, 12, 3, \underline{0}, 3, 12, 27, 48, 75, \dots\}.$$

Finally, it is incorrect to assume that a discrete-time signal is zero between two values of n . It is simply *not defined* for non-integer values.

Note 1.1 Voltage as a signal

In electrical engineering, the time-varying quantity is usually voltage (or sometimes current). Therefore, when we work with a signal, just think of it as a voltage changing over time. After we convert this signal from continuous-time to discrete-time, it becomes a sequence of numbers.

Unit Impulse and Unit Step Signals

Some basic signals play an important role in discrete-time signal processing. We discuss two of them below.

1. A *unit impulse* is a signal that is 0 everywhere, except at $n = 0$, where its value is 1. Mathematically, it is denoted as $\delta[n]$ and defined as

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases} \quad (1.1)$$

The unit impulse signal is shown in Figure 1.8.

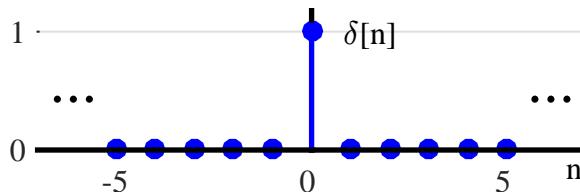


Figure 1.8: Unit impulse signal

2. A **unit step** signal is 0 for past ($n < 0$) values, while 1 for present ($n = 0$) and future ($n > 0$) values. To be precise, it is denoted as $u[n]$ and defined as

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \quad (1.2)$$

The unit step signal is shown in Figure 1.9.

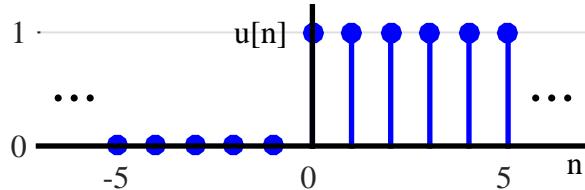


Figure 1.9: Unit step signal

Energy of a Signal

Plotting a discrete-time signal or writing it as a sequence of numbers seems good enough, but then how do we make a comparison between two signals? For example, in Figure 1.6b, we need to know how this player stands against other players participating in the tournament. It therefore seems that we should devise some measure of “strength” or “size” of a signal.

One simple method can be just adding all the values on the amplitude axis, which is our target dependent variable. So from Figure 1.6b, our player has a total number of 10 goals in the tournament. Now we can easily compare his performance with others, *provided that the signal always stays positive*.

Imagine another footballer who is so “energetic” that he just cannot resist possessing the ball during the game and sometimes even scores own goals, as shown in Figure 1.10. From his team’s point of view, his net total by adding all the goals is just 7. However, from an energy perspective in the field, he is much different than player 1 in Figure 1.6b.

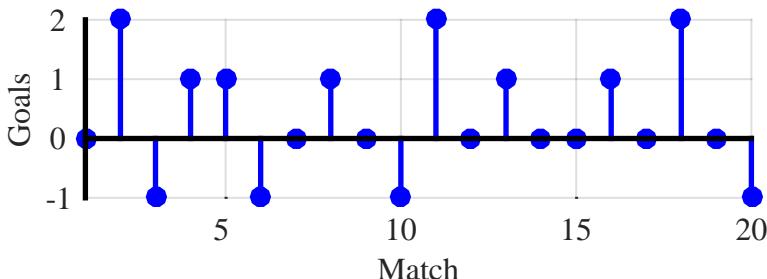


Figure 1.10: Player 2’s goals in the tournament

Therefore, a simple addition does not work for signals that assume both positive and negative values, such as a voltage varying with time because addition of both positive and negative values cancels and diminishes the calculated signal strength.

This suggests that strength of a signal can be measured by taking the absolute value of the signal and then adding all the values. Or square of the absolute value, or the fourth power of the absolute value, and so on. Due to its mathematical tractability (an exciting must-have for communication theorists), square of the absolute value is the preferred choice.

In light of the above, the energy of a discrete-time signal is defined as

$$E_s = \cdots + |s[-2]|^2 + |s[-1]|^2 + |s[0]|^2 + |s[1]|^2 + |s[2]|^2 + \cdots \\ = \sum_n |s[n]|^2 \quad (1.3)$$

where the term \sum_n denotes summation over all values of n .

Example 1.2

As an example, signal energy in Figure 1.6b is

$$E_{P1} = 2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 2^2 = 14$$

and in Figure 1.10, it is

$$E_{P2} = 2^2 + (-1)^2 + 1^2 + 1^2 + (-1)^2 + 1^2 + \dots \\ - 1^2 + 2^2 + 1^2 + 1^2 + 2^2 + -1^2 = 21$$

Using such a definition, the difference between the energy levels of the two players is truly reflected.

Similarly, energy in the signal of Figure 1.7 is infinite as the signal values extend from $-\infty$ to $+\infty$.

Even and Odd Signals

Some signals have specific properties which make analysis and computations simpler for us later. For example, a signal is called *even* (or symmetric) if

$$s[-n] = s[n] \quad \text{or} \quad s(-t) = s(t) \quad (1.4)$$

Flipping an even signal around amplitude axis results in the same signal. An example of an even signal is $s(t) = \cos(2\pi f_0 t)$ shown in Figure 1.11a. The quadratic signal $s[n] = n^2$ illustrated in Figure 1.7 is also an even signal.

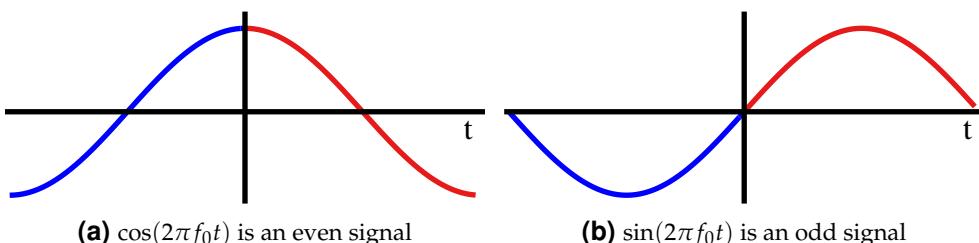


Figure 1.11: Examples of even and odd signals

On the other hand, a signal is called *odd* (or antisymmetric) if

$$s[-n] = -s[n] \quad \text{or} \quad s(-t) = -s(t) \quad (1.5)$$

An odd signal has symmetry around the origin. An example of an odd signal is $s(t) = \sin(2\pi f_0 t)$ drawn in Figure 1.11b, as well as $s[n] = -n$.

Periodic and Aperiodic Signals

A signal is *periodic* if it repeats itself after a certain period N .

$$s[n \pm N] = s[n] \quad \text{for all } n \quad (1.6)$$

Both $s[n] = \cos[2\pi f_0 n]$ and $s[n] = \sin[2\pi f_0 n]$ are examples of periodic signals if f_0 is a rational number.

If a signal does not repeat itself forever, there is no value of N that satisfies the above periodicity equation. Such a signal is known as *aperiodic*, an example of which is a unit step signal $u[n]$ and most other signals encountered in this text.

An example each of a periodic and an aperiodic signal is shown in Figure 1.12.

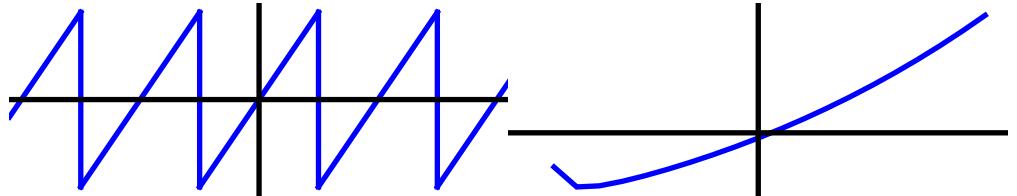


Figure 1.12: Examples of periodic and aperiodic signals

1.2 Transforming a Signal

Transforming a discrete-time signal – whether in time or amplitude – is certainly possible, and often in interesting ways. In practice, scaling and time shifting are the two most important signal modifications encountered. Scaling changes the values of dependent variable on amplitude-axis while time shifting affects the values of independent variable on time-axis.

Below we describe addition and multiplication of two signals as well as scaling and time shifting a signal in detail.

Addition

For addition of two discrete-time signals, say $x[n]$ and $y[n]$, add the two signals sample-by-sample: $z[n] = x[n] + y[n]$ for every n , e.g.,

$$\begin{aligned} z[0] &= x[0] + y[0] \\ z[1] &= x[1] + y[1] \\ &\dots \end{aligned}$$

and so on.

Multiplication

For multiplication, multiply the two signals sample-by-sample, as we did with addition: $x[n] \cdot y[n]$ for every n , e.g.,

$$z[0] = x[0] \cdot y[0]$$

$$z[1] = x[1] \cdot y[1]$$

.....

and so on.

Scaling

Scaling implies multiplying the signal amplitude by a constant number α resulting in $\alpha \cdot s[n]$, where α can be positive, negative, greater than or lower than 1. For example, Figure 1.13 shows a signal $s[n]$ scaled by 4 (compare the amplitudes of both signals on y-axis). One word of caution: in scaling, the whole signal gets scaled by the same value and hence scaling is different than sample-by-sample multiplication of two signals.

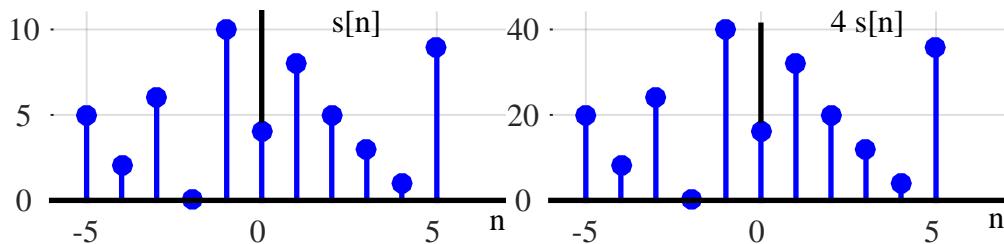


Figure 1.13: Scaling a signal $s[n]$ by 4

Time Shifting

A signal $s[n]$ can be shifted to the right or left by any amount m . There are two ways to look at time shifting a signal.

Conventional Method - Shifting the Signal

Shifting a discrete-time signal is usually described as follows.

Delay: For $s[n - m]$, a time shift results in a delay of $s[n]$ by m units of time. So plot $s[n - 2]$ by shifting $s[n]$ two units to the right (Figure 1.14a).

Advance: For $s[n + m]$, a time shift results in an advance of $s[n]$ by m units of time. So plot $s[n + 2]$ by shifting $s[n]$ two units to the left (Figure 1.14a).

This is simple to remember: Write the time shift as $n - m$. Shift right for $s[n - m]$ and shift left for $s[n + m]$.

There are two problems with the approach above.

1. It does not explain why we are doing what we are doing.

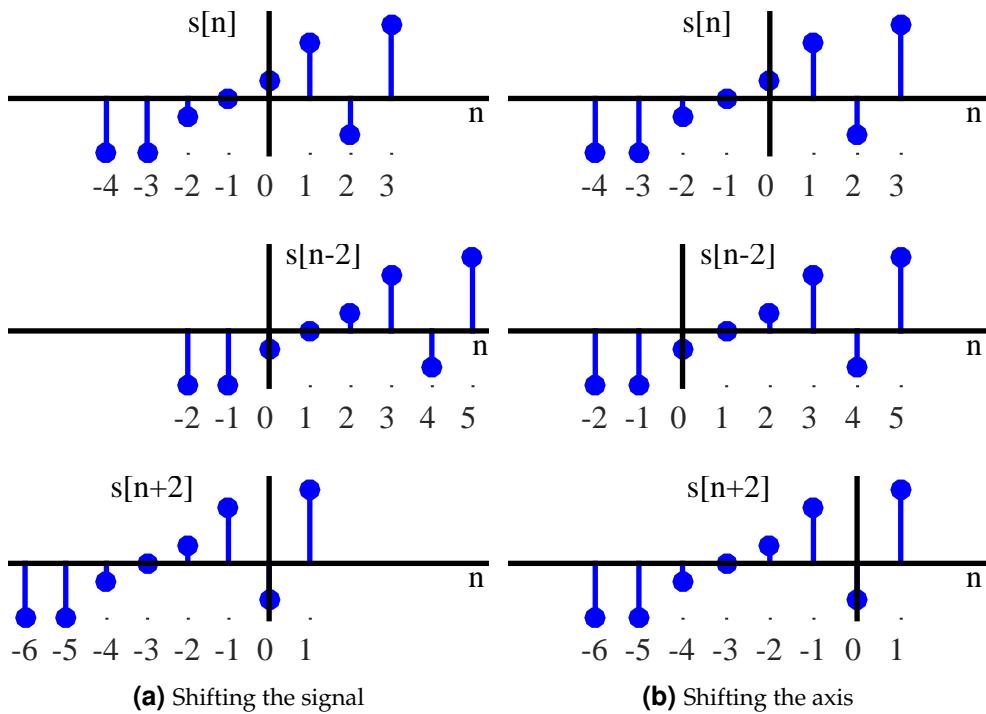


Figure 1.14: Two ways to look at time shifting a signal

2. A confusion might arise in the following way. The past samples of the sequence are to the left of the origin while the future samples are to the right. When we advance a signal by shifting left, a person might think that the future samples are on the left side. A similar argument holds for delaying a signal.

Intuitive Method - Shifting the Axis

This method is based on the earliest notions of time kept by humans: past, present and future. Replacing the word present with the NOW, consider the following quote:

“Realize deeply that the present moment is all you have. Make the NOW the primary focus of your life.”

Eckhart Tolle - The Power of Now

In signal processing applications, NOW is the time index 0 and we want to make it the primary focus of a signal's life, as shown in Figure 1.15. Looking from this perspective,

Delay: For $s[n - m]$, $n - m$ clearly implies traveling m units in the past. So plot $s[n - 2]$ by going 2 units back in time and making it the new NOW, i.e., time index 0 (Figure 1.14b).

Advance: For $s[n+m]$, $n+m$ obviously implies a future travel of m units. So plot $s[n+2]$ by going 2 units towards the future and making it the new NOW, i.e., time index 0 (Figure 1.14b).

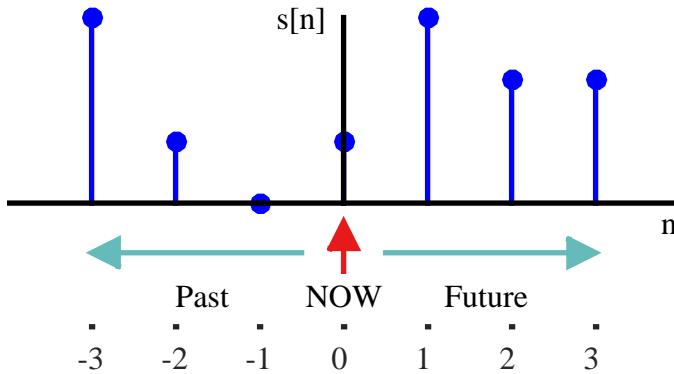


Figure 1.15: Past, NOW and future of a signal

Stated in another way, just look at the time-axis itself. Keep the discrete-time signal as it is, but move 0 time index m units to the left for $n - m$ and m units to the right for $n + m$, as shown in Figure 1.14b. Notice that index 0 of $s[n + 2]$ is at the same sample as index 2 of $s[n]$, which indicates accessing a future value. Although the difference is only how to look at the process, this is a simpler and more intuitive approach. This will be very helpful when we will discuss the topic of convolution in Section 2.4.

Note that as far as drawing time shifted signals is concerned, we will mostly use the first approach because it saves page space by aligning the time-axis indices right on top of one another.

Note 1.2 Addition of time shifted signals

Understanding time shifting a signal makes it easy to analyze seemingly complicated equations such as

$$r[n] = \sum_{m=-1}^2 s[n-m]$$

which is basically addition of $s[n-m]$ for $m = -1, 0, 1$ and 2 . We can simply break it down to a more recognizable form by substituting values of m as

$$r[n] = s[n+1] + s[n-0] + s[n-1] + s[n-2]$$

Now we can easily draw the time shifted versions of $s[n]$ and add them together to find the resultant signal. An example with a unit impulse signal $\delta[n]$ and its time shifted versions is shown in Figure 1.16.

Flipping or Time-Reversal

When the independent variable n is replaced by $-n$, the signal is *reflected* or *flipped* around the time origin $n = 0$ because the sample at $n = +1$ shifts to $n = -1$, sample at $n = -5$ shifts to $n = +5$, and so on. Figure 1.17 illustrates this concept by plotting $s[-n]$ and $s[-n+3]$.

Note that the rules of shifting to the left or right also become reverse due to the flipped time-axis, $-n$, itself. For example, from the perspective of intuitive method of shifting a signal, an axis of $-n$ instead of n suggests that the past becomes the future,

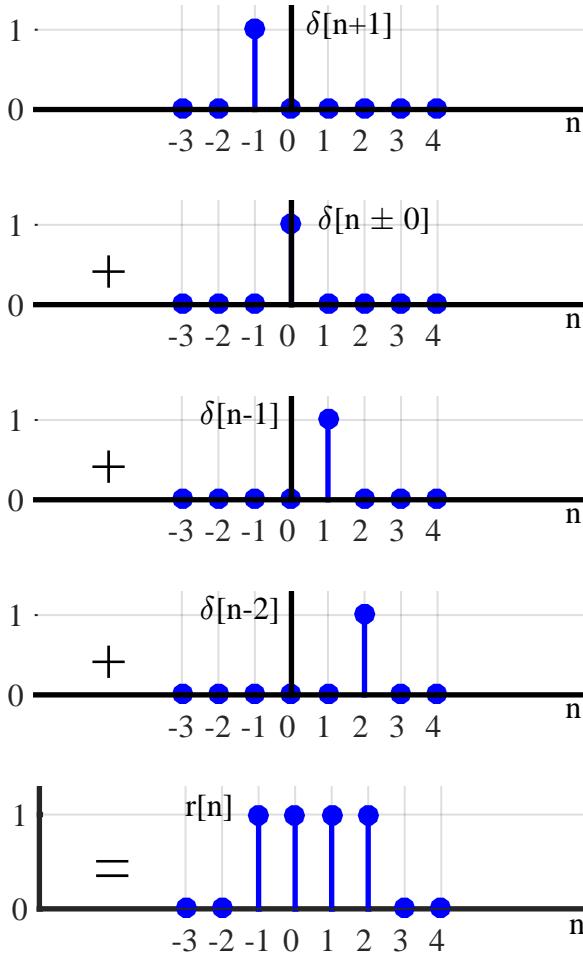


Figure 1.16: Computing $r[n] = \sum_{m=-1}^2 s[n-m]$ for $s[n] = \delta[n]$

and vice versa. Hence, $s[-n+3]$ first flips the signal around time origin, then travels 3 units to the *past* to label it the new NOW (time index 0).

Circular Shift

Circular shift is very similar to time shifting of a signal, except that we only focus on a segment of N samples for a circular shift whereas the available axis for a regular time shift is from $-\infty$ to $+\infty$.

If a signal $s[n]$ is circularly right shifted by m , the samples of the signal $s[n]$ that *fall off* to the right of length- N segment reappear at the start. Similarly, if $s[n]$ is circularly left shifted by m , the samples of the signal $s[n]$ that *fall off* to the left of length- N segment reappear from the end. Just like a video game character which disappears at one end of the screen to emerge from the other. An example for Pacman circularly shifting to the right is shown in Figure 1.18.

A simple rule to remember circular shift is “*what goes around comes around*”. There are two methods to implement a circular shift as follows.

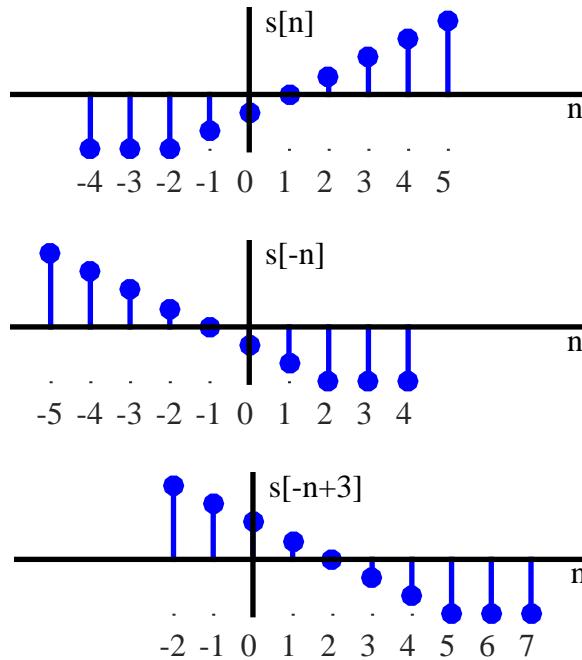


Figure 1.17: Flipping and time shifting a signal

Method 1

As circular shift is done with respect to a length- N segment, shifts are computed modulo N and denoted as $s[(n - m) \bmod N]$, where $(n - m) \bmod N$ means shifting the

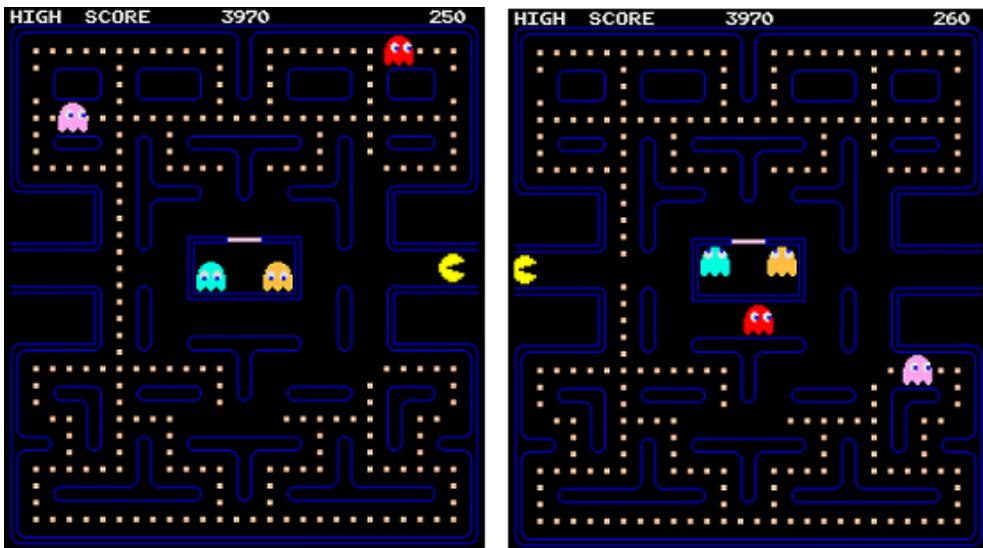


Figure 1.18: Circular right shift of Pacman on the game axis

signal as we normally would, but bringing them back in the range $0 \leq n \leq N - 1$ from the opposite side. For example, Figure 1.19 shows two examples of circular shifts where I want you to ignore the green dashed samples on both sides and consider only the indices n from 0 to 7.

- $s[(n - 1) \bmod 8]$ is a right shift of 1 for each sample, except the one at time index 7. Instead of going to index 8, it returns from the left at index 0.
- Again, $s[(n + 3) \bmod 8]$ is a left shift of 3 for most samples, except for samples at time indices 0, 1 and 2. Instead of going to negative time, these three samples return from the right at indices 5, 6 and 7, respectively.

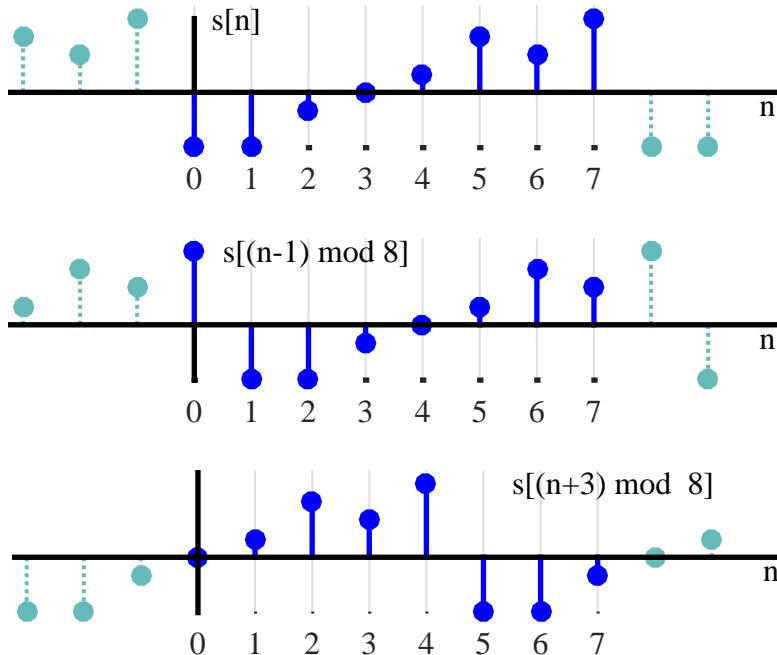


Figure 1.19: Circularly shifting a signal $s[(n - m) \bmod 8]$

As an exercise, try drawing $s[(n + 7) \bmod 8]$ and $s[(n - 5) \bmod 8]$. You will find that these shifts are actually the same as $s[(n - 1) \bmod 8]$ and $s[(n + 3) \bmod 8]$, respectively.

Method 2

Consider the green dashed samples in Figure 1.19. First, note that the signal $s[n]$ is repeated on both sides to make it periodic due to which samples at original indices $n = 5, 6$ and 7 appear before index $n = 0$ whereas samples at original indices $n = 0$ and 1 appear after $n = 7$. With this setting in place, a circular shift is just like a regular shift. Apply a regular right shift of 1, i.e., $s[n - 1]$, to the above signal and see that the result is $s[(n - 1) \bmod 8]$ shown in the middle of Figure 1.19. Similarly, apply a regular left shift of 3, i.e., $s[n + 3]$, to the above signal and check that the result is $s[(n + 3) \bmod 8]$ shown at the bottom of Figure 1.19.

Circular Flipping

An interesting question at this stage is: *what will be the result of a circularly flipped signal $s[(-n) \bmod N]$?*

The logic of circular shift stays the same. Like regular flipping, the sample at index 0 remains fixed. By method 1 above and ignoring the green dashed samples in Figure 1.19, the sample at index 7 should take a position at -7 but there is no negative indexing in circular notation and $(-7) \bmod 8 = -7 + 8 = 1$, which takes it to index 1. Figure 1.20 shows $s[(-n) \bmod N] = s[-n + N]$ for the same signal as above.

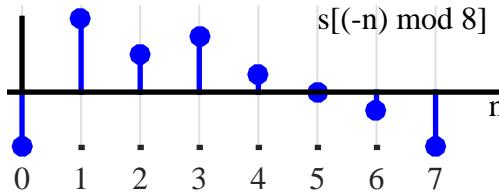


Figure 1.20: Circular flipping

By method 2 and taking into account the green dashed samples in Figure 1.19, a regular flip implies that the three samples to the left of index 0 are now present at the right of index 0 in Figure 1.20. The significance of circular shift will become clear when we discuss the Discrete Fourier Transform (DFT) later.

1.3 Dealing with Complex Numbers

Although complex notation is not complex to understand (see Chapter 8 in Ref. [4] for an excellent tutorial), one of the themes of this text is to avoid complex notation altogether.

A complex number V is defined as an ordered pair of real numbers in (x, y) -plane. In that respect, complex numbers can be considered as vectors with initial point on the origin $(0, 0)$. Addition of complex numbers is then similar to the addition of vectors in (x, y) -plane from this perspective.

However, multiplication is well defined for complex numbers while it is not defined for vectors – the dot product of two vectors is a scalar, not a vector, while the cross product of two vectors in a plane is a vector that is outside of that plane. The product of complex numbers, on the other hand, is a complex number – an extremely useful property.

For our purpose, using the complex numbers but still staying clear of the complex notation means that we focus on a 2-dimensional plane with x or real-axis named as I (which stands for *inphase*) and y or imaginary-axis named as Q (which stands for *quadrature*), as shown in Figure 1.21. In Section 1.4, we will learn why the x and y components are called inphase and quadrature, respectively.

V is a complex number in this IQ -plane with V_I and V_Q as its I and Q components, respectively. From Figure 1.21,

$$\begin{aligned} V_I &= |V| \cos \angle V \\ V_Q &= |V| \sin \angle V \end{aligned} \tag{1.7}$$

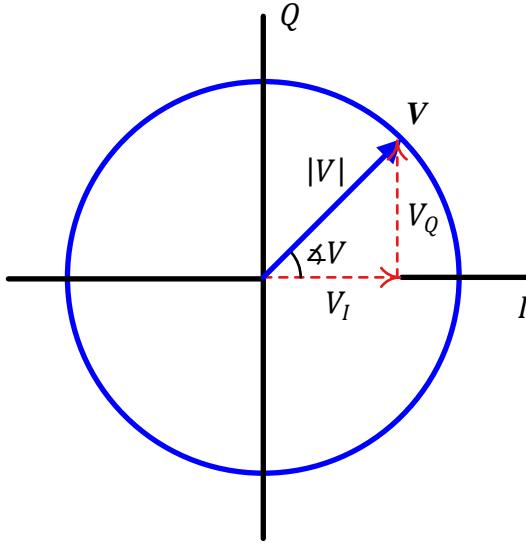


Figure 1.21: A complex number V in IQ -plane

where $|V|$ and $\angle V$ are the magnitude and angle of V with respect to I -axis, respectively.

Magnitude and Phase

In polar representation of complex numbers, the magnitude of V in an IQ -plane is defined from Eq (1.7) as

$$|V| = \sqrt{V_I^2 + V_Q^2} \quad (1.8)$$

where we have used the property $\cos^2 \theta + \sin^2 \theta = 1$. Defining the phase $\angle V$ is a little trickier. It is tempting to define it as $\tan^{-1} V_Q / V_I$. However, a problem with this is that

$$\begin{aligned} \tan^{-1} \frac{+V_Q}{+V_I} &= \tan^{-1} \frac{-V_Q}{-V_I} \quad \rightarrow \quad \text{in } [0, +\pi/2], \text{ quadrant I} \\ \tan^{-1} \frac{-V_Q}{+V_I} &= \tan^{-1} \frac{+V_Q}{-V_I} \quad \rightarrow \quad \text{in } [0, -\pi/2], \text{ quadrant IV} \end{aligned}$$

There is no way to differentiate whether V lies in quadrant I or III. The same holds for V lying in quadrant II or IV. On the other hand, Figure 1.21 tells us that V can lie in any quadrant and its phase should be in the range $[-\pi, \pi)$ as explained below.

- Quadrant I: $\angle V$ should be in $[0, \pi/2]$. $\tan^{-1} V_Q / V_I$ is good enough.
- Quadrant II: $\angle V$ should be in $[\pi/2, \pi]$. $\tan^{-1} V_Q / V_I$ is in quadrant IV and an adjustment by π is needed.
- Quadrant III: $\angle V$ should be in $[-\pi/2, -\pi]$. $\tan^{-1} V_Q / V_I$ is in quadrant I and an adjustment by π is needed.

- Quadrant IV: $\angle V$ should be in $[0, -\pi/2]$. $\tan^{-1} V_Q/V_I$ is good enough.

Similarly, from Figure 1.22,

- Case 1: When $V_I < 0$ and $V_Q = 0$, the phase of V should be π , not 0
- Case 2: When $V_I = 0$ and $V_Q > 0$, the phase of V should be $+\pi/2$
- Case 3: When $V_I = 0$ and $V_Q < 0$, the phase of V should be $-\pi/2$

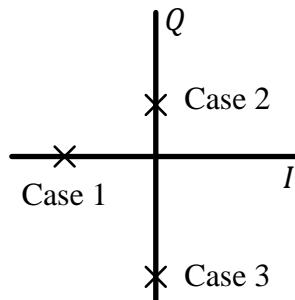


Figure 1.22: A few more cases $\tan^{-1}(\cdot)$ cannot handle

Taking into account all four quadrants, $\angle V$ is defined in terms of $\tan^{-1}(V_Q/V_I)$ as

$$\angle V = \begin{cases} \tan^{-1} \frac{V_Q}{V_I} & V_I > 0 \\ \tan^{-1} \frac{V_Q}{V_I} + \pi & V_I < 0 \text{ and } V_Q \geq 0 \\ \tan^{-1} \frac{V_Q}{V_I} - \pi & V_I < 0 \text{ and } V_Q < 0 \\ +\pi/2 & V_I = 0 \text{ and } V_Q > 0 \\ -\pi/2 & V_I = 0 \text{ and } V_Q < 0 \end{cases} \quad (1.9)$$

From here onwards, we will call this adjustment as a *four-quadrant inverse tangent* operation.

Addition and Multiplication Operations

The addition and multiplication rules for complex numbers are explained below.

Note 1.3 Operations in IQ-plane

Following rules apply to two complex numbers U and V in an IQ-plane, which is basically a simpler way to write complex additions and multiplications.

Addition: $W = U + V$ implies

$$\begin{array}{ll} I \rightarrow & W_I = U_I + V_I \\ Q \uparrow & W_Q = U_Q + V_Q \end{array} \quad (1.10)$$

Note that one complex addition results in two real additions, one each for I and Q , as shown in Figure 1.23.

Multiplication: $W = UV$ implies

$$|W| = |U|.|V| \quad \angle W = \angle U + \angle V \quad (1.11)$$

which in IQ form results in

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{array}{l} W_I = U_I V_I - U_Q V_Q \\ W_Q = U_Q V_I + U_I V_Q \end{array} \quad (1.12)$$

Note that one complex multiplication results in 4 real multiplications and 2 real additions. This is illustrated in Figure 1.23.

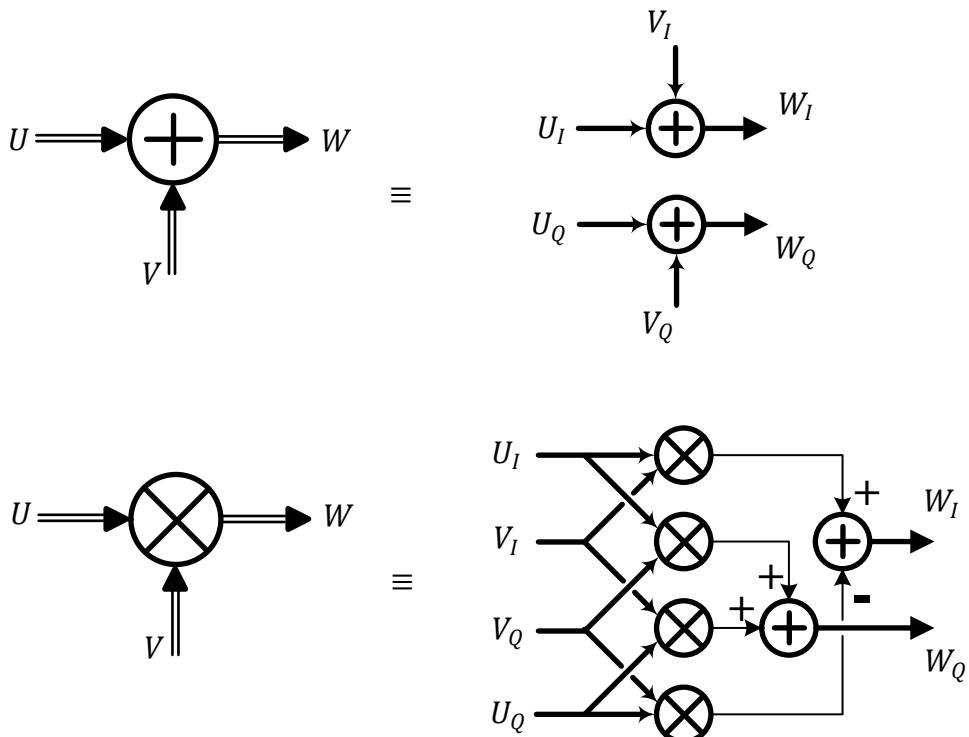


Figure 1.23: One complex addition results in two real additions, one each for I and Q . Also, one complex multiplication results in 4 real multiplications and 2 real additions

From Eq (1.12), *the multiplication rule* of complex numbers can be written as

Multiplying two complex numbers

$$\begin{aligned} & I \cdot I - Q \cdot Q \\ & Q \cdot I + I \cdot Q \end{aligned} \quad (1.13)$$

Addition rule above makes perfect sense: both I components are added together

as well as both Q components. The multiplication rule in Eq (1.13) seems a little strange though: *I is a difference between products of two aligned-axes terms (i.e., $I \cdot I - Q \cdot Q$), while Q is a sum of products of two cross-axes terms (i.e., $Q \cdot I + I \cdot Q$)*.

To see why multiplication of two complex numbers is perfectly logical, consider that

$$\begin{aligned} U_I &= |U| \cos \angle U & \text{and} & \quad U_Q = |U| \sin \angle U \\ V_I &= |V| \cos \angle V & \text{and} & \quad V_Q = |V| \sin \angle V \end{aligned}$$

Plugging in these values in Eq (1.13),

$$\begin{array}{ll} I \rightarrow & W_I = |U| \cos \angle U \cdot |V| \cos \angle V - |U| \sin \angle U \cdot |V| \sin \angle V \\ Q \uparrow & W_Q = |U| \sin \angle U \cdot |V| \cos \angle V + |U| \cos \angle U \cdot |V| \sin \angle V \end{array}$$

which leads to

$$\begin{array}{ll} I \rightarrow & W_I = |U||V| (\cos \angle U \cdot \cos \angle V - \sin \angle U \cdot \sin \angle V) \\ Q \uparrow & W_Q = |U||V| (\sin \angle U \cdot \cos \angle V + \cos \angle U \cdot \sin \angle V) \end{array}$$

Using the identities $\cos A \cos B - \sin A \sin B = \cos(A + B)$ and $\sin A \cos B + \cos A \sin B = \sin(A + B)$,

$$\begin{array}{ll} I \rightarrow & W_I = |U||V| \cos(\angle U + \angle V) \\ Q \uparrow & W_Q = |U||V| \sin(\angle U + \angle V) \end{array} \quad (1.14)$$

which is equivalent to Eq (1.11). Hence, *multiplication of two complex numbers is about multiplying their magnitudes and adding their phases*.

Raising a Complex Number to M^{th} Power

Another consequence of the definition of complex multiplication as in Eq (1.11) is that raising a complex number V to a power, say 2, generates

$$\begin{aligned} |W| &= |V| \cdot |V| = |V|^2 \\ \angle W &= \angle V + \angle V = 2 \cdot \angle V \end{aligned}$$

For a general power M ,

$$\begin{aligned} |W| &= |V|^M \\ \angle W &= M \cdot \angle V \end{aligned} \quad (1.15)$$

We can conclude that in an IQ -plane, raising a complex number to a certain power *raises* the magnitude to that power while *multiples* the phase with that number.

Phase Rotation

Rotating a complex number in IQ -plane by a phase θ seems very simple in complex notation but a bit complicated in IQ terms. *Rotation implies keeping the magnitude constant and adding θ to the angle of that complex number*, as shown in Figure 1.24. Let us see why.

To start, let us multiply a complex number V by a complex number U with magnitude 1 and angle θ . Using $|U| = 1$ and $\angle U = \theta$,

$$U_I = \cos \theta \quad \text{and} \quad U_Q = \sin \theta$$

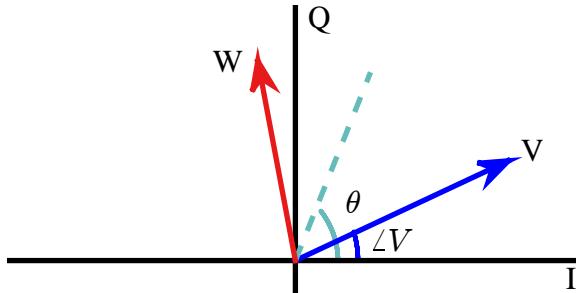


Figure 1.24: Rotation means adding θ to the angle with no change in magnitude

Plugging in the multiplication rule of complex numbers of Eq (1.13),

$$\begin{array}{ll} I \rightarrow & W_I = V_I \cdot \cos \theta - V_Q \cdot \sin \theta \\ Q \uparrow & W_Q = V_Q \cdot \cos \theta + V_I \cdot \sin \theta \end{array} \quad (1.16)$$

Using $V_I = |V| \cos \angle V$ and $V_Q = |V| \sin \angle V$,

$$\begin{array}{ll} I \rightarrow & W_I = |V| (\cos \angle V \cdot \cos \theta - \sin \angle V \cdot \sin \theta) \\ Q \uparrow & W_Q = |V| (\sin \angle V \cdot \cos \theta + \cos \angle V \cdot \sin \theta) \end{array}$$

Again using the identities $\cos A \cos B - \sin A \sin B = \cos(A + B)$ and $\sin A \cos B + \cos A \sin B = \sin(A + B)$,

$$\begin{array}{ll} I \rightarrow & W_I = |V| \cos(\angle V + \theta) \\ Q \uparrow & W_Q = |V| \sin(\angle V + \theta) \end{array} \quad (1.17)$$

which keeps the magnitude unchanged and adds the angle θ to the existing angle[†].

From Eq (1.16) and Eq (1.17), a fast rule for phase rotation can be devised as follows. For a complex number V and an angle $+\theta$ (i.e., anticlockwise rotation), *the phase rotation rule* states that

Anticlockwise rotation by $\theta \equiv$

$$\begin{array}{l} I \cdot \cos \theta - Q \cdot \sin \theta \\ Q \cdot \cos \theta + I \cdot \sin \theta \end{array} \quad (1.18)$$

This is recognized to be the same as the multiplication rule $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$. Similarly, for a complex number V and an angle $-\theta$ (i.e., clockwise rotation)

Clockwise rotation by $\theta \equiv$

$$\begin{array}{l} I \cdot \cos \theta + Q \cdot \sin \theta \\ Q \cdot \cos \theta - I \cdot \sin \theta \end{array} \quad (1.19)$$

[†]As an example of what happens when phase information is neglected, this is how my daughter writes some English letters:

$$L \rightarrow J, \quad V \rightarrow A, \quad Z \rightarrow \Sigma$$

The phase rotation rule above is important because we are not using complex notation in this text. That implies having on our disposal a quick way to recognize an equation if it rotates a complex number by an angle. The above two equations help fulfill that purpose for which we will see a lot of applications later, e.g., during the carrier phase synchronization.

Complex Conjugate

The *conjugate* V^* of a complex number V just changes the sign of the Q component. It is defined as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} \{V^*\}_I &= V_I \\ \{V^*\}_Q &= -V_Q \end{aligned} \quad (1.20)$$

Since magnitude is the sum of squares of I and Q components, it remains unchanged. On the other hand, phase is Q divided by I and hence a conjugation leads to a change of sign in phase. Consequently, an alternative definition of the conjugate of a complex number is

$$\begin{aligned} |V^*| &= |V| \\ \angle V^* &= -\angle V \end{aligned} \quad (1.21)$$

A significance of conjugate of a complex number arises from the fact that a complex number multiplied by its complex conjugate cancels the phase and produces its magnitude squared. Using the above relations in multiplication rule of Eq (1.11),

$$\begin{aligned} |V \cdot V^*| &= |V|^2 = V_I^2 + V_Q^2 \\ \angle(V \cdot V^*) &= 0 \end{aligned} \quad (1.22)$$

Another interesting consequence of defining the conjugate of a signal is the following identity set.

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} \{V + V^*\}_I &= V_I + V_I = 2V_I \\ \{V + V^*\}_Q &= V_Q - V_Q = 0 \end{aligned}$$

which generates two important results.

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} V_I &= \frac{1}{2} \{V + V^*\}_I \\ 0 &= \frac{1}{2} \{V + V^*\}_Q \end{aligned} \quad (1.23)$$

On the same note,

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} 0 &= \frac{1}{2} \{V - V^*\}_I \\ V_Q &= \frac{1}{2} \{V - V^*\}_Q \end{aligned} \quad (1.24)$$

1.4 The Concept of Frequency

A wireless signal from one device to another travels through the use of electromagnetic waves propagated by an antenna. Electromagnetic waves have different frequencies and one can pick up a specific signal by tuning a radio Rx to a specific frequency. But what is a frequency anyway?

A Complex Sinusoid

Consider again a complex number V previously shown in Figure 1.21 in an IQ -plane. Now here imagine V rotating anticlockwise in a circle at a constant rate with time as drawn in Figure 1.25. We note the following.

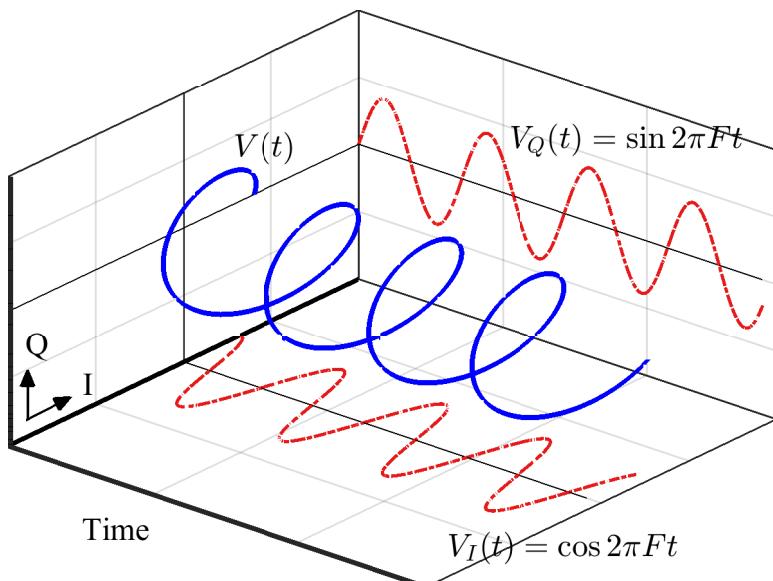


Figure 1.25: A complex sinusoid V rotating in time IQ -plane and generating a helix that projects two real sinusoids: $\cos 2\pi F t$ on I -axis and $\sin 2\pi F t$ on Q -axis

- Instead of a complex number V , now $V(t)$ can be treated as a signal with time as independent variable and we call it a *complex sinusoid*.
- A constant rate implies that in any given duration Δt , the change in phase $\Delta\theta$ is a constant. This constant is known as the angular velocity and is defined as the rate of change in phase of this complex sinusoid, just like velocity is the rate of change of displacement.

$$\text{angular velocity} = \frac{\Delta\theta}{\Delta t}$$

- This angular velocity results in $V(t)$ rotating in the time IQ -plane and generating a helix, as shown in Figure 1.25.

- The **frequency** of this complex sinusoid is defined as this angular velocity normalized by 2π so that the units are cycles/second instead of radians/second.

$$F = \frac{1}{2\pi} \cdot \frac{\Delta\theta}{\Delta t}$$

Note that as time passes, $V(t)$ is shown in Figure 1.25 as coming out of the page. When its projection from a 3-dimensional plane to a 2-dimensional plane formed by time and I -axis is drawn downwards, we get its I part $V_I(t) = \cos 2\pi F t$. Similarly, when the projection is drawn on a 2-dimensional plane formed by time and Q -axis (on the back of the page), it generates the Q part $V_Q(t) = \sin 2\pi F t$. Randomly choosing $\cos(\cdot)$ as our reference sinusoid, the I component is called **inphase** because it is in phase with $\cos(\cdot)$ while Q component is called **quadrature** because $\sin(\cdot)$ is in quadrature – i.e., 90° apart – with $\cos(\cdot)$.

In conclusion, a complex sinusoid with frequency F is composed of two real sinusoids

I	\rightarrow	$V_I(t) = \cos 2\pi F t$ $V_Q(t) = \sin 2\pi F t$	(1.25)
Q	\uparrow		

where F is the continuous frequency with units of cycles/second or Hertz (Hz). The direction of rotation is considered positive for anticlockwise rotation, and negative for clockwise rotation.

Amplitude A and phase θ are the two other parameters that characterize a sinusoidal signal, where A determines the maximum amplitude for the sinusoid, and θ determines the initial angular offset of $V(t)$ at $t = 0$.

Note 1.4 Sinusoid parameters

Remember that whenever you hear the word “frequency”, it is the frequency of a sinusoid $A \cos(2\pi F t + \theta)$ or $A \sin(2\pi F t + \theta)$. There are 3 parameters that characterize a sinusoid:

1. Frequency $\rightarrow F$
2. Phase $\rightarrow \theta$
3. Amplitude $\rightarrow A$

It is clear that if the complex sinusoid $V(t)$ rotates with a frequency F , it completes F cycles each second. In other words, it takes $1/F$ seconds to complete each cycle which is known as the time period T . Therefore, the frequency F is related to time period T as

$$F = \frac{1}{T}$$

and the range of continuous frequency values is

$$-\infty < F < \infty \quad (1.26)$$

Example 1.3

When one tunes to an FM radio station at 88 MHz, one is actually listening to a station broadcasting a radio signal at a carrier frequency of 88×10^6 Hz, which means that the transmitter is oscillating at a frequency of 88,000,000 cycles/second. Accordingly, that wave is completing one period in $T = 1/F = 11.4$ ns.

We find complex sinusoids everywhere such as in the form of a helical spring or a round staircase, see Figure 1.26.



Figure 1.26: Complex sinusoids are like a helical spring and a round staircase

What is a Negative Frequency?

Long ago, a lot of confusion clouded around negative numbers when numbers were used to count things. People could not understand how a person A can give -3 apples to a person B . It turned out that this can be accomplished by changing the direction of giving, i.e., A actually takes 3 apples from B .

Similarly, a frequency is usually defined as an inverse period $F = 1/T$ of a sinusoid. Naturally it becomes hard to visualize a negative frequency viewed as inverse period. Define it through the rate of rotation in anticlockwise direction of a complex sinusoid $V(t)$ as in Figure 1.25, and it is evident that a *negative frequency* simply implies rotation of $V(t)$ in a clockwise direction. For example, the complex sinusoid

$$\begin{aligned} I &\rightarrow \cos 2\pi(-F)t = \cos 2\pi F t \\ Q &\uparrow \quad \sin 2\pi(-F)t = -\sin 2\pi F t \end{aligned}$$

has a frequency of $-F$. Here, we have used the identities $\cos(-\theta) = \cos \theta$ and $\sin(-\theta) = -\sin \theta$.

Negative frequencies are a reality, just like negative numbers are a reality. Also note that from the definition of a complex conjugate, we can say that $V(t)$ is a complex sinusoid with frequency F and $V^*(t)$ is a complex sinusoid with frequency $-F$.

A Real Sinusoid

We have learned that a complex sinusoid rotating in time IQ -plane generates two real sinusoids. The question is: How to produce only one real sinusoid in complex IQ -plane?

Interestingly, just like a complex sinusoid is made up of two real sinusoids as in Eq (1.25), a real sinusoid can be produced by two complex sinusoids rotating in opposite directions to each other, $V(t)$ with a positive frequency F and $V^*(t)$ with a negative frequency $-F$. Let us see how.

Figure 1.27 draws two complex sinusoids, $V(t)$ with a frequency F shown as solid blue line while $V^*(t)$ with frequency $-F$ shown as dashed red line. Their I and Q components are similarly drawn as solid blue and dashed red lines, respectively. Observe that

- the I parts exactly fall on top of each other and hence are the same $\cos 2\pi F t$, whereas
- the Q parts carry exactly the same amplitude with opposite signs to each other and hence are $\sin 2\pi F t$ and $-\sin 2\pi F t$, respectively.

Consequently, when two such complex sinusoids are added together, the I parts add up point-by-point while the Q arms cancel out as a result of this addition.

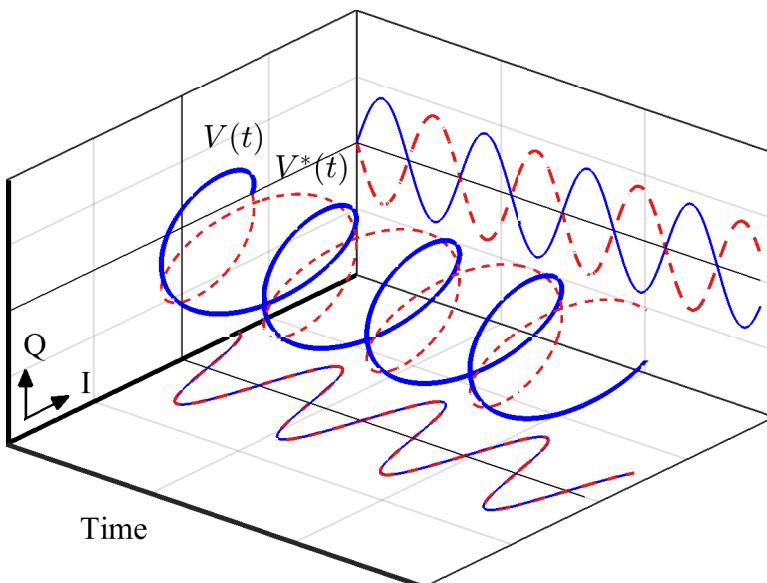


Figure 1.27: Two complex sinusoids $V(t)$ and $V^*(t)$ rotating in time IQ -plane

This *addition* is a graphical representation of what we got from Eq (1.23) before and reproduced below.

$$\begin{array}{ll} I \rightarrow & \cos 2\pi F t = \frac{1}{2} \{ V(t) + V^*(t) \} \\ Q \uparrow & 0 = \frac{1}{2} \{ V(t) + V^*(t) \} \end{array} \quad (1.27)$$

This figure is also a graphical representation of a sine wave that can be constructed in a similar manner by *subtracting* the two complex sinusoids as done in Eq (1.24) before.

$$\begin{array}{ll} I \rightarrow & 0 = \frac{1}{2} \{ V(t) - V^*(t) \} \\ Q \uparrow & \sin 2\pi F t = \frac{1}{2} \{ V(t) - V^*(t) \} \end{array} \quad (1.28)$$

This is somewhat similar to a double helix in a DNA molecule that carry the genetic instructions for our development in Figure 1.28. Maybe when the digital machines rise against us one day, they will think of us as one of their own kinds.



Figure 1.28: A DNA double helix

Frequency Domain

We have talked about frequency being the rate of rotation of a complex sinusoid in time IQ -plane, see Figure 1.25. It is evident that this rate of rotation can be changed from very slow (close to 0) to as fast as possible (close to $+\infty$). Also, a clockwise direction of rotation implies a negative frequency, and hence the complete range of frequencies of a complex sinusoid is from $-\infty$ to $+\infty$. When a signal or function is drawn in *frequency domain*, the graph actually shows the I and Q components of those complex sinusoids whose frequencies are present in that signal. We begin with the simplest case of a complex sinusoid, then move towards a real sinusoid and then handle the most general case of an arbitrary signal.

Complex sinusoid: Since one complex sinusoid has a single frequency, it is drawn as a single narrow impulse on the frequency IQ -plane in Figure 1.29. The impulse is on $+F$ if the direction of rotation is anticlockwise and on $-F$ when the direction of rotation is clockwise. The impulse is at $-F$ in the figure due to the clockwise rotation in time domain.

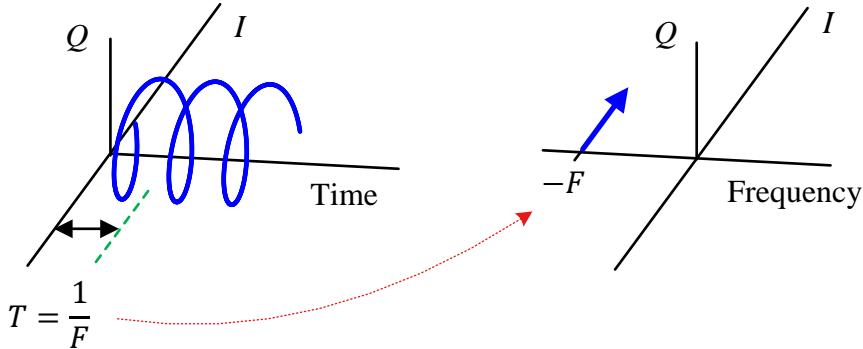


Figure 1.29: Representation of a complex sinusoid in frequency domain. A negative value of F implies clockwise rotation

Real cosine: From what we learned above in Eq (1.27), a frequency domain representation of a cosine wave should be two impulses, one at frequency $+F$ due to $V(t)$ and the other at frequency $-F$ contributed by $V^*(t)$. This is illustrated in Figure 1.30.

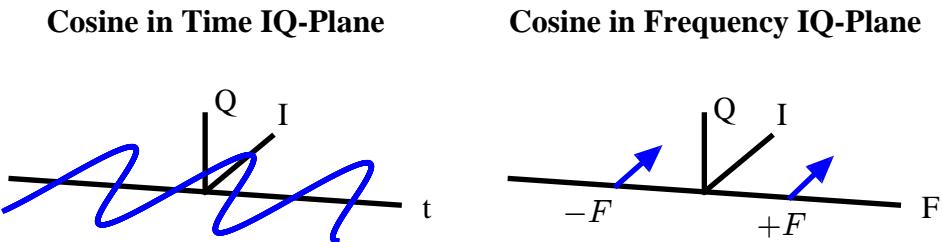


Figure 1.30: A cosine wave in Time IQ and Frequency IQ-planes

Real sine: The case of a sine wave is slightly trickier. For the subtraction of two complex sinusoids, start with Eq (1.28) that has a graphical representation in Figure 1.27. The result should be two impulses, one at frequency $+F$ due to $V(t)$ and the other at frequency $-F$ contributed by $V^*(t)$. However, the sign of $V^*(t)$ is negative as per Eq (1.28) and illustrated in Figure 1.31. Consequently, notice that the sine wave exists on the time Q -axis.

A question is: what is the frequency domain representation of a sine wave on time I -axis? First, we say that a transform is utilized to go from time to frequency domain (such as a Discrete Fourier Transform (DFT) covered later).

$$\text{Time Domain} \xrightarrow{\text{Transform}} \text{Frequency Domain}$$

Next, we observe that rotating the time waveform of Figure 1.31 clockwise by 90° , the sine wave can be put on time I -axis. This will correspondingly rotate the

Sine on Q-axis in Time IQ-Plane Sine in Frequency IQ-Plane

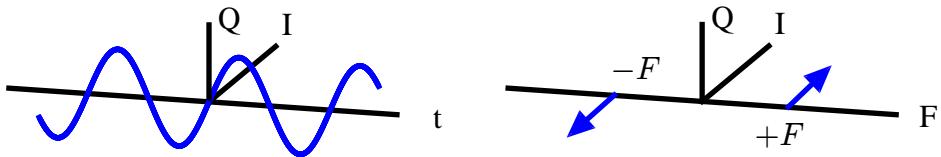


Figure 1.31: A sine wave in Time IQ and Frequency IQ -planes, residing on time Q -axis

frequency representation by 90° clockwise[†] as well. With this step, the frequency domain representation of a sine wave on time I -axis is plotted in Figure 1.32.

Sine on I-axis in Time IQ-Plane Sine in Frequency IQ-Plane

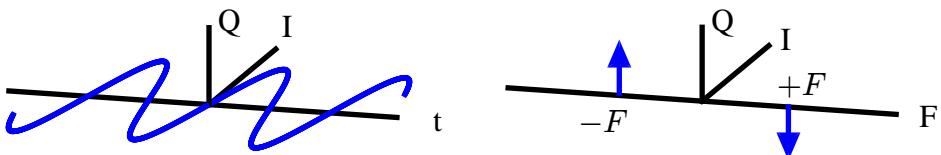


Figure 1.32: A sine wave in Time IQ and Frequency IQ -planes, residing on time I -axis

As a confirmation, we said that the inphase component implies a signal in phase with $\cos(\cdot)$. From the identity $\cos(A - 90^\circ) = \sin(A)$, we can see why a sine wave is represented with a -90° phase rotation in frequency domain!

Note 1.5 Time IQ -plane and Frequency IQ -plane

Visualizing the time and frequency IQ -planes is the single most important tool in understanding Digital Signal Processing (DSP) techniques. As we will see later, it will be of great help in devising algorithms for successful implementation of digital and wireless communications.

If I assume that you can learn only one concept from this book and nothing else, I will recommend you departing with this skill.

In the above discussion, we saw that a complex sinusoid is represented by a single impulse in frequency domain. Two such complex sinusoids in time (and hence impulses in frequency) combine to form a real sinusoid. In frequency domain, we actually plotted the magnitudes through lengths and phases through orientations of those complex sinusoids.

[†]We can justify this step through the concept of linearity. It is just a phase rotation by $\exp(-j\pi/2)$ on both sides of the Transform (which is a multiplication by $-j$).

Similar is the case for signals composed of more than two complex sinusoids. As such, these complex sinusoids form a basic unit of signal construction of any shape and each ‘tick’ on the frequency axis represents the frequency of one participating complex sinusoid. As more and more of them come together, they form a continuum in frequency domain that is illustrated as a continuous spectral graph.

A natural question arises at this stage: What about the signals having components other than these nice looking sinusoids? The answer is surprising: *There are hardly any!* Long ago, scientists figured out that most signals of practical interest can be considered as a sum of many sinusoids – possibly infinite – oscillating at different frequencies regardless of the signal shape. So an arbitrary signal $s(t)$ can be written as

$$\begin{aligned}s(t) &= a_0 \sin(2\pi F_0 t) + a_1 \sin(2\pi F_1 t) + a_2 \sin(2\pi F_2 t) + \dots \\ &= a_0 \sin(2\pi \frac{1}{T_0} t) + a_1 \sin(2\pi \frac{1}{T_1} t) + a_2 \sin(2\pi \frac{1}{T_2} t) + \dots\end{aligned}$$

where a_0, a_1, a_2, \dots , are amplitudes that determine the impact each sinusoid has on $s(t)$ while $1/T_0, 1/T_1, 1/T_2, \dots$ are their respective frequencies. If $s(t)$ has no sinusoid of frequency $1/T_k$, then the corresponding amplitude $a_k = 0$.

It is difficult to believe in such a statement for many signals with sharp edges like a square or triangular waveform. Nonetheless, the concept is still true and the number of sinusoids participating in construction of such a signal tends to infinity. We now examine a few examples of such phenomena from everyday life.



Figure 1.33: 4th dimension as depicted in the amazing movie Interstellar (2014)

- While the movie Interstellar captured the 4th dimension quite well (see Figure 1.33), have a look around wherever you are reading these lines and note that you can spot only 3 dimensions of space, namely x, y and z . Any point in space can be represented with a combination of just these 3 basis components. Exactly in a similar but non-trivial way, almost all practical signals can be represented by a linear combination of the complex sinusoids and hence they are known as the basis signals.

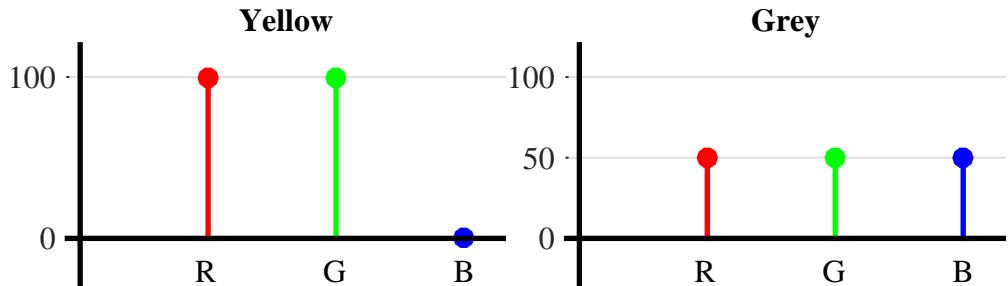


Figure 1.34: Contribution of red, green and blue in forming yellow and grey colours, respectively

- Another good analogy is the RGB model in which Red, Green, and Blue colours combine together in various ways to reproduce a broad range of different colours. Figure 1.34 shows the ‘spectral’ contents of yellow and grey colours formed by contributions [100% 100% 0%] and [50% 50% 50%] from red, green and blue colours, respectively.

In the context of signals, consider a square wave which is a periodic signal. The curves in Figure 1.35 show how it is approximated with integer multiples of a fundamental frequency $F = 1/T$ (the corresponding negative axis in frequency domain is not shown for simplicity). The slowly varying curve riding on the square wave in Figure 1.35a consists of the first two terms

$$s(t) = \sin(2\pi F t) + \frac{1}{3} \sin(2\pi 3F t) = \sin\left(2\pi \frac{1}{T} t\right) + \frac{1}{3} \sin\left(2\pi \frac{3}{T} t\right)$$

which is clearly quite inaccurate. However, we can increase this approximation with

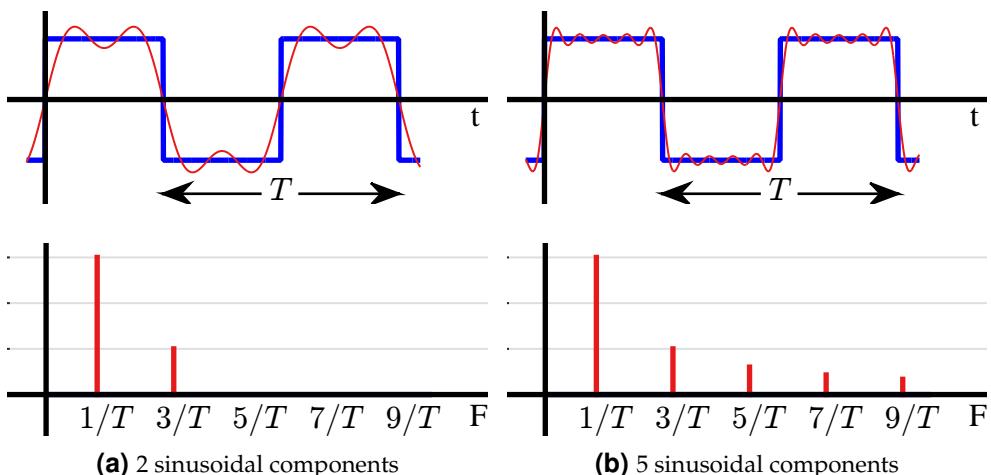


Figure 1.35: Approximation of a square wave through an increasing number of sinusoidal components becomes more accurate

just three more terms in the curve riding on the square wave of Figure 1.35b as

$$\begin{aligned}
 s(t) &= \sin(2\pi Ft) + \frac{1}{3} \sin(2\pi 3Ft) + \frac{1}{5} \sin(2\pi 5Ft) + \\
 &\quad \frac{1}{7} \sin(2\pi 7Ft) + \frac{1}{9} \sin(2\pi 9Ft) \\
 &= \sin\left(2\pi \frac{1}{T} t\right) + \frac{1}{3} \sin\left(2\pi \frac{3}{T} t\right) + \frac{1}{5} \sin\left(2\pi \frac{5}{T} t\right) + \\
 &\quad \frac{1}{7} \sin\left(2\pi \frac{7}{T} t\right) + \frac{1}{9} \sin\left(2\pi \frac{9}{T} t\right)
 \end{aligned} \tag{1.29}$$

that displays significantly closer behaviour. If we increase the number of sinusoids with respective decreasing amplitudes, we can improve this approximation even further. Sometimes it is better to present this idea in a 3D figure to grasp the whole concept. Figure 1.36 draws the approximation of the square wave containing 5 sinusoidal components on one side and the corresponding frequency domain representation on the other. Clearly, the meaning of all such waveforms adding together to generate the approximate square wave is more clear here.

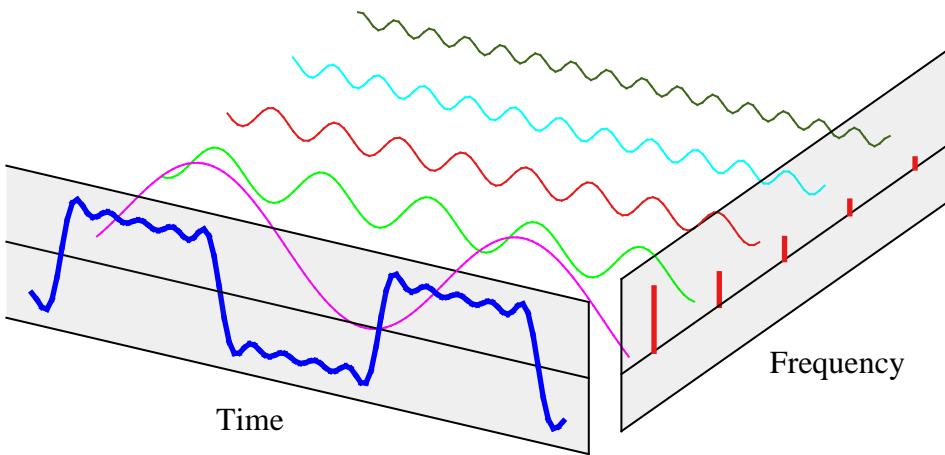


Figure 1.36: 5 sinusoidal components adding together to generate the approximate square wave in time domain. Compare with Figure 1.35b

Now if you are confident that you have understood this concept, I want to shake it a little for a deeper appreciation. There is *a slight misrepresentation* in Figure 1.36. The actual participants that add up are complex sinusoids, not the real sinusoids drawn in the figure. On the positive side of the spectrum, these complex sinusoids are rotating in an anticlockwise direction. On the negative side of the spectrum (not drawn), these are rotating in the clockwise direction. And the real sinusoids shown are just a special case due to this real and even square wave in time domain. All this would have been too complicated to draw in one such figure but we will gradually learn these concepts in this and the next chapter.

This representation of periodic signals is known as Fourier series. For aperiodic signals, the period T can be considered as infinite, the spacing between these frequencies diminishes and a continuous frequency domain graph is obtained as the Fourier Transform. In this text, we will not bother about Fourier series, Fourier Transform or Discrete-Time Fourier Transform. Instead, we will only focus on the Discrete Fourier Transform (DFT) that is actually computed in our digital machines.

Spectrum and Bandwidth

A frequency spectrum, or simply the *spectrum*, is just the range of all possible frequencies of electromagnetic radiation. A full continuous spectrum, for example, includes radio waves, microwaves, infrared, ultraviolet, x-rays and gamma rays. In the context of a signal, its spectrum contains the frequencies of complex sinusoids that sum up in time domain to form that signal.

In an ideal world, the *bandwidth* of a signal is the *range of frequencies* of complex sinusoids present in that signal. In other words, taking into account all the sinusoids, the bandwidth is simply the difference between the highest frequency F_H and the lowest frequency F_L in the spectrum of that signal.

$$\text{BW} = F_H - F_L \quad (1.30)$$

An ideal *band-limited signal* has a spectrum that is zero outside a finite frequency range $F_L \leq |F| \leq F_H$:

$$S(F) = \begin{cases} 0, & 0 \leq |F| \leq F_L \\ x, & |F_L| \leq |F| \leq |F_H| \\ 0, & F_H \leq |F| \leq \infty \end{cases} \quad (1.31)$$

Having said that, a signal can have a very low contribution from a sinusoid of a particular frequency but not completely zero. Skipping the mathematical proof, we present the following argument: remember that the concept of frequency is defined through complex sinusoids that are infinitely long in time domain. Since sinusoids exist only for a finite duration in real-world, the spectrum of these truncated sinusoids is not an impulse anymore. Instead, it spreads out in the entire frequency range from $-\infty$ to $+\infty$ and so does the ideal bandwidth of a time-limited signal. As a general rule, signals with fast irregular variations have wide bandwidth as a large number of sinusoids are required to construct such steep curves, while slowly varying signals are relatively smooth and few sinusoids are required to build them up.

Note 1.6 Time and frequency support

A signal cannot be limited in both time and frequency domains. For practical implementations, a signal must be time limited which makes it unlimited in frequency. Therefore, every real signal occupies an infinite amount of bandwidth. A band-limited signal is then referred to as a signal with most of its energy concentrated within a certain amount of frequency range. Practical definitions of bandwidth vary depending on that amount of energy.

For example, Federal Communications Commission (FCC) defines bandwidth as the band in which 99% of the signal power is contained. Another common definition is that everywhere outside the specified band, a certain attenuation (say, 60 dB) must be attained.

1.5 Sampling a Continuous-Time Signal

Most signals of our interest, e.g., wireless communication waveforms, are continuous-time signals as they have to travel through a real wireless channel. To process such a signal using Digital Signal Processing (DSP) techniques, it must be converted into a sequence of numbers. This can be done through the process of periodic sampling.

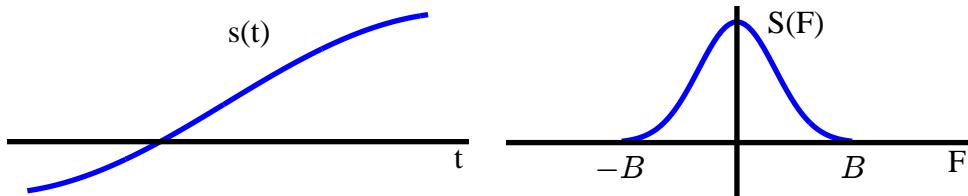


Figure 1.37: A continuous-time signal in time and frequency domains

Consider a band-limited continuous-time signal $s(t)$ and its frequency domain representation $S(F)$ with bandwidth B shown in Figure 1.37. A discrete-time signal $s[n]$ can be obtained by taking samples of $s(t)$ at equal intervals of T_S seconds. This process is shown in Figure 1.38 and mathematically represented as

$$s[n] = s(t) \Big|_{t=nT_S} \quad (1.32)$$

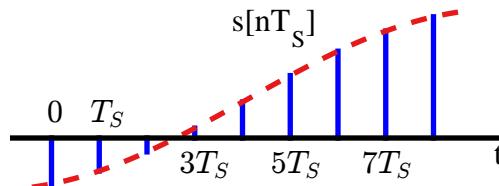


Figure 1.38: Sampling a continuous-time signal

The time interval T_S seconds between two successive samples is called the *sampling period* or *sample interval*, and its reciprocal is called the *sample rate* or *sampling frequency*.

$$F_S = \frac{1}{T_S} \quad (1.33)$$

Sample rate F_S is the most fundamental parameter encountered in DSP applications[†].

Let us find out what happens in the frequency domain as a result of the sampling process. We consider a continuous-time sinusoidal signal here because they can together form most signals of practical interest.

$$s(t) = A \cos(2\pi F t + \theta)$$

[†]There is a scaling factor of $1/T_S$ in frequency domain as a result of the sampling process which we ignore for most of this text.

Its sampled version at a rate $F_S = 1/T_S$ samples/second is

$$\begin{aligned}
 s[n] &= s(t)|_{t=nT_S} \\
 &= A \cos(2\pi F n T_S + \theta) = A \cos\left(2\pi F \frac{n}{F_S} + \theta\right) \\
 &= A \cos\left(2\pi \frac{F}{F_S} n + \theta\right)
 \end{aligned} \tag{1.34}$$

Note that F/F_S above is the frequency of a discrete-time sinusoid $s[n]$. Let us sample at the same rate F_S another sinusoid with continuous frequency $F + kF_S$, where $k = \pm 1, \pm 2, \dots$.

$$\begin{aligned}
 s(t) &= A \cos\left\{2\pi(F + kF_S)t + \theta\right\} \\
 s[n] &= A \cos\left\{2\pi(F + kF_S)nT_S + \theta\right\} \\
 &= A \cos\left(2\pi \frac{F + kF_S}{F_S} n + \theta\right) = A \cos\left(2\pi \frac{F}{F_S} n + 2\pi kn + \theta\right) \\
 &= A \cos\left(2\pi \frac{F}{F_S} n + \theta\right)
 \end{aligned} \tag{1.35}$$

due to the 2π periodicity of the sinusoid. This expression is exactly the same as the discrete-time sinusoid in Eq (1.34). It can be concluded that at the output of sampling process, it is impossible to distinguish between two discrete-time signals whose frequencies are F_S Hz apart.

The fact that a continuous frequency higher than $0.5F_S$ Hz appears similar to a frequency F_S Hz apart from itself can be understood in time domain from Figure 1.39. Observe that samples are taken at a rate such that both sinusoids pass through the same points. In fact, there are infinitely many sinusoids (F_S Hz apart) which pass through the same points, and hence become indistinguishable from each other after sampling.

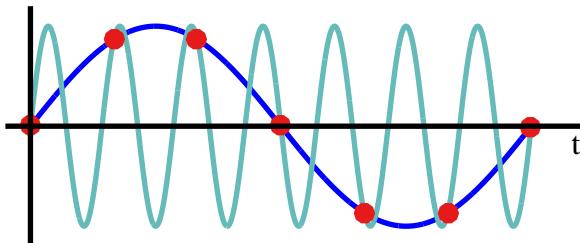


Figure 1.39: Two sinusoids spaced F_S Hz apart are indistinguishable after sampling

Therefore, for an arbitrary frequency F after sampling

$$\begin{aligned}
 F &= F + 1F_S = F - 1F_S \\
 &= F + 2F_S = F - 2F_S
 \end{aligned}$$

and so on. It is just like saying that any two angles 360° apart are all the same[†]. For example,

$$\begin{aligned} 30^\circ &= 30^\circ + 1(360^\circ) = 390^\circ \\ &= 30^\circ - 1(360^\circ) = -330^\circ \\ &= 30^\circ + 2(360^\circ) = 750^\circ \\ &= 30^\circ - 2(360^\circ) = -690^\circ \end{aligned}$$

Consequently all the following frequency *ranges* are the same:

$$\begin{array}{lll} \dots & & \\ -2.5F_S & \rightarrow & -1.5F_S \\ -1.5F_S & \rightarrow & -0.5F_S \\ & & \\ \boxed{-0.5F_S \quad \rightarrow \quad +0.5F_S} & & (1.36) \\ & & \\ 0.5F_S & \rightarrow & 1.5F_S \\ 1.5F_S & \rightarrow & 2.5F_S \\ \dots & & \end{array}$$

The range $-0.5F_S \rightarrow +0.5F_S$ is called the *baseband* or Nyquist band. The spectrum of the continuous-time signal $s(t)$ shown in Figure 1.37 is now drawn in Figure 1.40. We adopt the convention of indicating this zone within dotted red lines and drawing its spectral contents with solid lines while the spectral replicas with dashed lines.

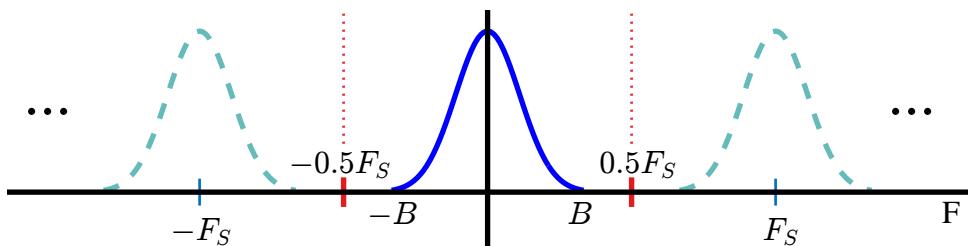


Figure 1.40: Spectrum after sampling. Dotted red lines indicate the baseband or Nyquist band

It is evident from Figure 1.40 that if a continuous-time signal has a bandwidth B greater than $0.5F_S$, it will appear as an alias of a lower frequency within the range $-0.5F_S \leq F < +0.5F_S$ and distort the signal. This *aliasing* is illustrated in Figure 1.41 for a signal whose bandwidth B extends beyond $F_S/2$.

[†]The reason a full circle has 360° , and not say 500° , is that the Sumerians of Mesopotamia who invented the writing system around 3000 BC used a combination of base-6 and base-10 numeral systems and they might have noticed that the sun completes one full annual circle ‘around the earth’ in approximately 360 days.

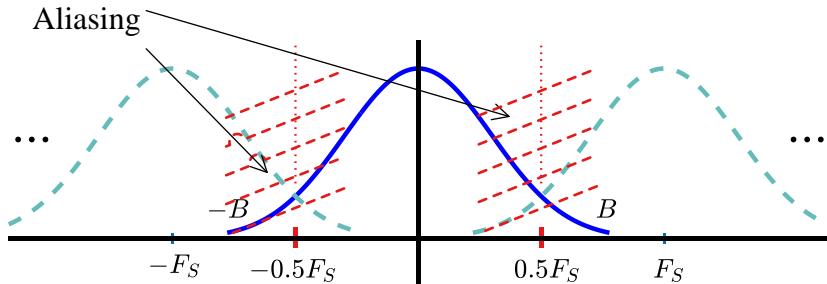


Figure 1.41: A signal whose bandwidth extends beyond the baseband experiences aliasing

Therefore, the highest frequency (or bandwidth) B a continuous-time signal $s(t)$ contains should be less than $0.5F_S$ to prevent any distortion in the sampled signal $s[n]$.

$$0.5F_S > B$$

or written in another form

$F_S > 2B$

(1.37)

This is known as the *sampling theorem*. As shown above, sampling in time domain at intervals of T_S creates periodicity in frequency domain with a period of $F_S = 1/T_S$. Therefore, a band-limited continuous-time signal with highest frequency (or bandwidth) B Hz can be uniquely recovered from its samples provided that the sample rate $F_S \geq 2B$ samples/second.

It is a common practice to call the exact frequency $2B$ as the *Nyquist rate* while $0.5F_S$ the *Nyquist frequency* or *folding frequency* (see Figure 1.41). Sampling theorem is one of the two most fundamental relations in digital signal processing, the other being the relationship between continuous and discrete frequencies in Eq (1.51) later.

A common example of aliasing is observable in what we call the wagon-wheel effect, in which a spoked wheel in a recorded video appears to rotate at a different rate as compared to its true speed of rotation. People living in regions of warm climates also see this effect with the blades of a ceiling fan in flickering light. In Figure 1.41, it is not just the spectrum that goes past $0.5F_S$ that is lost but the same spectral content reappears (aliased) as lower frequency. B. P. Lathi [5] mentions an interesting analogy to describe this effect. “The problem of aliasing is analogous to that of an army when a certain platoon has secretly defected to the enemy side but remains nominally loyal to their army. The army is in double jeopardy. First, it has lost the defecting platoon as an effective fighting force. In addition, during actual fighting, the army will have to contend with sabotage caused by the defectors and will have to use loyal platoon to neutralize the defectors. Thus, the army has lost two platoons to nonproductive activity.”

Note 1.7 Is aliasing always harmful?

Aliasing – the reemergence of frequencies higher than $0.5F_S$ within the range $-0.5F_S \leq F < +0.5F_S$ of baseband – is a consequence of violating the sampling theorem. *It may seem so but aliasing is not always bad.* In fact, there are three types of aliasing:

Harmful aliasing that distorts the signal and must be avoided for proper representation of a signal in discrete domain.

Useful aliasing that shifts the channel spectra up and down to our desired frequency through careful system design. This is employed in systems operating at multiple clock rates, known as multirate systems.

Harmless aliasing that is neither good nor bad for the system. This occurs, for example, during band-limited pulse shape design to avoid Inter-Symbol Interference (ISI) and sampling of Orthogonal Frequency Division Multiplexing (OFDM) systems.

Don't worry much if it sounds too confusing at this stage. We will cover everything in detail when the topic arises.

A final remark about sampling a continuous-time signal is that for a fixed time interval of data collection, a higher sample rate translates into a higher energy in the resulting discrete-time signal because there will be more samples present in that discrete-time sequence.

Next, we discuss the most important topic in digital and wireless communications, namely the correlation.

1.6 Correlation

Correlation is a foundation over which the whole structure of digital communications is built. In fact, *correlation is the heart of a digital communication system*, not only for data detection but for parameter estimation of various kinds as well. Due to this reason, we name it as *the Master Algorithm*, the one algorithm from which most of the detection and estimation problems in digital and wireless communication systems can be solved. Throughout this text, we will find recurring reminders of this fact.

As a start, we will soon discover that the Discrete Fourier Transform (DFT) is just a sum of term-by-term products between an input signal and a complex sinusoid which is actually a computation of correlation. Later, we will learn that to detect the transmitted bits at the receiver, correlation is utilized to select the most likely candidate. Moreover, estimates of timing, frequency and phase of a received signal are extracted through judicious application of correlation for synchronization, as well as channel estimates for equalization.

By definition, correlation is a measure of similarity between two signals. In our everyday life, we recognize something by running in our heads its correlation with what we know. Correlation plays such a vital and deep role in diverse areas of our life, be it science, sports, economics, business, marketing, criminology or psychology, that a complete book can be devoted to this topic.

“The world is full of obvious things which nobody by any chance ever observes.”

Holmes to Watson - The Hound of the Baskervilles

For all of Sherlock Holmes' inferences, his next step after observation was always correlation. For example, he accurately described Dr. James Mortimer's dog through correlating some observations with templates in his mind:

“ . . . and the marks of his teeth are very plainly visible. The dog’s jaw, as shown in the space between these marks, is too broad in my opinion for a terrier and not broad enough for a mastiff. It may have been – yes, by Jove, it is a curly-haired spaniel.”

Sherlock Holmes - The Hound of the Baskervilles



Figure 1.42: A statue of Sherlock Holmes was erected in 1999 near his fictitious address of 221B Baker Street in London perhaps in recognition of his services to the art of correlation

Next, we begin the discussion with real signals while the case of complex signals will be discussed later.

Correlation of Real Signals

The objective of correlation between two signals is to measure the degree to which those two signals are similar to each other. Mathematically, correlation between two signals $s[n]$ and $h[n]$ is defined as

$$\text{corr}[n] = \sum_{m=-\infty}^{+\infty} s[m]h[m-n] \quad (1.38)$$

where the above sum is computed *for each n* from $-\infty$ to $+\infty$. The process of correlation can be understood with the following steps.

- 1. Selecting the time shift:** First, notice from its definition above that there are two indices, n and m . The index m is the summation index and the index n is the variable used for the signal shift.

$$h[m-n]$$

- 2. Sample-by-sample product:** Keep $s[m]$ where it is and compute the product

$$s[m] \cdot h[m-n]$$

The only nonzero values will be at the indices m where the two signals overlap.

- 3. Sum:** Add all these sample-by-sample products to generate the correlation output at the shift n .

$$\text{corr}[n] = \sum_{m=-\infty}^{+\infty} s[m]h[m-n]$$

4. Repeat: Repeat the above process for all possible time shifts n .

Intuitively, this summation of sample-by-sample products computes the level of similarity between the two signals at each time shift. For example, if $s[n]$ is correlated with $s[n]$, the two signals are exactly the same for time shift 0 and the correlation output will be the sum of squared samples, i.e., energy in the signal $s[n]$.

An example of correlation between two signals

$$\begin{aligned}s[n] &= [2 \ -1 \ 1] \\ h[n] &= [-1 \ 1 \ 2]\end{aligned}$$

is shown in Figure 1.43. In the left column, the signal $s[m]$ is copied with shifted versions of $h[m]$ to demonstrate the overlap between the two, while the shift n is shown in the text box for each n . If you follow the steps described above, you will find the results $\text{corr}[n]$ in the text boxes on the right column of the figure.

$$r[n] = [4 \ 0 \ \underline{-1} \ 2 \ -1]$$

Since we said that the correlation is the heart of a digital communication system, we denote correlation operation by \heartsuit as

$$\text{corr}[n] = s[n] \heartsuit h[n] \quad (1.39)$$

Note that

$$s[n] \heartsuit h[n] \neq h[n] \heartsuit s[n] \quad (1.40)$$

This can be verified by plugging $p = m - n$ in Eq (1.38) which yields $m = n + p$ and hence

$$\begin{aligned}\sum_{p=-\infty}^{\infty} s[n+p]h[p] &= \sum_{p=-\infty}^{\infty} h[p]s[p+n] \\ &\neq \sum_{m=-\infty}^{\infty} h[m]s[m-n]\end{aligned}$$

Nevertheless, it can be deduced that Eq (1.38) is equivalent to

$$\text{corr}[n] = \sum_{m=-\infty}^{\infty} s[m+n]h[m] \quad (1.41)$$

Correlation of Complex Signals

Correlation between two complex signals $s[n]$ and $h[n]$ can be understood through writing Eq (1.38) in *IQ* form. However, another difference from convolution is that one signal is conjugated as

$$\begin{aligned}\text{corr}[n] &= s[n] \heartsuit h[n] \\ &= \sum_{m=-\infty}^{\infty} s[m]h^*[m-n]\end{aligned} \quad (1.42)$$

where conjugate of a signal, defined in Section 1.3, negates the *Q* arm of $h[m - n]$. We shortly see the need to introduce the conjugate operation in the second signal.

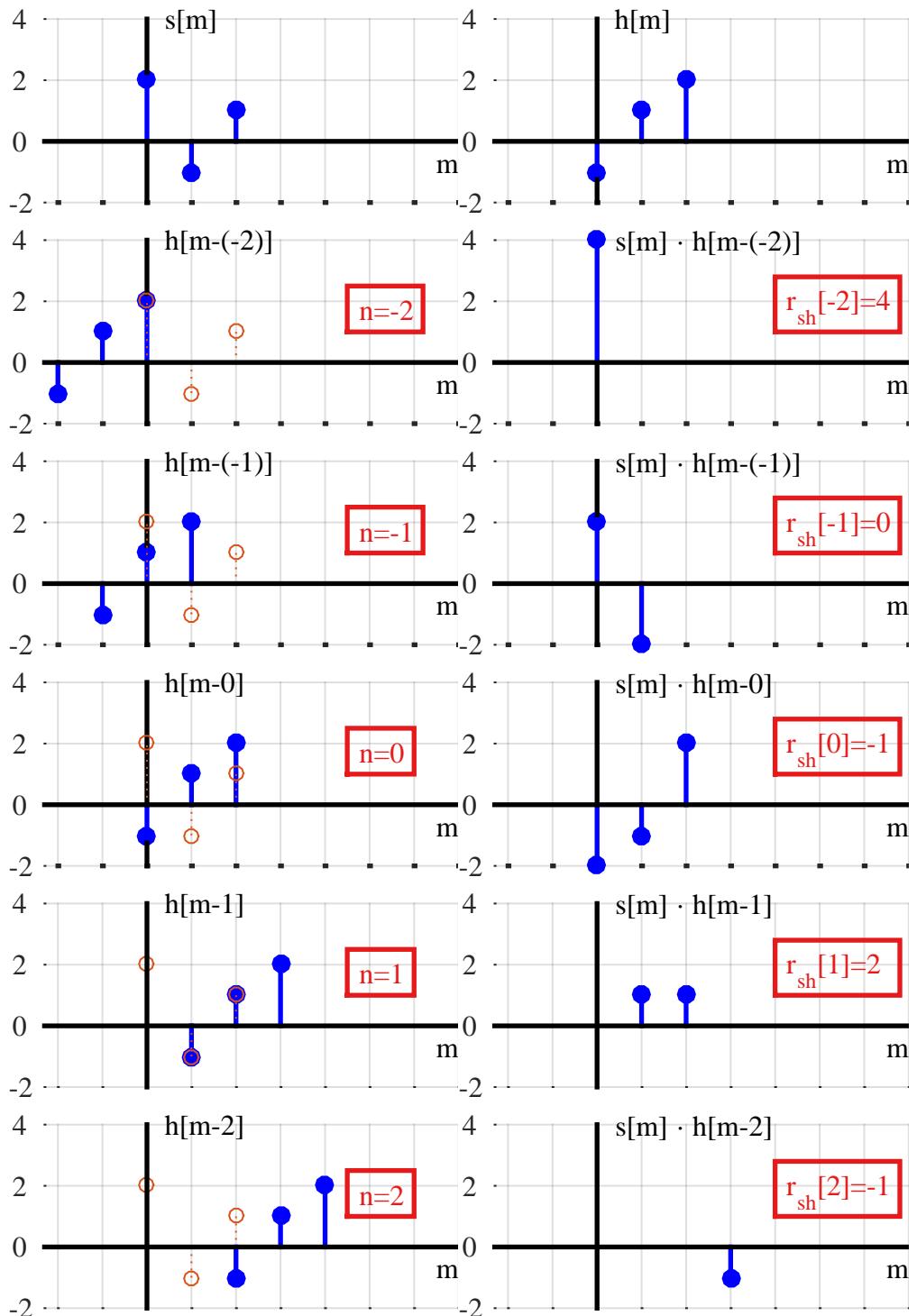


Figure 1.43: Correlation between $s[n]$ and $h[n]$

The above equation can be decomposed through utilizing the multiplication rule of complex numbers.

$$\begin{array}{ll} I \rightarrow & \text{corr}_I[n] = s_I[n] \heartsuit h_I[n] + s_Q[n] \heartsuit h_Q[n] \\ Q \uparrow & \text{corr}_Q[n] = s_Q[n] \heartsuit h_I[n] - s_I[n] \heartsuit h_Q[n] \end{array} \quad (1.43)$$

The actual computations can be written as

$$\begin{array}{ll} I \rightarrow & \text{corr}_I[n] = \sum_{m=-\infty}^{\infty} s_I[m]h_I[m-n] + \sum_{m=-\infty}^{\infty} s_Q[m]h_Q[m-n] \\ Q \uparrow & \text{corr}_Q[n] = \sum_{m=-\infty}^{\infty} s_Q[m]h_I[m-n] - \sum_{m=-\infty}^{\infty} s_I[m]h_Q[m-n] \end{array} \quad (1.44)$$

Due to the identity $\cos A \cos B + \sin A \sin B = \cos(A - B)$, a positive sign in I term indicates that phases of the two aligned-axes terms are actually getting subtracted. Obviously, the identity applies in above equations only if magnitude can be extracted as common term, but the concept of phase-alignment still holds. Similarly, the identity $\sin A \cos B - \cos A \sin B = \sin(A - B)$ implies that phases of the two cross-axes terms are also getting subtracted in Q expression. Hence, *a complex correlation can be described as a process* that

- computes 4 real correlations: $I \heartsuit I$, $Q \heartsuit Q$, $Q \heartsuit I$ and $I \heartsuit Q$
- *subtracts* by anti-aligning the 2 aligned-axes correlations ($I \heartsuit I + Q \heartsuit Q$) to obtain the I component
- subtracts the 2 cross-axes correlations ($Q \heartsuit I - I \heartsuit Q$) to obtain the Q component.

Now it can be inferred why a conjugate was required in the definition of complex correlation. The purpose of correlation is to extract the degree of similarity between two signals, and whenever A is close to B ,

$$\begin{aligned} \cos(A - B) &\approx 1 \\ \sin(A - B) &\approx 0 \end{aligned} \quad (1.45)$$

thus maximizing the correlation output. In words, the introduction of conjugate in the definition of correlation makes sense if we think of correlation between two copies of the same signal. Then, a negation in one's phase cancels out the other and perfectly aligns the signal components for maximum similarity.

1.7 Discrete Frequency

Having covered the transition from continuous to discrete time, a natural extension is to understand the notions of time – and later, frequency – in a discrete setting.

Discrete Time and the Sample Rate

An Analog to Digital Converter (ADC) samples a continuous-time signal to produce discrete-time samples. For a digital signal processor, this signal just resides in memory as a sequence of numbers. Consequently, the knowledge of the sample rate F_S is the key to signal manipulation in digital domain.

As far as time is concerned, one can easily determine the period or frequency of such a signal stored in the memory. For example, the period T in the sinusoid of Figure 1.44 is clearly 10 samples and sample time $T_S = 1/F_S$ can be employed to find its period in seconds.

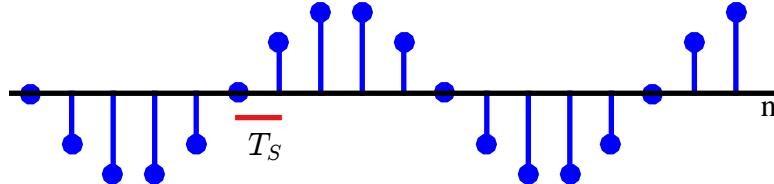


Figure 1.44: A discrete-time sinusoid

- For a sample rate of $F_S = 10 \text{ Hz}$, $T_S = 0.1 \text{ seconds}$, so

$$T = 10 \frac{\text{samples}}{\text{period}} \cdot 0.1 \frac{\text{seconds}}{\text{sample}} = 1 \text{ second}$$

and its frequency $F = 1/T = 1 \text{ Hz}$.

- For a sample rate of $F_S = 500 \text{ Hz}$, $T_S = 0.002 \text{ seconds}$, so

$$T = 10 \frac{\text{samples}}{\text{period}} \cdot 0.002 \frac{\text{seconds}}{\text{sample}} = 0.02 \text{ seconds}$$

with frequency $F = 1/T = 50 \text{ Hz}$.

As explained before, the samples of both sinusoids will be stored in memory as a sequence of numbers with no difference in discrete domain. Next, we turn our attention to the discrete frequency domain.

Discrete Frequency and the Sample Rate

Remember that the reason we work with discrete-time signals is that the finite memory of a processor can store only a fixed number of time values. Similarly, this finite memory can also store only a fixed number of frequency values and not an infinite range of F values.

For this reason, while we are at sampling in time domain, we also want to sample the frequency domain. Assume that a total of N samples are collected in time domain at a rate of $F_S = 1/T_S$, thus spanning a time duration of NT_S seconds. Then, the lowest frequency that can be represented by these N samples is the one by a sinusoid that completes *one full cycle* – and no more – during this interval of NT_S seconds. Consequently, its frequency is given by $1/(NT_S) \text{ Hz}$ and expressed as

$$\begin{aligned} I \rightarrow V_I[n] &= \cos 2\pi \frac{1}{NT_S} t \Big|_{t=nT_S} = \cos 2\pi \frac{1}{NT_S} nT_S = \cos 2\pi \frac{1}{N} n \\ Q \uparrow V_Q[n] &= \sin 2\pi \frac{1}{NT_S} t \Big|_{t=nT_S} = \sin 2\pi \frac{1}{NT_S} nT_S = \sin 2\pi \frac{1}{N} n \end{aligned}$$

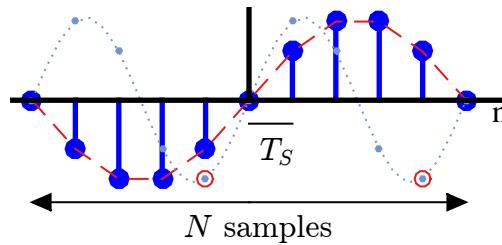


Figure 1.45: Q part of two discrete-time sinusoids with N samples, one with discrete frequency $1/N$ and the other with $2/N$

Hence, $1/N$ is the discrete frequency of this sinusoid whose Q part is drawn in Figure 1.45.

Now consider the equation

$$\frac{1}{NT_S} = \frac{F_S}{N} = F_S \cdot \frac{1}{N}$$

and observe the following.

- While the actual frequency is $F_S(1/N)$, the discrete frequency is $1/N$.
- Frequency resolution, determined by the lowest frequency that can be represented in such a discrete setting, is given by F_S/N .
- Viewed as $F_S \cdot 1/N$, we can get more discrete frequency samples that are integer multiples of $1/N$.

$$0, \frac{1}{N}, \frac{2}{N}, \dots$$

As a consequence, the frequencies of all such complex sinusoids are integer multiples of the fundamental frequency $1/N$. An example of Q component for $2/N$ is drawn in the background of Figure 1.45 with a dotted line that exhibits 2 periods within those N samples, indicating 2 cycles/ N samples or $2/N$ cycles/sample.

Extending this concept further, the complex sinusoids whose discrete frequencies are k/N complete k cycles in an interval of N samples (or k/N cycles per sample).

Orthogonality

Let us call the sinusoids with frequencies $1/N$ and $2/N$ as A and B , respectively. As we see now, they are orthogonal to each other. Orthogonality implies that their correlation computed at shift 0 is zero. In other words, the sum of their sample-by-sample products is zero. For a real sinusoid,

$$\sum_{n=0}^{N-1} \sin 2\pi \frac{1}{N} n \cdot \sin 2\pi \frac{2}{N} n = 0 \quad (1.46)$$

This can be verified from Figure 1.45. Observe that B has 2 periods within N samples. In its first period, the products of its samples are taken with negative samples of A . In

its second period, the same products are taken with positive samples of A with exactly the same magnitude. In this way, this sum turns out to be zero.

This is evident from circles drawn around two of its samples in Figure 1.45. Verify that the corresponding samples of the sinusoid A have opposite signs. Since the original workhorses of DSP are the complex sinusoids and not the real ones, we now analyze orthogonality in that context.

In Section 1.6, we found that the expression for correlation of two complex signals involves conjugation of the second signal. Therefore, to examine this correlation at shift 0 (i.e., orthogonality), we need to apply the multiplication rule of complex signals: due to the conjugation of the second signal, the I term of the result is $I \cdot I + Q \cdot Q$ and the Q term of the result is $Q \cdot I - I \cdot Q$. For two complex sinusoids with discrete frequencies k/N and k'/N , this correlation at lag 0 is

$$\begin{aligned} I &\rightarrow \sum_{n=0}^{N-1} \left[\cos 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k'}{N} n + \sin 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k'}{N} n \right] \\ Q &\uparrow \sum_{n=0}^{N-1} \left[\sin 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k'}{N} n - \cos 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k'}{N} n \right] \end{aligned}$$

Using the identities $\cos A \cos B + \sin A \sin B = \cos(A - B)$ and $\sin A \cos B - \cos A \sin B = \sin(A - B)$,

$$\begin{aligned} I &\rightarrow \sum_{n=0}^{N-1} \cos 2\pi \frac{k-k'}{N} n = \begin{cases} N & k = k' \\ 0 & k \neq k' \end{cases} \\ Q &\uparrow \sum_{n=0}^{N-1} \sin 2\pi \frac{k-k'}{N} n = 0 \end{aligned} \quad (1.47)$$

where we have used the fact the sum of samples of $\cos(\cdot)$ and $\sin(\cdot)$ above is zero due to an integer number of periods within N samples. Also, $\cos 0 = 1$ when $k = k'$ that adds up to N . This concept is illustrated in Figure 1.46.

We have proved the following result.

All complex sinusoids having frequencies as integer multiples of a fundamental frequency $F_S(1/N)$ are orthogonal to each other.

We now move towards drawing this discrete frequency axis.

Discrete Frequency Axis

With N discrete time domain samples, are there infinite complex sinusoids out there that are orthogonal to each other? The answer is no and we claim that their number is only N . To verify, let us explore one such option with a discrete frequency $N/N = 1$.

$$\sin 2\pi \frac{N}{N} n = \sin 2\pi n = 0$$

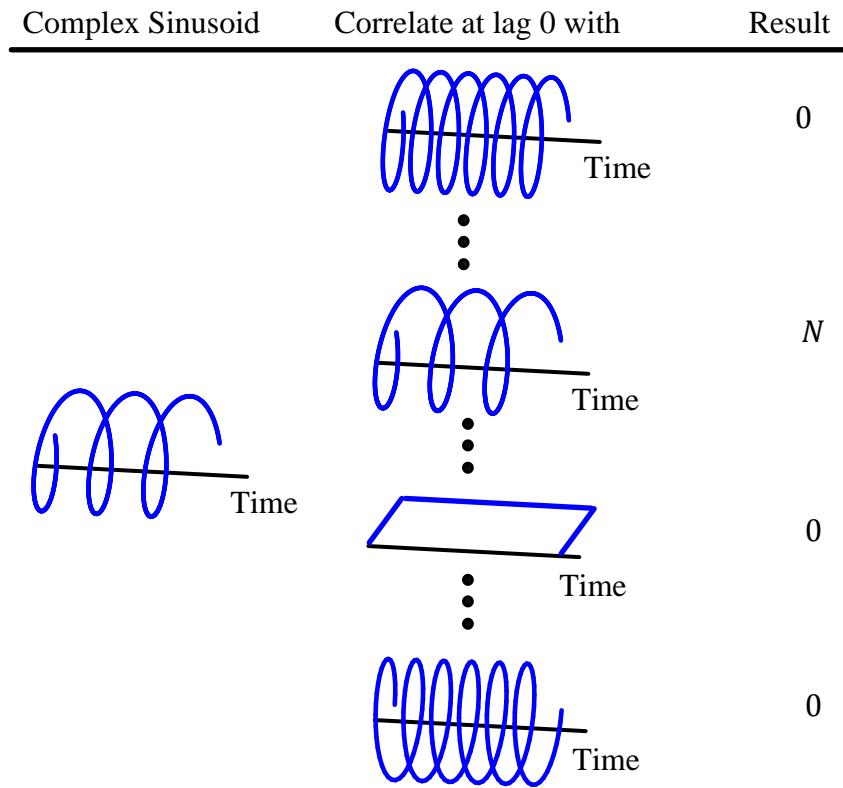


Figure 1.46: In N samples, a complex sinusoid with discrete frequency k/N is orthogonal to any other complex sinusoid with discrete frequency $k' \neq k$

Hence, a complex sinusoid with discrete frequency 0 is the same as the one with discrete frequency 1. For the next candidate $(N+1)/N$, use $\sin(A+B) = \sin A \cos B + \cos A \sin B$.

$$\begin{aligned} \sin 2\pi \frac{N+1}{N} n &= \sin 2\pi n \cos 2\pi \frac{1}{N} n + \cos 2\pi n \sin 2\pi \frac{1}{N} n \\ &= \sin 2\pi \frac{1}{N} n \end{aligned}$$

because $\sin 2\pi n = 0$ and $\cos 2\pi n = 1$. We conclude that there are only N distinct frequencies from 0 to $N-1$ (or $-N/2$ to $N/2-1$ as we will shortly see). Onwards, they just repeat themselves.

Interestingly, we learned that for N samples in discrete time domain, there are N samples in the discrete frequency domain as well! These N samples represent discrete frequencies of complex sinusoids that are integer multiples of one fundamental frequency $1/N$ and hence are all orthogonal to each other.

From N discrete frequencies found above, we could construct a discrete frequency axis with the following members.

$$0, \frac{1}{N}, \dots, \frac{N-1}{N}$$

which are exactly the same as

$$-1, -1 + \frac{1}{N}, \dots, \frac{-1}{N}$$

due their periodicity. However, it is due to the sampling theorem that we choose to work with an axes centered around 0. It was shown in Eq (1.36) of Section 1.5 that the unique range of continuous frequency F after sampling a signal is $-0.5F_S \leq F < +0.5F_S$, or

$$-0.5 \leq \frac{F}{F_S} < +0.5 \quad (1.48)$$

Since there are N discrete frequencies, $-0.5 \leq F/F_S < +0.5$ is divided into N equal intervals (assuming N is even) by sampling at instants

$$-0.5, -0.5 + \frac{1}{N}, \dots, -\frac{1}{N}, 0, \frac{1}{N}, \dots, +0.5 - \frac{1}{N} \quad (1.49)$$

to obtain the discrete frequency axis. Obviously, the discrete frequency resolution is $1/N$. So *discrete frequency* is basically the spectral content in the baseband sampled at N equally spaced frequency points (the term $+0.5$ is absent because it is the same as -0.5 owing to the axis periodicity).

Eq (1.49) can also be written as

$$\frac{-N/2}{N}, \frac{-N/2+1}{N}, \dots, -\frac{1}{N}, 0, \frac{1}{N}, \dots, \frac{N/2-1}{N} \quad (1.50)$$

Consequently, the index k of discrete frequency axis is given by $[-N/2, N/2 - 1]$, or

$$\begin{aligned} k &= -\frac{N}{2}, -\frac{N}{2} + 1, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1 \\ \frac{k}{N} &= -0.5, -0.5 + \frac{1}{N}, \dots, -\frac{1}{N}, 0, +\frac{1}{N}, \dots, +0.5 - \frac{1}{N} \end{aligned}$$

This is drawn in Figure 1.47. Comparing the above equation with Eq (1.48), we arrive at the relation between discrete frequency k/N and continuous frequency F as

$$\frac{k}{N} = \frac{F}{F_S} \quad (1.51)$$

The units of discrete frequency k/N from Eq (1.51) are

$$\frac{\text{cycles}}{\text{second}} \div \frac{\text{samples}}{\text{second}} = \frac{\text{cycles}}{\text{sample}}$$

Note 1.8 Discrete frequency axis

Eq (1.51) is one of the two most fundamental relations in digital signal processing, the other being the sampling theorem Eq (1.37). These are the two interfaces between continuous and discrete worlds.

Establishing the relationship between continuous and discrete frequencies gives the

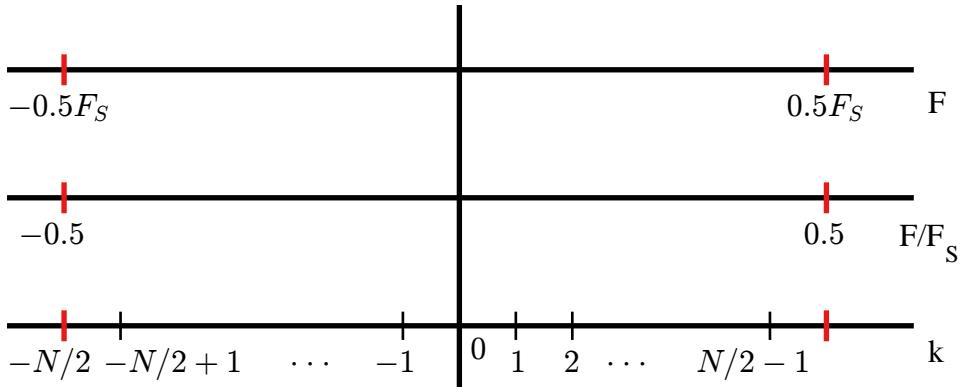


Figure 1.47: Sampling the axis in frequency domain

frequency domain as a sequence of numbers stored in a processor memory.

When in doubt about continuous and discrete frequency domains, refer to Eq (1.51)!

If the discrete frequency k/N and the sample rate F_S are known, the actual continuous frequency can be found from Eq (1.51).

$$F = F_S \cdot \frac{k}{N}$$

For example, with $F_S = 3$ kHz and $N = 32$,

$$\begin{aligned} k = 0 &\rightarrow F = 3000 \cdot \frac{0}{32} = 0 \text{ Hz} \\ k = 1 &\rightarrow F = 3000 \cdot \frac{1}{32} = 93.75 \text{ Hz} \\ k = 2 &\rightarrow F = 3000 \cdot \frac{2}{32} = 187.5 \text{ Hz} \end{aligned}$$

Each k is therefore called a *frequency bin* and the value N determines the number of input samples and the resolution of discrete frequency domain. Understanding Eq (1.37) and Eq (1.51) will make the further concepts much easier to grasp.

In conclusion, the range of unique discrete frequencies k/N is

$$-0.5 \leq \frac{k}{N} < +0.5 \quad (1.52)$$

This set of complex sinusoids is drawn in Figure 1.48 highlighting the discrete frequency index k . The negative indices of k illustrate the clockwise direction of rotation while the positive indices of k point towards the anticlockwise rotation. Observe that the rate of oscillation in discrete-time signals decreases with k/N going from -0.5 to 0 and increases from 0 to $+0.5$, remembering that $k/N = -0.5$ is the same as $k/N = +0.5$.

Keep this figure in mind for most of the applications of signal processing techniques as they represent a complete discrete frequency set. And it will help you in surprising ways!

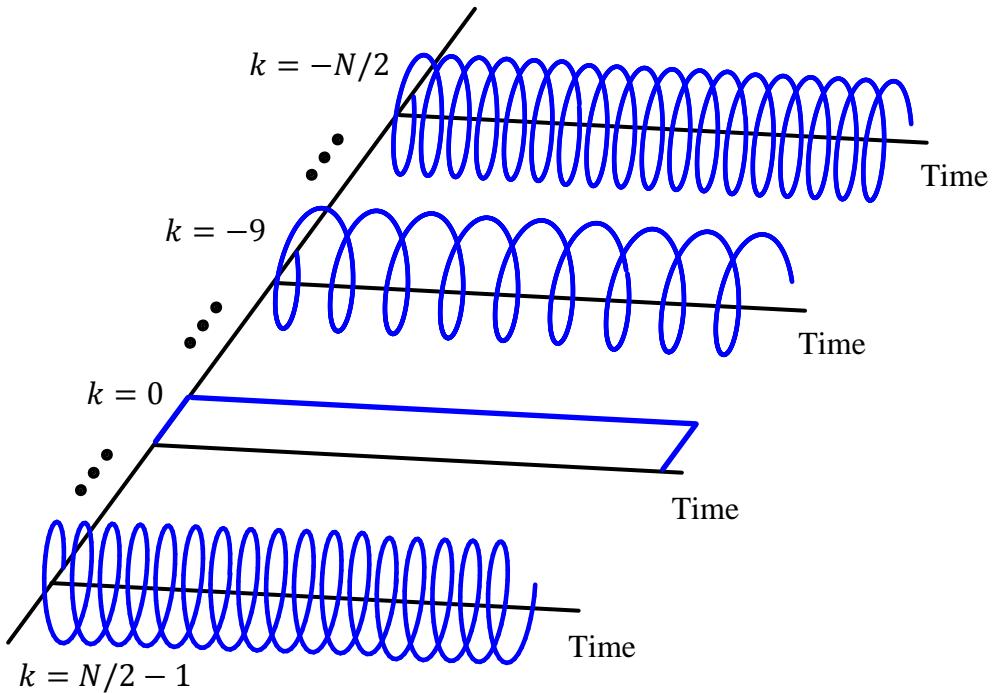


Figure 1.48: Complex sinusoids drawn to highlight the discrete frequency axis k on the left side

Some people are more comfortable with 2D figures instead of 3D ones. So this set of complex sinusoids with both I and Q components is shown in Figure 1.49 for $N = 8$.

Following are a few key observations in this figure.

- For $k = 0$, the I sinusoid is 1 because $\cos 0 = 1$, while Q sinusoid is 0 because $\sin 0 = 0$.
- Just like $k = 1$ implies that the I and Q waveforms, $\cos 2\pi \cdot 1/N$ and $\sin 2\pi \cdot 1/N$, complete one full cycle during $N = 8$ samples, each complex sinusoid with discrete frequency k/N spans k complete cycles in an interval of N samples.
- For negative values of k , I sinusoids remain unchanged while Q sinusoids change sign. In the context of complex sinusoids, this follows from the definition of negative frequencies as the ones with clockwise rotation.
- Finally, to understand what the units of discrete frequency ‘cycles/sample’ mean, consider for example the case of $k = 2$ in this figure. Observe that for $N = 8$, we have the discrete frequency equal to $k/N = 2/8 = 1/4$ cycles/sample. Notice that this sinusoid completes a quarter cycle from one sample to the next and hence the frequency of $1/4$ cycles/sample.

Next, we consider an example to highlight the impact of aliasing on a discrete-time signal and why it is not necessary for all continuous frequencies to translate into corresponding discrete frequencies.

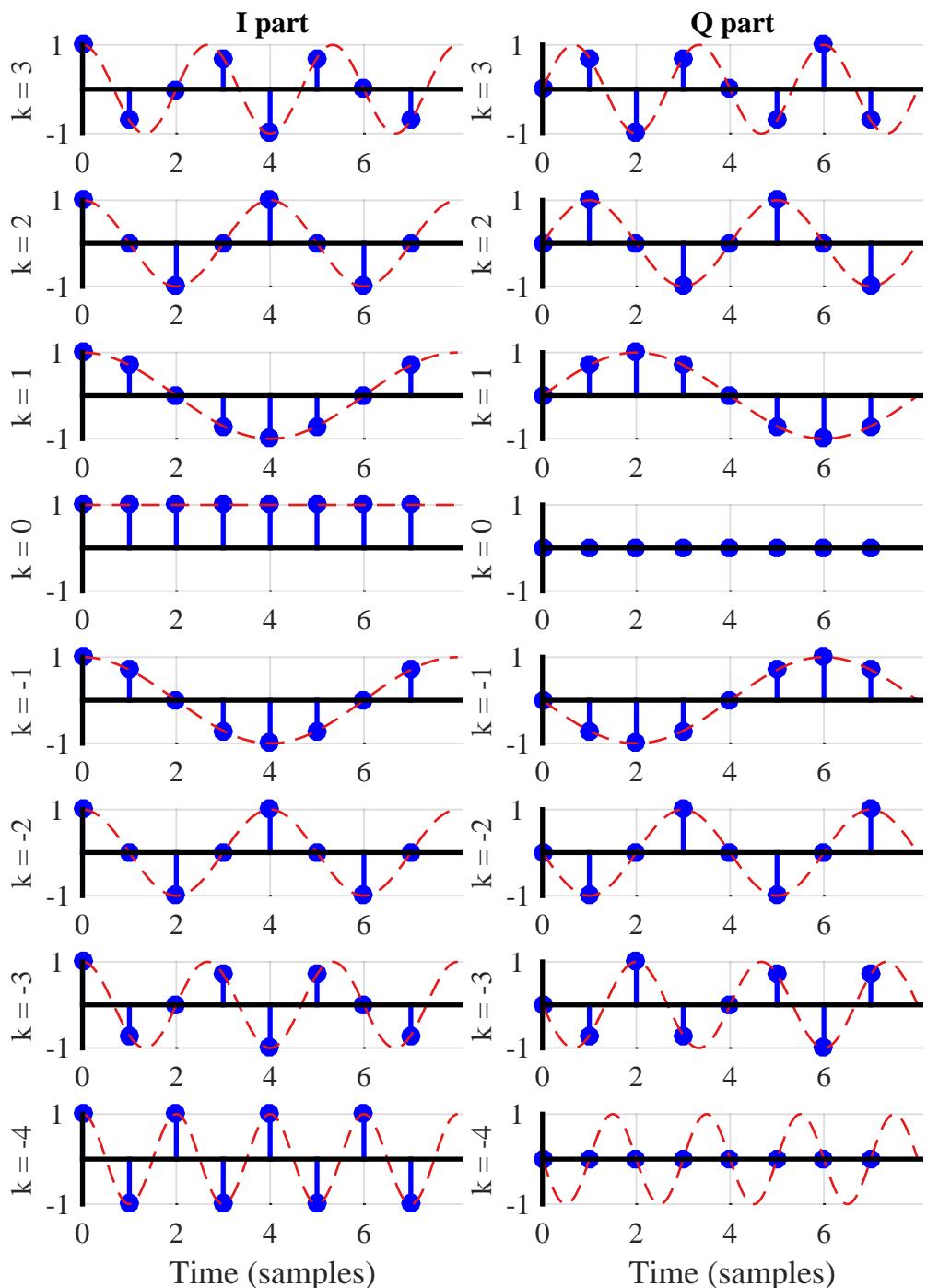


Figure 1.49: $N = 8$ complex sinusoids, the frequencies of which form ‘ticks’ on the discrete frequency axis

Example 1.4

Consider a signal

$$s(t) = 2 \cos(2\pi 1000t) + 7 \sin(2\pi 3000t) + 3 \cos(2\pi 6000t)$$

It is clear that continuous frequencies present in this signal are $F_1 = 1$ kHz, $F_2 = 3$ kHz and $F_3 = 6$ kHz. Since the maximum frequency is 6 kHz, the sampling theorem gives the Nyquist rate as 2×6 kHz = 12 kHz.

Now suppose that this signal is sampled at $F_S = 5$ kHz, the folding frequency is then given by $0.5F_S = 2.5$ kHz. Sampling the signal at equal intervals of $t = nT_S$

$$\begin{aligned} s[n] &= s(t)|_{t=nT_S} = s\left(\frac{n}{F_S}\right) \\ &= 2 \cos 2\pi \frac{1}{5}n + 7 \sin 2\pi \frac{3}{5}n + 3 \cos 2\pi \frac{6}{5}n \\ &= 2 \cos 2\pi \frac{1}{5}n + 7 \sin 2\pi \left(1 - \frac{2}{5}\right)n + 3 \cos 2\pi \left(1 + \frac{1}{5}\right)n \end{aligned}$$

where the adjustment is done to bring all discrete frequencies within the range -0.5 to 0.5 . Next,

$$\begin{aligned} s[n] &= 2 \cos 2\pi \frac{1}{5}n + 7 \sin 2\pi \left(-\frac{2}{5}\right)n + 3 \cos 2\pi \left(\frac{1}{5}\right)n \\ &= 5 \cos 2\pi \frac{1}{5}n - 7 \sin 2\pi \frac{2}{5}n \end{aligned}$$

If this signal is reconstructed in continuous-time domain, the frequencies present are $k/N = 1/5$ and $k/N = -2/5$. From Eq (1.51), these appear as continuous frequencies at $F_1 = 5 \times 1/5 = 1$ kHz and $5 \times -2/5 = -2$ kHz. Since only 1 kHz is less than $0.5F_S$ and hence within aliasing-free range, it is still there after sampling. The other two are above the folding frequency and hence aliased to $F_2 - F_S = 3 - 5 = -2$ kHz and $F_3 - F_S = 6 - 5 = 1$ kHz.

Two final remarks are in order here.

- Dealing with complex numbers through an *IQ* notation has several advantages. For example, it can avoid the standard complex expressions like $\exp(j\theta)$ at a cost of additional maths. More importantly, it describes how the mathematical operations are implemented in actual electronic circuits. Also, it reminds us how phase is equally important in signal analysis.
- In this text, every time I write the expression involving a discrete frequency, I express it either as $2\pi \frac{k}{N}n$ or $2\pi(k/N)n$, instead of $2\pi kn/N$ that is used in most texts. The former clearly indicates the presence of a discrete frequency k/N analogous to the variable F in $2\pi Ft$, while it is easy to lose this meaning in the latter.

1.8 The Discrete Fourier Transform (DFT)

Learned in the previous sections, the following three important concepts take us to the core of the Discrete Fourier Transform (DFT) idea.

- Regardless of the signal shape, most signals of practical interest can be considered as a sum of complex sinusoids oscillating at different frequencies.
- A set of N orthogonal complex sinusoids can be constructed within a span of N time domain samples.
- Each ‘tick’ or bin on the discrete frequency axis denotes the discrete frequency k/N of such a complex sinusoid.

To understand how a set of sinusoids with N discrete frequencies can sum up to a discrete-time signal of an arbitrary shape, we gave an example in Section 1.4 of the 3 dimensions of space, namely x , y and z . Any point in space can be represented with a combination of just these 3 basis components. In the same section, we also studied different colours arising from the participation of just three basic colours: Red, Green and Blue.

The question is: if we are given N samples of a discrete-time signal, how do we know the amplitude and phase contribution from each complex sinusoid in making up that signal? In the context of signal analysis, it turns out that the *Discrete Fourier Transform (DFT)* is the tool that finds out the frequency domain amplitudes and phases of N complex sinusoids present in a discrete-time signal of length N samples.

Note 1.9 The white light

In 1672, Newton demonstrated that the sunlight can be broken by a prism into 7 constituent colours of the rainbow. Assuming roughly the same intensity of each colour, therefore, a prism is a tool that tells us the intensity of each colour present in the white light. This is shown in Figure 1.50.

Next, imagine a tool that takes a colour input and tells us how much of RGB contribution is there. Referring to RGB Figure 1.34 for example, it can inform us that

$$\text{Yellow} = \begin{cases} 100\% & \text{Red} \\ 100\% & \text{Green} \\ 0\% & \text{Blue} \end{cases}$$

Or

$$\text{Grey} = \begin{cases} 50\% & \text{Red} \\ 50\% & \text{Green} \\ 50\% & \text{Blue} \end{cases}$$

Similarly, the DFT finds in a discrete-time signal the *amplitude and phase contributions* $S[k]$ from each of the N discrete-time complex sinusoids. These reference sinusoids given by k/N are then called *analysis frequencies*. See how DFT decomposes the signal and outputs their respective contributions in Figure 1.50.

As an example, a continuous-time square wave was shown in Figure 1.35 to be a summation of sinusoidal waves of different amplitudes. The DFT of such a square or rectangular signal is a particular function known as a sinc signal (as we will see later) and the DFT would have identified the amplitudes of each constituent frequency, e.g., 1, 1/3, 1/5, and so on depicted in Eq (1.29). Let us now turn towards a formal definition of the DFT.

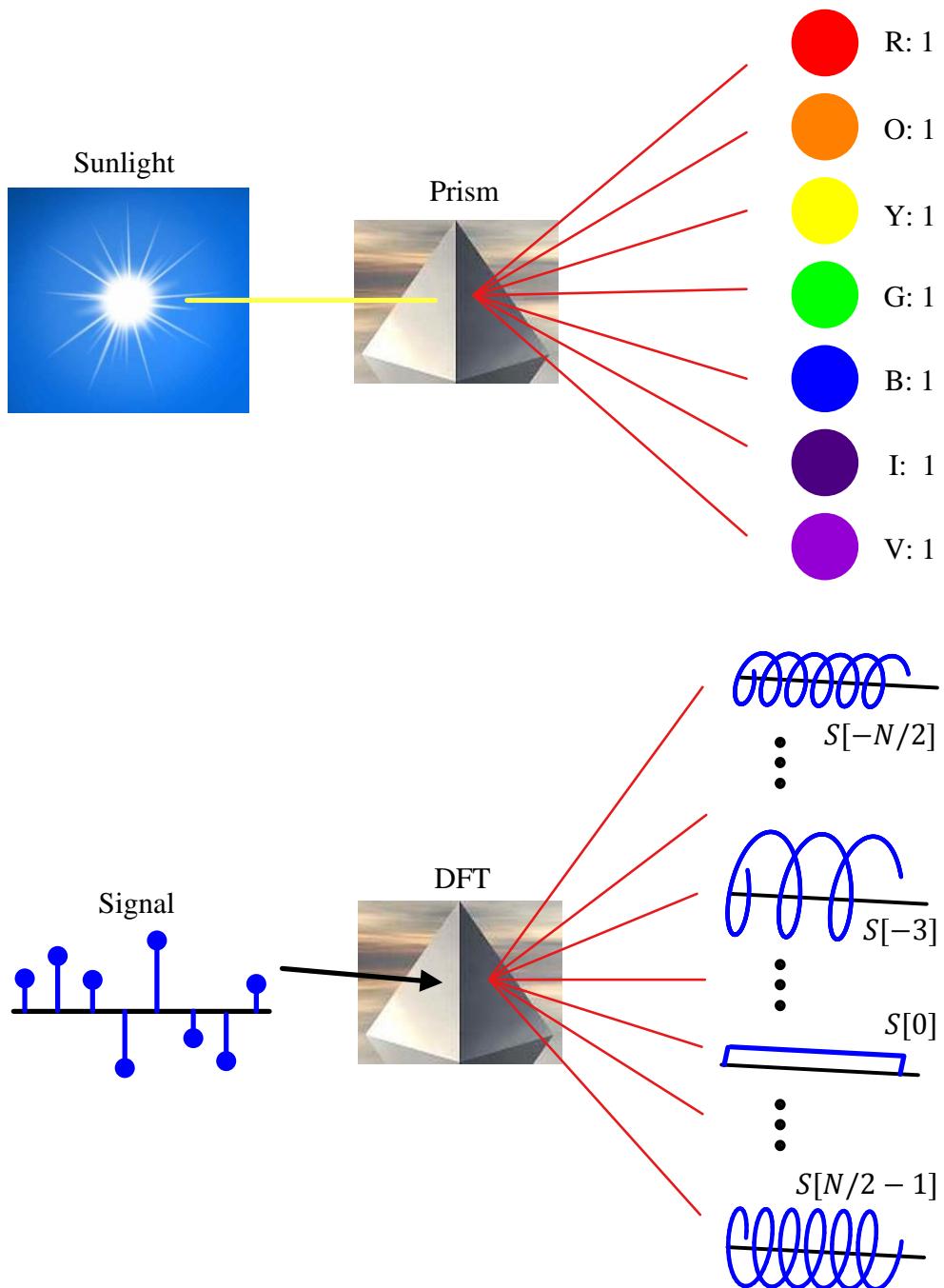


Figure 1.50: A DFT is like a prism

Definition

From the previous sections, let a discrete-time complex sinusoid be defined as $V[n]$.

$$\begin{array}{ll} I \rightarrow & V_I[n] = \cos 2\pi \frac{k}{N} n \\ Q \uparrow & V_Q[n] = \sin 2\pi \frac{k}{N} n \end{array}$$

What we actually want to do is correlate (at lag 0) a finite-length signal $s[n]$ with N such complex sinusoids with frequencies k/N . From correlation in Section 1.6, it suffices to say that we multiply $s[n]$ with the complex sinusoid $V^*[n]$ sample-by-sample and take the sum. Then, the DFT $S[k]$ is given by

$$S[k] = \sum_{n=0}^{N-1} s[n] V^*[n]$$

In terms of $s[n]$ with I and Q components $s_I[n]$ and $s_Q[n]$, the DFT $S[k]$ can be found through the application of multiplication rule of complex numbers.

$$\begin{array}{ll} I \rightarrow & S_I[k] = \sum_{n=0}^{N-1} \left[s_I[n] \cos 2\pi \frac{k}{N} n + s_Q[n] \sin 2\pi \frac{k}{N} n \right] \\ Q \uparrow & S_Q[k] = \sum_{n=0}^{N-1} \left[s_Q[n] \cos 2\pi \frac{k}{N} n - s_I[n] \sin 2\pi \frac{k}{N} n \right] \end{array} \quad (1.53)$$

for each k . In the above equation[†],

- k is the index of DFT output in frequency domain, $k = -N/2, \dots, -1, 0, 1, \dots, N/2 - 1$
- $S[k]$ is k^{th} DFT output component
- $S_I[k]$ and $S_Q[k]$ are I and Q components of the DFT $S[k]$
- n is the time domain index of input samples, $n = 0, 1, \dots, N - 1$
- $s[n]$ is the sequence of input samples
- N is the number of data samples in discrete time domain and number of bins in discrete frequency domain

Observe from the definition that each $S[k]$ output is computing the correlation at lag 0 between an input signal and a complex sinusoid whose frequency is k complete cycles in an interval of N samples (or k/N cycles per sample). Due to orthogonality, the correlation with all other $k' \neq k$ is zero, popping out the contribution in the signal only from the complex sinusoid with frequency k/N .

[†]In complex notation, this DFT is defined as

$$S[k] = \sum_{n=0}^{N-1} s[n] \exp \left(-j2\pi \frac{k}{N} n \right)$$

On the other hand, comparing with the phase rotation rule in Eq (1.19), we can observe that DFT operation is rotating a signal $s[n]$ clockwise.

$$\begin{aligned} I \cdot \cos(\cdot) + Q \cdot \sin(\cdot) \\ Q \cdot \cos(\cdot) - I \cdot \sin(\cdot) \end{aligned}$$

But what exactly does this mean?

Note 1.10 Beauty of correlation and orthogonality

Remember that frequency is defined as the rate of change of phase (sample-by-sample here). If $s[n]$ contains a component of frequency $+k/N$, sample-by-sample de-rotation by a complex sinusoid of the same frequency but opposite direction (i.e., $-k/N$) results in an output as a single number due to the corresponding phase canceling out at each step. That single number is naturally the spectral magnitude of the complex sinusoid with frequency k/N , an indicator of how much that sinusoid contributes to the construction of $s[n]$.

This single number contains a constant phase term as well since the DFT complex sinusoid starts with phase 0 but not the contributor complex sinusoid in the input signal. The whole process eventually tries to find the participation of every such complex sinusoid in construction of the examined signal.

The relation between a signal $s[n]$ and its Fourier Transform $S[k]$ is usually represented as

$$s[n] \xrightarrow{\mathcal{F}} S[k]$$

As a reminder, for a sample rate of $F_S = 30$ kHz and 64 point DFT, the fundamental frequency of the sinusoids is $F_S \cdot \pm 1/N = \pm 30,000/64 = \pm 468.75$ Hz represented by $k = \pm 1$. The other analysis frequencies are integer multiples of the fundamental frequency as

- $F_S \cdot \pm 2/N = \pm 937.5$ Hz represented by $k = \pm 2$
- $F_S \cdot \pm 3/N = \pm 1406.3$ Hz represented by $k = \pm 3$

and so on. Each frequency represents one complex sinusoid, or equivalently two real sinusoids.

Due to the periodicity of discrete frequency axis, DFT can also be defined for frequency bins $k = [0, N - 1]$. However, the range $k = [-N/2, N/2 - 1]$ is more natural than $k = [0, N - 1]$ since the former corresponds with our notion of a real sinusoid constituting one positive frequency and one negative frequency complex sinusoid, see Eq (1.27). As an example, having two impulses in discrete frequency domain at $k = 5$ and $k = N - 5$ does not seem to convey much information. However, viewing it as two impulses at $k = 5$ and $k = -5$ immediately makes one realize that the time domain signal is a real sinusoid with frequency $F_S \cdot 5/N$.

Inverse Discrete Fourier Transform (iDFT)

Just like DFT is used to convert a signal from time to frequency domain, Inverse Discrete Fourier Transform (iDFT) is used as an inverse process by converting a signal

from frequency to time domain. iDFT is defined as

$$\boxed{\begin{aligned} I \rightarrow \quad s_I[n] &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[S_I[k] \cos 2\pi \frac{k}{N} n - S_Q[k] \sin 2\pi \frac{k}{N} n \right] \\ Q \uparrow \quad s_Q[n] &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[S_Q[k] \cos 2\pi \frac{k}{N} n + S_I[k] \sin 2\pi \frac{k}{N} n \right] \end{aligned}} \quad (1.54)$$

Notice that unlike DFT where the summation is over time index n , summation in iDFT is over frequency index k . Moreover, as opposed to DFT, the complex sinusoids here have a anticlockwise direction of rotation[†], see phase rotation rule in Eq (1.18). iDFT is also a correlation like DFT as explained earlier. The amplitude scaling is $1/N$ to balance the factor N that arises from the definition of orthogonality in Eq (1.47) and Figure 1.46.

In words, iDFT equation is simpler to understand than the DFT equation since it is founded on our original premise that a signal can be represented as a sum of properly scaled complex sinusoids. From the definition of iDFT, observe that to reconstruct a signal $s[n]$, we have to sum N complex sinusoids from $k = -N/2$ to $N/2 - 1$ after weighting them with DFT coefficients $S[k]$. This is the same concept illustrated in Figure 1.35 where the time domain signal was made up of many sinusoids. The only difference is that in the case of continuous-time signals, the number of these sinusoids can go up to infinity.

DFT Periodicity

A natural consequence of DFT equation above is that $S[k]$ can be proved to be periodic with period N . Notice that the complex sinusoids making up the DFT output are periodic in k because

$$\begin{aligned} I \rightarrow \quad \cos 2\pi \frac{k+N}{N} n &= \cos \left\{ 2\pi \frac{k}{N} n + 2\pi n \right\} = \cos 2\pi \frac{k}{N} n \\ Q \uparrow \quad \sin 2\pi \frac{k+N}{N} n &= \sin \left\{ 2\pi \frac{k}{N} n + 2\pi n \right\} = \sin 2\pi \frac{k}{N} n \end{aligned}$$

Since the discrete frequency axis is itself periodic, i.e., the frequency k/N and $(k+N)/N = k/N + 1$ are the same, the DFT $S[k]$ is the same as $S[k+N]$. Thus,

$$S[k+N] = S[k] \quad (1.55)$$

It is not a surprising result that the DFT is a periodic signal in frequency domain due to the following reasoning. When we sample a continuous-time signal to obtain a discrete-time signal, infinite spectral replicas emerge that are spaced at integer multiples of the sample rate F_S . When we obtain the discrete frequency axis by sampling

[†]In complex notation, this iDFT is defined as

$$s[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} S[k] \exp \left(+j2\pi \frac{k}{N} n \right)$$

the baseband from $-0.5F_S$ to $+0.5F_S$ at N equally spaced intervals, we are also implicitly sampling those spectral aliases and the DFT of a signal becomes periodic in frequency domain.

Note 1.11 Is DFT input periodic?

Interestingly, similar trigonometric identities used to find $S[k + N]$ can be utilized to obtain $s[n + N]$ through the definition of iDFT in Eq (1.54) as

$$s[n + N] = s[n] \quad (1.56)$$

Although the DFT of any signal $s[n]$ can be taken without any regard to periodicity, whether the DFT structure assumes $s[n]$ being periodic with a period N or not is a matter of debate in DSP community. Some consider it as a logical conclusion of sampling in frequency domain while some others have perfectly reasonable viewpoints of treating it as an aperiodic signal. Either approach is consistent with other insofar as the signal behaviour and interpretation of DFT output is concerned.

I personally consider the other approach more intuitive but it involves concepts like convolution and windowing which we have not covered yet and will encounter later. Since a common theme of this text is simplicity and periodic approach is simpler to follow, Eq (1.56) will be our guide. All it says is the following:

“Although we know nothing about what happens with $s[n]$ outside the range 0 to $N - 1$, sampling the frequency axis at N discrete points from $-N/2$ to $N/2 - 1$ creates replicas of $s[n]$ in time domain, just like sampling in time domain created replicas of its spectrum $S(F)$ in frequency domain.”

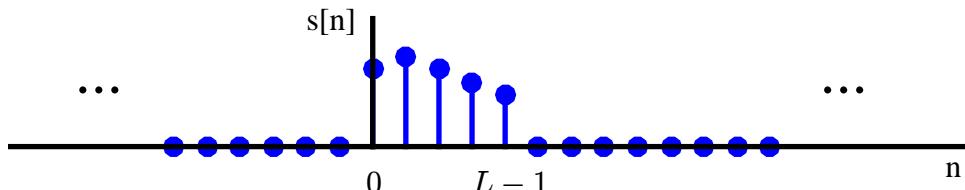
Figure 1.51 shows an aperiodic signal $s[n]$ and its periodic extension. As long as the DFT length N is greater or equal to the signal length L , the DFT accurately represents the sampled frequency response of the input signal and there is no distortion. Nevertheless, for $N < L$, the DFT spectrum is no longer its actual sampled frequency domain representation and there will be *aliasing in time domain*. Obviously, common sense also dictates that taking DFT of a truncated input signal ($N < L$) should produce inaccurate results.

A major implication of this periodic extension is that sequence shifts for DFT purpose actually result in circular shifts. Assume $N = L$ in Figure 1.51. Then, a right shift of 1 will knock the last sample at position $n = N - 1$ out of the observation window $[0, N - 1]$ to the new position $n = N$. Moreover, it will bring the last sample from its previous extension at $n = -1$ inside the observation window to $n = 0$. Conceptually, this is nothing but a circular shift within the window $[0, N - 1]$. This DFT observation window is shown as red dotted lines within 0 to $N - 1$.

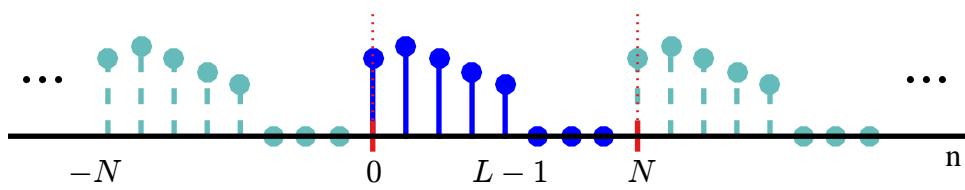
DFT Approximation of the Actual Spectrum

A natural question at this stage is: how well does the DFT – a sampled frequency function of a sampled time domain signal – approximates the underlying Fourier transform – a continuous frequency function of a continuous time domain signal? An initial guess is that it is not much accurate because the continuous Fourier transform is an integral from $-\infty$ to ∞ while the DFT is a finite sum.

The actual answer is that this approximation is quite well. However, there are two possible sources of error.



(a) Aperiodic signal $s[n]$ of length L



(b) Periodic extension of $s[n]$ for $N \geq L$

Figure 1.51: Sampling the frequency axis creates aliases in time domain

Truncation error: Clearly, any signal $s[n]$ that does not have a finite length must be truncated to accommodate finite length of the DFT.

Aliasing error: According to sampling theorem, the sample rate must be at least twice than the maximum frequency contained in $s[n]$. Aliasing can occur if the signal is not band-limited.

We discussed before that every real signal occupies an infinite amount of bandwidth, as a signal cannot be limited in both time and frequency domains. To avoid truncation errors, a signal must be time-limited which makes its spectrum unlimited causing aliasing errors. To avoid aliasing errors, a signal must be ideally band-limited with a sufficient sample rate, but such a signal cannot have finite duration causing truncation errors.

For many cases of practical interest, the amount of these errors is tolerable: with a reasonably band-limited signal that means having very small energy outside a certain frequency band, we can make the signal time-limited and sample it with an adequate sampling rate. That makes the DFT a great tool for analyzing real continuous-time signals.

Fast Fourier Transform (FFT)

A Fast Fourier Transform (FFT) is *neither* another type of Fourier transform, *nor* an approximation to the DFT. An FFT is just a faster method of computing the DFT of a signal by exploiting redundancy in DFT equations. In fact, it was FFT that made the Fourier analysis possible for majority of signal processing applications. For example, if FFT of a signal with length $N = 10^6$ takes 1 second on a computing device, the DFT on the same platform will take approximately 28 hours!

It is then no surprise that FFT is described as “the most important numerical algorithm of our lifetime”. In this text, we will not derive the exact FFT formulation and instead will use a software routine to compute the FFT when required. In Matlab and Python, for example, the command `fft(·)` serves this purpose.

1.9 Effect of Time Shift in Frequency Domain

One day my daughter Varda asked me, “How many hours have passed?” When I inquired about the start time, she had no idea.

Just like the zero of a measuring tape, a zero reference for time plays a crucial role in analyzing the signal behaviour in time and frequency domains. Until now, we assumed that reference time 0 coincides with the start of a sine and a cosine wave to understand the frequency domain. Later, we will deal with symbol timing synchronization problem in single-carrier systems and carrier frequency synchronization problem in multicarrier systems, both of which address the problem of finding this reference to prevent signal distortion. To solve those problems, we study the signals with arbitrary time shifts and want to know their effects on frequency domain behaviour.

To begin with, let us shift a cosine signal by $T/4$ to the right where T is its time period $T = 1/F$.

$$\cos 2\pi F \left(t - \frac{T}{4} \right) = \cos \left(2\pi F t - 2\pi F \frac{T}{4} \right) = \cos \left(2\pi F t - \frac{\pi}{2} \right) = \sin 2\pi F t$$

where a time shift of $-T/4$ is seen to impart a phase shift of $-\pi/2$ in frequency domain. This makes sense because if a full period T of the two complex sinusoids summing up in time domain to produce a cosine spans 360° , then a shift of $-1/4$ of 360° is -90° . This is illustrated in Figure 1.52.

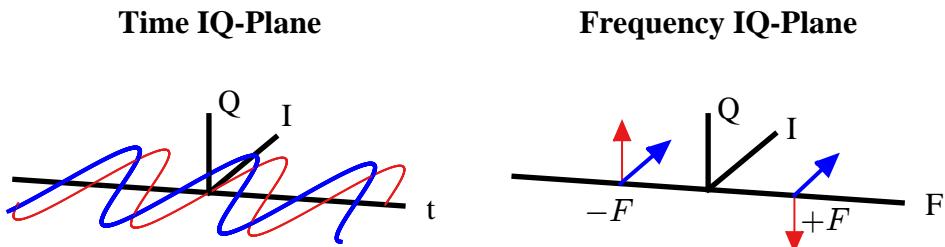


Figure 1.52: A cosine wave shifted by $T/4$ in time IQ becomes a sine wave. In frequency IQ -plane, the corresponding phase shift is $\pi/2$

This is why you often see a phase description for a real cosine wave as in Figure 1.53. Taken as the reference times, these phases are due to the orientation of the two complex sinusoids in frequency IQ -plane and present a more interesting view (although simple trigonometric formulas like $\cos 0 = 1$, $\cos \pi/2 = 0$ can also be applied to see this phase relationship).

Now evaluate a discrete-time complex sinusoid for a general time shift of n_0 samples. Since this is in context of the DFT, the shift here is circular even if not explicitly stated.

$$I \rightarrow \cos 2\pi \frac{k}{N} (n + n_0) = \cos \left(2\pi \frac{k}{N} n + 2\pi \frac{k}{N} n_0 \right)$$

$$Q \uparrow \quad \sin 2\pi \frac{k}{N} (n + n_0) = \sin \left(2\pi \frac{k}{N} n + 2\pi \frac{k}{N} n_0 \right)$$

There are two ways to look into this expression.

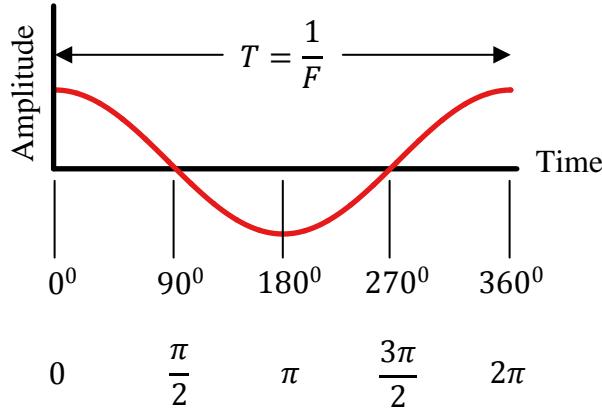


Figure 1.53: Phase description of a real cosine wave originates from phases of two complex sinusoids

Effect of Time Shift on DFT - Magnitude and Phase

This complex sinusoid can be seen to undergo a phase shift equal to $2\pi(k/N)n_0$. Since the DFT of a signal is just a combination of N such complex sinusoids with distinct values of k/N , the phase of the DFT *considering each k* becomes a linear function of discrete frequency k/N .

$$\text{phase shift} = 2\pi \frac{k}{N} n_0 = \text{constant} \cdot \frac{k}{N}$$

where the constant term, or slope, depends on the circular time shift n_0 . Also, the bigger the index k , the larger the phase shift is. Such a phase plot is drawn in Figure 1.54 for a positive n_0 .

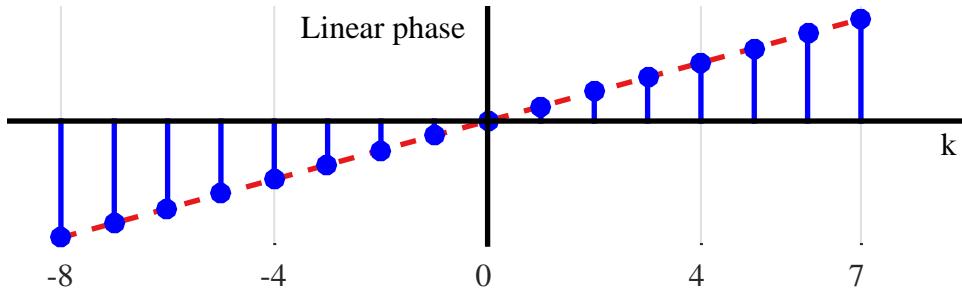


Figure 1.54: A phase plot varying linearly with frequency index k drawn for a positive n_0

In conclusion, the DFT of $\tilde{s}[n] = s[(n + n_0) \bmod N]$ has its magnitude unchanged. However its phase is rotated by $2\pi(k/N)n_0$ *for each k* . We denote the rotated DFT by $\tilde{S}[k]$.

$$\begin{aligned} |\tilde{S}[k]| &= |S[k]| \\ \angle \tilde{S}[k] &= \angle S[k] + 2\pi \frac{k}{N} n_0 \end{aligned} \tag{1.57}$$

We can summarize the above result as

$$\boxed{\text{Time shift } s[(n \pm n_0) \bmod N] \longrightarrow \pm 2\pi \frac{k}{N} n_0 \text{ Phase shift}} \quad (1.58)$$

This is the view mostly taken while describing the effect of a time shift and it serves well for the mathematical purpose. In my opinion, the effect of a time shift can be better visualized through *IQ* plots that follow next.

Effect of Time Shift on DFT - *I* and *Q*

A more interesting view comes from focusing on *I* and *Q* samples of the frequency domain. Let us shift an impulse signal $\delta[n]$ by n_0 . Plugging $s[n] = s_I[n] = \delta[(n + n_0) \bmod N]$ in the DFT definition in Eq (1.53) and using the fact that it is 1 at $n = -n_0$ and 0 everywhere else,

$$\begin{aligned} I &\rightarrow S_I[k] = \sum_{n=0}^{N-1} \cos 2\pi \frac{k}{N} (-n_0) = N \cos 2\pi \frac{k}{N} n_0 \\ Q &\uparrow S_Q[k] = \sum_{n=0}^{N-1} -\sin 2\pi \frac{k}{N} (-n_0) = N \sin 2\pi \frac{k}{N} n_0 \end{aligned}$$

for each k which represents a *frequency domain complex sinusoid* with amplitude N ! Let us explore it with the help of an example.

Example 1.5

Figure 1.55 shows a unit impulse signal $s[n]$ and its DFT $S[k]$ along with its circularly time shifted version $s[(n - 1) \bmod 5]$ and its DFT $\tilde{S}[k]$. This DFT is computed shortly in Section 1.10.3.

Note that phase shift for each frequency bin k is different for each k . To be exact, $\Delta\theta(k) = 2\pi(k/N) \cdot (-1)$. For $k = -2$ to $k = 2$ and $N = 5$, it turns out to be

$$\begin{aligned} k = 0 &\rightarrow \Delta\theta(0) = 2\pi \frac{0}{5} \cdot (-1) \times \frac{180^\circ}{\pi} = 0^\circ \\ k = \pm 1 &\rightarrow \Delta\theta(\pm 1) = 2\pi \frac{\pm 1}{5} \cdot (-1) \times \frac{180^\circ}{\pi} = \mp 72^\circ \\ k = \pm 2 &\rightarrow \Delta\theta(\pm 2) = 2\pi \frac{\pm 2}{5} \cdot (-1) \times \frac{180^\circ}{\pi} = \mp 144^\circ \end{aligned}$$

The phase rotations of 72° and 144° are illustrated in the figure. Also observe that for a right shift, the phase rotations are clockwise for positive k and anticlockwise for negative k .

To understand this frequency domain complex sinusoid, recall that a time domain complex sinusoid rotating at frequency F was shown to generate a helix in time *IQ*-plane. Such a complex sinusoid has a frequency domain representation that consists of a single impulse at $+F$. This is how we defined each ‘tick’ on the frequency axis.

Time domain complex sinusoid \longleftrightarrow Frequency domain impulse

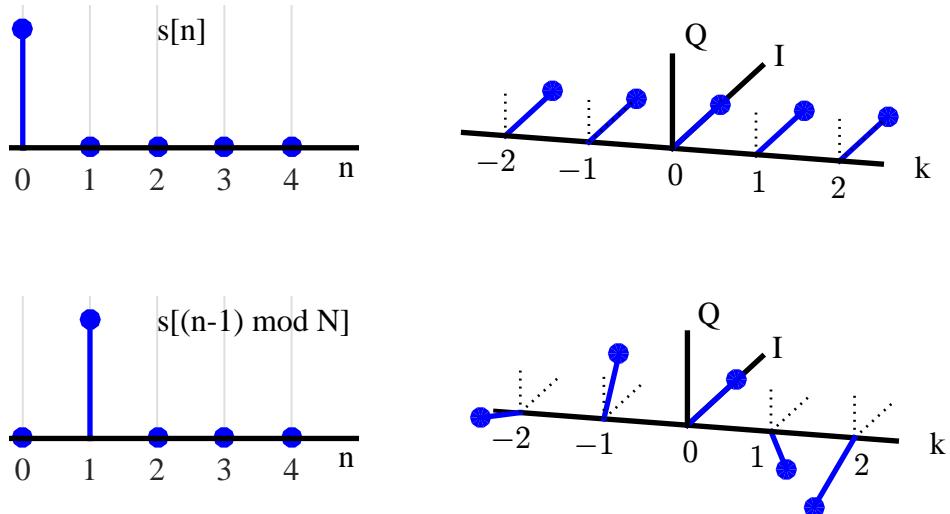


Figure 1.55: Sample shift in time generates phase shift in frequency

Since time and frequency are dual of each other, the time domain impulse in this example drawn in Figure 1.55 must have a corresponding frequency domain complex sinusoid.

$$\text{Time domain impulse} \longleftrightarrow \text{Frequency domain complex sinusoid} \quad (1.59)$$

And that is exactly what a phase shift of $2\pi(k/N)n_0$ represents as a function of k . This is shown for $n_0 = -1$ in Figure 1.56, a redrawn version of the bottom right of Figure 1.55. The important point here is that the frequency index k is the variable of this frequency domain complex sinusoid. On the other hand, time shift n_0 is an inverse period of this sinusoid and plays the role similar to frequency F in the expression $2\pi F t$.

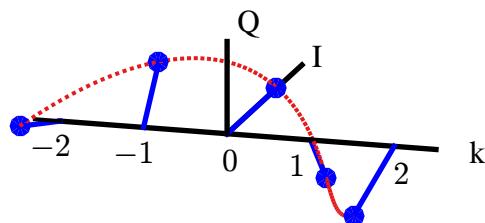


Figure 1.56: A redrawn version of the bottom right subfigure of Figure 1.55 to show a frequency domain complex sinusoid arising from a shifted impulse in time

$$\text{phase shift} = 2\pi(n_0) \frac{k}{N} = 2\pi(\text{inverse period}) \frac{k}{N}$$

Now let us apply the time shift to a time domain complex sinusoid. Using the identities $\cos(A + B) = \cos A \cos B - \sin A \sin B$ and $\sin(A + B) = \sin A \cos B + \cos A \sin B$,

$$\begin{array}{ll} I \rightarrow & \cos 2\pi \frac{k}{N} (n + n_0) = \cos 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n_0 - \\ & \quad \sin 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n_0 \\ Q \uparrow & \sin 2\pi \frac{k}{N} (n + n_0) = \sin 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n_0 + \\ & \quad \cos 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n_0 \end{array}$$

From the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$, we can see that in frequency domain, our original complex sinusoid is again being multiplied with another *frequency domain complex sinusoid* given by

$$\begin{array}{ll} I \rightarrow & V_I[n] = \cos 2\pi \frac{k}{N} n_0 \\ Q \uparrow & V_Q[n] = \sin 2\pi \frac{k}{N} n_0 \end{array}$$

Finally, what happens when N analysis frequencies are present (the complex sinusoids that form the DFT)? Extending the same concept proved above and noting that a DFT is a combination of N complex sinusoids, the DFT of $\tilde{s}[n] = s[(n + n_0) \bmod N]$ is given by the following expression which is also straightforward to prove through DFT definition.

$$\begin{array}{ll} I \rightarrow & \tilde{S}_I[k] = S_I[k] \cos 2\pi \frac{k}{N} n_0 - S_Q[k] \sin 2\pi \frac{k}{N} n_0 \\ Q \uparrow & \tilde{S}_Q[k] = S_Q[k] \cos 2\pi \frac{k}{N} n_0 + S_I[k] \sin 2\pi \frac{k}{N} n_0 \end{array} \quad (1.60)$$

i.e., the DFT is multiplied with a frequency domain complex sinusoid as n_0 is fixed and k varies[†]. This is the most important relation to understand for visualizing time and frequency domain transformations of a signal and we summarize it as

Shift in one domain \longrightarrow Multiplication by a complex sinusoid in the other domain

(1.61)

The magnitude and direction of rotation for these frequencies is symbolically shown for a positive n_0 in Figure 1.57.

[†]In terms of complex signals, this expression is written for a signal $s(t)$ as

$$\begin{aligned} s(t) &\longrightarrow S(F) \\ s(t - t_0) &\longrightarrow S(F) \cdot \exp(-j2\pi F t_0) \end{aligned}$$

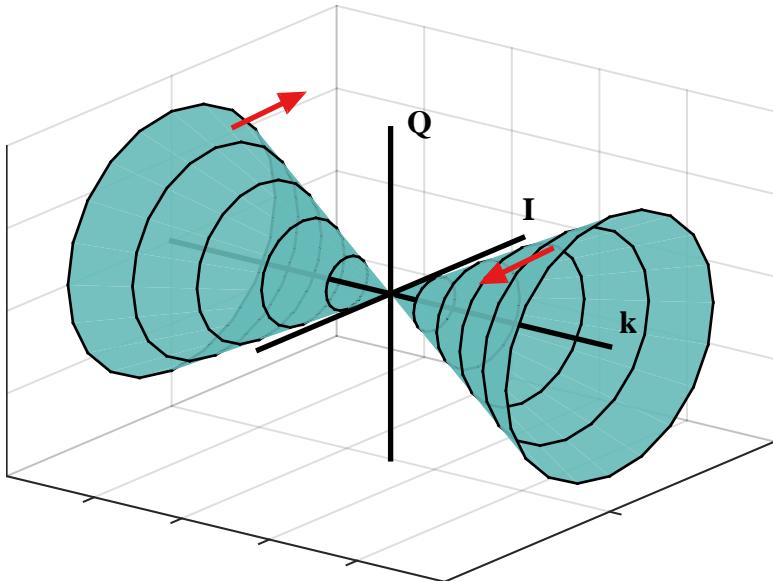


Figure 1.57: Symbolic representation of linear phase rotation that changes with frequency index k for a positive n_0

Significance of Linear Phase

As far as a linear phase is concerned, it preserves the waveform shape due to the following reason. A simple delay ensures the waveform integrity in time domain. When the signal is composed of constituent sinusoids, all of them need to be delayed by the same time (and not by the same phase). Intuitively, if sinusoids with different frequencies get delayed by the same number of samples, then they naturally end up with different phases at the end of that common duration, as illustrated in Figure 1.58. However, each of those phases is the *common delay* multiplied with that frequency. The overall sum of the sinusoids with different frequencies still remains the same.

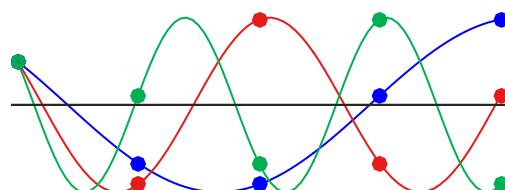


Figure 1.58: A linear phase implies that all sinusoids experience phase shifts given by the product of their frequency with the delay duration

On the other hand, if the phase is non-linear, then the delay introduced in the waveform is not proportional to the frequency, thus different constituent sinusoids get delayed by varying amounts of time which distorts the signal shape, commonly known as phase distortion. In wireless communications, the target is to maintain a linear phase of the signal for proper detection.

True insights can only be developed through grasping the implications of phase

rotations as well as time and frequency domain complex sinusoids. Next, we discuss a few examples of DFTs of some basic signals that will help not only understand the Fourier transform but will also be useful in comprehending the concepts discussed further.

1.10 DFT Examples

We begin with the example of the DFT of a rectangular signal.

1.10.1 A Rectangular Signal

A rectangular sequence, both in time and frequency domains, is by far the most important signal encountered in DSP applications. One of the reasons for this is that any signal with a finite duration, say T seconds, in time domain (that all practical signals have) can be considered as a product between a possibly infinite signal and a rectangular sequence with duration T seconds. This is called *windowing* and a rectangular sequence is the simplest form of a window. Windowing is a key concept in signal conditioning for the application of DSP techniques.

Let us compute the N -point DFT of a length- L rectangular sequence

$$s_I[n] = \begin{cases} 1, & n_s \leq n \leq n_s + L - 1 \\ 0, & \text{otherwise} \end{cases} \quad (1.62)$$

shown in Figure 1.59, where $N \geq L$.

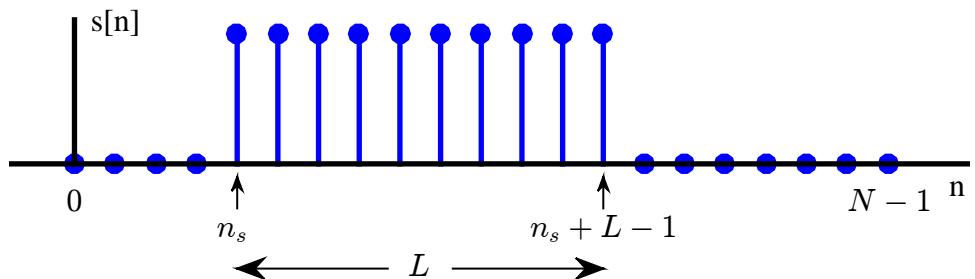


Figure 1.59: A general rectangular sequence with length $L < N$

Since this sequence is real with no Q component in time domain, the I component in frequency domain from the DFT definition can be expressed as

$$\begin{aligned} I &\rightarrow S_I[k] = \sum_{n=0}^{N-1} s_I[n] \cos 2\pi \frac{k}{N} n = \sum_{n=n_s}^{n_s+L-1} \cos 2\pi \frac{k}{N} n \\ &= \sum_{n=n_s}^{n_s+L-1} \cos 2\pi \frac{k}{N} n \frac{\sin \pi \frac{k}{N}}{\sin \pi \frac{k}{N}} \end{aligned} \quad (1.63)$$

Let $\theta = 2\pi k/N$ and using the identity $2 \cos A \sin B = \sin(A + B) - \sin(A - B)$, we get

$$\begin{aligned} I &\rightarrow S_I[k] = \sum_{n=n_s}^{n_s+L-1} \cos n\theta \frac{\sin \theta/2}{\sin \theta/2} \\ &= \frac{1}{2 \sin \theta/2} \sum_{n=n_s}^{n_s+L-1} \left[\sin \left(n + \frac{1}{2} \right) \theta - \sin \left(n - \frac{1}{2} \right) \theta \right] \\ &= \frac{1}{2 \sin \theta/2} \left[\sin \left(n_s + L - \frac{1}{2} \right) \theta - \sin \left(n_s - \frac{1}{2} \right) \theta \right] \end{aligned}$$

where all the other terms in the summation cancel out. Using the same identity as above

$$\begin{aligned} I &\rightarrow S_I[k] = \frac{1}{\sin \theta/2} \cos \left(n_s + \frac{L-1}{2} \right) \theta \cdot \sin \left(\frac{L}{2} \right) \theta \\ &= \frac{\sin L\theta/2}{\sin \theta/2} \cos \left(n_s + \frac{L-1}{2} \right) \theta \end{aligned}$$

Plugging in the expression back for θ ,

$$I \rightarrow S_I[k] = \frac{\sin \pi Lk/N}{\sin \pi k/N} \cos \left[2\pi \frac{k}{N} \left(n_s + \frac{L-1}{2} \right) \right] \quad (1.64)$$

Similarly, Q component can be found from the DFT definition as well.

$$Q \rightarrow S_Q[k] = \sum_{n=0}^{N-1} -s_I[n] \sin 2\pi \frac{k}{N} n = \sum_{n=n_s}^{n_s+L-1} -\sin 2\pi \frac{k}{N} n \quad (1.65)$$

Following the same procedure as I component, and using another identity $\sin A \sin B = 0.5 \{ \cos(A - B) - \cos(A + B) \}$, the Q component of its DFT is given by

$$Q \rightarrow S_Q[k] = -\frac{\sin \pi Lk/N}{\sin \pi k/N} \sin \left[2\pi \frac{k}{N} \left(n_s + \frac{L-1}{2} \right) \right] \quad (1.66)$$

Using the above equations and for an all-ones rectangular sequence $L = N = 16$, Figure 1.60 draws the I and Q parts of the resulting DFT. Notice that when the rectangular sequence covers all the available time domain from 0 to $N - 1$, its DFT is just a single impulse at frequency 0.

Despite its simplicity, there is a lot of information hidden and several interesting conclusions to be drawn here for which we continue further discussion below.

Magnitude and Phase

Applying the definitions of magnitude and phase to Eq (1.64) and Eq (1.66) and using $\cos^2 A + \sin^2 A = 1$, we get the magnitude and phase of the DFT of a rectangular signal.

$$\begin{aligned} |S[k]| &= \frac{\sin \pi Lk/N}{\sin \pi k/N} \\ \angle S[k] &= -2\pi \frac{k}{N} \left(n_s + \frac{L-1}{2} \right) \end{aligned} \quad (1.67)$$

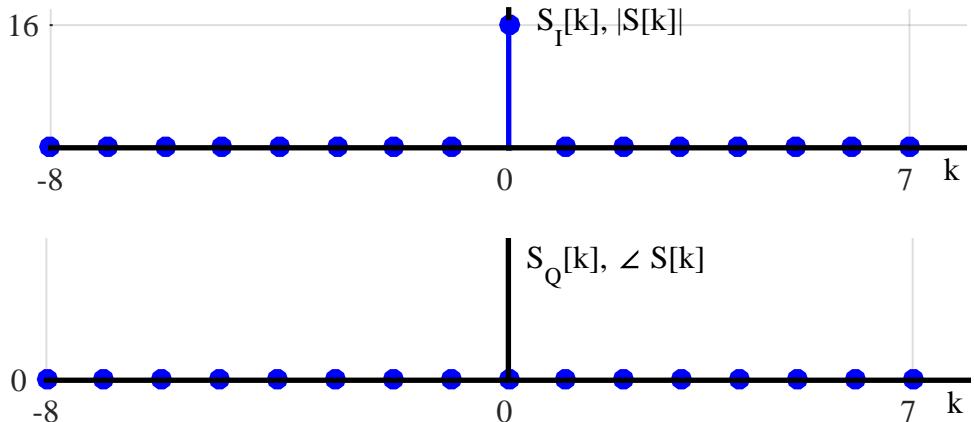


Figure 1.60: Magnitude and phase (as well as I and Q parts) of an all-ones DFT for $L = N = 16$

With $L = N = 16$, Figure 1.60 also displays the magnitude and phase plots for the DFT of a rectangular signal which in this case are similar to the I and Q plots, respectively. How a specific value of n_s affects the phase plot is discussed in Section 1.9. In general, magnitude-phase and IQ plots convey information in different manners, many examples of which we will encounter throughout this text.

DFT, Frequency Domain Sampling and Leakage

For DFT of a rectangular signal, the IQ equations are given in Eq (1.64) and (1.66), and the magnitude-phase relations in Eq (1.67). The corresponding figures are drawn in Figure 1.60 with single impulses at bin 0.

The question is: How can such relatively complicated equations generate such simple plots? The answer is that it was a special case due to the signal being an all-ones rectangular sequence in time domain. Let us now consider $L < N$ as shown in Figure 1.61 for $L = 7$, $N = 16$ and

$$n_s = -\frac{L-1}{2} = -3$$

This choice of n_s makes the signal symmetric around zero position. Also, this is the same as $n_s = N - (L-1)/2 = 13$ due to DFT input periodicity. Just like before,

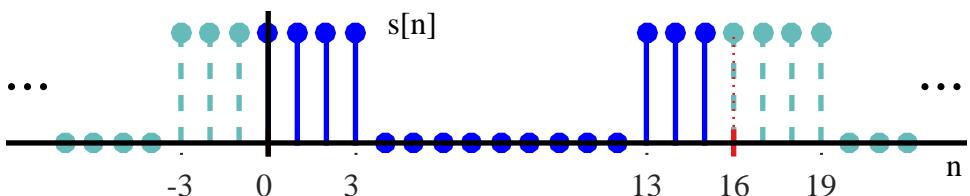


Figure 1.61: The rectangular sequence in time domain for $n_s = -(L-1)/2$ from DFT computation perspective

verify through a software routine like `fft()` that this even signal with an inphase only component has a DFT that is also even and has only an inphase part.

Next, for the same signal with $L = 7$ and $N = 16$, let $n_s = -(L - 1)/2 - 1 = -4$. The sequence can be imagined as shifted to the left by 1 from its symmetric position around zero. I and Q plots from Eq (1.64) and Eq (1.66) are shown in Figure 1.62 and the magnitude and phase plots from Eq (1.67) are drawn in Figure 1.63. A clear shape of *sinc signal* defined as

$$\text{sinc}(k) = \frac{\sin(\pi L k / N)}{\sin(\pi k / N)} \quad (1.68)$$

is visible in magnitude plot which is sampled at discrete frequencies k/N .

Now we can recognize that the plots in Figure 1.60 were also sinc functions but *sampled at the peak value at bin 0 and at zero crossings for all other bins*. Have a careful look at the sinc signal and understand its characteristics that are shortly coming next. The ability to visualize these sinc signals and their corresponding sample points makes all the difference to one's DSP comprehension!

To comprehend the reason why energy for this signal appeared in other frequency bins as well, recall that the DFT finds the contributions of complex sinusoids with analysis frequencies $F = F_S \cdot k/N$ for $k = -N/2$ to $N/2 - 1$ in an input signal, which are all integer multiples of a fundamental frequency F_S/N . Due to their orthogonality, the DFT is able to find each k^{th} contribution independently of all others. And if the input signal contains contributions from only a frequency k/N , the DFT output is proportional to the magnitude at the contributing frequency bin and zero at the remaining bins.

In the case of an all-ones rectangular sequence with $L = N = 16$, it is actually a

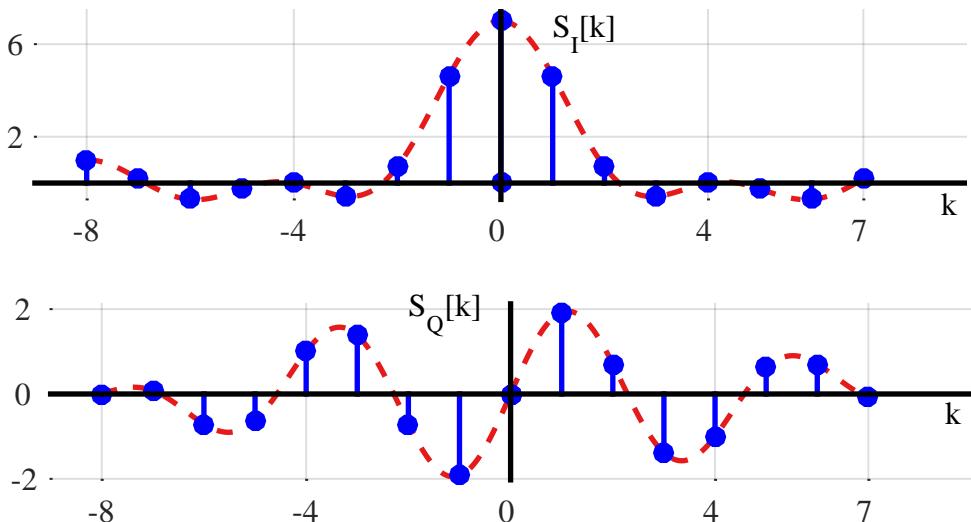


Figure 1.62: I and Q components of the DFT of a rectangular signal for $L = 7$, $N = 16$ and $n_s = -(L - 1)/2 - 1$

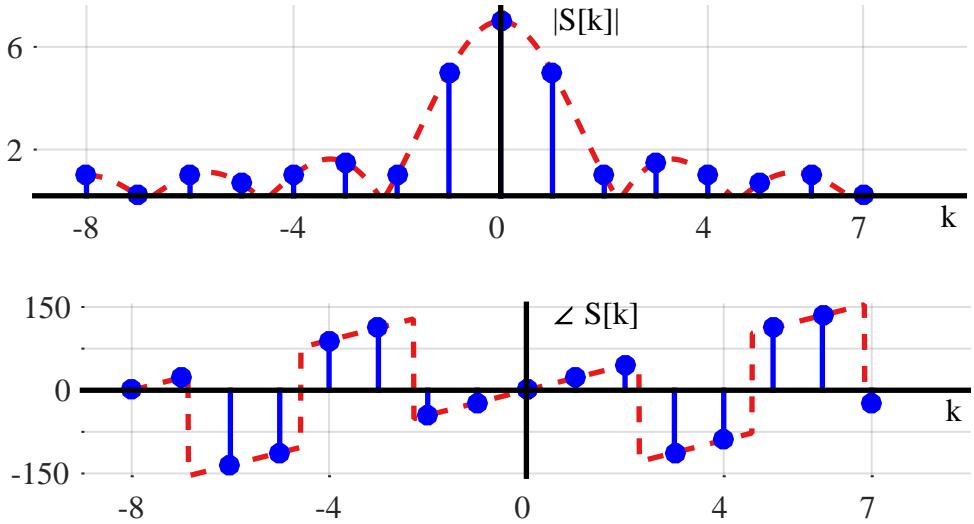


Figure 1.63: Magnitude and phase of the DFT of a rectangular signal for $L = 7$, $N = 16$ and $n_s = -(L - 1)/2 - 1$

single complex sinusoid at discrete frequency $0/N$,

$$\begin{aligned} I \rightarrow \quad \cos 2\pi \frac{k}{N} n &= \cos 2\pi \frac{0}{N} n = 1, & n = 0, \dots, N-1 \\ Q \uparrow \quad \sin 2\pi \frac{k}{N} n &= \sin 2\pi \frac{0}{N} n = 0, & n = 0, \dots, N-1 \end{aligned}$$

and hence orthogonal to all the remaining $N - 1$ complex sinusoids. For this reason, we have its DFT as a single impulse at $k = 0$ and zeros for all other k , see Figure 1.60.

On the other hand, its truncated version with $L = 7$, $N = 16$ is not a single complex sinusoid. It is actually made up of all N sinusoids just like a periodic square wave we encountered in Figure 1.35 and hence the energy in all N bins.

In a practical scenario, the input data sequence is not bound to contain energy precisely at a set of given analysis frequency bins. If the input has a signal component at some intermediate frequency between these integer multiples of F_S/N , say $1.3F_S/N$, the orthogonality condition does not hold and this input signal shows up to some degree in all N output bins of our DFT. This is known as *DFT leakage*.

When we perform the DFT on real-world finite-length time sequences, DFT leakage is an unavoidable phenomenon. Let us construct an example to observe this in detail.

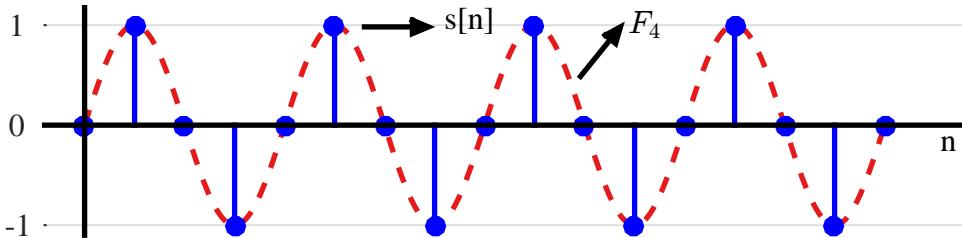
Example 1.6

Assume a signal with frequency 4 kHz at a sample rate of $F_S = 16$ kHz.

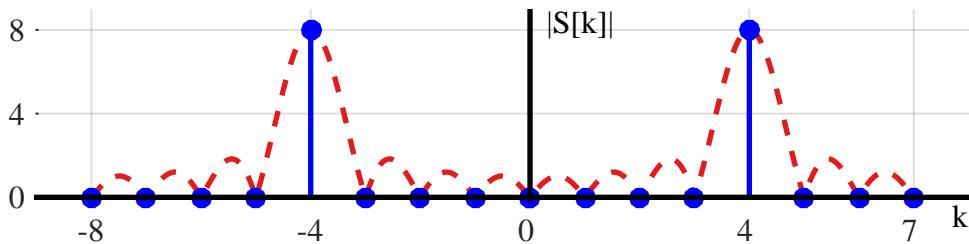
$$s[n] = \sin 2\pi \frac{4}{16} n$$

where $N = 16$. The solid blue curve is shown in Figure 1.64a in time domain along with one analysis frequency $F_4 = F_S \cdot k/N = 16000 \cdot 4/16 = 4$ kHz. Note that both the input frequency and the analysis frequency F_4 complete four full cycles during the sampling interval and hence its DFT, plotted in Figure 1.64b, has two impulses at bins

4 and -4 with zero input from all the other bins. *The shape of the actual curve is the same sinc function* but it is sampled in frequency domain at just the right points, i.e., integer frequency bins due to the integer number of cycles of the input in time domain.



(a) Signal with input frequency 4 kHz has integer number of cycles during the interval, just like the analysis frequency F_4



(b) All the energy is in bin 4 with no energy in any other bin

Figure 1.64: No DFT leakage

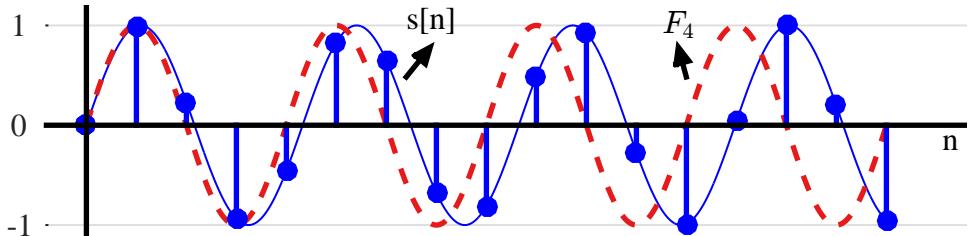
Now let us change the input signal slightly by making the input frequency equal to 3.7 kHz as shown in Figure 1.65a.

$$s[n] = \sin 2\pi \frac{3.7}{16} n$$

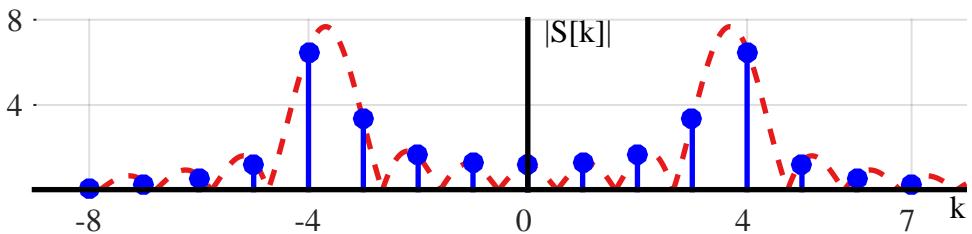
Observe that the analysis frequency F_4 completes four full cycles during that interval, while the frequency 3.7 kHz in the original signal $s[n]$ does not have integer number of cycles over 16-sample interval, violating the orthogonality and causing the input energy to leak into all the other DFT output bins as shown in Figure 1.65b.

The $k = 3$ bin is not zero because the sum of the products of the input sequence and $k = 3$ analysis frequency is no longer zero (same is also true for, say, $k = 4$). This leakage causes any input signal whose frequency is not exactly at a DFT bin center to leak into all of the other DFT output bins. *The shape of the sinc function did not change* as compared to a 4 kHz input but it is sampled in frequency domain at non-ideal points, i.e., fractional frequency bins due to non-integer number of cycles of the input in time domain.

Why does the leakage happen? Imagine the process of correlation at lag 0 as a sampling process in frequency domain where a particular analysis frequency ‘samples’ the spectrum. With a non-integer multiple of the fundamental frequency F_s/N , this frequency domain sampling does not occur at zero crossings thus introducing



(a) Signal with input frequency 3.7 kHz has incomplete cycle during the interval, unlike the closest analysis frequency F_4



(b) Most of the energy is in 4th bin with the rest leaking into remaining bins

Figure 1.65: DFT leakage

interference from all N participants.

In conclusion, the signals with frequencies other than the set k/N are not periodic in the observation window and are suddenly terminated without completing the final cycle in full. This discontinuity violates the orthogonality relation yielding a non-zero result and becomes responsible for spectral contributions over the entire set of frequencies k/N .

Mainlobe Peak

The DFT of a rectangular signal has a mainlobe centered about the point $k = 0$. Its amplitude cannot be found by plugging in $k = 0$ in sinc signal of Eq (1.67) because both the numerator and denominator become zero resulting in undefined 0/0 expression. Although a mathematical trick known as L'Hôpital's rule can be applied to compute it, we take an easier course. *The peak amplitude of the mainlobe is L* because at $k = 0$, the DFT definition dictates that

$$I \rightarrow S_I[0] = \sum_{n=0}^{N-1} s_I[n]$$

$$Q \uparrow S_Q[0] = \sum_{n=0}^{N-1} s_Q[n]$$

This is because $\cos 2\pi(0/N)n = 1$ and $\sin 2\pi(0/N)n = 0$. In words, the DFT $S[0]$ is the sum of all original samples. As $s_Q[n] = 0$ in this example, the sum of L unity-valued samples appears as $S_I[0] = L$. Consequently, the peak value is seen to be $L = 16$ in Figure 1.60 and $L = 7$ in Figure 1.63.

Mainlobe Width and Zero Crossings

The width of the mainlobe is defined by zero crossings of the curve. Since $\sin \pm \pi = 0$, the first zero crossing occurs when the numerator argument of sinc signal in Eq (1.67), $\sin \pi Lk/N$, is equal to π . All other zero crossings are integer multiples of the first. So the mainlobe width is given by the value of the first zero crossing k_{zc} as

$$\begin{aligned}\pi L \frac{k_{zc}}{N} &= \pi \\ k_{zc} &= \frac{N}{L}\end{aligned}\quad (1.69)$$

This value is confirmed from $N/L = 1$ in Figure 1.60 and $N/L = 16/7 = 2.29$ in Figure 1.63. A zero crossing right at sample 1 in Figure 1.60 illustrates that it was sampled at peak value for bin 0 and at zero for all other bins.

DFT Phase

Most of the discussion until now was around the magnitude plots. Let us understand the concept of phase through the DFT of a rectangular signal. Now the phase plot in Figure 1.63 for $L = 7$, $N = 16$ and $n_s = -(L - 1)/2 - 1$ has been drawn in Figure 1.66 after unwrapping 180° phase jumps which arose due to changing I and Q signs. The resulting phase is a straight line with a positive slope equal to 22.5° . Where did this number come from?

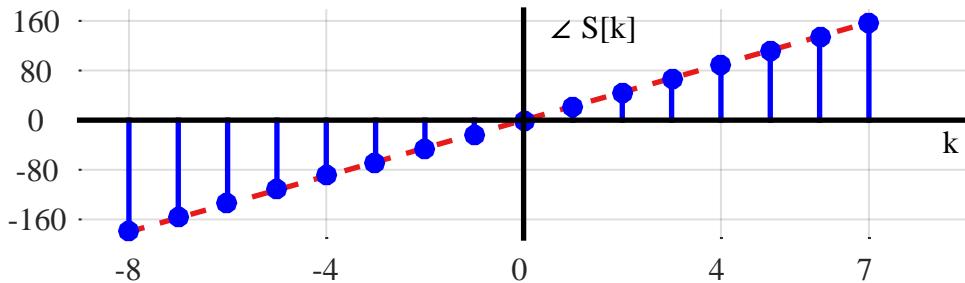


Figure 1.66: DFT phase in Figure 1.63 for $L = 7$, $N = 16$ and $n_s = -(L - 1)/2 - 1$ with 180° jumps unwrapped

We learned in Eq (1.58) of Section 1.9 that a time shift of n_0 samples results in phase rotation of $2\pi(k/N)n_0$ for each k , the sign of which depends on the sign of the time shift. For time traveling in the past (i.e., a negative shift), the phase rotation is also negative while for time traveling in the future (i.e., a positive shift), the phase rotation is also positive.

Now in our example, an even symmetric signal would have been obtained for $n_s = -(L - 1)/2 = 3$. Then, $n_0 = 0$ and the phase plot would be 0 or π as shown in Figure 1.67a.

However, for $n_s = -(L - 1)/2 + 1$, it is a right shift (e.g., from -3 to -2 for $L = 7$) of that even symmetric signal by 1 sample. Thus, plugging $n_0 = -1$ in the expression $2\pi(k/N)n_0$ gives the phase rotation $\Delta\theta(k)$ for each frequency bin as

$$\Delta\theta(k) = 2\pi \frac{k}{N} n_0 = -2\pi \frac{k}{N}$$

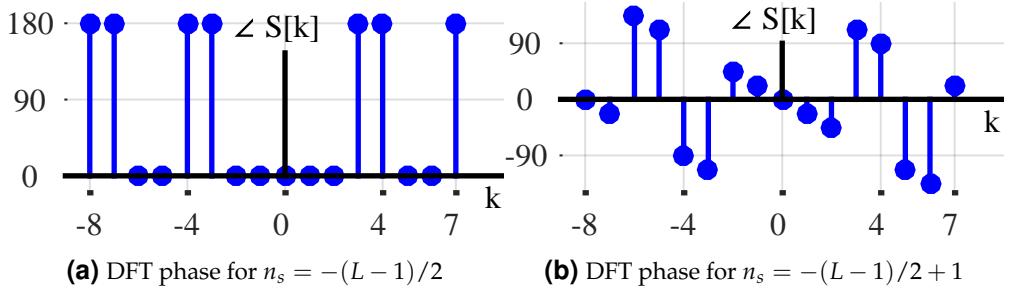


Figure 1.67: Phase plots for different time shifts

for each k from $-N/2$ to $N/2 - 1$. For $N = 16$,

$$\begin{aligned}\Delta\theta(0) &= 2\pi \frac{0}{N} \times \frac{180^\circ}{\pi} = 0 \\ \Delta\theta(\pm 1) &= 2\pi \frac{\pm 1}{N} \times \frac{180^\circ}{\pi} = \pm 22.5^\circ \\ \Delta\theta(\pm 2) &= 2\pi \frac{\pm 2}{N} \times \frac{180^\circ}{\pi} = \pm 45^\circ \\ \Delta\theta(\pm 3) &= 2\pi \frac{\pm 3}{N} \times \frac{180^\circ}{\pi} = \pm 67.5^\circ \\ &\dots \\ &\dots\end{aligned}$$

A linear slope of -22.5° is thus verified from Figure 1.67b when adjusted for 180° phase jumps arising from sinc amplitude changing signs.

1.10.2 A General Sinusoid

The DFT of a general sinusoid can be derived similarly by plugging the expression of a complex sinusoid in DFT definition and following the same procedure as in the rectangular sequence example.

Nevertheless, having understood the above concepts, we can see how the magnitude and phase plots of the DFT of a general sinusoid look like by an easier method. As an all-ones rectangular sequence is nothing but a complex sinusoid with frequency 0 with length $L = N$, the DFT magnitude of a general complex sinusoid with frequency k' (which can be a non-integer) can be found by putting $L = N$ (as it spans the whole length), $n_s = 0$ (as it starts at 0) and $k - k'$ instead of k in the sinc signal of Eq (1.67) (to bring sinc center to zero) as

$$|S[k]| = \frac{\sin \pi(k - k')}{\sin \pi(k - k')/N}$$

$$\angle S[k] = -2\pi \frac{k - k'}{N} \left(\frac{N - 1}{2} \right)$$

(1.70)

Similarly, the DFT of a real sinusoid $\cos[2\pi(k'/N)n]$ can be seen as the sum of the

above expression with a similar expression replacing $k - k'$ with $k + k'$ because each real sinusoid consists of two complex sinusoids scaled by half, see Eq (1.27).

For example, consider a signal shown in Figure 1.68 consisting of two sinusoids at 1 kHz and 2 kHz as

$$\begin{aligned}s(t) &= s_1(t) + s_2(t) \\ &= \sin(2\pi 1000t) + 0.75 \sin(2\pi 2000t + 120^\circ)\end{aligned}$$

It can be seen that $s(t)$ has only an I component with zero Q component. Sampling

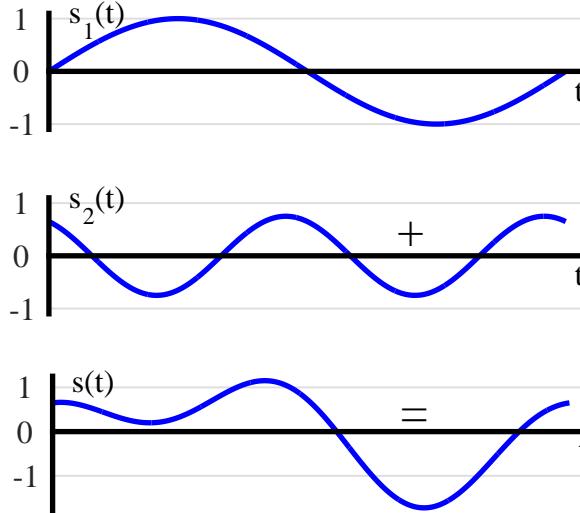


Figure 1.68: A signal $s(t)$ composed of two sinusoids

the above signal at a rate of $1/T_S = F_S = 8$ kHz produces

$$\begin{aligned}s[n] &= \sin(2\pi 1000nT_S) + 0.75 \sin(2\pi 2000nT_S + 120^\circ) \\ &= \sin\left(2\pi \frac{1}{8}n\right) + 0.75 \sin\left(2\pi \frac{2}{8}n + 120^\circ\right)\end{aligned}\quad (1.71)$$

Our DFT analysis frequencies $F = F_S(k/N)$ for an 8 point DFT are

$$\begin{aligned}k = 0 \Rightarrow 8000 \cdot \frac{0}{8} &= 0 \text{ kHz} & k = \pm 1 \Rightarrow 8000 \cdot \frac{\pm 1}{8} &= \pm 1 \text{ kHz} \\ k = \pm 2 \Rightarrow 8000 \cdot \frac{\pm 2}{8} &= \pm 2 \text{ kHz} & k = \pm 3 \Rightarrow 8000 \cdot \frac{\pm 3}{8} &= \pm 3 \text{ kHz}\end{aligned}$$

So in this case, bin 1 corresponds to an actual frequency of 1 kHz, bin 2 to 2 kHz, and so on. The DFT $S[k]$ – both IQ and magnitude phase – is plotted in Figure 1.69.

From magnitude plot of this figure, observe that the DFT has detected two real sinusoids in this signal because the impulses at bins 1 and -1 indicate the presence of two complex sinusoids that combine to form one real sinusoid at a frequency $8000 \cdot 1/8 = 1$ kHz. A similar argument holds for the other frequency of 2 kHz.

DC or Average Value

The DC, or average, value of this signal $\frac{1}{N} \sum s[n]$ is 0 which is evident from Eq (1.71) as the average value of a sinusoid over an integral number of cycles is 0.

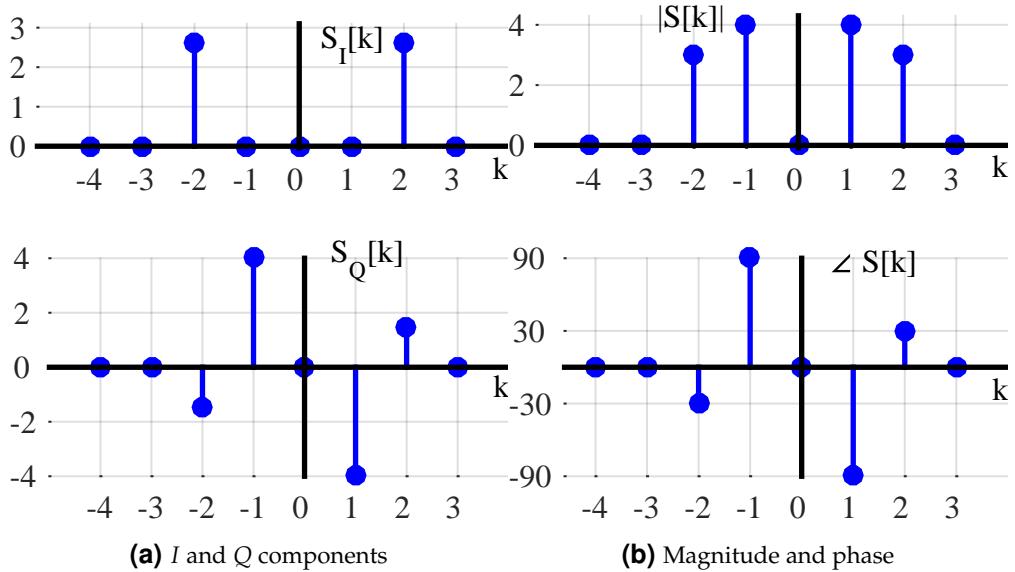


Figure 1.69: DFT of $s[n]$ composed of two sinusoids

Magnitude

When an input signal contains a complex sinusoid of peak amplitude A with an integral number of cycles over N input samples, *the output magnitude of the DFT for that particular sinusoid is AN* . This can be checked by plugging in the expression for a complex sinusoid with amplitude A_0 into the DFT definition.

$$\begin{aligned} I &\rightarrow \sum_{n=0}^{N-1} \left[A_0 \cos 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n + A_0 \sin 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n \right] \\ Q &\uparrow \sum_{n=0}^{N-1} \left[A_0 \sin 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n - A_0 \cos 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n \right] \end{aligned}$$

Using the identities $\cos A \cos B + \sin A \sin B = \cos(A - B)$ and $\sin A \cos B - \cos A \sin B = \sin(A - B)$,

$$\begin{aligned} I &\rightarrow \sum_{n=0}^{N-1} A_0 \cos 0 = A_0 N \\ Q &\uparrow \sum_{n=0}^{N-1} A_0 \sin 0 = 0 \end{aligned}$$

Thus, the output magnitude of the DFT is proved as $A_0 N$.

For a real sinusoid, $s_Q[n]$ is 0 and only the first term in I part of the above equation survives. Since $\cos^2 A = 1/2(1 + \cos 2A)$, *the output magnitude of the DFT for a real sinusoid is $A_0 N/2$* .

Considering the fact that $s[n]$ is composed of two real sine waves with amplitudes 1 and 0.75, we can see in Figure 1.69b that 1 kHz sine wave has a magnitude $1 \times N/2 = 4$ and 2 kHz sine wave has a magnitude $0.75 \times N/2 = 3$.

Phase

To decode the phase plot of Figure 1.69b, we will have to go all the way back to Figure 1.32 where to generate a sine wave on time I -axis, we saw in 3 dimensions how the two impulses, representing two complex sinusoids at $+F$ and $-F$, are rotated by an angle of -90° and $+90^\circ$, respectively. In other words, the phase reference is cosine on the I axis.

Therefore, bin 1 frequency in Figure 1.69b at $8000 \cdot 1/8 = 1$ kHz has a phase of -90° . From the identity $\cos(A - 90^\circ) = \sin(A)$, it is indeed a cosine with a phase of -90° . The corresponding phase at bin -1 is obviously $-(-90^\circ) = +90^\circ$.

As for the second term $0.75 \sin\{2\pi(2/8)n + 120^\circ\}$, it is a sine at 2 kHz with a phase shift of 120° , or a cosine with a phase of $-90^\circ + 120^\circ = +30^\circ$ as shown in Figure 1.69b for bin 2.

Referring to its time domain plot in Figure 1.68, the first sinusoid $s_1(t)$ is *positioned around the origin* according to its -90° cosine (or 0° sine) phase, while the second sinusoid $s_2(t)$ is positioned around the origin according to its 30° cosine (or 120° sine) phase. As examined before, through the phase plot, the DFT in fact finds the time alignments of all the sinusoids at bin frequencies $F_S \cdot k/N$.

Do remember that the plots shown in Figure 1.69b are – once again – sinc functions overlapping each other. However, since the signal consists of two exact analysis frequencies, the frequency domain is sampled at the precise locations of zero crossings. That being the case, the sinc function becomes invisible and looks like two sets of impulses only.

1.10.3 The Unit Impulse

Having known the DFT of a rectangular signal, we have two ways to find the Fourier transform of a unit impulse.

$$s[n] = \delta[n]$$

Time-Frequency Duality: A closer look at DFT and iDFT equations reveal that the forward and inverse transform are almost identical, except the scaling factor $1/N$ and direction of rotations of analysis sinusoids. Therefore, time and frequency domains are dual and DFT of a signal in time domain can be derived by the iDFT of a signal in frequency domain. For example, a single impulse at frequency bin 0 in Figure 1.60 is an all-ones rectangular sequence in time domain. By duality, the inverse is also true: a single impulse in time domain corresponds to an all-ones rectangular sequence in frequency domain.

Rectangular signal: From Figure 1.59, it is evident that a unit impulse is also a rectangular sequence with length $L = 1$. Therefore,

$$\begin{array}{ll} I \rightarrow & S_I[k] = |S[k]| = \frac{\sin \pi k / N}{\sin \pi k / N} = 1 \\ Q \uparrow & S_Q[k] = \angle S[k] = 0 \end{array}$$

Since the starting sample $n_s = 0$, the signal is symmetric and the angle is equal to 0. This DFT was graphically illustrated on the top right of Figure 1.55.

A Shifted Unit Impulse

The DFT of a shifted unit impulse can be derived from the result that the DFT of a unit impulse is an all-ones sequence.

$$\tilde{s}[n] = \delta[(n + n_0) \bmod N]$$

Then, we utilize Eq (1.60) as

$$\begin{aligned} I &\rightarrow \tilde{S}_I[k] = \cos 2\pi \frac{k}{N} n_0 \\ Q &\uparrow \quad \quad \quad \tilde{S}_Q[k] = \sin 2\pi \frac{k}{N} n_0 \end{aligned} \quad (1.72)$$

which is nothing but a complex sinusoid discovered earlier in Figure 1.56.

1.10.4 The Sampling Sequence

The sampling sequence is a sequence of unit impulses repeating with a period M within our observation window (and owing to DFT input periodicity, outside the observation window as well). Figure 1.70 illustrates the sampling sequence in time domain for $N = 15$ and $M = 3$.

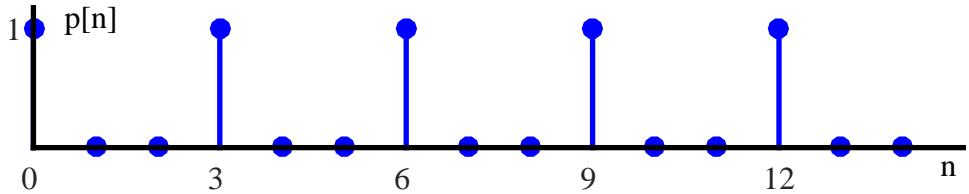


Figure 1.70: Sampling sequence in time domain for $N = 15$ and $M = 3$

To compute its DFT, consider that $p_I[n] = p[n]$ and $p_Q[n] = 0$. Then,

$$I \rightarrow P_I[k] = \sum_{n=0}^{N-1} p[n] \cos 2\pi \frac{k}{N} n$$

Since $p[n] = 0$ except when $n = M$, we can write $p[Mm] = 1$ where $m = 0, 1, \dots, N/M - 1$. We are assuming N/M as an integer here because otherwise $p[n]$ will not be periodic outside the observation window. So in the above example,

$$m = 0, 1, 2, 3, 4 \quad n = Mm = 0, 3, 6, 9, 12$$

and

$$I \rightarrow P_I[k] = \sum_{m=0}^{N/M-1} \cos 2\pi \frac{k}{N} Mm = \sum_{m=0}^{N/M-1} \cos 2\pi \frac{k}{N/M} m$$

Note that the above equation is very similar to Eq (1.63) encountered while computing the DFT of a rectangular sequence. Here, we have

$$n_s \rightarrow 0 \quad N \rightarrow \frac{N}{M} \quad L \rightarrow \frac{N}{M}$$

Consequently, the DFT of the sampling sequence can be written similar to the DFT of a rectangular sequence as

$$\begin{aligned}|P[k]| &= \frac{\sin \pi k}{\sin \pi k M/N} \\ \angle P[k] &= 0\end{aligned}\quad (1.73)$$

Magnitude and Phase

The sequence is plotted in frequency domain in Figure 1.71 and consists of M impulses, where both I and Q components are shown for $M = 3$ and $N = 15$. While it seems that the spectrum should be zero for all k due the numerator being a multiple of π for all k , we shortly see why this is not true. Furthermore, as explained above, due to the sequence being real and symmetric with respect to zero in time domain, the phase is zero in frequency domain and hence the magnitude plot is the same as the I part.

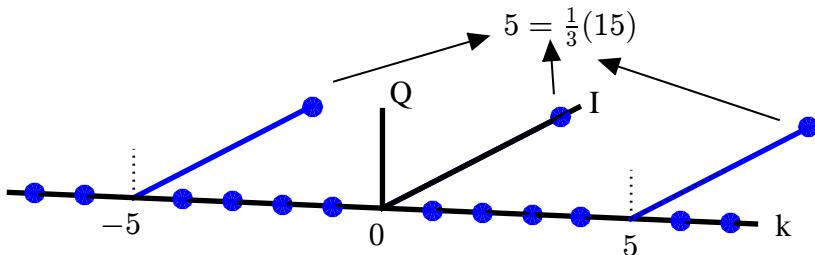


Figure 1.71: Sampling sequence in frequency domain for $N = 15$ and $M = 3$

Mainlobe Peak

In the DFT of a rectangular signal, we saw that both the numerator and denominator become equal to zero for $k = 0$. In that case, the peak of the mainlobe was equal to L , the length of the sequence. We found this value by plugging $k = 0$ in the DFT definition and deduced that the result is just a sum of all time domain samples because $\cos 2\pi k/N = 1$ and $\sin 2\pi k/N = 0$. Here in this case, the peak value can easily be seen as the sum of N/M unit impulses and hence equal to $N/M = 5$.

Remember from Section 1.10.2 that the DFT of a complex sinusoid with amplitude A_0 in time has a magnitude $A_0 N$ in frequency domain. Here, the spectral replicas have a peak magnitude of 5 instead of 15 which suggests that *the actual spectrum has been scaled down by a factor of $1/M = 1/3$* due to the effect of intermediate zero-valued samples. This is shown in Figure 1.71.

Mainlobe Positions

There are multiple impulses in frequency domain here: more than one as in DFT of an all-ones rectangular sequence but less than N as in the DFT of a unit impulse. Observe from Eq (1.73) that when the denominator is non-zero, the spectrum is clearly zero due to integer k . When both the numerator and denominator are zero, the peak

values are N/M as above. To find these locations, check where the denominator is zero, i.e., an integer multiple of π , as well.

$$\pi k_{\text{peak}} \frac{M}{N} = \text{integer} \cdot \pi$$

$$k_{\text{peak}} = \text{integer} \cdot \frac{N}{M}$$

In our example, these peaks turn out to be 0 and ± 5 as shown in Figure 1.71 due to the chosen values of N and M . In most DSP applications, we are interested in a frequency axis from -0.5 to 0.5 . In that case, these positions are normalized by N and hence given by integer multiples of $1/M = 1/3$.

If you guessed that these impulses in frequency domain are actually sinc functions sampled at just the peak and zero values, you are right (compute a length-32 DFT of this sequence instead of length-15, for example).

A Shifted Sampling Sequence

What happens when the sampling sequence studied above is shifted by one or more samples as in Figure 1.72a. We have already seen in Section 1.9 that a shifting a signal in time domain produces a multiplication of its spectrum with a complex sinusoid with inverse period equal to that shift. Subsequently, we can see the DFT of a right shifted by 1 sampling sequence through applying either Eq (1.58) or Eq (1.61), i.e., the DFT of a sampling sequence is phase rotated by $-2\pi(k/N)$. This is shown in Figure 1.72b where the phase shift can be recognized as

$$\pm 2\pi \frac{k}{N} n_0 = \pm 2\pi \frac{5}{15} (-1) = \mp 120^\circ$$

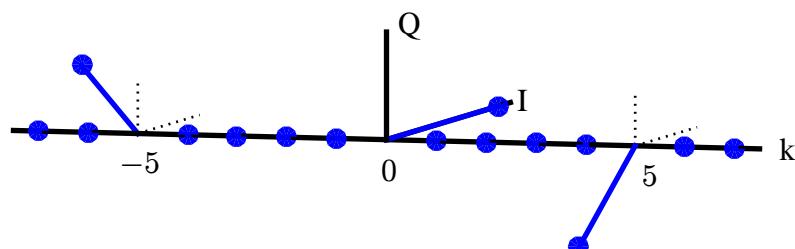
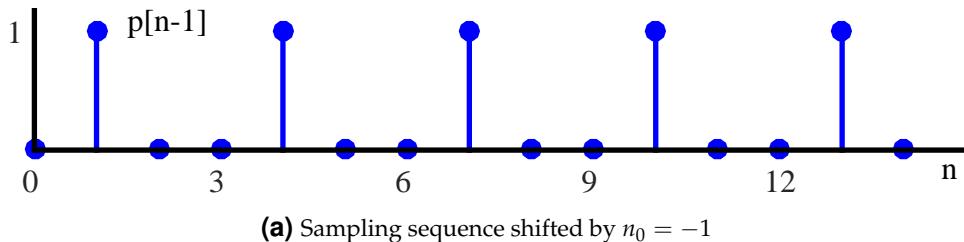


Figure 1.72: Spectrum of a sampling sequence shifted in time domain consists of phase rotated impulses

as compared to Figure 1.71. This concept will be helpful when we describe the channel selection of a wireless signal through a polyphase filterbank implementation in Chapter 10.

1.11 A Digital Signal

We have talked about obtaining a discrete-time signal through sampling the time-axis and obtaining a discrete-frequency set through sampling the frequency axis. The same concept can be applied to the amplitude-axis, where the signal amplitude can be sampled to take only a finite set of discrete values. This discrete-time discrete-valued signal is called a *digital signal*, as opposed to an *analog signal* that is continuous in time and continuous in amplitude.

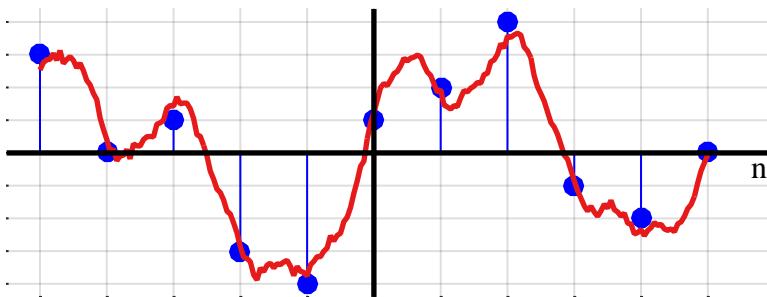


Figure 1.73: Obtaining a digital signal through slicing the y-axis

Figure 1.73 shows how a digital signal having amplitudes over a fixed set of values can be obtained through slicing the underlying continuous amplitudes. For example, an amplitude of 2.2 can be rounded to 2, 1.4 to 1 and so on depending on the desired resolution.

Computers can only work with digital signals because discrete-time signals – though defined only for finite values on time-axis – can have infinite values on the amplitudes-axis. Just as computer memory is finite and can store only a known number of time values, its width is also finite (e.g., 8 bits) and can store only a fixed number of amplitudes (e.g., for an 8-bit wide memory, we can have $2^8 = 256$ values for amplitudes).

1.12 The Small Picture

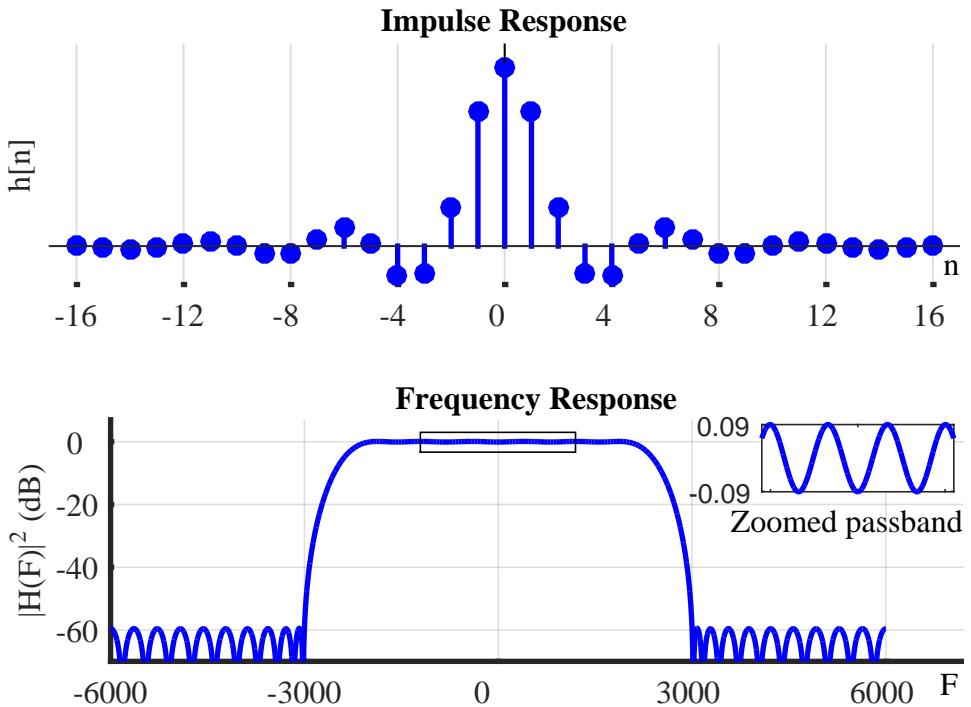
There are 26 letters in English language as well as numerous rules for connecting them to form words. In reference to Figure 1.55, the language of *basic* DSP is much simpler.

1. There is only one letter: a sample at time 0 (that can be scaled by any number). The DFT of that sample is an all ones sequence in frequency domain, which in reality is a complex sinusoid with frequency 0.
2. There is one major rule: a time shift n_0 in that sample generates a phase shift in frequency which in reality is a complex sinusoid with inverse period n_0 .

Later in Section 2.4, we will learn that any discrete-time sequence can be thought of as a summation of scaled samples shifted in time.

Chapter 2

Introduction to Systems



“A complex system that works is invariably found to have evolved from a simple system that worked.”

Gall's Law

A device or algorithm that performs some prescribed operations on an input signal to generate an output signal is called a *system*. As shown in Figure 2.1, communicating from a point *A* to a point *B* through a wireless medium is a game of three systems:

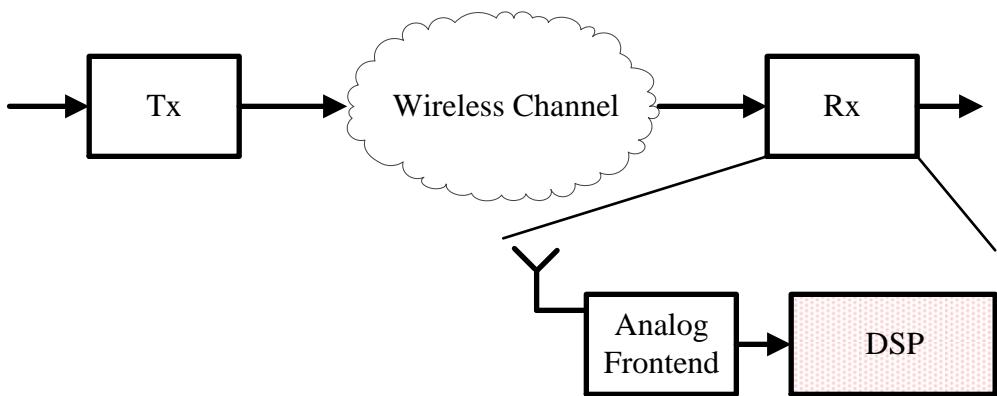


Figure 2.1: Wireless communication is a game of 3 systems

Tx: The Tx system generates the signal to be transmitted over the air or any other medium.

Wireless channel: While the wireless channel might look like an enemy, it is the system that carries our Tx signal to the Rx. Through extensive measurement campaigns, scientists have studied this channel in quite significant details. Consequently, our transmission strategy at the Tx and the Rx is based to cope with the obstacles it poses in an optimal manner.

Rx: An ideal Rx should perform an exact inverse of the operations carried out at the Tx. However, due to the wireless channel bringing a weak and distorted signal at its doorstep, the task of the Rx system is much harder and consists of several additional operations. Solving these problems through subsystems in an efficient manner is what communication is all about. Two of these subsystems are shown in Figure 2.1, namely an analog frontend that conditions the incoming signal to prepare for digital processing and a Digital Signal Processing (DSP) engine that applies efficient algorithms in a computing language to recover the Tx data. This is a Software Defined Radio (SDR).

In wireless communications and other DSP applications, we often want to modify a generated or acquired signal. In Section 1.2 for example, we saw how a signal can be scaled and time shifted, or added and multiplied with another signal. These are all examples of simple systems. Our main focus in this text is on a particular class of systems which are linear and time-invariant.

2.1 Linear Systems

A *linear system* implies that if two inputs are scaled and summed together to form a new input, the new output of the system is also a scaled sum of their individual outputs.

Scaling: For scaling to hold, if

$$\text{input } s_1[n] \longrightarrow r_1[n] \text{ output}$$

then

$$\text{input } \alpha s_1[n] \longrightarrow \alpha r_1[n] \text{ output}$$

where α is any scalar.

Addition: When two such inputs are added together, the output should be the sum of their individual outputs as

$$\text{input } s_1[n] + s_2[n] \longrightarrow r_1[n] + r_2[n] \text{ output}$$

A linear system combines the above two properties, namely scaling and addition, as

$$\text{input } \alpha_1 s_1[n] + \alpha_2 s_2[n] \longrightarrow \alpha_1 r_1[n] + \alpha_2 r_2[n] \text{ output} \quad (2.1)$$

Below, we discuss examples of a linear and a non-linear system.

Example 2.1

Consider a system

$$r[n] = \frac{3}{7} \cdot s[n]$$

The output of this system, as a response to an input $1 \cdot s_1[n] = 1 \cdot \sin(2\pi 0.1n)$, is

$$r_1[n] = \frac{3}{7} \sin(2\pi 0.1n)$$

Similarly, its response to a different signal $2 \cdot s_2[n] = 2 \cdot \sin(2\pi 0.3n)$ is

$$r_2[n] = \frac{3}{7}(2) \sin(2\pi 0.3n) = \frac{6}{7} \sin(2\pi 0.3n)$$

When this system is given the input $1 \cdot s_1[n] + 2 \cdot s_2[n]$, the output is

$$\begin{aligned} r[n] &= \frac{3}{7} \left\{ 1 \cdot \sin(2\pi 0.1n) + 2 \cdot \sin(2\pi 0.3n) \right\} \\ &= \frac{3}{7} \sin(2\pi 0.1n) + \frac{6}{7} \sin(2\pi 0.3n) \\ &= r_1[n] + r_2[n] \end{aligned}$$

Hence, it is a linear system.

On the other hand, when the same input $1 \cdot s_1[n] + 2 \cdot s_2[n]$ is given to another system

$$r[n] = s^2[n]$$

and using the identity $\sin(A)\sin(B) = 0.5\{\cos(A - B) - \cos(A + B)\}$, the output is

$$\begin{aligned}
 r[n] &= \{1 \cdot \sin(2\pi 0.1n) + 2 \cdot \sin(2\pi 0.3n)\}^2 \\
 &= \sin^2(2\pi 0.1n) + 4 \cdot \sin^2(2\pi 0.3n) + 4 \sin(2\pi 0.1n) \sin(2\pi 0.3n) \\
 &= \sin^2(2\pi 0.1n) + 4 \cdot \sin^2(2\pi 0.3n) + 2 \cdot \cos(2\pi 0.2n) - 2 \cdot \cos(2\pi 0.4n) \\
 &\neq r_1[n] + r_2[n] \\
 &= \sin^2(2\pi 0.1n) + 4 \cdot \sin^2(2\pi 0.3n)
 \end{aligned}$$

Clearly, it is a non-linear system.

We can deduce from the above example that if the x and y axes are drawn on the ground as independent variables, then the traces of the great pyramids of Gaza and the popular Sydney Opera House in Figure 2.2 can be related to linear and non-linear systems, respectively.



Figure 2.2: Great pyramids of Gaza and Sydney Opera House

Note 2.1 Nature and linearity

Wherever you are reading these lines, have a look around and you will find that most of the things with straight lines are made by humans such as books, tables and buildings. And most of the natural stuff is non-linear or curved such as plants, mountains and meteors. This is an indication that human and natural designs follow different game plans and our products are an attempt to extract linearity out of an inherently non-linear world. While human designs work reasonably well, perhaps this is also one of the reasons we have barely scratched the tip of the iceberg when it comes to the cruel field of biology, particularly making changes to our own bodies. In this regard, we will have to chart serious non-linear territories to comply with nature and then control it.

From Example 2.1, it is also clear that input sinusoids do not interact with each other in linear systems, and hence output frequencies were the same as the input frequencies. In a non-linear system, however, input sinusoids interacted with each other to produce frequencies that were not present in either of the input signal, as shown in Figure 2.3 for $r[n] = s^2[n]$. Note from $|S[k]|$ that $s[n]$ actually consists of only two frequencies at bins 1 and 2, but the output $[r[n]]$ of the system is composed of other frequencies as well shown in $|R[k]|$.

The Discrete Fourier Transform (DFT) is a linear operation as it is evident from its definition involving complex sinusoids. Any scaling and addition of two or more input signals will result in a DFT output that is a scaled and summed version of their

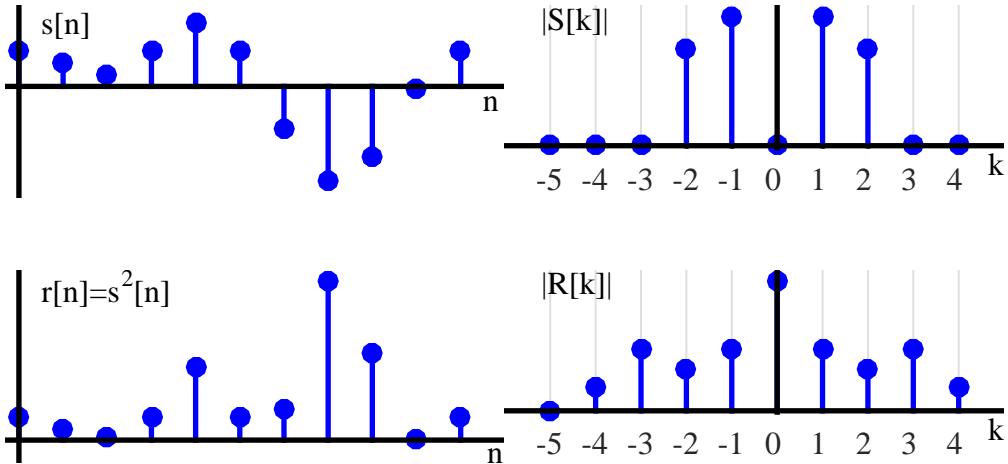


Figure 2.3: Extra frequency components in a non-linear system

individual DFT outputs.

2.2 Time-Invariant Systems

Time invariance roughly means that the output is independent of the time at which the input is applied. For example, if a human body is considered as a system, Usain Bolt would run at different speeds at different times of the day. In fact, the best time for peak physical performance is late afternoon or early evening in which a disproportionate number of Olympic records have been broken. Had this system been time-invariant, the athletes would have displayed a similar performance regardless of the time of the day when the competition was held.

Technically, a system is *time-invariant* if shifting the input sequence on time axis leads to an equivalent shift of the output sequence along the time axis, with no other changes. So if an input $s[n]$ generates an output $r[n]$, then

$$\text{input } s[n - n_0] \longrightarrow r[n - n_0] \text{ output}$$

In other words, if an input signal is delayed by some amount n_0 , the output also just gets delayed by n_0 and remains exactly the same otherwise. Whether a system is time-invariant or not can be determined by the following steps.

1. Delay the input signal $s[n]$ by n_0 samples to make it $s[n - n_0]$. Find the output $r_1[n]$ for this input.
2. Delay the output signal $r[n]$ by n_0 samples to make it $r[n - n_0]$. Call the new output $r_2[n]$.
3. If $r_1[n] = r_2[n]$, the system is time-invariant. Otherwise, it is time-variant.

Example 2.2

Consider a system

$$r[n] = s[n] + s[n - 3]$$

We will follow the steps above to determine whether this system is time-invariant or not.

1. Delaying the input signal by n_0 samples, the output in response to $s[n - n_0]$ is

$$r_1[n] = s[n - n_0] + s[n - n_0 - 3]$$

2. Delaying the output signal by n_0 samples, we get

$$r_2[n] = s[n - n_0] + s[n - 3 - n_0]$$

3. Since $r_1[n] = r_2[n]$, this system is time-invariant.

Now consider another system

$$r[n] = ns[n]$$

and apply the same steps.

1. Delaying the input signal by n_0 samples, the output in response to $s[n - n_0]$ is

$$r_1[n] = ns[n - n_0]$$

2. Delaying the output signal by n_0 samples, we get

$$r_2[n] = (n - n_0)s[n - n_0]$$

3. Since $r_1[n] \neq r_2[n]$, this system is time-variant.

In other words, a system is time-invariant if the order of delaying does not matter, i.e.,

$$s[n] \rightarrow \text{Delay} \rightarrow \text{System} \rightarrow \text{Output } r_1[n]$$

is the same as

$$s[n] \rightarrow \text{System} \rightarrow \text{Delay} \rightarrow \text{Output } r_2[n]$$

Note 2.2 Linear Time-Invariant (LTI) System

A system that is both linear and time-invariant is, not surprisingly, called [*Linear Time-Invariant \(LTI\)*](#) system. It is a particularly useful class of systems that not only truly represents many real-world systems but also possesses an invaluable benefit of having a rich set of tools available for its design and analysis.

2.3 System Characterization

After learning some properties of the system, the question is: how should we characterize a system in both time and frequency domains? We now see that impulse response and frequency response are the tools to characterize a system in time domain and frequency domain, respectively. We begin with a description of the impulse response.

Impulse Response

As the name says, *impulse response* is just the output response of a system to a unit impulse $\delta[n]$ as an input. Impulse response is also a signal sequence and it is usually denoted as $h[n]$.

As an example, when the strings of a guitar are plucked, the back and forth vibrations disturb the surrounding air molecules for a certain amount of time: this is an impulse response. In a very similar manner, *when a system is 'kicked' by a unit impulse signal*, the whole sequence of samples it generates can be viewed as its impulse response.

For instance, if a system just delays any input signal by 3 samples, the impulse response $h[n]$ will be $\delta[n]$ shifted by 3 samples.

$$h[n] = \delta[n - 3]$$

Of course, impulse response $h[n]$ of a real-world system is much more complex. An example is illustrated in Figure 2.4 where both $h_I[n]$ and $h_Q[n]$ are drawn to highlight the fact that the impulse response $h[n]$ of the system is, in general, a complex signal of time.

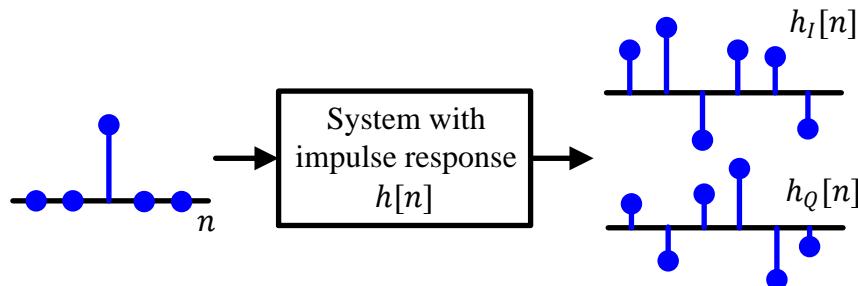


Figure 2.4: Impulse response $h[n]$ is system output in response to input $\delta[n]$

Frequency Response

From the definition of impulse response, it is straightforward to guess that frequency response of a system is defined as *the DFT of its impulse response*.

$$h[n] \xrightarrow{\text{F}} H[k]$$

From the definition of DFT,

$$\begin{aligned} I &\rightarrow H_I[k] = \sum_{n=0}^{N-1} \left[h_I[n] \cos 2\pi \frac{k}{N} n + h_Q[n] \sin 2\pi \frac{k}{N} n \right] \\ Q &\uparrow \quad \quad \quad H_Q[k] = \sum_{n=0}^{N-1} \left[h_Q[n] \cos 2\pi \frac{k}{N} n - h_I[n] \sin 2\pi \frac{k}{N} n \right] \end{aligned} \tag{2.2}$$

The question is what this response means in frequency domain. Remember from Section 1.10.3 that the DFT of a single impulse in time domain is an all-ones rectangular sequence in frequency domain, which is nothing but N complex sinusoids of equal magnitudes.

So when an impulse is input to a system in time domain, a sequence of N complex sinusoids with equal magnitudes are input to the system in frequency domain. What comes out is the comprehensive spectral behaviour of the system! This is a beautiful result because a system can thus be probed in frequency domain at all frequencies with equal intensity. Figure 2.5 shows an example of a system frequency response.

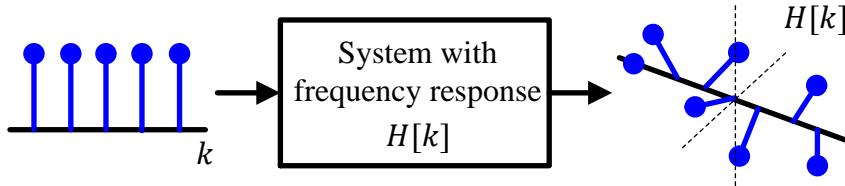


Figure 2.5: Frequency response $H[k]$ of the system in response to complex sinusoids at all N frequencies

The frequency response $H[k]$ is, in general, a complex signal of frequency. However, instead of drawing separate I and Q parts like $h[n]$ of Figure 2.4, a 3D frequency response is drawn due to the simplicity of visualizing the signals in frequency domain in this manner. Now we come to two questions that arise from the discussion so far.

1. What happens at the output in frequency domain as a result of a complex sinusoidal input?
2. By this stage, you would have noticed a dominant use of complex sinusoids not only in this text but throughout the DSP literature. What makes these signals so significant?

We answer these questions next in Section 2.4.2 after discussing convolution.

2.4 Convolution

At last, we have arrived at the stage where we can talk about the biggest test DSP beginners face: understanding convolution. After learning about a system and its impulse response, one wonders if there is any method through which the output signal of a system can be determined for any given input signal, not just a unit impulse. **Convolution** is the answer to that question, provided that the system is linear and time-invariant (LTI).

2.4.1 Regular Convolution

We start with real signals and LTI systems with real impulse responses. The case of complex signals and systems will be discussed later.

Decomposition of a Signal into Scaled Unit Impulses

Assume that we have an arbitrary signal $s[n]$. Then, $s[n]$ can be decomposed into a scaled sum of shifted unit impulses through the following logic. Multiply $s[n]$ with a unit impulse shifted by m samples as $\delta[n - m]$. Since $\delta[n - m]$ is equal to 0 everywhere

except at $n = m$, this would multiply all values of $s[n]$ by 0 when n is not equal to m and by 1 when n is equal to m . So the resulting sequence will have an impulse at $n = m$ with its value equal to $s[m]$. This process is clearly illustrated in Figure 2.6. This can

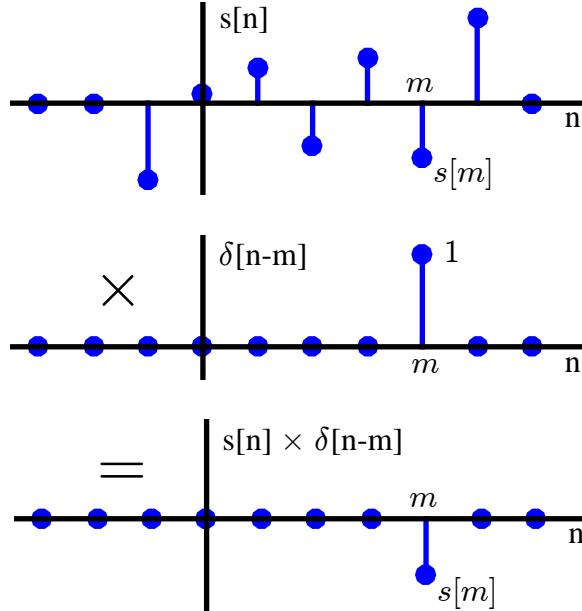


Figure 2.6: Multiplication of a signal $s[n]$ with a unit impulse shifted by m

be mathematically written as

$$s[n]\delta[n - m] = s[m]\delta[n - m]$$

Repeating the same procedure with a different delay m' gives

$$s[n]\delta[n - m'] = s[m']\delta[n - m']$$

We say that *the value $s[m']$ is extracted at this instant*. Therefore, if this multiplication is repeated over all possible delays $-\infty < m < \infty$, and all produced signals are summed together, the result will be the sequence $s[n]$ itself.

$$\begin{aligned} s[n] &= \cdots + s[-2]\delta[n+2] + s[-1]\delta[n+1] + \\ &\quad s[0]\delta[n] + s[1]\delta[n-1] + s[2]\delta[n-2] + \cdots \\ &= \sum_{m=-\infty}^{\infty} s[m]\delta[n-m] \end{aligned} \tag{2.3}$$

In summary, the above equation states that $s[n]$ can be written as a summation of scaled unit impulses, where each unit impulse $\delta[n - m]$ has an amplitude $s[m]$. An example of such a summation is shown in Figure 2.7.

Response of an LTI System to Scaled Unit Impulses

Now consider what happens when such a signal is given as an input to an LTI system with an impulse response $h[n]$.

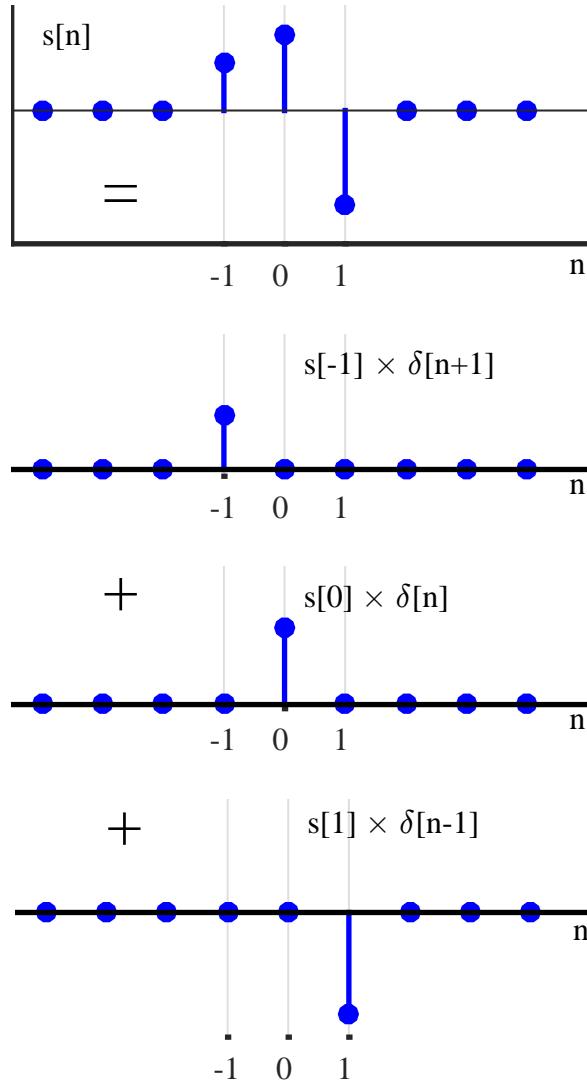


Figure 2.7: A signal $s[n]$ broken down into scaled and shifted impulses

- The impulse response – response to a unit impulse $\delta[n]$ – of the LTI system is $h[n]$.
- By virtue of time-invariance, the system response to a shifted unit impulse $\delta[n - m]$ is $h[n - m]$.
- By virtue of scaling property of linearity, the system response to a scaled and shifted unit impulse $s[m] \cdot \delta[n - m]$ is $s[m] \cdot h[n - m]$.
- By virtue of addition property of linearity, the system response to a sum of shifted unit impulses is the sum of their respective outputs.

input	→	output	
$\delta[n]$	→	$h[n]$	
$\delta[n - m]$	→	$h[n - m]$	(time-invariance)
$\alpha_1 \cdot \delta[n - m]$	→	$\alpha_1 \cdot h[n - m]$	(linearity)
$\alpha_2 \cdot \delta[n - m']$	→	$\alpha_2 \cdot h[n - m']$	(linearity)
$\alpha_1 \cdot \delta[n - m] + \alpha_2 \cdot \delta[n - m']$	→	$\alpha_1 \cdot h[n - m] + \alpha_2 \cdot h[n - m']$	(LTI)

This leads to an input-output sequence as

$$\begin{array}{cccc}
 + & \dots & \xrightarrow{\hspace{2cm}} & \dots \\
 + & s[-2] \cdot \delta[n+2] & \xrightarrow{\hspace{2cm}} & s[-2] \cdot h[n+2] \\
 + & s[-1] \cdot \delta[n+1] & \xrightarrow{\hspace{2cm}} & s[-1] \cdot h[n+1] \\
 + & s[0] \cdot \delta[n] & \xrightarrow{\hspace{2cm}} & s[0] \cdot h[n] \\
 + & s[1] \cdot \delta[n-1] & \xrightarrow{\hspace{2cm}} & s[1] \cdot h[n-1] \\
 + & s[2] \cdot \delta[n-2] & \xrightarrow{\hspace{2cm}} & s[2] \cdot h[n-2] \\
 + & \dots & \xrightarrow{\hspace{2cm}} & \dots \\
 = & s[n] & \xrightarrow{\hspace{2cm}} & \sum_{m=-\infty}^{\infty} s[m]h[n-m]
 \end{array}$$

During the above procedure, we have worked out the famous convolution equation that describes the output $r[n]$ for an input $s[n]$ to an LTI system with impulse response $h[n]$:

$$r[n] = \sum_{m=-\infty}^{\infty} s[m]h[n-m] \quad (2.4)$$

where the above sum is computed *for each n* from $-\infty$ to $+\infty$. In this text, convolution operation is denoted by $*$ as

$$r[n] = s[n] * h[n] \quad (2.5)$$

Note that unlike correlation,

$$s[n] * h[n] = h[n] * s[n] \quad (2.6)$$

$$= \sum_{m=-\infty}^{\infty} h[m]s[n-m]$$

This can be verified by plugging $p = n - m$ in Eq (2.4) which yields $m = n - p$ and hence $\sum_{p=-\infty}^{\infty} s[n-p]h[p]$.

Note 2.3 Convolution notation

Remember that although both convolution and conjugate of a signal are denoted by an $*$, the difference is always visible in the context. When used for convolution, it appears

as an operator between two signals, e.g., $s[n] * h[n]$. On the other hand, when used for conjugation, it appears as a superscript of a signal, e.g., $h^*[n]$.

Convolution is a very logical and simple process but many DSP learners can find it confusing due to the way it is explained. Next, we will describe a conventional method and another more intuitive approach to computing the convolution output.

Conventional Method

After defining the convolution equation, its implementation is suggested through the following steps. For every individual time shift n ,

Flip: Arrange the convolution equation as

$$r[n] = \sum_{m=-\infty}^{\infty} s[m]h[-m+n]$$

and consider the impulse response as a function of variable m . Flip $h[m]$ about $m = 0$ to obtain $h[-m]$.

Shift: To obtain $h[-m+n]$ for time shift n , shift $h[-m]$ by n units to the right for positive n and left for negative n .

Multiply: Point-wise multiply the sequence $s[m]$ by sequence $h[-m+n]$ to obtain a *product sequence* $s[m] \cdot h[-m+n]$.

Sum: Sum all the values of the above product sequence to obtain the convolution output at time n .

Repeat: Repeat the above steps for every possible value of n .

An example of convolution between two signals

$$\begin{aligned}s[n] &= [2 \ -1 \ 1] \\ h[n] &= [-1 \ 1 \ 2]\end{aligned}$$

is shown in Figure 2.8. In the left column, the signal $s[m]$ is copied with shifted versions of $h[-m]$ to demonstrate the overlap between the two, while the shift n is shown in the text box for each n . If you follow the steps described above, you will find the result $r[n]$ in the text boxes on the right column of the figure.

$$r[n] = [-2 \ 3 \ 2 \ -1 \ 2]$$

Note a change in signal representation above. The actual signals $s[n]$ and $h[n]$ are a function of time index n but the convolution equation denotes both of these signals with time index m . On the other hand, n is used to represent the time shift of $h[-m]$ before point-wise multiplying it with $s[m]$. The output $r[n]$ is a function of time index n , which was that shift applied to $h[-m]$.

Next, we turn to a more intuitive method where flipping a signal is not required.

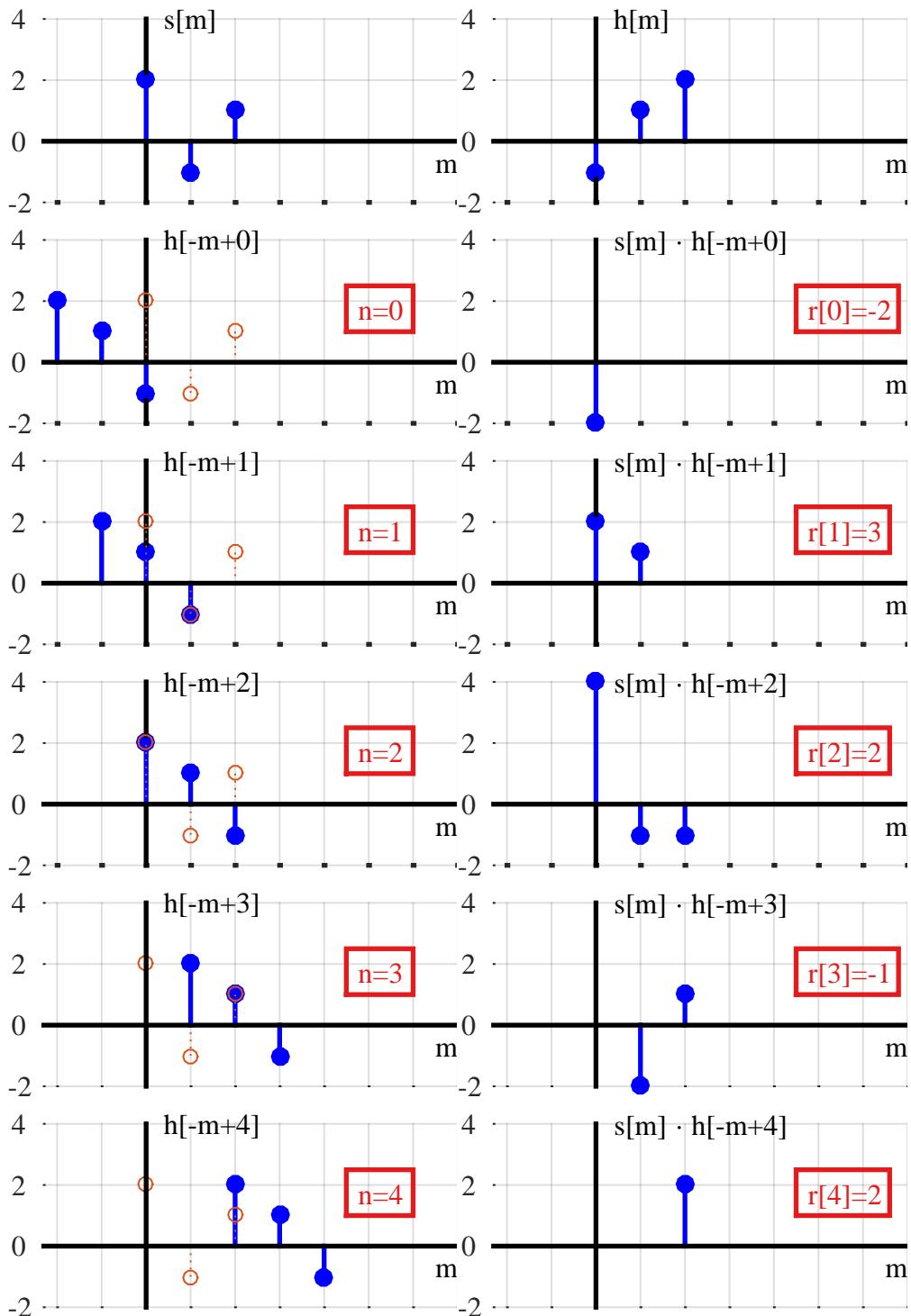


Figure 2.8: Conventional Method for convolution between $s[n]$ and $h[n]$

Intuitive Method

There is another method to understand convolution. In fact, it is built on the derivation of convolution equation (2.4), i.e., find the output $r[n]$ as

$$\begin{aligned}
 r[n] = & \cdots + \\
 & s[-2] \cdot h[n+2] + \\
 & s[-1] \cdot h[n+1] + \\
 & s[0] \cdot h[n] + \\
 & s[1] \cdot h[n-1] + \\
 & s[2] \cdot h[n-2] + \\
 & \cdots
 \end{aligned} \tag{2.7}$$

Let us solve the same example as in Figure 2.8, where

$$\begin{aligned}
 s[n] &= [2 \ -1 \ 1] \\
 h[n] &= [-1 \ 1 \ 2]
 \end{aligned}$$

Table 2.1: Intuitive method for convolution between $s[n]$ and $h[n]$

n	0	1	2	3	4
$s[n]$	2	-1	1		
$s[0] \cdot h[n]$	2(-1)	2(1)	2(2)		
$s[1] \cdot h[n-1]$	0	-1(-1)	-1(1)	-1(2)	
$s[2] \cdot h[n-2]$	0	0	1(-1)	1(1)	1(2)
Output	-2	3	2	-1	2

Such a method is illustrated in Table 2.1 and in Figure 2.9 where the result is produced from column-wise summation. From an implementation point of view, there is no difference between this and the conventional method.

Note 2.4 The curious case of a flipped impulse response

The Curious Case of Benjamin Button was a 2008 American film based on a remarkable account of a man living his life in reverse. He was born an old man and got younger with time until he became an infant at his death.

Running the impulse response backwards in time to perform convolution reminds me of that story. Instead of a curious case, it has been an interesting convention in DSP

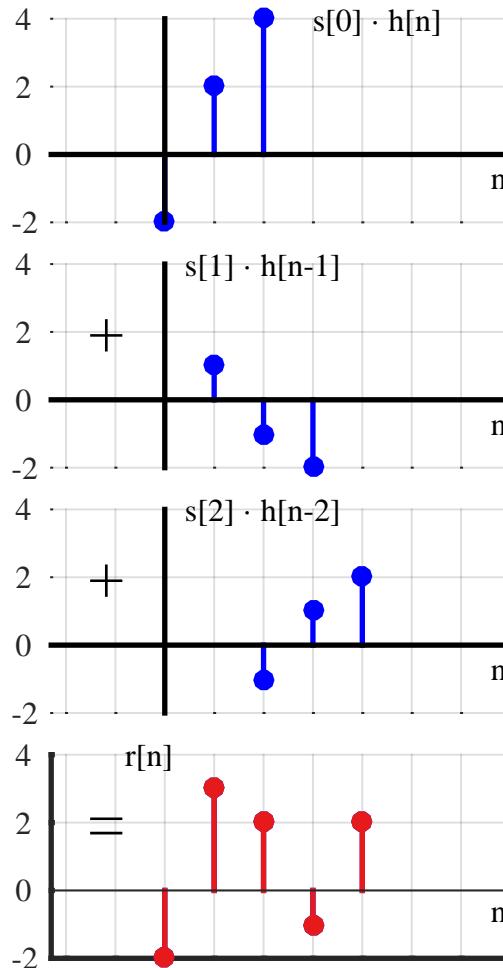


Figure 2.9: Intuitive method for convolution between $s[n]$ and $h[n]$

history to regard convolution between two signals as the flipped version of one signal sliding over the other. The textbooks have justified it as a necessary mathematical trick for finding LTI system response, thus creating confusion for DSP beginners.

As we saw from the intuitive method above, flipping is essentially not there at all. Flipping only occurs in the universe of time index m , not in the universe of time index n . And it is at each n where we are finding every new output.

To further understand everything in terms of time index n , consider the following quote:

“Nothing has happened in the past; it happened in the Now. Nothing will ever happen in the future; it will happen in the Now.”

The Power of Now

One can infer from the above quote – buried in mathematical details of convolution – is that *the NOW, represented by n in convolution equation $\sum_m s[m]h[n-m]$, is itself a moving pointer* which actually keeps changing during the process. Time ticks and with each passing tick, we are able to *access* one unit into the future where a new input comes and kicks out another impulse response (scaled by input amplitude at that instant) from the system.

Since no practical input can be applied before the NOW, let us start from time index 0.

0. The input $s[0]$ contributes $s[0] \cdot h[0]$ in the NOW, which becomes the first output sample at time 0. It actually sets in motion the complete impulse response scaled by its value $(s[0] \cdot h[1], s[0] \cdot h[2], \dots)$ but those will occur in the future.
1. The new NOW is time index 1. Input $s[1]$ arrives and launches a complete impulse response of its own, but only $s[1] \cdot h[0]$ happens in the NOW. Also, having moved 1 unit into the future implies $s[0] \cdot h[1]$ has arrived at this time. So the output at time 1 is $s[1] \cdot h[0] + s[0] \cdot h[1]$.
2. Continuing in this way, it can be seen that during the convolution operation, the NOW n advances with every tick. Each such n activates a new impulse response which of course starts with $h[0]$, and hence the expression $s[n] \cdot h[0]$. Only those contributions from the past inputs can be summed here which contribute in the present moment.

Obviously, $h[1]$ will be the contribution from an input occurring at time $n - 1$, $h[2]$ from $n - 2$, and so on. In summary, the argument of $h[\cdot]$ is nothing but a length of time separation of every individual input from the NOW (time index n) – owing to this memory, it acts to find out the effect of every past input into the NOW.

Properties

Some important properties of convolution are as below.

Length: From the above figures, we find that the length of the resultant signal $r[n]$ is 5, when the lengths of both $s[n]$ and $h[n]$ are 3 samples. This is a general rule:

$$\text{Length}\{r[n]\} = \text{Length}\{s[n]\} + \text{Length}\{h[n]\} - 1 \quad (2.8)$$

due to one signal completely sliding past the other during convolution.

First sample: As evident from the flipping operation, location of the first sample of the result is given by

$$\begin{aligned} \text{First sample location}\{r[n]\} &= \text{First sample location}\{s[n]\} + \\ &\quad \text{First sample location}\{h[n]\} \end{aligned} \quad (2.9)$$

Unit impulse: Since a single unit impulse $\delta[n]$ can only generate a single instance of impulse response $h[n]$, convolution of an impulse response by a unit impulse is the impulse response itself. This result is general and holds for any signal $s[n]$.

$$s[n] * \delta[n] = s[n] \quad (2.10)$$

Convolution of Complex Signals

Convolution between a complex signal $s[n]$ input to a system with complex impulse response $h[n]$ can be understood through writing the convolution in IQ form. It is defined in a similar manner as

$$r[n] = s[n] * h[n]$$

Employing the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$

$$\begin{aligned} I &\rightarrow r_I[n] = \sum_{m=-\infty}^{\infty} s_I[m]h_I[n-m] - \sum_{m=-\infty}^{\infty} s_Q[m]h_Q[n-m] \\ Q &\uparrow r_Q[n] = \sum_{m=-\infty}^{\infty} s_Q[m]h_I[n-m] + \sum_{m=-\infty}^{\infty} s_I[m]h_Q[n-m] \end{aligned} \quad (2.11)$$

This can be written as a combination of individual real convolutions.

$$\begin{aligned} I &\rightarrow r_I[n] = s_I[n] * h_I[n] - s_Q[n] * h_Q[n] \\ Q &\uparrow r_Q[n] = s_Q[n] * h_I[n] + s_I[n] * h_Q[n] \end{aligned} \quad (2.12)$$

Due to the identity $\cos A \cos B - \sin A \sin B = \cos(A + B)$, a negative sign in I expression indicates that phases of the two aligned-axes terms are actually adding together. Obviously, the identity applies in above equations only if magnitude can be extracted as common term, but the concept of phase-alignment still holds. Similarly, the identity $\sin A \cos B + \cos A \sin B = \sin(A + B)$ implies that phases of the two cross-axes terms are also adding together in Q expression. Hence, *a complex convolution can be described as a process* that

- computes 4 real convolutions: $I * I$, $Q * Q$, $Q * I$ and $I * Q$
- *adds* by phase aligning the 2 aligned-axes convolutions ($I * I - Q * Q$) to obtain the I component
- adds the 2 cross-axes convolutions ($Q * I + I * Q$) to obtain the Q component.

These computations are shown in Figure 2.10 where a double arrow denotes a complex operation.

Computing Correlation through Convolution

Recall the definition of correlation from Eq (1.38) reproduced below.

$$\text{corr}[n] = \sum_{m=-\infty}^{+\infty} s[m] \cdot h[m-n] = \sum_{m=-\infty}^{+\infty} s[m] \cdot h[-(n-m)]$$

For this reason, the correlation can be defined in terms of convolution as

$$\text{corr}[n] = s[n] * h[-n] \quad (2.13)$$

From a viewpoint of conventional method, correlation between two signals can be computed through efficient methods devised for computing convolution, except that there is no flipping of one signal. This is because $h[-n]$ flips the signal once and convolution flips it again, hence bringing the original signal back.

From the viewpoint of intuitive method, it is clear that a negative sign with the NOW, $-n$, turns the future into past, and the past into future. Consequently, the last sample of the original signal arrives first, since it has become the farthest past.

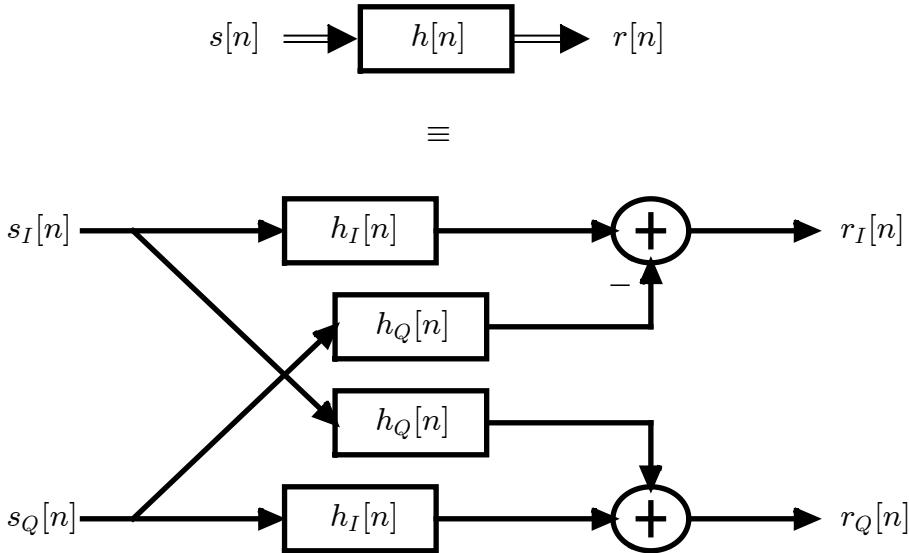


Figure 2.10: Convolution between two complex signals $s[n]$ and $h[n]$

2.4.2 Complex Sinusoids and the Grand Scheme of Things

In Section 2.3, we raised two questions about the frequency domain output for a complex sinusoidal input and their significance. To investigate these questions, let us excite an LTI system with a real impulse response $h[n]$ (i.e., $h_Q[n] = 0$) by an input complex sinusoid $V[n]$. A similar result holds for a complex impulse response.

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} V_I[n] &= \cos 2\pi \frac{k}{N} n \\ V_Q[n] &= \sin 2\pi \frac{k}{N} n \end{aligned}$$

where k/N is an arbitrary discrete frequency in the range $[-0.5, +0.5)$. The output is given by convolution of the input and system impulse response as

$$r[n] = V[n] * h[n] = \sum_{m=0}^{N-1} V[m] h[n-m] = \sum_{m=0}^{N-1} h[m] V[n-m]$$

where the last equality is due to commutative property of convolution. The I and Q components of the output are then expressed as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} r_I[n] &= \sum_{m=0}^{N-1} h[m] \cdot \cos 2\pi \frac{k}{N} (n-m) \\ r_Q[n] &= \sum_{m=0}^{N-1} h[m] \cdot \sin 2\pi \frac{k}{N} (n-m) \end{aligned}$$

Expanding by identities $\cos(A - B) = \cos A \cos B + \sin A \sin B$ and $\sin(A - B) = \sin A \cos B - \cos A \sin B$,

$$\begin{aligned} I &\rightarrow r_I[n] = \left\{ \sum_{m=0}^{N-1} h[m] \cdot \cos 2\pi \frac{k}{N} m \right\} \cos 2\pi \frac{k}{N} n + \\ &\quad \left\{ \sum_{m=0}^{N-1} h[m] \cdot \sin 2\pi \frac{k}{N} m \right\} \sin 2\pi \frac{k}{N} n \\ Q &\uparrow r_Q[n] = - \left\{ \sum_{m=0}^{N-1} h[m] \cdot \sin 2\pi \frac{k}{N} m \right\} \cos 2\pi \frac{k}{N} n + \\ &\quad \left\{ \sum_{m=0}^{N-1} h[m] \cdot \cos 2\pi \frac{k}{N} m \right\} \sin 2\pi \frac{k}{N} n \end{aligned}$$

For a real $h[n]$ from Eq (2.2) where $h[n] = h_I[n]$,

$$\begin{aligned} I &\rightarrow r_I[n] = H_I[k] \cos 2\pi \frac{k}{N} n - H_Q[k] \sin 2\pi \frac{k}{N} n \\ Q &\uparrow r_Q[n] = H_Q[k] \cos 2\pi \frac{k}{N} n + H_I[k] \sin 2\pi \frac{k}{N} n \end{aligned}$$

From the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$, the above equation can clearly be seen as a product between $H[k]$ and input complex sinusoid $V[n]$ as

$$r[n] = H[k] \cdot V[n] \quad (2.14)$$

The most amazing thing about the above equation is that the output is just a scaled version of the input $V[n]$! This is because $H[k]$ is a complex constant with respect to time index n . We conclude that if given as an input, *the same complex sinusoid appears at the output* scaled by the system frequency response at discrete frequency k/N . Since $H[k]$ is a complex constant, we can write Eq (2.14) as

$$\begin{aligned} I &\rightarrow r_I[n] = |H[k]| \cdot \cos \left(2\pi \frac{k}{N} n + \angle H[k] \right) \\ Q &\uparrow r_Q[n] = |H[k]| \cdot \sin \left(2\pi \frac{k}{N} n + \angle H[k] \right) \end{aligned} \quad (2.15)$$

The significant conclusion from the above equation is that *the output response of an LTI system to a complex sinusoid at its input is nothing but the same complex sinusoid at the same frequency but with different amplitude and phase*. The following note answers the questions asked at the beginning of this section.

Note 2.5 Why complex sinusoids?

The output of an LTI system to a complex sinusoid is the same complex sinusoid but with altered amplitude and phase. To find out how an LTI system responds in frequency

domain, the most logical method then is to probe the system at its input with a complex sinusoid at every possible frequency. This implies forming an input signal with N complex sinusoids of equal magnitudes. In this manner, the output amplitudes and phases at all available frequency bins are examined and checked how a system deals with each possible complex sinusoid. This is exactly what we had as an input to measure the frequency response in Figure 2.5. The magnitude and phase of the frequency response are called *magnitude response* and *phase response*, respectively.

It can also be concluded from the above discussion that impulse response in time domain and frequency response in frequency domain are two equivalent descriptions of a system.

2.4.3 Circular Convolution

After understanding the regular convolution both intuitively and mathematically, we want to see the interpretation of this process in frequency domain. Specifically, what is the DFT of $r[n] = s[n] * h[n]$? To answer this question, we want to apply the DFT definition to the convolution Eq (2.4), reproduced below.

$$r[n] = \sum_{m=-\infty}^{\infty} s[m]h[n-m]$$

However, we saw in Section 1.8 that both discrete time and discrete frequency domains are defined within a range comprising of N samples only. Figure 1.51 also demonstrated that sampling in frequency axis by the DFT creates time aliases in time domain and a major implication of this periodic extension is that sequence shifts for DFT purpose actually result in circular shifts. That leads us to circular convolution.

Circular Convolution in Time Domain

Circular convolution between two signals, denoted by \circledast , is very similar to regular convolution except that the flipping and time shifts are circular.

$$\begin{aligned} r[n] &= s[n] \circledast h[n] \\ &= \sum_{m=0}^{N-1} s[m]h[(n-m) \bmod N] \end{aligned} \quad (2.16)$$

Instead of working out the steps as we did in convolution, we just illustrate this process through Figure 2.11 of the same signals

$$\begin{aligned} s[n] &= [2 \ -1 \ 1] \\ h[n] &= [-1 \ 1 \ 2] \end{aligned}$$

You would have noticed one problem here: different choices of N will lead to different outputs.

$N > 5$: The length of the regular convolution output is $3 + 3 - 1 = 5$ and it was demonstrated in Figure 2.9. When $N > 5$, even if the circular flip and shifts are applied,

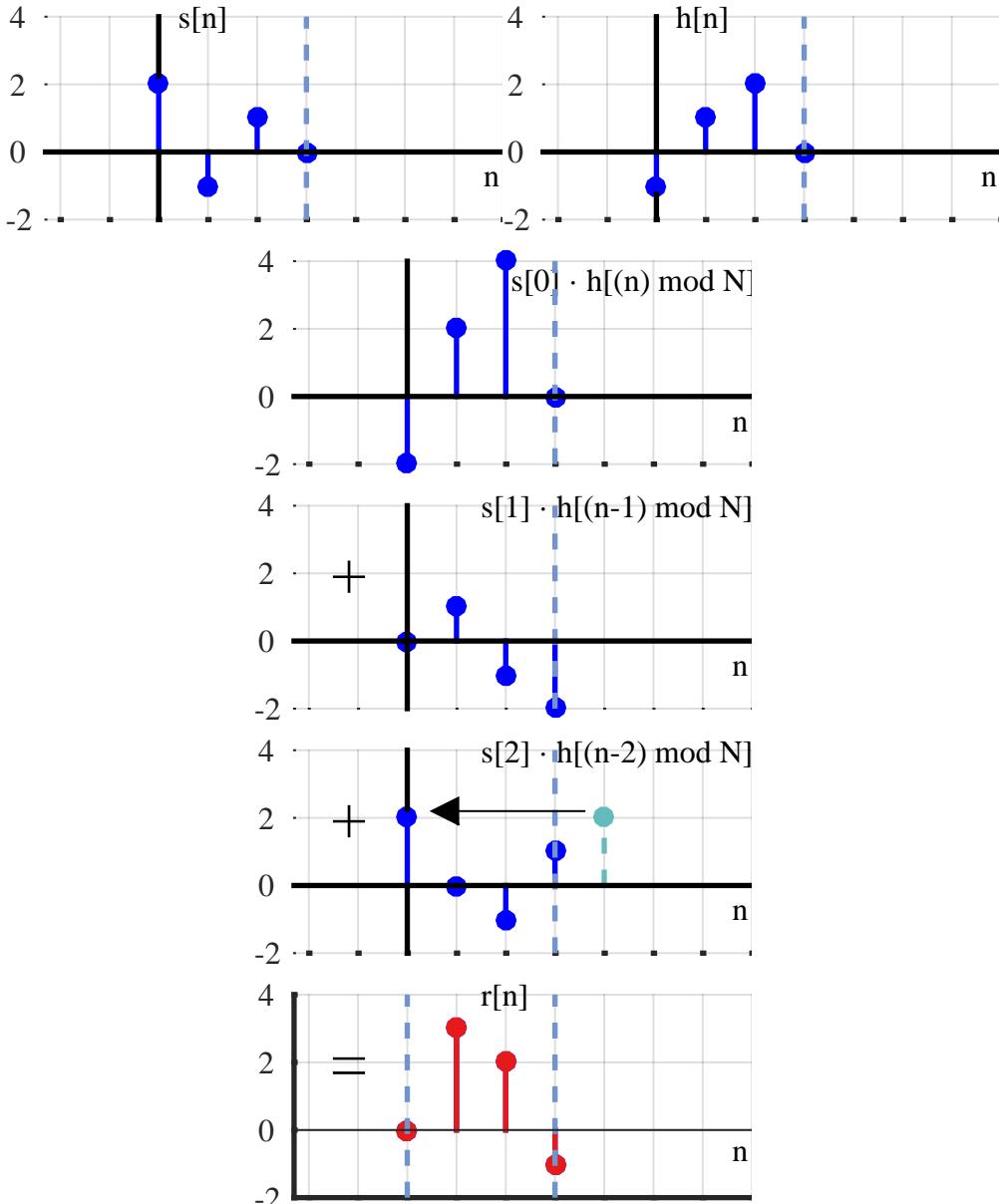


Figure 2.11: Circular convolution between $s[n]$ and $h[n]$ for $N = 4$

it will only result in zeros crossing the boundary of $N - 1$ from the right and entering towards the left. Consequently, the result of the circular convolution is the same as the regular convolution.

$$r[n] = [-2 \ 3 \ 2 \ -1 \ 2] \quad (2.17)$$

N = 4: For $N = 4$, we need to insert a zero at the right of both $s[n]$ and $h[n]$ before applying the circular flip and shifts. For the intuitive method, this is demonstrated

in Figure 2.11 where the output of the circular convolution is produced from column-wise summation.

$$r[n] = [0 \ 3 \ 2 \ -1]$$

Interestingly, the circular convolution output can be obtained from the regular convolution by circular shifting and addition after the procedure. For example, only one sample in Eq (2.17) is out of the boundary set by $N = 4$ which needs to be circularly brought back in from the left side. By adding this value 2 to the value -2 at $n = 0$, we get the same $r[n] = [0 \ 3 \ 2 \ -1]$.

$N = 3$: In this case, the whole procedure revolves around just 3 samples. Verify that the output is

$$r[n] = [-3 \ 5 \ 2]$$

Just like before, this result can be seen from utilizing the regular convolution. In Eq (2.17), two samples, -1 and 2 , are out of the boundary of $N = 3$ which can be brought back in from the left side and added to the values -2 and 3 at $n = 0$ and $n = 1$, respectively.

An everyday example of circular convolution is the dissimilarity between the solar and lunar years which consist of approximately 365 and 354 days, respectively. Due to this ≈ 11 days difference, the remaining part of the lunar year continues from Dec 31 to Jan 1 of the next year and the partial overlap keeps shifting each year. This is illustrated in Figure 2.12.

Circular Convolution in Frequency Domain

Let us compute the DFT $R[k]$ of the resultant signal $r[n]$. We focus on real signals as the derivation for complex signals is similar but a bit more complicated. Since $r[n]$ is a real signal in our case, $r_I[n] = r[n]$ and $r_Q[n] = 0$, and the I component in frequency domain $R_I[k]$ from the DFT definition can be expressed as

$$\begin{aligned} I &\rightarrow R_I[k] = \sum_{n=0}^{N-1} r_I[n] \cos 2\pi \frac{k}{N} n \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s[m] h[(n-m) \bmod N] \cos 2\pi \frac{k}{N} n \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s[m] h[(n-m) \bmod N] \cos 2\pi \frac{k}{N} (n-m+m) \end{aligned}$$

Using the identities $\cos(A+B) = \cos A \cos B - \sin A \sin B$,

$$\begin{aligned} I \rightarrow R_I[k] &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s[m] h[(n-m) \bmod N] \cos 2\pi \frac{k}{N} (n-m) \cos 2\pi \frac{k}{N} m \\ &\quad - \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} s[m] h[(n-m) \bmod N] \sin 2\pi \frac{k}{N} (n-m) \sin 2\pi \frac{k}{N} m \end{aligned}$$

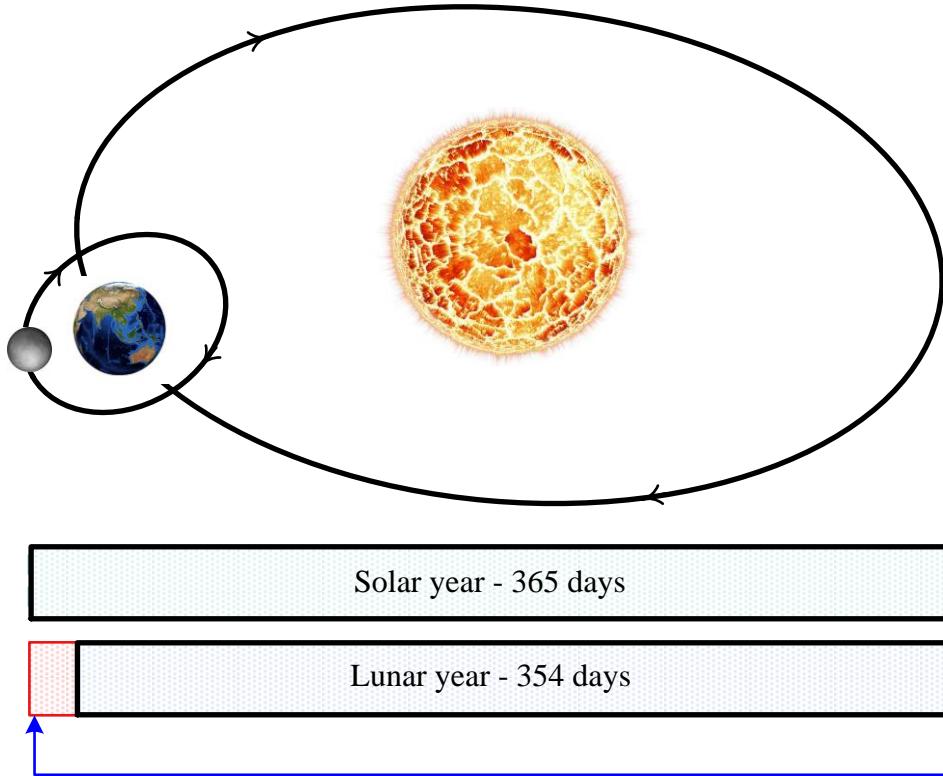


Figure 2.12: Circular convolution between solar and lunar years

The above equation can be rearranged as

$$I \rightarrow R_I[k] = \sum_{m=0}^{N-1} s[m] \cos 2\pi \frac{k}{N} m \sum_{n=0}^{N-1} h[(n-m) \bmod N] \cos 2\pi \frac{k}{N} (n-m) \\ - \sum_{m=0}^{N-1} s[m] \sin 2\pi \frac{k}{N} m \sum_{n=0}^{N-1} h[(n-m) \bmod N] \sin 2\pi \frac{k}{N} (n-m)$$

Since both $s[n]$ and $h[n]$ are real for the purpose of this derivation, while $\cos(\cdot)$ and $\sin(\cdot)$ are periodic signals, we get

$$I \rightarrow R_I[k] = S_I[k] \cdot H_I[k] - S_Q[k] \cdot H_Q[k]$$

A similar computation for Q component yields

$$Q \rightarrow R_Q[k] = S_Q[k] \cdot H_I[k] + S_I[k] \cdot H_Q[k]$$

Combining both equations above and comparing with multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$,

$$R[k] = S[k] \cdot H[k] \quad (2.18)$$

Therefore, we arrive at an extremely important result that forms the backbone of most DSP applications.

Note 2.6 Convolution \xrightarrow{F} Multiplication

For discrete signals, circular convolution between two signals in time domain is equivalent to sample-by-sample multiplication of those signals in frequency domain.

$$s[n] \circledast h[n] \xrightarrow{F} S[k] \cdot H[k] \quad (2.19)$$

The above relation is true for both real and complex signals.

The inverse relation also holds. For discrete signals, multiplication of two time domain signals results in their circular convolution in frequency domain.

$$s[n] \cdot h[n] \xrightarrow{F} \frac{1}{N} \cdot S[k] \circledast H[k] \quad (2.20)$$

Circular Correlation in Frequency Domain

Recall that the correlation between two complex signals involves conjugation of the second signal because a negation in one's phase cancels out the other and perfectly aligns the signal components for maximum similarity. Here, utilizing circular convolution in Eq (2.13), we can write

$$\text{corr}[n] = s[n] \circledast h^*[(-n) \bmod N] \quad (2.21)$$

which leads to

$$s[n] \circledast h^*[(-n) \bmod N] \xrightarrow{F} S[k] \cdot H^*[k] \quad (2.22)$$

The relation between $h^*[(-n) \bmod N]$ and $H^*[k]$ will be established through the DFT definition in Section 2.9. We conclude that the product of the DFT $S[k]$ with the DFT $H[k]$ produces their circular convolution in time domain while the product of $S[k]$ with $H^*[k]$ produces their circular correlation in time domain.

Cross and Auto-Correlation

The correlation above is defined between two different signals and hence is naturally called *cross-correlation*. When a signal is correlated with itself, it is called *auto-correlation*. It is defined by setting $h[n] = s[n]$ in Eq (2.21) as

$$\begin{aligned} \text{corr}[n] &= s[n] \circledast s^*[(-n) \bmod N] \\ &= \sum_{m=-\infty}^{\infty} s[m]s^*[(-m-n) \bmod N] \end{aligned} \quad (2.23)$$

An example of auto-correlation of a rectangular signal is drawn in Figure 2.13 for N greater than twice the rectangle length L so that a regular shift and a circular shift become the same. As a consequence of one rectangle sliding over another sample-by-sample, its auto-correlation is a triangular pulse shape with duration clearly from $-(L-1)$ to $L-1$.

- For $n = -L$, there is no overlap of the rectangular signal with itself and hence the auto-correlation is zero.

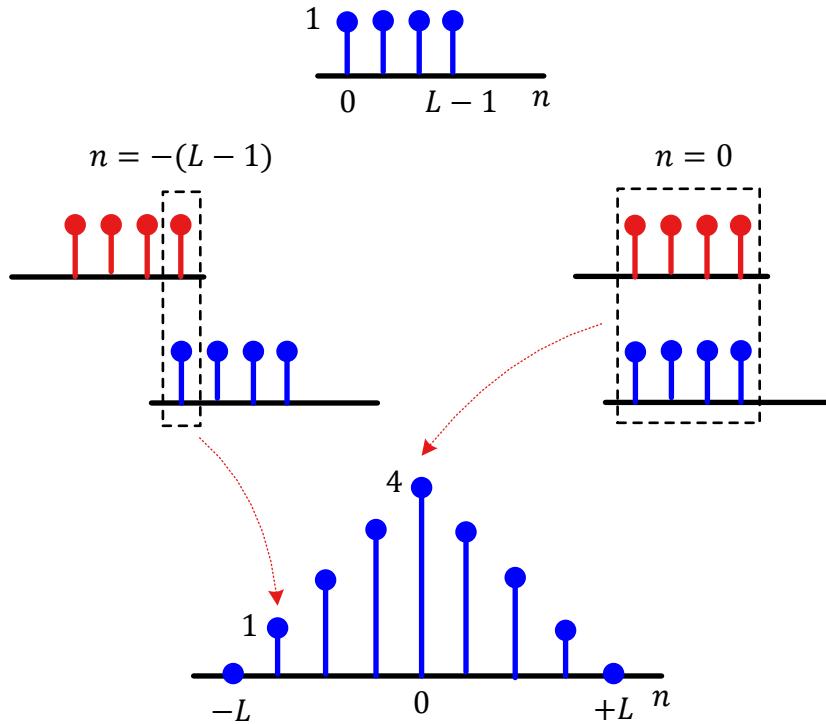


Figure 2.13: Auto-correlation of a rectangular pulse. Observe the output for different time shifts with peak at $n = 0$

- For $n = -(L - 1)$, there is an overlap of one sample that appears at the output.
- For $n = 0$, the rectangles are completely aligned and the maximum correlation occurs.

From this last point, an interesting fact to note is that

$$\text{corr}[0] = \sum_{m=-\infty}^{\infty} s[m]s^*[m] = \sum_{m=-\infty}^{\infty} |s[m]|^2 = E_s \quad (2.24)$$

which is the energy of the signal $s[n]$. This is illustrated as the peak value 4 in Figure 2.13. Since correlation is a measure of resemblance, this indicates that a signal resembles itself the most. The lower peaks on the sides can be thought of coming from the correlation of a signal with the same structure as the actual signal in some ways (e.g., rectangular) but not in others (e.g., with time shifts), just like your siblings resemble you more than anyone else outside your home but not in every respect.

Remember that another signal can have a large amount of energy such that the result of its cross-correlation with $s[n]$ can be greater than the auto-correlation of $s[n]$, which intuitively should not happen. Normalized correlation $\overline{\text{corr}}[n]$ is defined in Eq (2.25) in a way that the maximum value of 1 can only occur for correlation of a

signal with itself.

$$\begin{aligned}\overline{\text{corr}}[n] &= \frac{\sum_{m=-\infty}^{\infty} s[m]h^*((m-n) \bmod N)}{\sqrt{\sum_{m=-\infty}^{\infty} |s[m]|^2} \cdot \sqrt{\sum_{m=-\infty}^{\infty} |h[m]|^2}} \\ &= \frac{1}{\sqrt{E_s} \cdot \sqrt{E_h}} \sum_{m=-\infty}^{\infty} s[m]h^*((m-n) \bmod N)\end{aligned}\quad (2.25)$$

Now regardless of the energy in the other signal, its normalized cross-correlation with another signal cannot be greater than the normalized auto-correlation of a signal due to both energies appearing in the denominator.

Energy Spectral Density

Taking the DFT of auto-correlation of a signal and utilizing Eq (2.22), we get

$$\text{corr}[n] \xrightarrow{\mathcal{F}} S[k] \cdot S^*[k] = |S[k]|^2 \quad (2.26)$$

The expression $|S[k]|^2$ is called the *Energy Spectral Density* because as we will learn in Section 2.9, the signal energy in time domain is related to that in frequency domain through Parseval's relation.

$$E_s = \sum_{n=0}^{N-1} |s[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |S[k]|^2 \quad (2.27)$$

Thus, energy of a signal can be obtained by summing the energy $|S[k]|^2$ in each frequency bin (up to a normalizing constant $1/N$). Accordingly, $|S[k]|^2$ can be termed as energy per spectral bin, or energy spectral density.

Figure 2.14 plots the energy spectral density for a rectangular signal whose auto-correlation was illustrated in Figure 2.13. If the shape looks familiar, it is because the DFT $S[k]$ a rectangular signal is a sinc function, and hence its energy spectral density $|S[k]|^2$ is sinc squared.

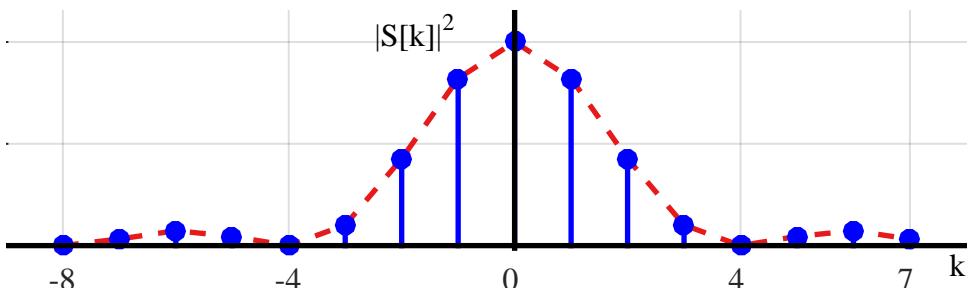


Figure 2.14: Energy spectral density of a rectangular signal whose auto-correlation was illustrated in Figure 2.13

From the above discussion, there are two ways to find the spectral density of a signal:

1. Take the magnitude squared of the DFT of a signal.
2. Take the DFT of the signal auto-correlation.

Next, we study the design of Finite Impulse Response (FIR) filters that are used in several blocks during the Tx and Rx stages of the communication system.

2.5 Finite Impulse Response (FIR) Filters

At this stage, we know that most signals of practical interest can be considered as a sum of complex sinusoids oscillating at different frequencies. The amplitudes and phases of these sinusoids shape the frequency contents of that signal and are drawn through magnitude response and phase response, respectively. In DSP, a design objective is to modify the frequency contents of an input signal in a prescribed manner to obtain a desired output. This operation is called *filtering* and it is the most fundamental operation in the whole field of DSP.

It is possible to design a system, or filter, such that some desired frequency contents of the signal can be passed through by specifying $H(F) \approx 1$ for those values of F , while other frequency contents can be suppressed by specifying $H(F) \approx 0$ (although the idea is right, this is not a straightforward task as we shortly see). The concept is similar to a water filter that treats impurities in the water and an oil filter that removes contaminants from the engine oil. The terms lowpass, highpass and bandpass then make intuitive sense where low, high and band refer to desired spectral contents that need to be passed while blocking the rest.

Filter Frequency Response

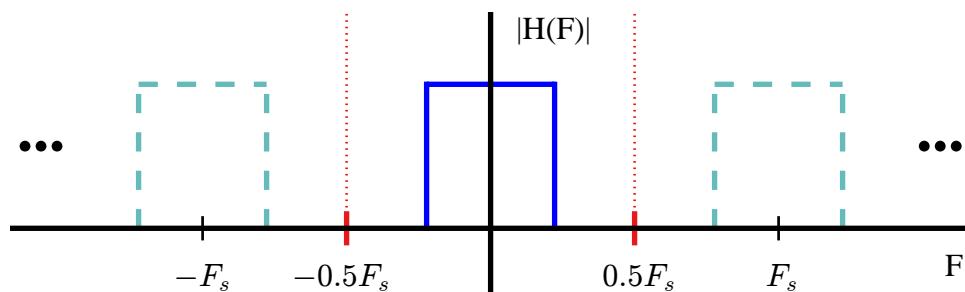


Figure 2.15: An ideal lowpass filter has a brickwall response

Figure 2.15 shows the magnitude response $|H(F)|$ (as a function of continuous frequency) of an ideal lowpass filter. A lowpass filter passes frequencies near 0 while blocks the remaining frequencies. In a continuous frequency world, the middle filter is all that exists. However, in a sampled world, the frequency response of the filter – just like a sampled signal – repeats at intervals of F_s (sampling frequency) and there are infinite spectral replicas on both sides shown with dashed lines in Figure 2.15. The baseband of the sampled frequency response is from $-0.5F_s$ to $+0.5F_s$ shown within dotted red lines.

On the other hand, a highpass filter passes both positive and negative frequencies near $0.5F_s$ while blocking everything else, as observed through the symmetry around

$\pm 0.5F_s$ in Figure 2.16. Similarly, bandpass filters only allow a specific portion of the spectrum within these two extremes.

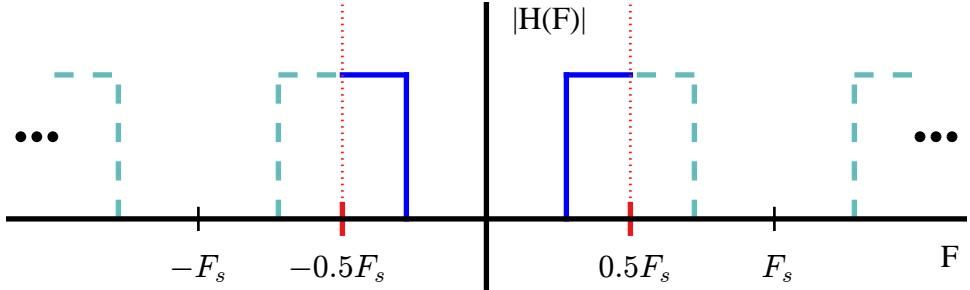


Figure 2.16: An ideal highpass filter

FIR Filter Design

Since a filter is a system that modifies the spectral contents of a signal, it naturally has an impulse response as well. When the length of this impulse response is finite, the filter is called a *Finite Impulse Response (FIR)* filter that is a discrete-time sequence and its samples are called filter coefficients.

We could accomplish this task by simply computing the inverse transform of the ideal frequency response $H(F)$. Since ideal filters have unity passband and zero stopband and a direct transition between these two (like a rectangular signal), this finite frequency domain support produces an impulse response which is infinite and hence non-realizable. Why? Because the inverse transform of a rectangular signal in continuous frequency domain is a sinc signal with an infinite support in time domain.

Assume that this inverse transform is still obtained and sampled to produce a discrete-time signal $h[n]$. Now when it is truncated on the two sides, it is equivalent to multiplication of the ideal infinite $h[n]$ by a rectangular window $w_{\text{Rect}}[n]$ as

$$h_{\text{Truncated}}[n] = h[n] \cdot w_{\text{Rect}}[n]$$

As mentioned previously, *windowing* is a process of pruning an infinite sequence to a finite length. We know that multiplication in time domain is convolution in frequency domain and that the spectrum of a time domain rectangular window is a sinc signal. Thus, convolution between the spectra of the ideal filter and this sinc signal generates the actual frequency response $H(F)$ of the filter.

$$H(F) = \text{Ideal brickwall response} * \text{sinc}(F)$$

This is drawn in Figure 2.17. This frequency domain sinc signal has a mainlobe as well as decaying oscillations that extend from $-\infty$ to ∞ .

- Convolution with these decaying oscillations is the reason why a realizable FIR filter always exhibits ripple in the passband and in the stopband.
- The mainlobe of the sinc signal – when convolved with the brickwall spectrum – generates a transition bandwidth between the passband and the stopband. Naturally, this transition bandwidth is equal to the width of the sinc mainlobe, which in turn is a function of the length of the rectangular window in time.

Eventually, truncation with a window causes an FIR filter to have an impulse response $h[n]$ of finite duration.

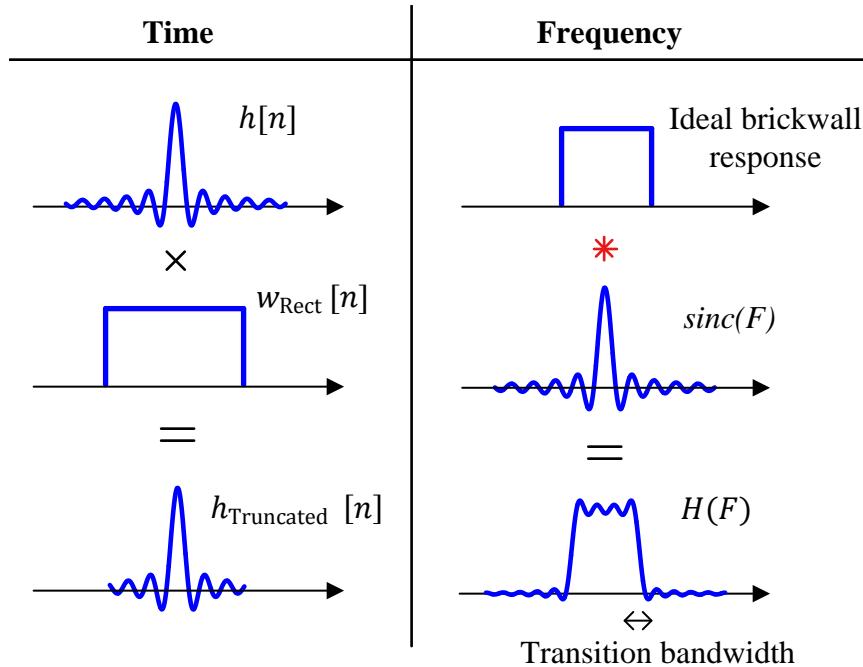


Figure 2.17: Impulse response truncation process in time domain and its consequences in frequency domain

Sidelobes appearing in the stopband of the filter, transition bandwidth and ripple in the passband characterize the performance of a filter. These parameters are usually specified in terms of *decibels (or dBs)*. The filter magnitude in dB is given by

$$|H(F)|_{dB} = 20 \log_{10} |H(F)|$$

It should be remembered that the multiplicative factor 20 above is replaced by 10 for power conversions (because power is a function of voltage or current squared). The *advantage of converting magnitudes to dBs* for plotting purpose is that very small and very large quantities can be displayed clearly on a single plot.

A filter is designed in accordance with the parameters in Table 2.2 and shown graphically in Figure 2.18.

FIR filter design basically requires finding the values of filter taps (or coefficients) that translate into a desired frequency response. Many software routines are available to accomplish this task. A standard method for FIR filter design is the *Parks-McClellan algorithm*. The Parks-McClellan algorithm is an iterative algorithm for finding the optimal FIR filter such that the maximum error between the desired frequency response and the actual frequency response is minimized. Filters designed this way exhibit an equiripple behavior in their frequency responses and are sometimes called *equiripple filters*. The term equiripple implies equal ripple within the passband and within the stopband, but not necessarily equal in both bands.

Next, we design an FIR filter with the help of an example.

Table 2.2: Filter specification parameters

F_S	Sample rate
F_{pass}	Passband edge frequency
F_{stop}	Stopband edge frequency
δ_{pass}	Maximum passband ripple
δ_{stop}	Maximum stopband ripple

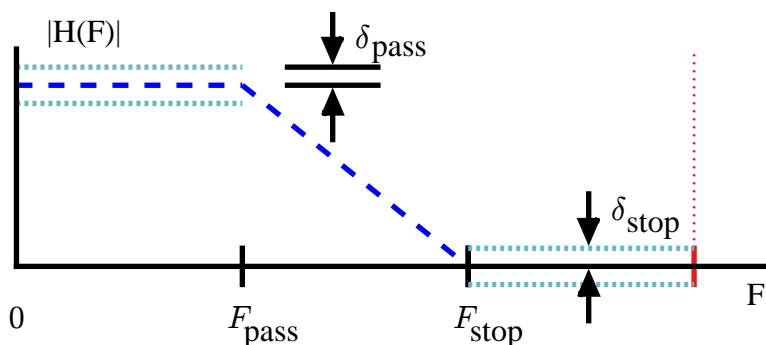


Figure 2.18: Parameter specification for sampled lowpass filter

Example 2.3

We design a lowpass FIR filter through Parks-McClellan algorithm in Matlab that meets the following specifications.

Sample rate $F_S = 12 \text{ kHz}$
 Passband edge $F_{\text{pass}} = 2 \text{ kHz}$
 Stopband edge $F_{\text{stop}} = 3 \text{ kHz}$
 Passband ripple $\delta_{\text{pass,dB}} = 0.1 \text{ dB}$
 Stopband ripple $\delta_{\text{stop,dB}} = -60 \text{ dB}$

First, noting from Figure 2.18 that the maximum amplitude in the passband of frequency domain is $1 + \delta_{\text{pass}}$, we can convert dB units into linear terms as

$$\begin{aligned}
 20 \log_{10} (1 + \delta_{\text{pass}}) &= \delta_{\text{pass,dB}} \\
 1 + \delta_{\text{pass}} &= 10^{\delta_{\text{pass,dB}}/20} \\
 \delta_{\text{pass}} &= 0.012
 \end{aligned}$$

For the stopband,

$$\begin{aligned}
 20 \log_{10} \delta_{\text{stop}} &= \delta_{\text{stop,dB}} \\
 \delta_{\text{stop}} &= 10^{-60/20} = 0.001
 \end{aligned}$$

A Matlab function `firpm()` returns the coefficients of a length $N + 1$ linear phase FIR filter. For a trial value of $N = 29$,

$$\begin{aligned} h &= \text{firpm}(N - 1, [0 \ F_{\text{pass}} \ F_{\text{stop}} \ 0.5F_S]/(0.5F_S), [1 \ 1 \ 0 \ 0]) \\ &= \text{firpm}(28, [0 \ 2000 \ 3000 \ 6000]/6000, [1 \ 1 \ 0 \ 0]) \end{aligned}$$

where the vector $[1 \ 1 \ 0 \ 0]$ specifies the desired amplitude at band edges of passband and stopband, respectively. The normalization by $0.5F_S$ is of no significance as it is just a requirement of this Matlab function. Furthermore, the filter is designed with $N + 1$ coefficients by Matlab and that is why we used $N - 1$ as its argument.

If you try this little code, you will see that the result thus obtained had stopband levels of only around -45 dB and a passband ripple of 0.05 dB. The sidelobes do not meet the 60 dB attenuation while the passband ripple exceeds the requirement of 0.1 dB. Therefore, N can be increased until 60 dB attenuation is achieved for $N = 41$. To avoid excessive filter length increase which increases the processor workload, different weighting factors can be employed for passband and stopband ripples as well. For example, a weighting factor of $1 : 10$ for passband to stopband ($[1 \ 10]$ in vector form) yields,

$$h = \text{firpm}(28, [0 \ 2000 \ 3000 \ 6000]/6000, [1 \ 1 \ 0 \ 0], [1 \ 10])$$

In this case, the sidelobe level reached -55 dB, a 10 dB improvement for the same length $N = 29$ when there were no band weights. Next, the filter order is gradually increased until it meets the desired specifications at $N = 33$, a difference of 8 taps from $N = 41$ and no weighting factors. Such trial and error approach is often necessary to design a filter with given specifications.

The impulse response $h[n]$ and frequency response $20 \log_{10} |H(F)|$ of the final filter are drawn in Figure 2.19. Note the stopband attenuation of 60 dB and passband ripple within 0.1 dB.

We learned in the above example that the higher the number of taps, the more closely the filter follows an expected response (e.g., lowpass, highpass, bandpass, etc.) at a cost of increased computational complexity. Therefore, the defining feature of an FIR filter is the number of coefficients L_{FIR} .

While the design example seemed simple, there are several things to consider when designing a filter such as the DSP platform, hardware, processor memory, clock speed, dynamic range and, according to fred harris, your spouse's birthday. Therefore, a filter should always be designed from a system point of view instead of an isolated component in a system, i.e., in addition to filtering the input signal, several other secondary tasks can also be folded into the design. This helps in reduced computational load, an example of which we will see in Chapter 10.

Convolution

Since the FIR filter with fixed coefficients is an LTI system, the output is a convolution between its impulse response $h[n]$ and the input signal $s[n]$. For an FIR filter of length L_{FIR} ,

$$r[n] = \sum_{m=0}^{L_{\text{FIR}}-1} h[m]s[n-m] \quad (2.28)$$

Figure 2.20 illustrates a 5 tap FIR filter. To implement convolution, an FIR filter is built of a bunch of multipliers, adders and delay elements. A delay element is just a clocked

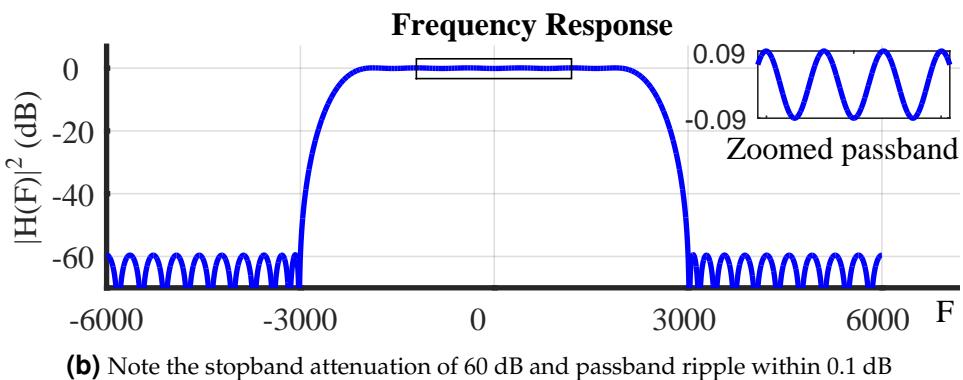
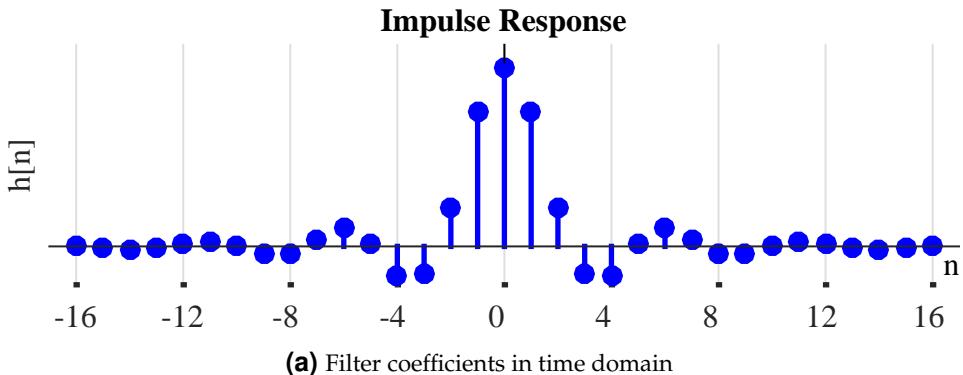


Figure 2.19: Time and frequency response of a lowpass FIR filter designed with Parks-McClellan algorithm for $N = 33$

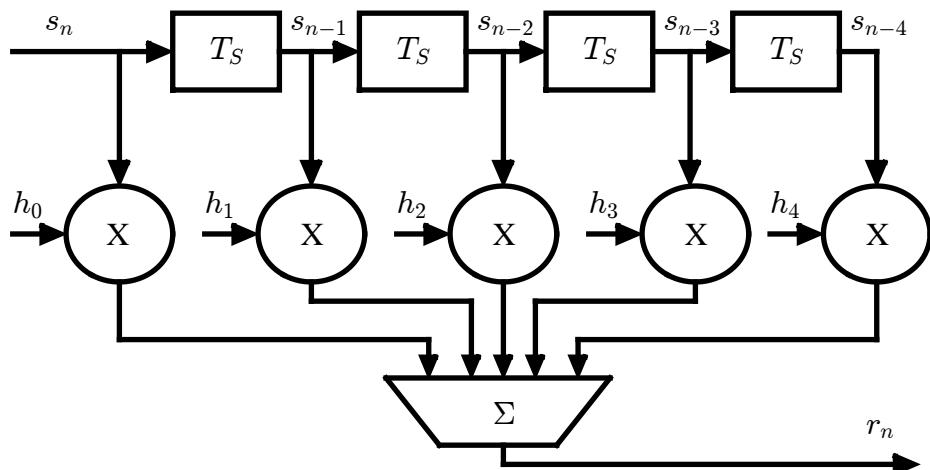


Figure 2.20: Filtering a signal $s[n]$ with an FIR filter with impulse response $h[n]$. Observe the convolution between the two sequences appearing at the output

register to store the input values and is represented with a T_S symbol (a single sample delay).

Taking into account the structure in Figure 2.20, this 5 tap filter output can be mathematically written as

$$r[n] = h[0]s[n] + h[1]s[n - 1] + h[2]s[n - 2] + h[3]s[n - 3] + h[4]s[n - 4]$$

At each clock cycle, samples from input data and filter taps are multiplied and the outputs of all the multipliers are added to form a single output. On the next clock cycle, the input data samples are shifted by 1 to accommodate a new arriving sample relative to filter taps, and the process continues.

Due to the series of the delay elements, this structure is commonly known as a *tapped delay-line filter* or a *transversal filter*. If we had a series of two-input adders, each of the two inputs to the adder is coupled with a fixed coefficient multiplier to form a Multiply-Accumulate (MAC). The other option is a transposed structure of the same equation in which the registers operate as partial sum accumulators.

Group Delay, Linear Phase and Coefficient Symmetry

Assume that a signal $s[n]$ needs to be lowpass filtered without any distortion. It seems that the task of the filter with frequency response $H[k]$, then, is

- to remove all the spectral contents outside the signal bandwidth, and
- to pass the spectral contents within the signal bandwidth unharmed, i.e., it should have as flat a passband as possible.

However, in this description, we are ignoring the role of phase response of the filter which also has the capability of harming the waveform shape, even when it is not suspected by focusing at the filter magnitude response only. Consequently, we can write the frequency response of the filter $H[k]$ as

$$|H[k]| = \begin{cases} 1 & -\frac{k_{\text{pass}}}{N} < k < +\frac{k_{\text{pass}}}{N} \\ 0 & \text{elsewhere} \end{cases}$$

$$\angle H[k] = ?$$

where k_{pass}/N is the passband frequency. Due to this magnitude response, the output of this lowpass filter is the same as the input signal $s[n]$. The question is: what kind of phase response should it have such that the waveform shape is preserved after the filtering operation?

One fact we know is that like all practical systems, the filtering operation is going to introduce a delay between the input and the output signals. Now recall from Section 1.9 that the DFT $\tilde{S}[k]$ of a signal $s[(n - n_0) \bmod N]$ has an unaltered magnitude while its phase is rotated by $-2\pi(k/N)n_0$ *for each k*.

$$|\tilde{S}[k]| = |S[k]|$$

$$\angle \tilde{S}[k] = \angle S[k] - 2\pi \frac{k}{N} n_0$$

Also remember that the output of a filter in frequency domain is the product of the input and filter spectra. In complex operations, it is the product of magnitude responses and an *addition* of phase responses.

From the above two facts, we can deduce that within k of interest, if the filter has a phase response given by

$$\angle H[k] = -2\pi \frac{k}{N} n_0 \quad (2.29)$$

where n_0 is some delay, then the output signal will be exactly a delayed version of the input signal. The phase response seen here is a linear function of discrete frequency k/N . If the phase is non-linear, then the delay introduced in the waveform is not proportional to the frequency, thus different constituent sinusoids get delayed by different amounts of time distorting the signal shape.

To find this delay n_0 , the length of the output signal $r[n]$ due to convolution between $s[n]$ and $h[n]$ is given by Eq (2.8) which is repeated below.

$$\text{Length}\{r[n]\} = \text{Length}\{s[n]\} + \text{Length}\{h[n]\} - 1$$

which is longer than the input by $\text{Length}\{h[n]\} - 1$ samples. Assuming an odd filter length, we can say that every FIR filter causes the output signal to be delayed by a number of samples given by

$$n_0 = \frac{\text{Length}\{h[n]\} - 1}{2} \quad (2.30)$$

This is known as *group delay* of the filter. For our purpose, it corresponds to the peak of the impulse response in the middle (see Figure 2.19a) and hence the factor of 2 in the denominator. As a result, n_0 samples of the output at the start – when the filter response is moving into the input sequence – can be discarded. Similarly, the last n_0 samples – when the filter is moving out of the input sequence – can be discarded as well.

Group delay in this context is very similar to many everyday examples such as a warmup before getting into full swing exercise and stretches in the end, start up and shut down times of a computer and autumn/spring seasons before and after the actual long winters/summers.

In more precise terms, group delay is defined as the negative of the derivative[†] of the phase response with respect to the frequency. On the other hand, phase delay is defined as the negative of the phase response divided by the frequency. For a linear phase filter from Eq (2.29), both are the same.

$$-\frac{1}{2\pi} \frac{d\angle H(F)}{dF} = n_0 = -\frac{1}{2\pi} \frac{\angle H[k]}{k/N}$$

Maintaining an exact linear phase in an FIR filter is a straightforward task (but not in general filter design) as follows. Assume that an FIR filter with an odd length has symmetric coefficients.

- Symmetry in coefficients implies that for each sample at time n_0 , there is a similar sample at time $-n_0$, as the signal is centered at 0.
- These two samples are two impulses in time domain which form a real sinusoid in frequency domain that represents the spectral amplitude with zero phase.

[†]We shortly explain the derivative of a signal.

- Therefore, the only phase comes from the delay that arises from shifting the first sample of the impulse response to time 0. A shift in time implies a linear phase, see Eq (1.58).

This is why you will usually see a symmetric impulse response of the FIR filters and this is one of the main reasons for widespread use of FIR filters in DSP applications.

2.6 Some Useful FIR Filters

Now we will study some types of FIR filters that will be useful in our applications of Rx design in later chapters. We begin with a moving average filter.

Moving Average Filter

A commonly used filter in DSP applications is a moving average filter. In today's world with extremely fast clock speeds of the general purpose microprocessors, it seems strange that an application would require simple operations. But that is exactly the case with most embedded systems that run on limited battery power and consequently host small microcontrollers. For filtering applications, a moving average filter can be implemented with only a few adders and delay elements. It is most helpful when a relatively flat passband response and substantial suppression of stopband sidelobes are less important than having a basic lowpass filtering functionality.

As the name implies, a length- M moving average filter averages M input signal samples to generate one output sample. Mathematically, it can be written as

$$r[n] = \frac{1}{M} \sum_{m=0}^{M-1} s[n-m] \quad (2.31)$$

$$\begin{aligned} &= \sum_{m=0}^{M-1} \frac{1}{M} \cdot s[n-m] \\ &= \sum_{m=0}^{M-1} h[m]s[n-m] \end{aligned} \quad (2.32)$$

where as usual, $s[n]$ is the input signal, $r[n]$ is the system output and $h[n]$ is the impulse response of the moving average filter given by

$$h[n] = \frac{1}{M}, \quad n = 0, 1, \dots, M-1 \quad (2.33)$$

The above discussion leads us to the following observations.

Impulse response: From Eq (2.32), the impulse response of a moving average filter is a rectangular pulse which is drawn in Figure 2.21 for $M = 5$ and $M = 11$. In terms of filtering, a constant amplitude implies that it treats all input samples with equal significance. This results in a smoothing operation of a noisy input, see Ref. [6], while a scaling by $1/M$ ensures a unity gain.

To see an example of noise reduction in time domain, consider a pulse in Figure 2.22 in which random noise is added. Observe that a longer filter performs a better smoothing action on the input signal. However, the longer length translates into wider edges during the transition.

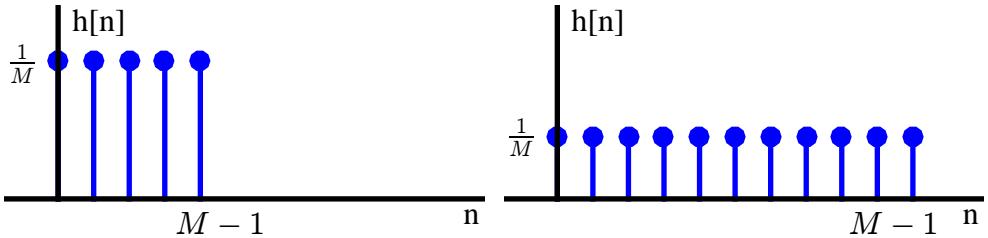


Figure 2.21: Coefficients of a moving average filter in time domain

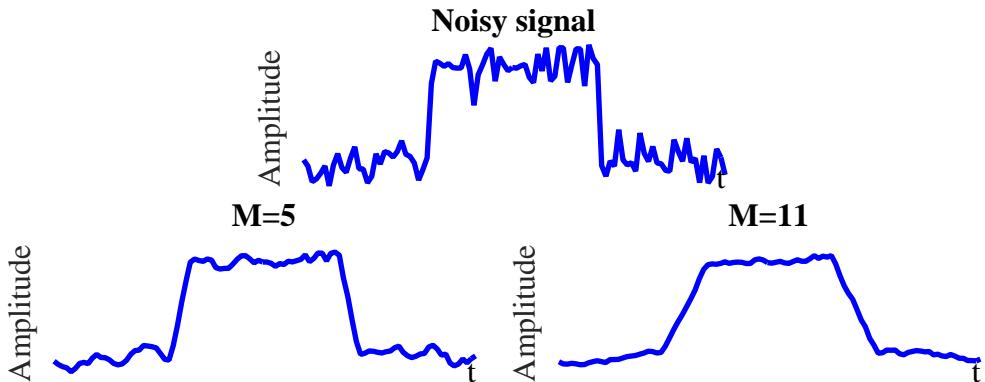


Figure 2.22: A noisy signal and its smoothed versions for two different filter lengths

Magnitude response: As a consequence of its flat impulse response, the frequency response of a moving average filter is the same as the frequency response of a rectangular pulse, i.e., a sinc signal. The magnitude response of such a filter is drawn in Figure 2.23. It is far from an ideal brickwall response of an ideal lowpass filter but the magnitude response still has a shape that passes low frequencies with little distortion and attenuates high frequencies to some extent. Due to the inverse relationship between time and frequency domains, a longer filter in time domain produces a narrower frequency response. This can be observed in the impulse and frequency responses for filter lengths $M = 5$ and $M = 11$.

Due to this insufficient suppression of sidelobes, it is probably the worst low-pass filter one can design. On the other hand, as described earlier, its simplicity of implementation makes it useful for many applications requiring high sample rates including wireless communication systems. In such conditions, a cascade of moving average filters is implemented in series such that each subsequent filter inflicts more suppression to higher frequencies of the input signal.

Discrete-Time Integrator

An integrator is a very important filter that proves useful in implementation of many blocks of a communication receiver. In continuous-time case, an integrator finds the area under the curve of a signal amplitude. On the other hand, a discrete-time system deals with just the signal samples and hence a discrete-time integrator serves the purpose of *collecting a running sum of past samples* for an input signal. Looking at an

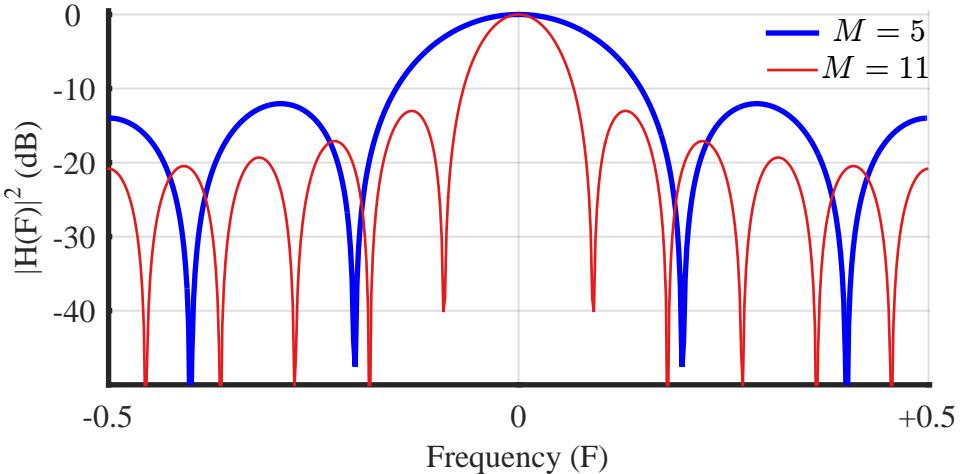


Figure 2.23: Magnitude response of a moving average FIR filter in frequency domain for $M = 5$ and $M = 11$. A filter longer in time domain produces a response narrower in frequency domain

infinitesimally small scale, this is the same as computing the area under the curve of a signal sampled at an extremely high rate. For the following discussion, we assume a normalized sample time $T_S = 1$.

For an input $s[n]$ and output $r[n]$, there are three main methods to implement a discrete-time integrator.

Forward difference: The forward difference integrator is realized through the following relation.

$$\begin{aligned} r[n] &= \sum_{i=-\infty}^n s[i] = \sum_{i=-\infty}^{n-1} s[i] + s[n] \\ &= r[n-1] + s[n] \end{aligned} \quad (2.34)$$

For obvious reasons, the running sum $r[n-1]$ is a common component in all types of integrators; the distinct factor is the term added to $r[n-1]$ as a replacement for area under the curve. For a forward difference integrator, this additional term is $s[n]$: the current input. This is drawn in Figure 2.24 where T_S represents a delay of one sample time. Notice that the forward difference integrator computes its output *after the arrival of the current sample*, i.e., at time n .

The block diagram shown in the figure will be used to implement a forward difference integrator in the loop filter of a Phase Locked Loop (PLL) later.

Backward difference: The backward difference integrator is realized through the following relation.

$$r[n] = r[n-1] + s[n-1] \quad (2.35)$$

Here, the term added to $r[n-1]$ is the previous input $s[n-1]$. This is also illustrated in Figure 2.24. In contrast to the forward difference case, the backward difference integrator can compute its output *after the last sample*, i.e., at time $n-1$.

Forward difference

$$r[n] = r[n - 1] + s[n]$$

Backward difference

$$r[n] = r[n - 1] + s[n - 1]$$

Area = width . height

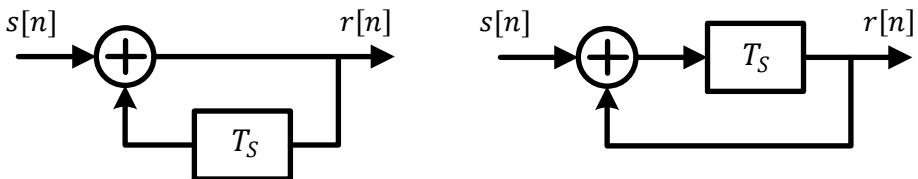
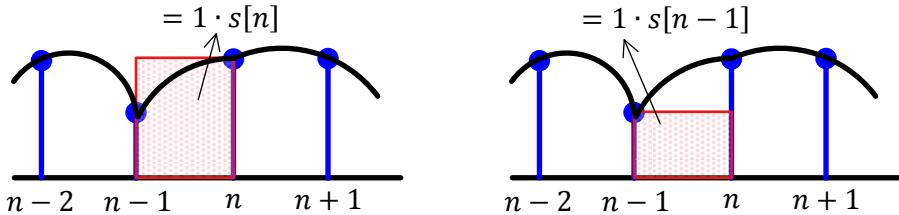


Figure 2.24: A discrete-time integrator implemented through a forward difference and a backward difference technique

The block diagram shown in the figure will be used to implement a backward difference integrator in a Numerically Controlled Oscillator (NCO) of a Phase Locked Loop (PLL) later.

Average of the backward and forward difference: A third integrator can also be implemented as

$$r[n] = r[n - 1] + \frac{1}{2} (s[n - 1] + s[n])$$

which can be seen as the average of the backward and forward difference.

Note 2.7 Why is integrator a lowpass filter?

An integrator maintains a running sum of past samples. In time domain, the summation of a large number of values tends towards a mean value. When some numbers are small and some are large, their running sum naturally pulls these extremes towards the middle, thus smoothing them out. Large variations represent high frequencies and smoothing out such variations is the function of a lowpass filter.

To verify this fact in frequency domain, first consider that when an impulse is given as input to an integrator in time domain, it again forms a running sum of the impulse, which is a unit step signal. Hence, its impulse response is a unit step signal. The unit step signal is very wide in time domain, so it must be narrow in frequency domain. Consequently, it filters out the high frequency terms and passes only the low frequency content. We can say that it acts as a lowpass filter.

Differentiator

In many applications of signal processing, computing the derivative of a signal in real time is required. For the applications of signal processing in synchronization stages later, for example, the derivative of a signal needs to be computed. If you do not know about differentiation, just remember that the derivative at a point is defined as the slope of the line tangent to the curve at that point. A filter that outputs the derivative of its input signal is a main component in some blocks of a communication receiver such as carrier frequency synchronization and symbol timing synchronization.

Differentiation is shown in Figure 2.25 where the derivative of a signal $s(t)$ at time t is defined as

$$\frac{d}{dt}s(t) = \frac{s(t + \Delta) - s(t - \Delta)}{2\Delta} \quad \text{as } \Delta \rightarrow 0$$

As Δ becomes small, both $s(t + \Delta)$ and $s(t - \Delta)$ approach $s(t)$ to yield the slope at time t .

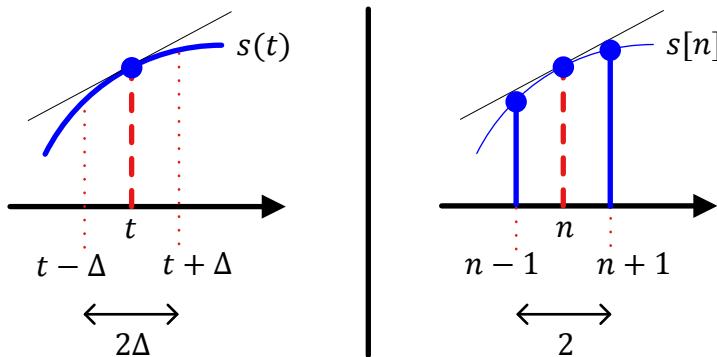


Figure 2.25: Computing the derivative of a signal in continuous-time and discrete-time cases

This tangent indicates the rate of change of the signal. As an example, consider $\sin \theta_\Delta$ in the left column of Figure 2.26. Just as $\sin \theta_\Delta$ starts at zero, its *rate of change* is maximum at zero. Then, as $\sin \theta_\Delta$ climbs to its maximum, this rate of change keeps decreasing until it reaches zero. But that is exactly what $\cos \theta_\Delta$ looks like. This is shown in dashed green line in the figure. We conclude that the maximum of $\sin \theta_\Delta$ occurs where its rate of change $\cos \theta_\Delta$ is zero! A similar argument holds for the signal $\cos \theta_\Delta$ and its rate of change $-\sin \theta_\Delta$ as illustrated in the right column of Figure 2.26.

$$\begin{aligned}\sin \theta_\Delta &\rightarrow \cos \theta_\Delta \\ \cos \theta_\Delta &\rightarrow -\sin \theta_\Delta\end{aligned}$$

Combining the above arguments, the maximum of a function can be found by finding its rate of change and equating it to zero.

The question is how to implement this operation in a simpler discrete-time approach. For the following discussion, we assume a normalized sample time $T_S = 1$.

A discrete-time system deals with just the signal samples. A widely used approximation to compute the derivative of a waveform at a time n is known as the first central

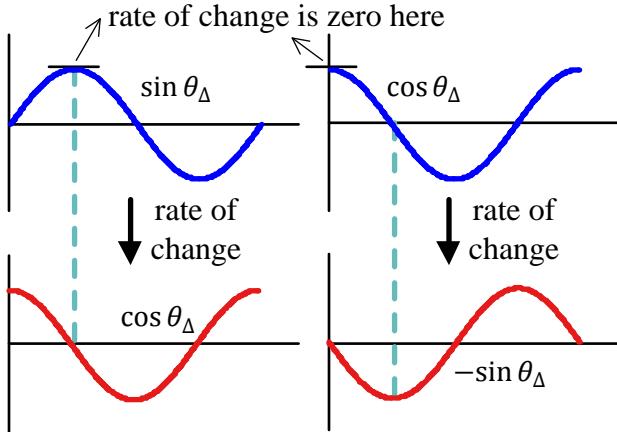


Figure 2.26: Left column: relationship between $\sin \theta_\Delta$ and its rate of change. Right column: relationship between $\cos \theta_\Delta$ and its rate of change

difference. Here, the slope is found by using the values of the signal at adjacent time instants $n - 1$ and $n + 1$, as drawn in Figure 2.25. For a signal $s[n]$,

$$\dot{s}[n] = \frac{s[n+1] - s[n-1]}{2}$$

The above equation can be written as

$$\begin{aligned}\dot{s}[n] &= \frac{1}{2}s[n+1] - \frac{1}{2}s[n-1] \\ &= \frac{1}{2} (+1 \cdot s[n+1] + 0 \cdot s[n] - 1 \cdot s[n-1]) \\ &= \sum_{m=-1}^{+1} h[m]s[n-m]\end{aligned}$$

Therefore, the impulse response of this filter is given by

$$h[n] = \frac{1}{2} \{ +1, 0, -1 \} \quad (2.36)$$

and is drawn in Figure 2.27. It is critical to remember that $h[n]$ uses only two samples of the input signal to compute the derivative and ignores all the remaining samples. Consequently, the output thus generated is a poor approximation of the true derivative of the underlying continuous-time signal. Nonetheless, it serves the purpose well enough if the input signal $s[n]$ is at least five times oversampled as compared to the minimum limit set by the Nyquist theorem.

2.7 Sample Rate Conversion

In Section 1.5, the process of sampling a continuous-time signal was discussed in detail and subsequently the sampling theorem was derived. In many applications,

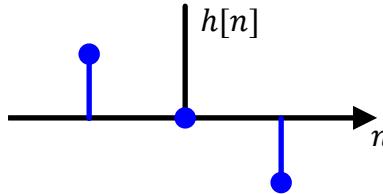


Figure 2.27: Coefficients of a discrete-time filter that roughly approximates the derivative of its input

resampling an already digitized signal is mandatory for an efficient system design. In wireless communications, sample rate conversion is typically utilized for upconversion and downconversion to a desired frequency, filtering stages in the digital frontend and for carrier and timing synchronization during signal acquisition.

In discrete domain, the sample rate can be

- reduced by periodically discarding intermediate samples called *downsampling*, or
- increased by periodically inserting additional samples within a sequence called *upsampling*.

Both downsampling and upsampling are intrinsically tied to a class of filters called multirate filters. A *multirate filter* is a digital filter that contains a mechanism to increase or decrease the sample rate while processing discrete-time signals. We discuss both downsampling and upsampling below.

When a continuous-time signal is sampled, there are no restrictions over the phase of the sampling clock with respect to time index of the signal. However, in discrete domain, the output of the resampling process inherently depends on the alignment of the sampling sequence with respect to its time origin. Later in this text, we will see several examples of the utilization of this property.

Resampling in discrete domain can be conceptually understood through the help of a discrete-time sampling sequence. An example of such a sequence was discussed in Section 1.10.4 where its DFT was also derived. We redraw Figure 1.70 and Figure 1.71 here as Figure 2.28 to aid in the discussion below.

In addition, we restate the general rules for the DFT of a sampling sequence with length N and period P , and utilize these rules during the subsequent discussion.

Mainlobe Positions: There a total of P spectral copies evenly spaced at integer multiples of $\pm N/P$ in the baseband.

Mainlobe Peaks: Each spectral copy is scaled such that the peak values are N/P .

2.7.1 Downsampling

Downsampling is the process of reducing the sample rate by an integer factor and it is also known as decimation. A signal $s[n]$ can be downsampled by a factor of P by retaining every P^{th} sample and discarding the remaining samples. The new slower sample rate is $1/P$ of the original faster sample rate. The time and frequency domain visualizations of downsampling stages are now explained.

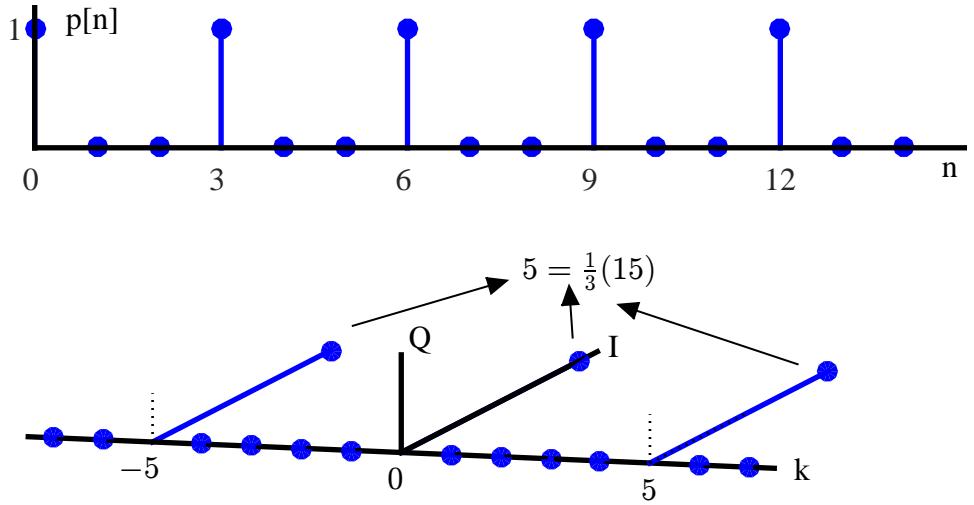


Figure 2.28: Sampling sequence in time and frequency domains for $N = 15$ and $P = 3$

Time Domain View

A 3 : 1 downsampling operation is graphically illustrated for time domain in Figure 2.29.

- Initially, assume that $s[n]$ is the only signal in our baseband and the bandwidth of

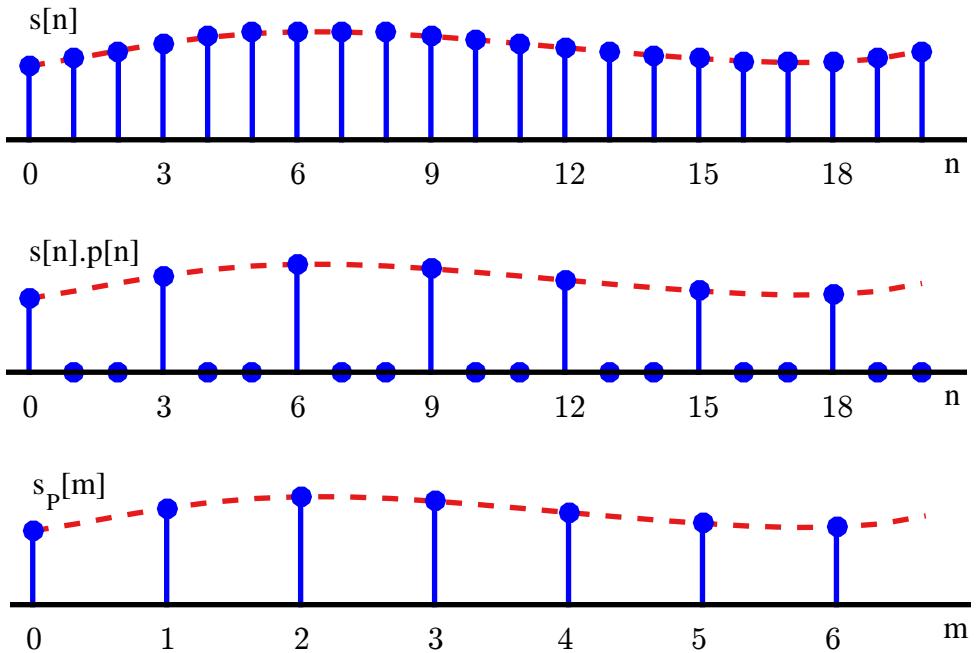


Figure 2.29: Downsampling $s[n]$ by $P = 3$ in time domain

$s[n]$ is less than or equal to $1/P$ of the original sample rate (otherwise, a lowpass filter is required which will soon be described in frequency domain view).

- Since the signal bandwidth is less than or equal to $1/P$ of the original sample rate, a corresponding reduction in sample rate can be achieved by keeping the samples with indices $0, P, 2P, \dots$ and discarding $P - 1$ samples after each such retained sample.
- Such a process can be *imagined* as multiplying $s[n]$ with a sampling sequence $p[n]$ with period P and then throwing off the intermediate zero samples. In practice, such zero values are not computed through multiplication, as they are immediately going to be discarded anyway. It is a common question if there is even a need to imagine this multiplication. The answer is that this process gives us a convenient route towards viewing its frequency domain representation as we see next.

Frequency Domain View

What happens in frequency domain is fairly interesting which can be explained with the help of 3 : 1 downsampling operation graphically illustrated in Figure 2.30.

- First, note that when we downsample a signal to a lower sample rate, there is a risk of going below the limit imposed by sampling theorem that can induce aliasing. Moreover, adjacent spectra might need to be cleaned up as well. For these reasons, we should apply a lowpass filter prior to downsampling which ensures that our signal contains no significant energy above the new lower sample rate.
- Since multiplication in one domain is equivalent to convolution in the other domain, the multiplication in time between $s[n]$ and sampling sequence $p[n]$ with period P induces a convolution in frequency between $S[k]$ and $P[k]$. Remember that there is an additional factor of $1/N$ for discrete signals:

$$s[n] \cdot h[n] \xrightarrow{\mathcal{F}} \frac{1}{N} \cdot S[k] \circledast H[k]$$

We know from Figure 2.28 that the spectrum of $P[k]$ consists of P impulses at frequency bins $k = 0$ and integer multiples of $k = \pm N/P$. Furthermore, these spectral replicas are scaled down by a factor of $1/P$. Finally, convolution of a signal with a unit impulse is the signal itself.

Combining all these facts, the result of this convolution is the emergence of P spectral copies of $S[k]$, one at frequency bin $k = 0$ and others at $\pm \text{integer} \cdot N/P$, all having a spectral amplitude of N/P .

- The final step is to reduce the bandwidth by a factor of P , which in frequency domain can be taken as a rescaling of the frequency axis by P with the baseband around $k = 0$. The $P - 1$ extra spectral copies arising as a result of above convolution now act as spectral aliases around the new sample rate. That is why these are now shown as shaded in Figure 2.30.

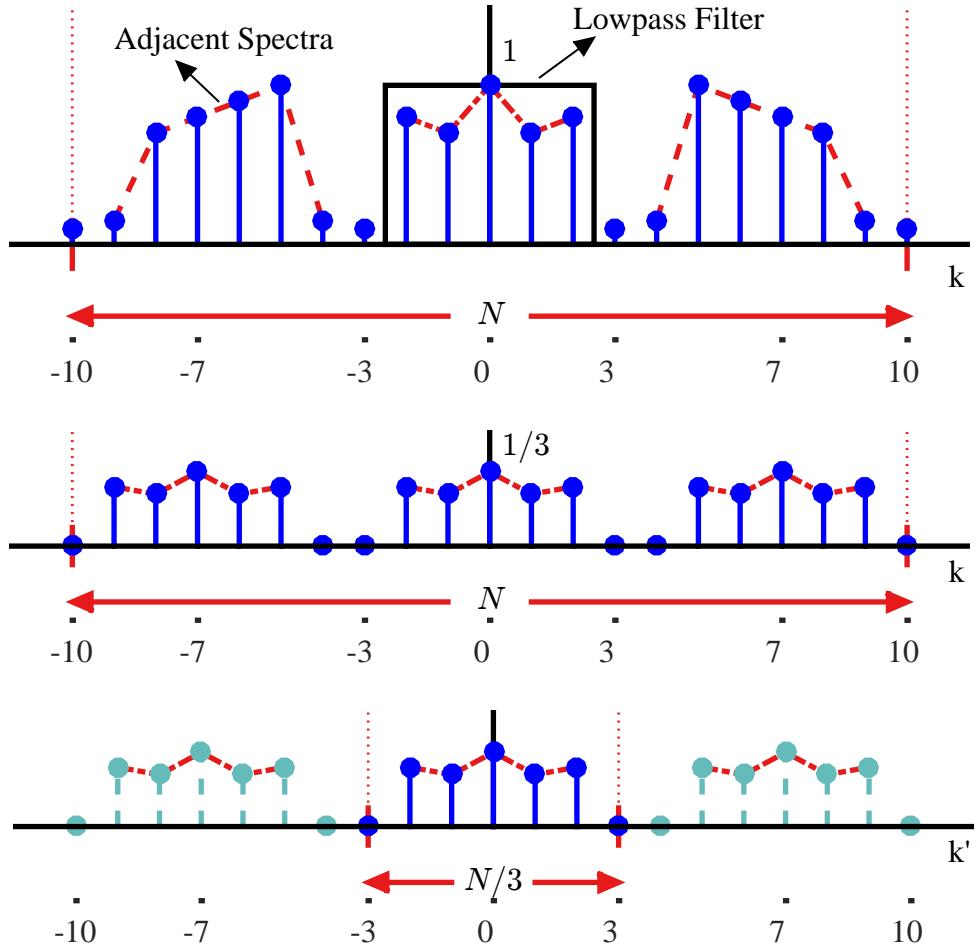


Figure 2.30: Downsampling $s[n]$ by $P = 3$ in frequency domain

Note 2.8 A wider bandwidth?

Many DSP resources draw the spectrum of the downsampled signal only within the new discrete frequency axis and new labeling, e.g., N/P in the last subplot of Figure 2.30 becomes the new ' N '. Since the new N is now drawn on the same scale as the previous signal, sometimes this causes confusion to new DSP learners in regards to how the spectrum got wider by applying the downsampling operation.

Once you downsample a signal, *the actual analog signal bandwidth does not change*. Nonetheless, the discrete frequency axis on which we draw the first spectrum no longer holds. We have a new discrete frequency axis that is scaled by a factor of P , i.e., the bandwidth only changes in the discrete frequency domain. I personally do not like drawing the wider bandwidth figure. Proper labelling of the corresponding axes while keeping the signal bandwidth the same in all figures is a better approach in my opinion.

Block Diagram

A block diagram of filtering and $P : 1$ downsampling is shown in Figure 2.31. Here, $H[k]$ is the lowpass filter that limits the bandwidth of the initial signal $s[n]$ and cleans any neighboring spectra around it. This diagram simply shows what operations need to be performed. How a downampler is implemented is left for Section 10.2.5 where we describe a polyphase filterbank in the context of the digital frontend of a software defined radio.

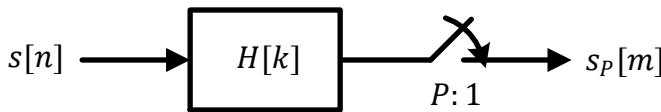


Figure 2.31: Block diagram of the filter-downsample operation

Effect on Amplitude and Energy

A discussion about amplitude and energy as a result of downsampling operation is in order. We start with Parseval's relation that relates the signal energy in time domain to that in frequency domain (it will be explained in Section 2.9).

$$E_s = \sum_{n=0}^{N-1} |s[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |S[k]|^2 \quad (2.37)$$

To make this discussion easier to follow, we refer to downsampling a rectangular signal of length N and amplitude A , which is also a complex sinusoid of frequency 0. The DFT of such a signal is given by a unit impulse of height AN which was derived in Section 1.10.1. Also, its energy is given by

$$E_s = \sum_{n=0}^{N-1} A^2 = A^2 N \quad (2.38)$$

which can be verified in frequency domain by Parseval's relation above.

$$E_s = \frac{1}{N} \cdot \left(\dots + 0 + 0 + \underbrace{AN}_{0^{th} \text{ bin}} + 0 + 0 + \dots \right)^2 = A^2 N \quad (2.39)$$

More complicated signals can be decomposed as sum of many different complex sinusoids and the basic concept discussed here can be extended.

Time Domain

The whole downsampling procedure for this signal is shown in Figure 2.32 in which the following steps can be traced. Relate the title to the figure (e.g., T3 is the third figure in time column).

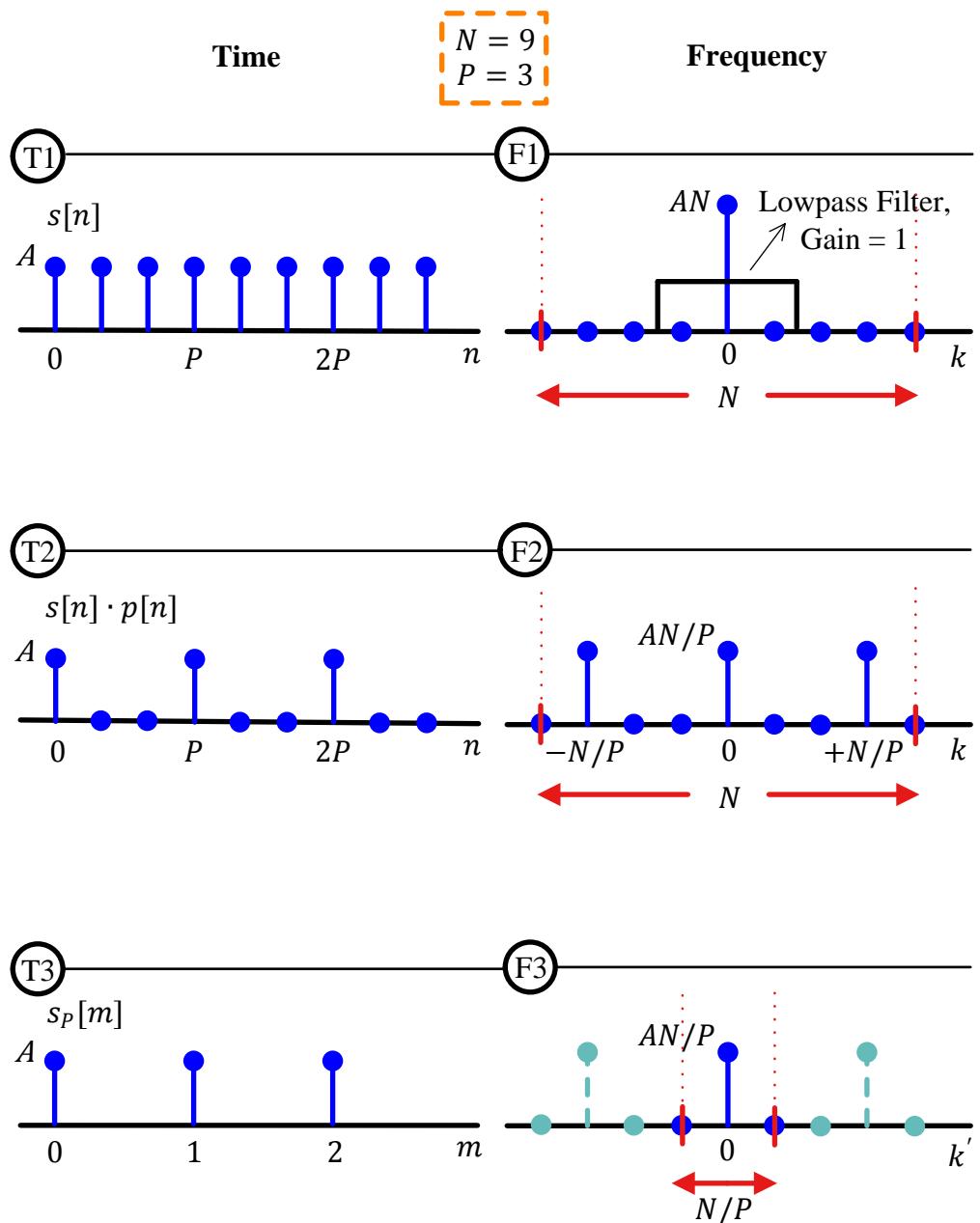


Figure 2.32: A $P = 3$ downsampling example for a rectangular signal with length $N = 9$

T1, T3: Notice that the time domain amplitude of the samples of a downsampled signal remains unchanged.

$$\text{Amplitude } \{s_P[m]\} = \text{Amplitude } \{s[n]\} \quad (2.40)$$

T2: On the other hand, the energy of the downsampled signal is lesser than the energy of the original signal due to zeroing the intermediate samples. Assuming an amplitude of A in $s[n]$, the energy in every P samples of $s[n] \cdot p[n]$ is A^2 . Since there are N/P such repetitions, the total energy in $s[n] \cdot p[n]$ is

$$\frac{N}{P} A^2 = \frac{1}{P} \cdot A^2 N \quad (2.41)$$

T3: To get $s_P[m]$, $P - 1$ zero-valued samples out of every P samples are thrown away, leaving the energy the same as in T2. Consequently,

$$E_{s_P} = \frac{1}{P} \cdot A^2 N = \frac{1}{P} \cdot E_s \quad (2.42)$$

a loss by a factor of $1/P$ compared with Eq (2.38). Obviously, the relation is exact for this example, but approximate for realistic signals and gets more accurate with large number of samples.

Frequency Domain

To verify these results in frequency domain, again refer to subfigures in Figure 2.32. For example, F2 refers to the second subfigure in frequency column. Also, there is a $1/N$ factor in energy calculations in frequency domain due to Parseval's relation in Eq (2.37). This factor arises due to how we define the DFT.

F1: Consider that the energy in the actual signal is $A^2 N$, see Eq (2.39).

F2: Also, recall from the DFT of the sampling sequence in Figure 2.28 that its spectral impulses bear an amplitude equal to N/P . When a spectral shape is convolved with these impulses, the shape gets replicated P times at integer multiples of $k = \pm N/P$, and it bears a spectral magnitude equal to N/P as well. The energy in this intermediate figure is

$$\frac{1}{N} \cdot \underbrace{\sum_{k=1}^{P-1} \left(\frac{AN}{P} \right)^2}_{\text{due to } P \text{ spectral replicas}} = \frac{1}{P} \cdot A^2 N$$

as in its time domain counterpart on the left, see Eq (2.41).

F3: In a total of P spectral copies, we keep only 1 and discard $P - 1$ neighbors by reducing the sample rate. In this process, we essentially retain only $1/P$ of the spectral energy. This is shown at the bottom of Figure 2.32 where owing to new signal length and DFT size N/P (instead of N),

$$E_{s_P} = \frac{1}{N/P} \cdot \left(\frac{AN}{P} \right)^2 = \frac{1}{P} \cdot A^2 N = \frac{1}{P} \cdot E_s$$

which is the same as found in Eq (2.42) during time domain analysis.

2.7.2 Upsampling

The procedure for upsampling a signal can be explained as a dual to downsampling. Note that upsampling a signal is also known as interpolation.

Time Domain View

A 1 : 3 upsampling operation is graphically illustrated for time domain in Figure 2.33.

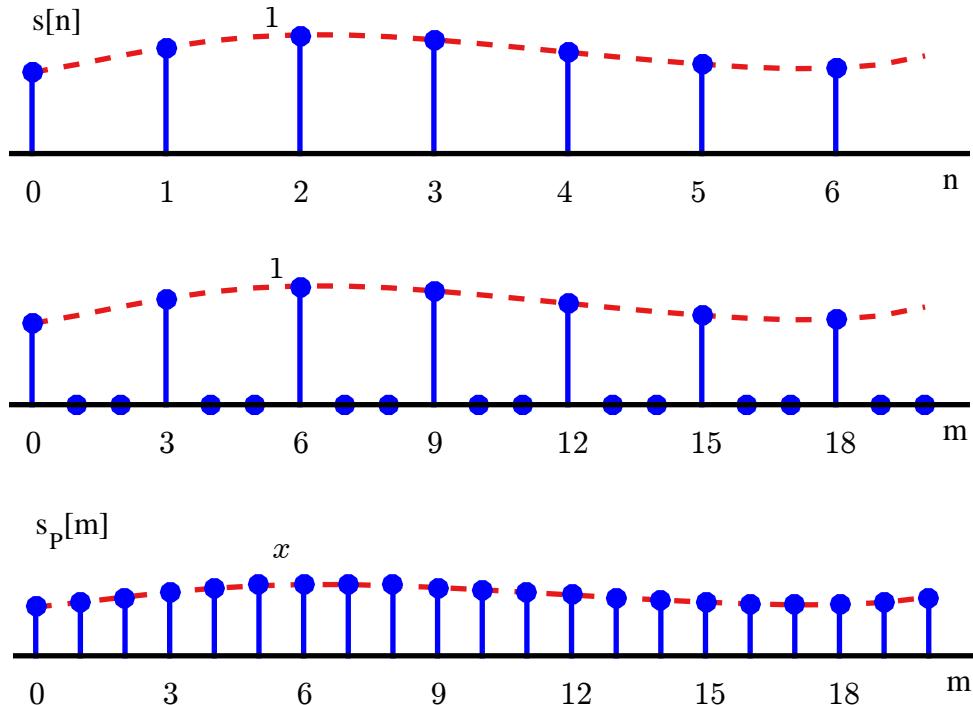


Figure 2.33: Upsampling $s[n]$ by $P = 3$ in time domain

- Starting from a signal $s[n]$, the sample rate is increased by a factor of P through inserting $P - 1$ zero-valued samples between the original input samples.
- In this new time axis, the original samples lie at indices $0, P, 2P, \dots$.
- These samples can be interpolated to obtain a smooth curve with an increased sample rate. There are many techniques to implement this interpolation, one of which we describe next in frequency domain analysis.

Frequency Domain View

Let us turn our attention towards the frequency domain through a 1 : 3 upsampling operation in Figure 2.34.

- Starting from a signal $s[n]$, its spectrum contains replicas at intervals of N frequency samples (F_S in continuous-time).

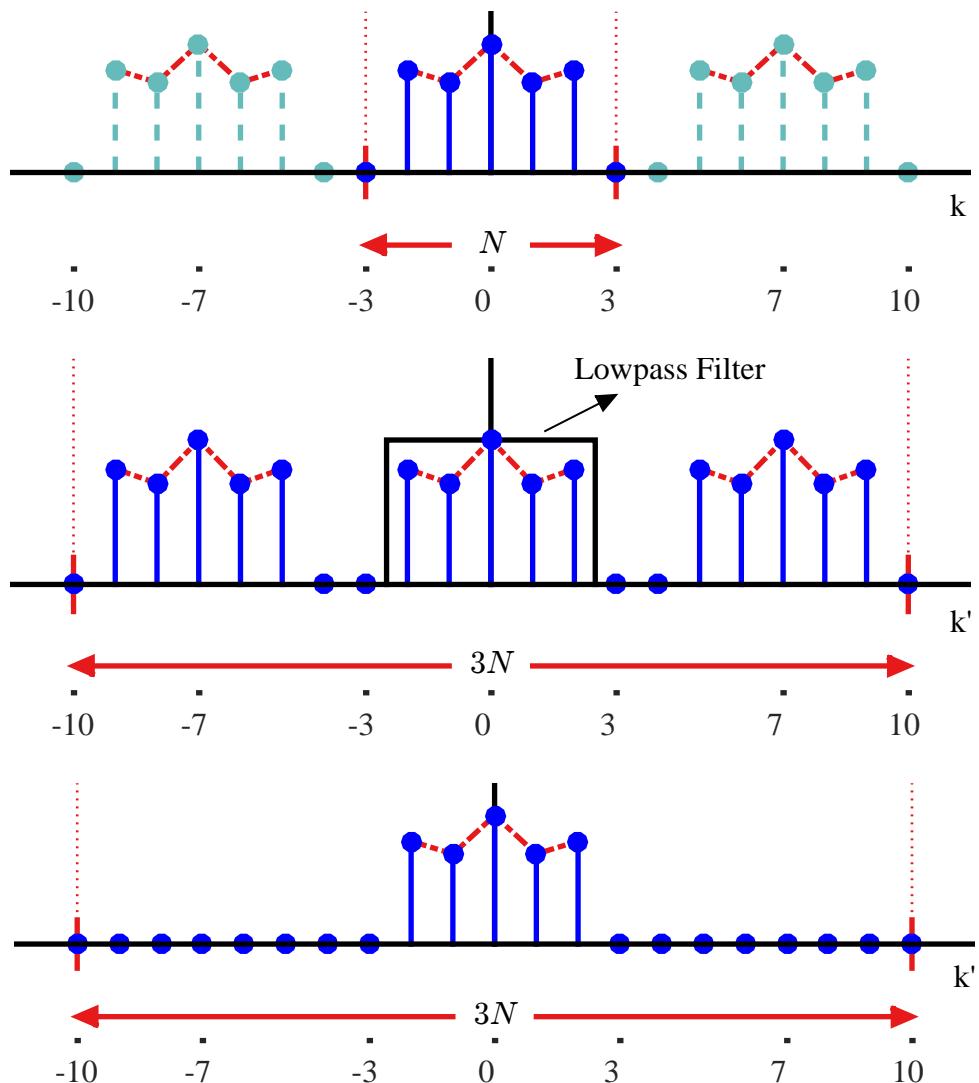


Figure 2.34: Upsampling $s[n]$ by $P = 3$ in frequency domain

- An increase in sample rate causes the baseband to take in $P - 1$ spectral copies at multiples of the original sample rate. Specifically note the change in the dotted red lines at zone edge since our baseband is now changed.
- A lowpass filter can now be employed to reject these $P - 1$ spectral replicas leaving only the central spectrum. This filtering restores the original spectrum alone in the now wider baseband.

In time domain, it has the effect of interpolating between the samples because multiplication with a rectangular filter in frequency domain is equivalent to convolving with a sinc function in time domain. This sinc function has integer spaced zero crossings due to which its convolution with a sampled signal creates the output exactly as the input at original time locations and a combination of those samples in between.

Block Diagram

A block diagram of $1 : P$ upsampling and filtering is shown in Figure 2.35. Here, $H[k]$ is the lowpass filter that rejects the neighboring spectral copies around the central spectrum. This diagram simply shows what operations need to be performed. How an upsampler is implemented is left for Section 7.13 and Section 10.1.1 where we describe a polyphase filterbank in the context of the timing synchronization and digital frontend of a digital receiver, respectively.

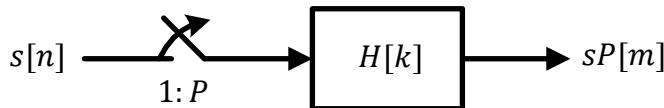


Figure 2.35: Block diagram of the upsample-filter operation

Effect on Amplitude and Energy

As with downsampling, a discussion about amplitude and energy as a result of up-sampling operation is in order. Again, we refer to upsampling a rectangular signal of length N , which is also a complex sinusoid of frequency 0 with height A . The whole upsampling procedure for this signal is shown in Figure 2.36.

Time Domain

Let us trace the following steps from different subfigures in Figure 2.36.

T1, T2: First, the energies in both the original and zero-inserted signal are the same, $E_s = A^2 N$.

T2, T3: Notice that the energy to raise the intermediate samples from dead has to come from somewhere. One possibility is that this comes from the original non-zero samples whose energy gets spread out to all P samples. As a result, the time domain amplitude of the samples of an upsampled signal gets reduced to x , which is unknown at this stage.

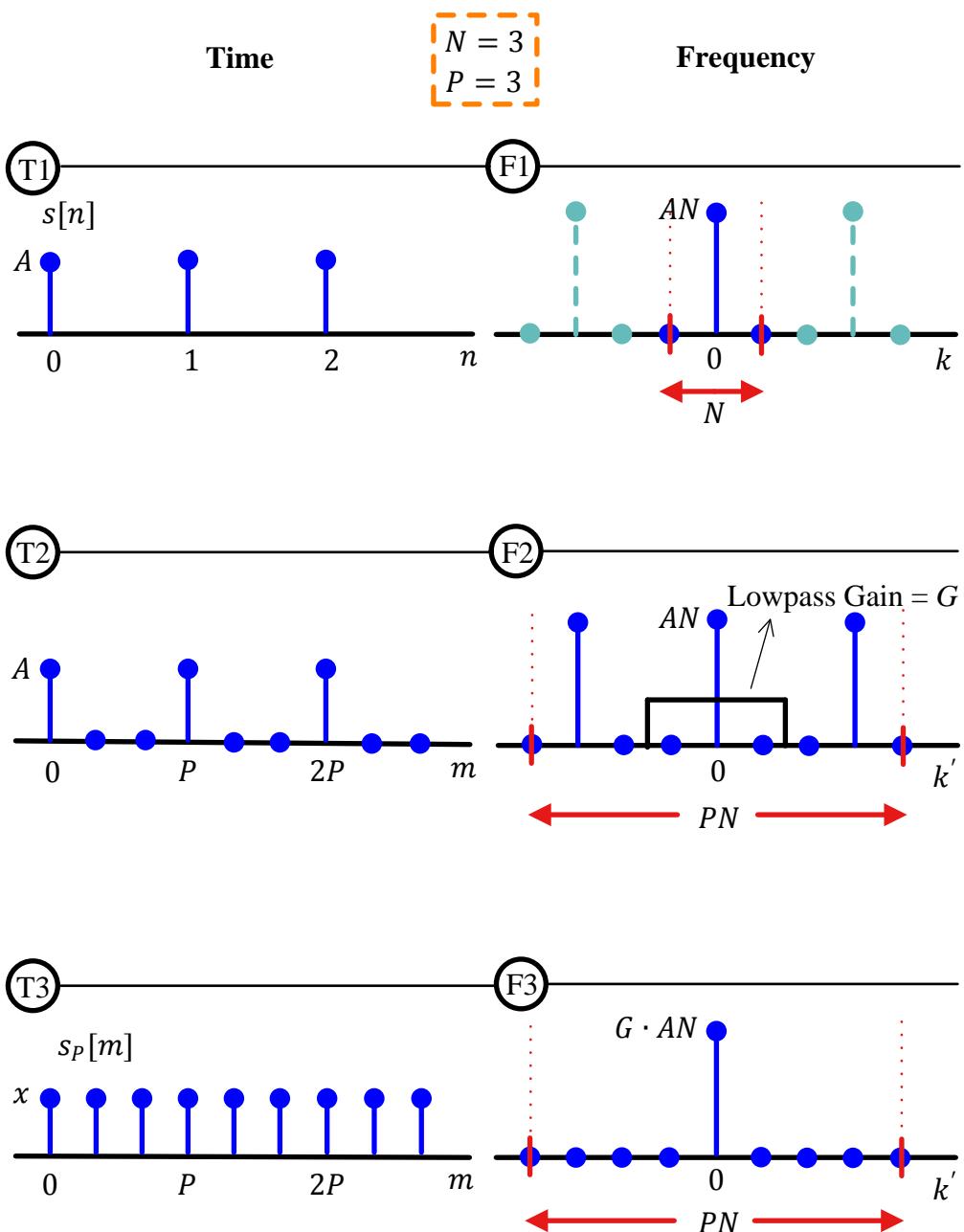


Figure 2.36: A $P = 3$ upsampling example for a rectangular signal with length $N = 3$

T3: To relate it to the energy of the upsampled signal, note that

$$E_{sp} = \underbrace{x^2 + x^2 + \cdots + x^2}_{\text{PN terms}} = PN \cdot x^2 \quad (2.43)$$

If we want to have the same energy as before,

$$E_{sp} = E_s \Rightarrow PNx^2 = A^2N$$

which leads to

$$x = \frac{A}{\sqrt{P}} \quad \text{for equal energies } E_s \text{ and } E_{sp} \quad (2.44)$$

Frequency Domain

To verify these results in frequency domain, again refer to subfigures in Figure 2.36.

F1: Using Parseval's relation in Eq (2.37), consider that the energy in the actual signal is $(1/N) \cdot (AN)^2 = A^2N$.

F2: When the sample rate is increased, the signal length and DFT size increase from N to PN and the energy in the intermediate figure is

$$\frac{1}{PN} \cdot \underbrace{P}_{\text{due to } P \text{ spectral replicas}} \cdot (AN)^2 = A^2N$$

as in its time domain counterpart on the left.

F3: Finally, let the gain of the lowpass filter be denoted by G . Then, we try three possible values for G : 1, \sqrt{P} and P . The energy in the lowpass filtered spectrum at the bottom subfigure becomes

$$\begin{aligned} G = 1 &\Rightarrow E_{sp} = \frac{1}{PN} \cdot (1 \cdot AN)^2 = \frac{1}{P}(A^2N) = \frac{1}{P} \cdot E_s \\ G = \sqrt{P} &\Rightarrow E_{sp} = \frac{1}{PN} \cdot (\sqrt{P} \cdot AN)^2 = A^2N = E_s \\ G = P &\Rightarrow E_{sp} = \frac{1}{PN} \cdot (P \cdot AN)^2 = P \cdot A^2N = P \cdot E_s \end{aligned} \quad (2.45)$$

In the above derivations, we used the fact that the definition of energy in frequency domain has a factor of horizontal span in the denominator. This horizontal span is PN instead of N for upsampled signal. Also, the relations are approximate for realistic signals and get more accurate with increasing number of samples.

Filter Gain Selection

The final task is to find x in time domain of the upsampled signal which can be found by noting that

- Energy in the upsampled time domain signal is PNx^2 , see Eq (2.43).

- For different filter gains, Eq (2.45) derives the corresponding expressions in frequency domain.
- The above two energies in time and frequency domains should be equal as is clear from the bottom of Figure 2.36.

Let us apply different gain values to see their effect.

$G = 1$: For gain G equal to 1,

$$PNx^2 = \frac{1}{P}(A^2N) \Rightarrow x = \frac{A}{P}$$

Here, one can think that the original samples sacrifice a certain part of their energy to raise the zero-injected samples from dead.[†]

$G = \sqrt{P}$: Now for a gain G equal to \sqrt{P} ,

$$PNx^2 = A^2N \Rightarrow x = \frac{A}{\sqrt{P}}$$

which is the same as derived in Eq (2.44) in time domain analysis. In this case, the upsampled signal has P times more samples as compared to the original signal and hence their amplitude is reduced by a square-root factor to maintain the same *energy* (remember that the energy is defined as the sum of squares of sample absolute values). We conclude that to maintain the same energy, the gain should be \sqrt{P} .

$G = P$: Finally, for a gain G equal to P ,

$$PNx^2 = PA^2N \Rightarrow x = A$$

We can infer that an upsampled signal with a similar amplitude as the original signal has P times more samples and hence P times more energy. To maintain the same *time domain amplitude* in the upsampled signal as before, the filter should be designed with a gain P . In many applications, maintaining the same amplitude after increasing the sample rate is desirable.

These results are summarized in Table 2.3.

[†]In regular DSP applications, a DC gain of unity for a filter is often used. To see why, assume that the filter impulse response is $h[n]$ with length N . A filter gain is the DC value at the DFT bin 0 which is

$$H[0] = \sum_{n=0}^{N-1} h[n] = 1$$

Now for a constant input $s[n] = b$, we have

$$r[n] = s[n] * h[n] = b \sum_{n=0}^{N-1} h[n] = b$$

So there is no change in the DC component of any signal and the same *dynamic range* is maintained throughout the signal processing stages. This is important in fixed-point processing where a gain greater than 1 can cause the registers to overflow while a gain less than 1 implies that the signal is being more quantized than necessary.

Table 2.3: Effect of filter gain on amplitude and energy for upsampling a signal for $A = 1$

Filter Gain	Amplitude	Energy E_{sp}
1	$\frac{1}{P}$	$\frac{1}{P} \cdot E_s$
\sqrt{P}	$\frac{1}{\sqrt{P}}$	E_s
P	1	$P \cdot E_s$

Exercise 2.1

In many GNU Radio blocks such as a ‘Low Pass Filter’ and a ‘High Pass Filter’, an input of gain is required along with a decimating or interpolating option. Although Table 2.3 describes an upsampling case, a similar concept can be employed in choosing the appropriate filter gain for any application.

As a final remark, the filtering operations required during downsampling and upsampling can be made efficient by keeping track of the zero valued samples and eliminating their multiplications. One such option is known as a polyphase filterbank and we defer their explanation when their application arises during polyphase clock synchronization in Section 7.13 and Tx and Rx architectures in Chapter 10.

2.8 Additive White Gaussian Noise (AWGN)

The performance of a digital communication system is quantified by the probability of bit detection errors in the presence of *thermal noise*. In the context of wireless communications, the main source of thermal noise is addition of random signals arising from the vibration of atoms in the receiver electronics.

The term additive white Gaussian noise (AWGN) originates due to the following reasons:

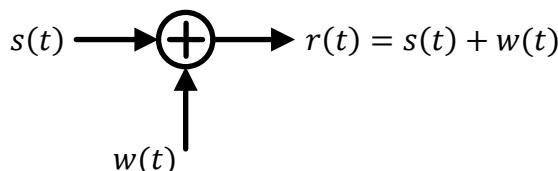


Figure 2.37: Additive property of AWGN

Additive: The noise is additive, i.e., the received signal is equal to the transmitted signal plus noise. This gives the most widely used equality in communication systems.

$$r(t) = s(t) + w(t) \quad (2.46)$$

which is shown in Figure 2.37. Moreover, this noise is statistically independent of the signal. Remember that the above equation is highly simplified due to neglecting every single imperfection a Tx signal encounters, except the noise itself.

White: Just like the white color which is composed of all frequencies in the visible spectrum, white noise refers to the idea that it has uniform power across the whole frequency band. As a consequence, the Power Spectral Density (PSD) of white noise is constant for all frequencies ranging from $-\infty$ to $+\infty$, as shown in Figure 2.38.

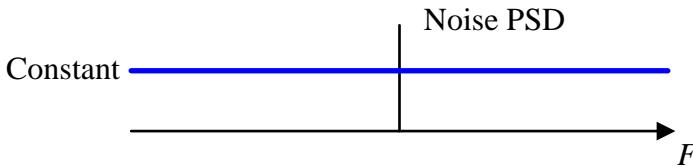


Figure 2.38: White noise has constant power spectral density across all frequencies

Nyquist investigated the properties of thermal noise and showed that its power spectral density is equal to $k \times T$, where k is a constant and T is the temperature in Kelvin. As a consequence, the noise power is directly proportional to the equivalent temperature at the receiver frontend and hence the name *thermal noise*. Historically, this constant value indicated in Figure 2.38 is denoted as $N_0/2$ Watts/Hz.

When we view the constant spectral density[†] as a rectangular sequence, its iDFT must be a unit impulse. Furthermore, in Eq (2.26), we saw that the iDFT of the spectral density is the auto-correlation function of the signal. Combining these two facts, an implication of a constant spectral density is that the auto-correlation of the noise in time domain is a unit impulse, i.e., it is zero for all non-zero time shifts. This is drawn in Figure 2.39.

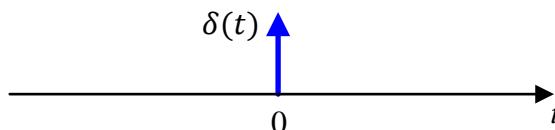


Figure 2.39: Autocorrelation of white noise is an impulse

In words, *each noise sample in a sequence is uncorrelated with every other noise sample in the same sequence*. Therefore, mean value of a white noise is zero.

Gaussian: The probability distribution of the noise samples is Gaussian with a zero mean, i.e., in time domain, the samples can acquire both positive and negative values and in addition, the values close to zero have a higher chance of occurrence while the values far away from zero are less likely to appear. This is shown

[†]We do not discuss random sequences here, so this discussion is just for a general understanding.

in Figure 2.40. As a result, the time domain average of a large number of noise samples is equal to zero.

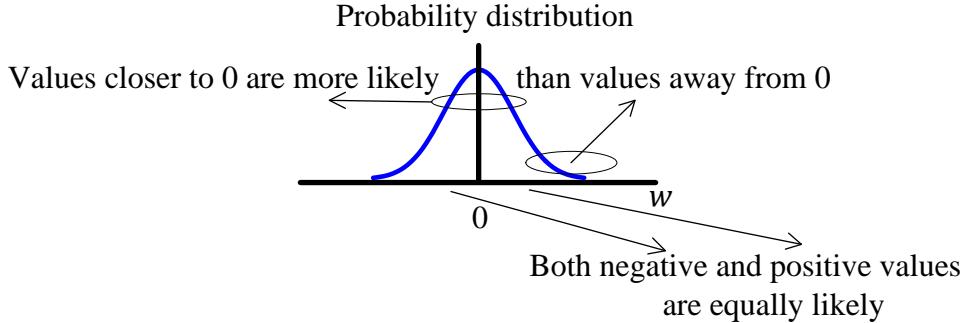


Figure 2.40: Implications of Gaussian distribution of noise

In reality, the ideal flat spectrum from $-\infty$ to $+\infty$ is true for frequencies of interest in wireless communications (a few kHz to hundreds of GHz) but not for higher frequencies. Nevertheless, every wireless communication system involves filtering that removes most of the noise energy outside the spectral band occupied by our desired signal. Consequently *after filtering*, it is not possible to distinguish whether the spectrum was ideally flat or partially flat outside the band of interest. To help in mathematical analysis of the underlying waveforms resulting in closed-form expressions – a holy grail of communication theory – it can be assumed to be flat before filtering as shown in Figure 2.38.

For a discrete signal with sampling rate F_S , the sampling theorem dictates that the bandwidth of a signal is constrained by a lowpass filter within the range $\pm F_S/2$ to avoid aliasing. For the purpose of calculations, this filter is an ideal lowpass filter with

$$H(F) = \begin{cases} 1 & -F_S/2 < F < +F_S/2 \\ 0 & \text{elsewhere} \end{cases}$$

The resulting in-band power is shown in red in Figure 2.41, while the rest is filtered out.

As with all densities, the value N_0 is the amount of noise power P_w per unit bandwidth B .

$$N_0 = \frac{P_w}{B} \quad (2.47)$$

For the case of real sampling, we can plug $B = F_S/2$ in the above equation and the noise power in a sampled bandlimited system is given as

$$P_w = N_0 \cdot \frac{F_S}{2} \quad (2.48)$$

Thus, the noise power is directly proportional to the system bandwidth at the sampling stage.

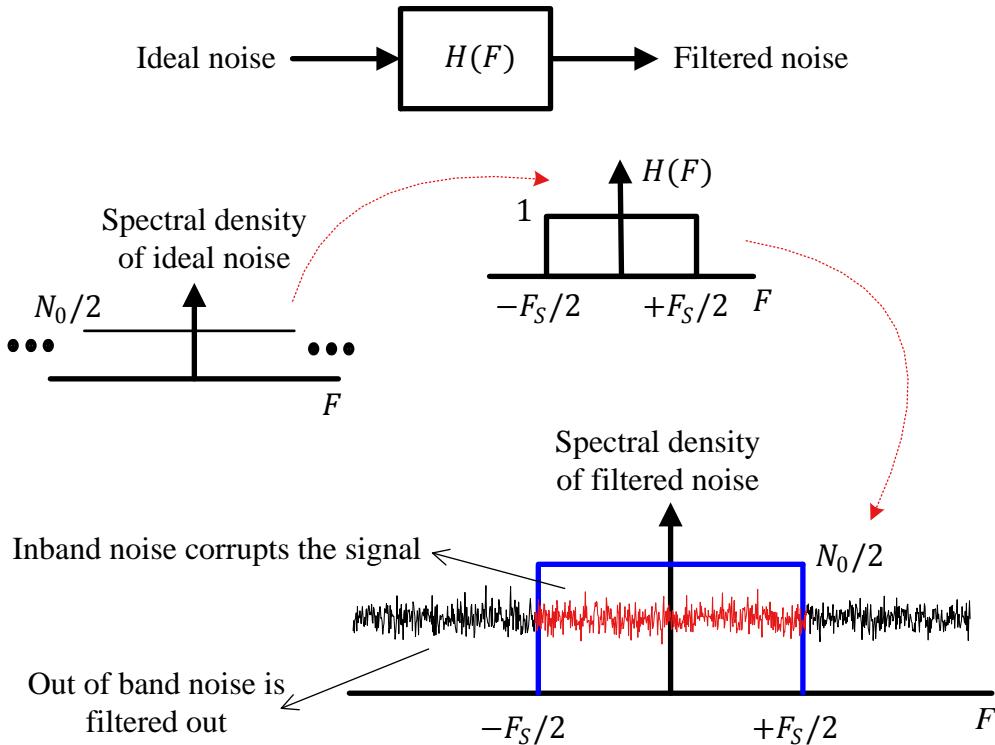


Figure 2.41: Computing the noise power within specified bandwidth

Note 2.9 Fooling the Randomness

As we will see in Chapter 3, a communication signal has a lot of structure in its construction. However, due to Gaussian nature of noise acquiring both positive and negative values, the result of adding a large number of noise-only samples tends to zero.

$$\sum_n w[n] \rightarrow 0 \quad \text{for large } n$$

Therefore, when a noisy signal is received, the Rx exploits this structure through correlating with what is expected. In the process, it estimates various unknown parameters and detects the actual message through averaging over a large number of observations. This correlation brings the order out while averaging drives the noise towards zero.

An AWGN channel is the most basic model of a communication system. Some examples of systems operating largely in AWGN conditions are space communications with highly directional antennas and some point-to-point microwave links. While several imperfections are still present (e.g., carrier and timing offsets), the underlying signal structure stays intact leading to a simpler Rx design.

2.9 Appendix

Based on what we have learned so far, some important properties of the DFT are summarized in Table 2.4 below. Any of these relations can be derived by plugging in the time domain expression in the DFT definition of Eq (1.53).

Table 2.4: DFT Properties

Property	Time Domain	Frequency Domain
Linearity	$\alpha_1 s[n] + \alpha_2 h[n]$	$\alpha_1 S[k] + \alpha_2 H[k]$
Time reversal	$s[(-n) \bmod N]$	$S[(-k) \bmod N]$
Complex conjugate	$s^*[n]$	$S^*[(-k) \bmod N]$
Time reversal + complex conjugate	$s^*[(-n) \bmod N]$	$S^*[k]$
Circular convolution	$s[n] \circledast h[n]$	$S[k] \cdot H[k]$
Multiplication	$s[n] \cdot h[n]$	$\frac{1}{N} S[k] \circledast H[k]$
Circular correlation	$s[n] \circledast h^*[-n]$	$S[k] \cdot H^*[k]$
Parseval's relation	$\sum_{n=0}^{N-1} s[n]h^*[n]$ $\sum_{n=0}^{N-1} s[n] ^2$	$\frac{1}{N} \sum_{k=0}^{N-1} S[k]H^*[k]$ $\frac{1}{N} \sum_{k=0}^{N-1} S[k] ^2$
Symmetry	real $s[n]$	$S[(N - k) \bmod N] = S^*[k]$
Time periodicity	$s[n + N] = s[n]$	—
Frequency periodicity	—	$S[k + N] = S[k]$
Circular time shift	$s[(n \mp n_0) \bmod N]$	$ \cdot = S[k] $ $\angle = \angle S[k] \mp 2\pi \frac{k}{N} n_0$

Some of the examples are derived below.

Time Reversal and Complex Conjugation

From the time reversal and complex conjugation property in Table 2.4,

$$s^*[(-n) \bmod N] \xrightarrow{\text{F}} S^*[k] \quad (2.49)$$

To see why it is so, plug the expression $s^*[(-n) \bmod N]$ in DFT definition and let us call this DFT as $\tilde{S}[k]$. Recall that in the complex conjugate, the I part stays the same while Q reverses

in sign. Moreover, operation $\bmod N$ is implicitly assumed here even if not stated.

$$\begin{array}{ll} I \rightarrow & \tilde{S}_I[k] = \sum_{n=0}^{N-1} s_I[-n] \cos 2\pi \frac{k}{N} n - s_Q[-n] \sin 2\pi \frac{k}{N} n \\ Q \uparrow & \tilde{S}_Q[k] = - \sum_{n=0}^{N-1} s_Q[-n] \cos 2\pi \frac{k}{N} n - s_I[-n] \sin 2\pi \frac{k}{N} n \end{array}$$

Next, a change of variable from n to u is introduced such that $-n = u$. The limits remain the same due to the DFT input and output periodicity. Also, using the identities $-\cos A = \cos(-A)$ and $-\sin A = \sin(-A)$,

$$\begin{array}{ll} I \rightarrow & \tilde{S}_I[k] = \sum_{u=0}^{N-1} s_I[u] \cos 2\pi \frac{k}{N} u + s_Q[u] \sin 2\pi \frac{k}{N} u \\ Q \uparrow & \tilde{S}_Q[k] = - \sum_{u=0}^{N-1} s_Q[u] \cos 2\pi \frac{k}{N} u + s_I[u] \sin 2\pi \frac{k}{N} u \end{array}$$

which can be written as

$$\begin{array}{ll} I \rightarrow & \tilde{S}_I[k] = \sum_{u=0}^{N-1} s_I[u] \cos 2\pi \frac{k}{N} u + s_Q[u] \sin 2\pi \frac{k}{N} u \\ Q \uparrow & \tilde{S}_Q[k] = - \left[\sum_{u=0}^{N-1} s_Q[u] \cos 2\pi \frac{k}{N} u - s_I[u] \sin 2\pi \frac{k}{N} u \right] \end{array}$$

Comparing with the DFT definition in Eq (1.53), observe that the I part is the same while the Q part contains a negative sign. Thus,

$$\tilde{S}[k] = S^*[k] \quad (2.50)$$

In conclusion, the DFT of a time-reversed and complex conjugated signal is given by the complex conjugate of its DFT.

Conjugate Symmetry

In many of the figures encountered so far in this text, we observed some kind of symmetry in DFT outputs. More specifically, I parts of the DFT had even symmetry while the Q components were odd symmetric in many plots. Similarly, magnitude plots were even symmetric and phase plots had odd symmetry. *This is true only for real input signals* for which quadrature component is zero.

Such kind of symmetry is called *conjugate symmetry* defined as

$$S[N-k] = S^*[k] \quad (2.51)$$

which implies

$$\begin{array}{ll} I \rightarrow & S_I[N-k] = S_I[k] \\ Q \uparrow & S_Q[N-k] = -S_Q[k] \end{array} \quad (2.52)$$

or

$$\begin{aligned} |S[N-k]| &= |S[k]| \\ \angle S[N-k] &= -\angle S[k] \end{aligned} \quad (2.53)$$

i.e., the I component is even symmetric while the Q part is odd symmetric. To see why real signals have a conjugate symmetric DFT, refer to the DFT definition. For a real signal, $s_Q[n]$ is zero and

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} S_I[k] &= \sum_{n=0}^{N-1} s_I[n] \cos 2\pi \frac{k}{N} n \\ S_Q[k] &= - \sum_{n=0}^{N-1} s_I[n] \sin 2\pi \frac{k}{N} n \end{aligned} \quad (2.54)$$

Now $S[N-k]$ is defined as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} S_I[N-k] &= \sum_{n=0}^{N-1} s_I[n] \cos 2\pi \frac{N-k}{N} n \\ S_Q[N-k] &= \sum_{n=0}^{N-1} -s_I[n] \sin 2\pi \frac{N-k}{N} n \end{aligned}$$

Using the identities $\cos(A-B) = \cos A \cos B + \sin A \sin B$, $\sin(A-B) = \sin A \cos B - \cos A \sin B$, $\cos 2\pi n = 1$, $\sin 2\pi n = 0$, $\cos(-A) = \cos A$, and $\sin(-A) = -\sin A$, we get

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} S_I[N-k] &= \sum_{n=0}^{N-1} s_I[n] \cos 2\pi \frac{k}{N} n \\ S_Q[N-k] &= \sum_{n=0}^{N-1} s_I[n] \sin 2\pi \frac{k}{N} n \end{aligned} \quad (2.55)$$

Eq (2.54) and Eq (2.55) satisfy the definition of conjugate symmetry in Eq (2.52) and the proof is complete.

As an example, observe an even symmetry in I part of the DFT of a rectangular sequence in Figure 1.62 and magnitude in Figure 1.63. Similarly, an odd symmetry can be observed from the same figures for Q part and phase, respectively.

Note 2.10 Effect of symmetry on plots

For a *real* input signal, due to an even symmetry in DFT I and magnitude plots while odd symmetry in DFT Q and phase plots, it is quite normal to discard the negative half of all these plots for real signals, with the understanding that the reader knows their symmetry properties.

Finally, observe that for real and even signals, the DFT is purely real and even, as Q part is 0. This can be verified from Eq (2.54) because Q term is then a product of an even signal and an odd signal $\sin 2\pi(k/N)n$ resulting in an odd signal. Half the values in an odd signal are the same as the other half but of opposite sign, and their sum is zero. Also, the I part is a product of an even signal $s_I[n]$ and an even signal $\cos 2\pi(k/N)n$, generating an even output.

As an easy rule, whenever you see an even symmetric signal *around 0* whether in time or frequency domain, remember that its transform will be real and will only consist of an I component. On the other hand, if the signal is not even symmetric whether in time or in frequency domain, its transform will contain a Q component and hence will be complex in general.

Odd Symmetry

To use later, we now prove that a real and odd *spectrum*, i.e., with only an I component that is an odd function of frequency, has an odd quadrature time component with a zero I part.

Referring to the iDFT definition and using the fact that $S_Q[k]$ is zero,

$$\begin{aligned} I &\rightarrow s_I[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} S_I[k] \cos 2\pi \frac{k}{N} n \\ Q &\uparrow s_Q[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} S_I[k] \sin 2\pi \frac{k}{N} n \end{aligned}$$

Using the identity $\cos(-A) = \cos A$ in addition to $S_I[k]$ being odd, we can see that for a fixed n , half of the terms in I part have one sign while the other half possess a reverse sign thus summing up to zero. On the other hand, $\sin(-A) = -\sin A$ and hence for a fixed n , when $S_I[k]$ reverses its sign, $\sin(\cdot)$ also reverses its sign, resulting in a nonzero Q component. While $S_I[k]$ stays the same for all n , $\sin(\cdot)$ reverses its sign for half of the values of n that leads to an odd response. We will apply this result during our discussion on band edge filters in Chapter 7.

Parseval's Relation

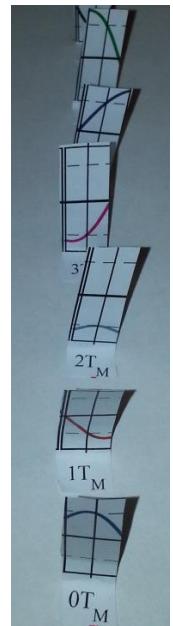
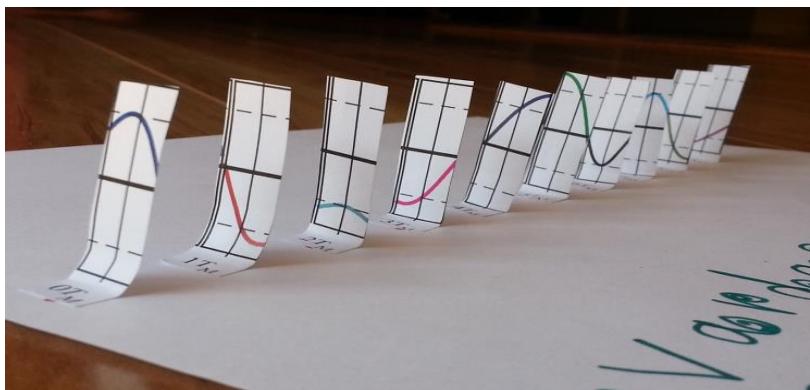
One of the most important properties of DFT we use over and over again is the Parseval's relation that relates the energy of a signal in time domain with that in frequency domain.

$$E_s = \sum_{n=0}^{N-1} |s[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |S[k]|^2$$

In words, energy of a signal in time domain is equal to its energy in frequency domain scaled by $1/N$. The intuitive reason for this scaling factor in frequency domain is that the DFT output for a complex sinusoid with amplitude A_0 is $A_0 N$, an N time magnification. The rest of the energy in frequency domain stays equal to the time domain, as demonstrated in Section 2.7.

Chapter 3

Digital Communication with Linear Modulations



“The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.”

Claude Shannon - *A Mathematical Theory of Communication* [7]

Our main purpose is to transfer digital information - which is a sequence of 0's and 1's - from one system to another through a communication channel. Let us return for a moment to the concept behind simple digital logic where logic 0 can be assigned to one voltage level while logic 1 to another. Provided that the static discipline is followed, all our system electronics has to do is to be able to *differentiate* between these two levels.

Of course, digital circuits also have real voltages and real currents which are never exact. The infinite voltage levels are just broken into two discrete regions. Hence, they remain close to that state when the real world noise is introduced, which makes the digital circuits very robust and noise tolerant. This breaking down of real numbers on a continuous scale into discrete regions is the basis of the whole digital revolution.

This particular philosophy is very close to human nature too. Think about it: we like to classify everyday stuff into clearly defined boxes. For example, day and night, black and white, good and bad. The reality, however, is continuous in the sense that the brightness of light varies throughout the 24 hours, there are multiple (perhaps 50) shades of gray, and people are much more complex than being simply good and bad. Nevertheless, this labeling works for our immediate purposes.

Digital communication follows the same principle but allows for multiple discrete regions. Let us explore why and how. We explain these concepts from more of a simple viewpoint without rigorous proofs. The text by Proakis [3] is highly recommended for a mathematics savvy reader.

3.1 A Simple Communication System

In every digital communication system, whether it is a home DSL or cellular data plan, one fundamental parameter we always hear about is the data rate. Data rate is simply the number of bits transmitted and received during one second, also called the *bit rate* denoted by R_b bits/second (b/s). Stated in another way, the source produces one bit during every $1/R_b = T_b$ seconds.

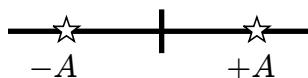


Figure 3.1: Two voltage levels representing 0 and 1

Since these bits consist of 0's and 1's, we start with assigning them two distinguishable voltage levels $-A$ and $+A$, an example of which is shown in Figure 3.1. Such a representation is called a *constellation*. Our ultimate purpose is to transmit a sequence of these voltage levels according to the bit stream at the Tx side. Then, we design the system such that these voltage levels are received undistorted in every

Table 3.1: Bit-symbol mappings, received signal after noise addition and receiver decisions

m	0	1	2	3
b	1	0	1	0
$a[m]$	+2	-2	+2	-2
$r[m]$	+2.2	-0.4	+1.1	-2.4
$\hat{a}[m]$	+2	-2	+2	-2
\hat{b}	1	0	1	0

manner at the Rx side so that the Rx can perform the inverse mapping from voltage to bits and find out the original bit stream.

Based on this reasoning, a very simple digital communication system can be built as follows: At the Tx side, logic 0 and 1 of a bit sequence \underline{b} are mapped to levels $-A$ and $+A$ of a *symbol* sequence $a[m]$, respectively, at discrete time intervals $m = 0, 1, 2, \dots$. This process is called bit-symbol mapping and the time index m counts each such *symbol time*, T_M . A symbol is basically a mapping from the binary logic levels to signal levels into the real world. The constellation in Figure 3.1 illustrates the relationship between bits in \underline{b} and symbols $a[m]$ as

$$a[m] = \begin{cases} -A, & b = 0 \\ +A, & b = 1 \end{cases} \quad (3.1)$$

The block diagram of such a system is shown in Figure 3.2 and for a bit sequence 1010, the generated symbols for $A = 2$ are shown in Table 3.1.

The received signal $r[m]$ is corrupted by the addition of random noise $w[m]$ on the Rx side and given as

$$r[m] = a[m] + w[m]$$

As a consequence of noise addition, the received symbols are scattered around the actual symbol values as illustrated in the decision block of Figure 3.2. Common sense dictates that symbol decisions $\hat{a}[m]$ should be taken according to the following rule:

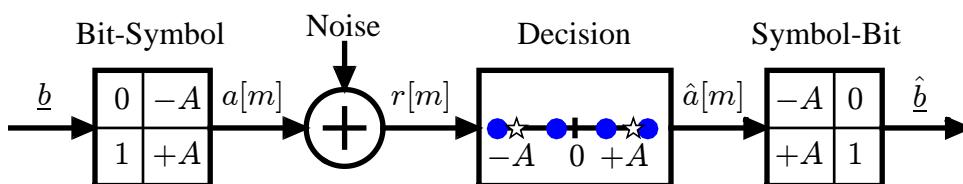


Figure 3.2: A binary digital communication system

Note 3.1 Minimum distance rule

If $r[m]$ is closer to $-A$, it is decided that symbol $-A$ was sent at time m , and if it closer to $+A$, it is taken as symbol $+A$. To construct a general intuitive rule, the point in their middle is $(-A + A)/2 = 0$, and hence

$$\hat{a}[m] = \begin{cases} -A, & r[m] < 0 \\ +A, & r[m] > 0 \end{cases} \quad (3.2)$$

Such a decision rule is applied to compute $\hat{a}[m]$ in Table 3.1 for $A = 2$ where we luckily encounter no errors. The above relation is actually a manifestation of the minimum distance rule. What is not evident here, and we will find out later in this chapter, is that we are actually performing a correlation with the two possible options and choosing the more likely one. Since we quoted Sherlock Holmes to introduce correlation in Section 1.6, we recall another of his quote in this context:

“... we balance probabilities and choose the most likely.”

Sherlock Holmes - The Hound of the Baskervilles

Finally, the subsequent symbol-bit mapping generates bit 0 for symbol $-A$ and bit 1 for symbol $+A$ to form the estimated bit sequence \hat{b} . Notice that the digital communication system constructed above is sending and receiving one bit in every symbol time T_M .

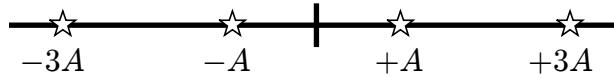
3.2 Packing More Bits in One Symbol

Digital electronics are constrained to work on only two levels by electronic switches which in the simplest case are either on or off. For many reasons, practical digital communication systems require quite complicated signal processing workload both at the Tx and Rx ends that can be performed only by a device more intelligent than an electronic switch, such as an Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA), Digital Signal Processor (DSP) or a General Purpose Processor (GPP).

Departing from mapping a 0 to one symbol and 1 to another, note that if our Rx can differentiate between two signal levels like a switch, it can certainly differentiate between more than two signal levels as well. Being able to do so can result in sending more data during the same amount of time. For example, Figure 3.3 shows four different symbols, $-3A$, $-A$, $+A$, and $+3A$. To develop a fair assignment of bits to symbols, consider a shift from a binary number system to a decimal number system. Clearly 0, 1, 2 and 3 can represent those four symbols with the help of the mapping shown in Figure 3.3.

In this kind of a system, two bits can be collected during every unit of time (the symbol time T_M), converted into their decimal equivalent and a symbol out of $-3A$, $-A$, $+A$, and $+3A$ can be accordingly selected for transmission. Note that $+3A$ and $-3A$ are chosen instead of, say, $+2A$ and $-2A$ to keep an equal distance between all the symbols.

Armed with this idea, the digital communication system of Figure 3.2 can be modified to accommodate 4 symbols as shown in Figure 3.4. A serial-to-parallel (S/P) converter at the Tx collects multiple bits at a time. These bits, converted into their decimal



00	0	$+3A$
01	1	$+A$
10	2	$-A$
11	3	$-3A$

Figure 3.3: Bit to symbol mapping for 4 symbols

representation, form an address to a unique symbol in a bit-symbol mapper called a *Look-Up Table (LUT)*. After the addition of noise in the channel, the Rx performs its decisions according to the minimum distance rule, maps the resulting symbols back into bits and uses a parallel-to-serial (P/S) converter to generate back the target bit stream. Finding bit sequence from the estimate of symbol sequence is a matter of one-to-one mapping. Once the symbol decisions are formed, the resultant bits can automatically be known through an LUT.

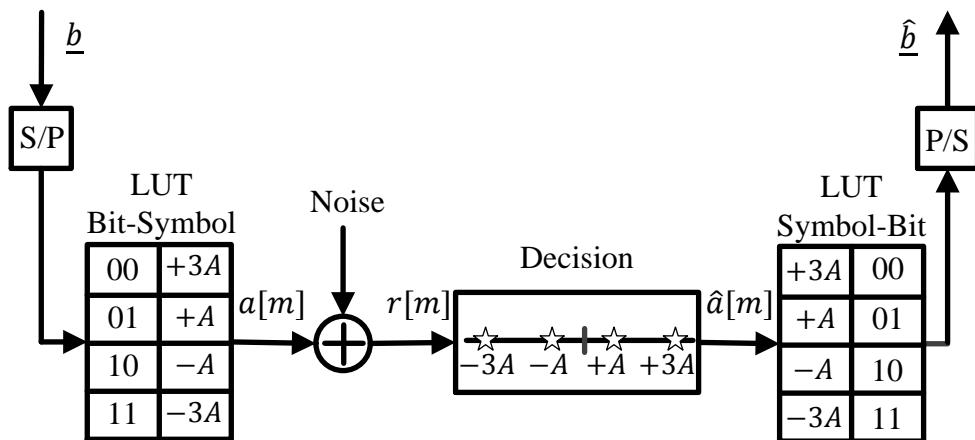


Figure 3.4: A 4 symbol communication system

Extending the above procedure, a certain number of bits can be combined and represented by one symbol to transmit information during every T_M seconds. Combining two bits generates $2^2 = 4$ symbols. Extending the same reasoning, $2^b = M$ symbols are required to transmit b bits of information. Hence, *an M-symbol scheme*

transmits $\log_2 M$ bits of information every T_M seconds. As an example, we need 8 different symbols to send 3 such bits.

In summary, a digital communication system sends a discrete set of numbers and tries to map the received values to (hopefully) the same discrete set of numbers. At this stage, the question arises: can these numbers - which are just symbols in the form of levels - be sent over **real** channels? The answer is no, and the next section explains what happens instead.

3.3 Modulation: From Numbers to Signals

The purpose of digital communications is to send digital data across a channel, some examples of which are

- wireless
- telephone lines
- coaxial cable
- optical fiber
- Ethernet
- USB
- chips on a printed circuit board.

Considering the examples shown in Figure 3.5, clearly neither a bit sequence nor a symbol sequence can be transmitted on their own through these channels – as they are nothing more than a set of numbers. This leads us to the idea of converting the numbers into a signal waveform as an appropriate tool that can travel down the channel and carry the required information – just like a train running on its track and carrying the load. For this purpose, the symbol sequence needs to be converted into an analog waveform that suits the characteristics of that particular channel. Let us start with a simple example.

Assume a rectangular pulse shape [†] $p(nT_S)$ shown in Figure 3.6 whose length is equal to one **symbol interval** in continuous time domain.

$$T_M = L \cdot T_S$$

During each symbol interval T_M , some characteristic of $p(nT_S)$ can be altered to make it a function of the symbol sequence $a[m]$. This process is called **modulation** which



Figure 3.5: Some real world channels

[†]Technically, it is called a Non Return to Zero (NRZ) pulse shape but we will continue using the term rectangular for ease of visualization.

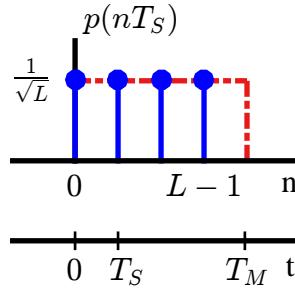


Figure 3.6: A rectangular pulse shape

is why the subscript M is chosen in T_M . The amplitude of the pulse is one such candidate that can be varied according to the symbol values.

Note that the pulse energy without any normalization constant is given by $\sum_{n=0}^{L-1} 1^2 = L$. To make the system independent of the pulse amplitude itself, it can be normalized with $1/\sqrt{L}$ as shown in Figure 3.6, thus turning it into a unit energy pulse.

$$E_p = \sum_{n=0}^{L-1} \left(\frac{1}{\sqrt{L}} \right)^2 = 1 \quad (3.3)$$

Let us now relate the bit rate R_b with the symbol rate $1/T_M$ and sample rate F_S . Typically, data bits 0 and 1 are generated with equal probability at a rate of $R_b = 1/T_b$ b/s. In our simple scenario, a 0 is represented by symbol $-A$ and a 1 by symbol $+A$ and the symbol rate is denoted as $R_M = 1/T_M$ (here, T_b and T_M are equal due to a single bit for every symbol duration). Subsequently, a pulse shape at a sample rate of $F_S = 1/T_S$ is generated with L number of samples.

$$L = \frac{T_M}{T_S} = \frac{F_S}{R_M} \quad (3.4)$$

This parameter L is known as *samples/symbol*, or sample rate normalized to symbol rate, or oversampling factor. It is standard practice to select an integer L so that the sample rate is an integer multiple of the symbol rate.

Now the amplitude of the pulse $p(nT_S)$ can be multiplied with $-A$ for transmitting a 0 and with $+A$ for transmitting a 1. As a consequence, two scaled versions of that pulse shape $s_0(nT_S)$ and $s_1(nT_S)$ are created whose amplitudes bear the symbol levels $-A$ and $+A$ representing 0 and 1, respectively. This amplitude scaling of the pulse $p(nT_S)$ according to the symbol value is called *Pulse Amplitude Modulation (PAM)* and is illustrated in Figure 3.7.

This can be seen as a natural result of holding the amplitude constant at a symbol level for the duration of a symbol T_M . Mathematically,

$$\begin{aligned} s_0(nT_S) &= -A \cdot p(nT_S) \\ s_1(nT_S) &= +A \cdot p(nT_S) \end{aligned} \quad (3.5)$$

During the transmission, the signal is corrupted by additive White Gaussian noise (AWGN) with power spectral density $N_0/2$ Watt/Hz. The received signal can be ex-

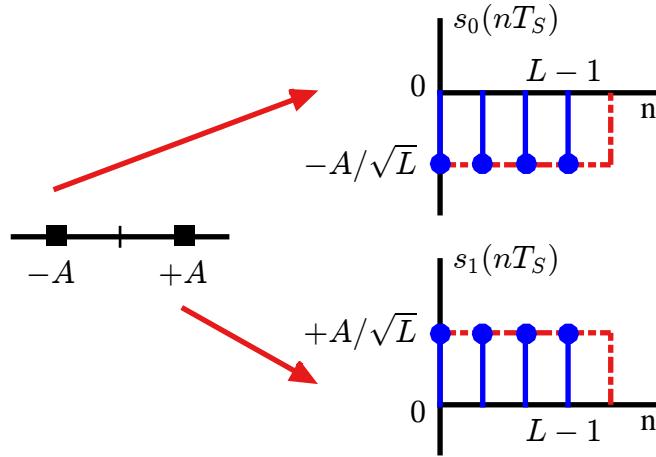


Figure 3.7: From numbers to signals: Scaling the amplitude of a rectangular pulse shape with symbols $-A$ and $+A$

pressed as

$$r(nT_S) = \begin{cases} s_0(nT_S) + w(nT_S), & b = 0, \\ s_1(nT_S) + w(nT_S), & b = 1, \end{cases} \quad 0 \leq n \leq L - 1$$

The task of the receiver is to decide the most probable symbol in the interval $0 \leq n \leq L - 1$ and hence determine whether a 0 or a 1 was transmitted.

Note 3.2 Getting numbers back from signals

A natural question at this stage is the following: Basically, digital communication is all about numbers but how will we get those numbers back from such signals for detection purpose? *In the answer to this question lies most of the theory and practice of digital communication systems.*

To accomplish this task, we need a simple mechanism through which we can map all the possible information in a symbol interval $0 \leq t \leq T_M$ back to a number. That number should be an estimate of the symbol and consequently can be used to detect the bit pattern as discussed in the next section.

3.4 Demodulation: From Signals Back to Numbers

Remember that in Section 1.6, we claimed that *correlation is the heart of a digital communication system*. We now apply the concept of correlation to figure out the transmitted message which in this case is embedded in the amplitude of the received signal. Similarly later in the text, we will encounter numerous applications of the correlation concept to solve synchronization and equalization problems.

It was also discussed there that correlation of a signal with proper normalization is maximum with itself and lesser for all the other signals. Since the number of possible signals is limited in a digital communication system, we can use the correlation

between incoming signal $r(nT_S)$ and possible choices $s_0(nT_S)$ and $s_1(nT_S)$ in a digital receiver. Consequently, a decision can be made in favor of the one with highest correlation. It turns out that the theory of maximum likelihood detection formalizes this conclusion that it is the optimum receiver for equal energy signals.

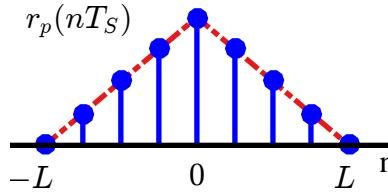


Figure 3.8: Auto-correlation $r_p(nT_S)$ of a rectangular pulse shape

Before correlating the received signal with possible choices, it is helpful to see the correlation of a rectangular pulse shape with itself – called the auto-correlation $r_p(nT_S)$ and drawn earlier in Figure 2.13. Here, Figure 3.8 illustrates the auto-correlation $r_p(nT_S)$ of the rectangular pulse shape whose time support is equal to a symbol duration, i.e., from 0 to $L - 1$ samples. As a consequence of one rectangle sliding over another sample-by-sample, its auto-correlation is a triangular pulse shape with duration clearly from $-(L - 1)$ to $L - 1$. From the definition of correlation in Eq (1.38),

$$r_p(nT_S) = p(nT_S) \diamond p(nT_S) = \sum_i p(iT_S)p(iT_S - nT_S) \quad (3.6)$$

Since correlation is similar to convolution with one signal flipped and flipping does not alter the length of a signal, the length of the correlation output is the same as convolution output, i.e., $L + L - 1 = 2L - 1$.

Eq (3.6) is extremely important in the design of communication systems and this term, $r_p(nT_S)$, will appear over and over again in any communications text.

Note 3.3 Convenience of linearity

Notice that our focus is on linear modulation, i.e., the signal is generated by linear scaling of a basic pulse shape to vary its amplitude as $\pm A, \pm 3A$, etc. as expressed in Eq (3.5). Therefore, the effect of scaling the pulse at the Tx will appear as the same scaling at the Rx without affecting the pulse shape or its auto-correlation. This leads to the conclusion that we have two choices available:

- correlate the incoming signal with all possible transmitted signals such as $s_0(nT_S)$ and $s_1(nT_S)$ here, or
- correlate the incoming signal with the pulse shape only, since $s_0(nT_S)$ and $s_1(nT_S)$ are just the scaled by $-A$ or $+A$ versions of the same pulse shape. With this strategy, the peak of the auto-correlation will bear the amplitude scaling $-A$ or $+A$ done at the Tx.

If you have some confusion regarding this point, don't worry. We will shortly see it graphically in Section 3.5 during detection of pulse amplitude modulated signals.

Now adopting the latter strategy above, let us compute the correlation between the

received signal $r(nT_S)$ and the pulse shape $p(nT_S)$ as

$$z(nT_S) = \sum_{i=-\infty}^{\infty} r(iT_S)p(iT_S - nT_S) \quad (3.7)$$

for all possible lags n . The question then is to find the optimum sampling instant: the value of n that gives the best result through maximizing the correlation output. For both $r(nT_S) = s_0(nT_S)$ and $r(nT_S) = s_1(nT_S)$ with zero noise, Figure 3.9 draws the correlations for many different lags n .

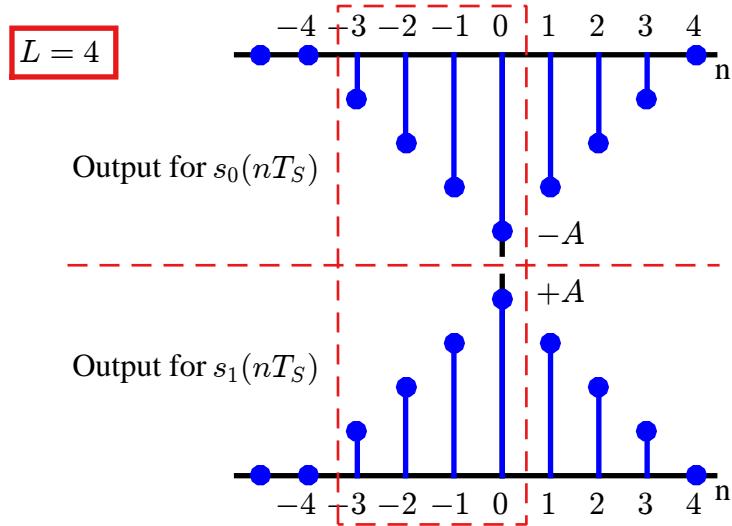


Figure 3.9: Correlation outputs of pulse shape $p(nT_S)$ for different lags n

For $L = 4$, imagine a rectangular pulse normalized by $1/\sqrt{4}$ sliding sample-by-sample over the Rx signal. This Rx signal is another rectangular pulse with a scaling factor of $-A/\sqrt{4}$ or $+A/\sqrt{4}$. Then, the correlation outputs $z(nT_S)$ vary as

$$\begin{array}{lll} & \vdots & \\ n = -4 & \rightarrow & z(-4T_S) = 0 \\ n = -3 & \rightarrow & z(-3T_S) = \pm A/4 \\ n = -2 & \rightarrow & z(-2T_S) = \pm 2A/4 \\ n = -1 & \rightarrow & z(-1T_S) = \pm 3A/4 \end{array}$$

$$n = 0 \quad \rightarrow \quad z(0T_S) = \pm 4A/4 = \pm A$$

$$\begin{aligned}
n = +1 &\rightarrow z(+1T_S) = \pm 3A/4 \\
n = +2 &\rightarrow z(+2T_S) = \pm 2A/4 \\
n = +3 &\rightarrow z(+3T_S) = \pm A/4 \\
n = +4 &\rightarrow z(+4T_S) = 0 \\
&\vdots
\end{aligned}$$

It is evident that at time instant $n = 0$, the output for $r(nT_S) = s_0(nT_S)$ is $-A$ and the output for $r(nT_S) = s_1(nT_S)$ is $+A$. This time instant is the point of maximum overlap where the received signal $r(nT_S)$ is completely aligned with the pulse shape. For symbol detection, only this particular sample of correlation output is needed.

$$\hat{a}[m] = \begin{cases} -A, & z(0T_S) < 0 \\ +A, & z(0T_S) > 0 \end{cases}$$

The rest of the samples can simply be discarded. This is the process of *demodulation* that maps a signal back to a number.

3.4.1 Matched Filter in Time Domain

Observe from Figure 3.9 that the time base of our demodulator starts at $n = -L$ samples or $-T_M$ seconds (the point where the upward slope of auto-correlation begins). However, a real receiver cannot begin processing a signal before it arrives! That would require having an ability to perceive events in the future: ask *the Oracle*.

Therefore, a Rx can only start processing the received signal at time $n = 0$. The peak or maximum of the auto-correlation shifts by the same amount of time which is $n = L$ samples or T_M seconds.

We just found in the last section that the maximum overlap is the only sample within a symbol time T_M that is required out of $L = T_M/T_S$ samples. Let us substitute $n = L = T_M/T_S$ in Eq (3.7) as

$$\begin{aligned}
z(T_M) &= z(nT_S) \Big|_{n=L=T_M/T_S} = \sum_i r(iT_S) p(iT_S - nT_S) \Big|_{n=L=T_M/T_S} \\
&= \sum_i r(iT_S) p(-(nT_S - iT_S)) \Big|_{n=L=T_M/T_S} \\
&= r(nT_S) * p(-nT_S) \Big|_{n=L=T_M/T_S}
\end{aligned} \tag{3.8}$$

We conclude that the process of correlation can be implemented as *convolution with a filter* whose impulse response $h(nT_S)$ is a flipped version of the actual pulse shape $p(nT_S)$. Most texts say that this is true only as far as the point of maximum overlap is concerned but they define the processing of a correlator in a specific manner. We discuss this point in detail during a comparison of a matched filter with a correlator.

Combining the above two facts, not only that this filter $h(nT_S)$ starts at $n = 0$ but also the filter output reaches its maximum at time T_M seconds. This flipped and delayed version of the template pulse shape is called *the matched filter* given by

$$h(nT_S) = p(-nT_S + T_M)$$

Above statement is true because the pulse has real coefficients. For complex signals, complex conjugation is also required which follows from how the correlation of complex signals is defined.

$$h(nT_S) = p^*(-nT_S + T_M) \quad (3.9)$$

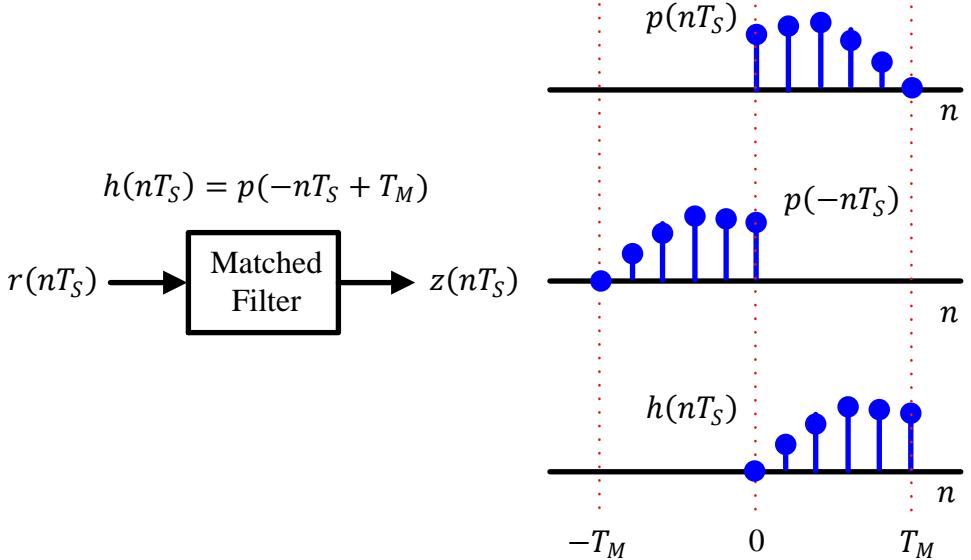


Figure 3.10: A matched filter

Figure 3.10 draws a matched filter for an example pulse shape. To find the matched filter output, consider that in the absence of noise, the received signal is

$$r(nT_S) = \pm A \cdot p(nT_S)$$

From Eq (3.8),

$$\begin{aligned} z(T_M) &= \pm A \cdot p(nT_S) * p(-nT_S) \Big|_{n=L=T_M/T_S} \\ &= \pm A \cdot 1 = \pm A \end{aligned}$$

where we utilized the fact that convolution of a signal with its flipped version is its auto-correlation which is unity at the middle point (L^{th} sample) as shown in Figure 3.8. Thus, the output $z(T_M)$ reflects the information embedded in the amplitude of the signal. Keep in mind that the above procedure describes the demodulation of a single symbol. When a whole symbol sequence is pulse shaped and transmitted (discussed in the later sections), L samples delay of the matched filter is implicitly assumed and not necessarily mentioned every time.

A question that immediately arises from the above derivation is the following: how do we know for sure that the remaining $L - 1$ samples are of little use, as far as the process of demodulation is concerned? We answer this question in the next subsection during the matched filter viewpoint in frequency domain.

Note 3.4 A natural approach

For an intuitive understanding of matched filtering approach, consider the following argument: we know that correlation can be implemented through convolution if one of the signals is flipped beforehand. During convolution with the received signal, this flipped version is folded again, hence bringing the original expected signal back and ready to be correlated.

By definition, a matched filter is a linear filter that maximizes the output signal-to-noise ratio (SNR) if the signal is buried in AWGN. *Since the incoming signal structure is already known*, the matched filter can be easily constructed through time-reversing and delaying this known signal for maximizing the output correlation.

As an example in everyday life, carefully watch out for the opinions you hold. In general, we are always defending the viewpoints we already have and rejecting (filtering out) either the unknown or what we decided to oppose in the past. That is matched filtering^a!

^aPaul Graham has written a great article on this topic titled “Keep your identity small”

3.4.2 Matched Filter in Frequency Domain

In frequency domain, the matched filter has an interesting view as well. Using $L = T_M/T_S$, we can write the matched filter expression as

$$\begin{aligned} h(nT_S) &= p^*(-(nT_S - T_M)) = p^*\left(-\left(n - \frac{T_M}{T_S}\right)T_S\right) \\ &= p^*(-(n-L)T_S) \end{aligned}$$

In terms of discrete-time, this signal can then be given as

$$h[n] = p^*[-(n-L)]$$

To view this signal in frequency domain, we need two pieces of information.

- In Section 2.9, we proved that the DFT of a time-reversed and complex conjugated signal $s^*[(-n) \bmod N]$ is given by the complex conjugate of its DFT, i.e.,

$$s^*[(-n) \bmod N] \xrightarrow{\mathcal{F}} S^*[k] \quad (3.10)$$

So the magnitude of the DFT stays the same while the phase becomes negative due to conjugate operation in k .

- The time shift of an input signal $s[(n-n_0) \bmod N]$ results in a corresponding phase shift $\Delta\theta(k) = -2\pi(k/N)n_0$ *at each frequency* of its DFT $S[k]$ with no change in magnitude.

Combining the above two facts and implicitly assuming a circular shift, the N -point DFT of the matched filter impulse response $h[n]$ can be deduced.

$$|H[k]| = |P[k]|$$

$$\angle H[k] = -\angle P[k] - 2\pi \frac{k}{N} L$$

(3.11)

So the DFT of the matched filter has a magnitude that is the same as the magnitude of the underlying pulse while a phase shift that is negative of the phase of the pulse DFT (which arises from taking the complex conjugate), plus an additional factor proportional to the symbol time T_M seconds, or L samples.

What happens when the received signal is passed through the matched filter? First, in the presence of zero noise, the Rx signal is expressed as

$$r(nT_S) = \pm A \cdot p(nT_S)$$

whose spectrum can be written as

$$\begin{aligned} |R[k]| &= A \cdot |P[k]| \\ \angle R[k] &= \angle A + \angle P[k] \end{aligned}$$

where $\angle A$ is 0 if A is positive and π if A is negative. At the output of the matched filter, the convolution in time domain is multiplication in frequency domain leading to the following results.

Magnitude and Phase

Utilizing Eq (3.11), the magnitude of the matched filter output $Z[k]$ can be written as

$$|Z[k]| = |R[k]| \cdot |H[k]| = A \cdot |P[k]| \cdot |P[k]| = A \cdot |P[k]|^2 \quad (3.12)$$

Consider that for a number b ,

$$\begin{aligned} b^2 > b &\quad \text{if } b > 1 \\ b^2 < b &\quad \text{if } 0 < b < 1 \end{aligned}$$

Therefore, the matched filter has high gain at frequencies where $P[k]$ is large and low gain at frequencies where $P[k]$ is small. It can be deduced that the matched filter enhances the strong spectral components and reduces the weak spectral components in $P[k]$.

On the other hand, the phase of $Z[k]$ takes the form

$$\begin{aligned} \angle Z[k] &= \angle R[k] + \angle H[k] = \angle A + \angle P[k] - \angle P[k] - 2\pi \frac{k}{N} L \\ &= \angle A - 2\pi \frac{k}{N} L \end{aligned} \quad (3.13)$$

where we can notice the following.

Phase cancellation: The phase of the incoming signal $\angle P[k]$ is canceled by the matched filter phase $-\angle P[k]$. This phase adjustment enables all the spectral components to align at the sampling time instant.

Tx time reference: The additional phase of $-2\pi(k/N)L$ comes due to a delay of L samples from the Tx time reference 0. Since the Rx is only interested in the value of the symbol and not the Rx location (in reality, there are several additional delays that are added to the incoming signal such as the propagation delay and the Tx/Rx frontend processing in both analog and digital domains), this delay is irrelevant and it can set its time reference 0 at the peak of the auto-correlation. Then, the phase term $-2\pi(k/N)0$ disappears.

Figure 3.11 illustrates an example signal template and a corresponding matched filter where the solid blue lines represent the signal template while the dotted red lines show the matched filter. Notice the same magnitude on each spectral line but exactly the opposite phase. This is the reason complex conjugation is required for designing a matched filter for complex signals.

Conjugate operation aligns the phases in frequency domain

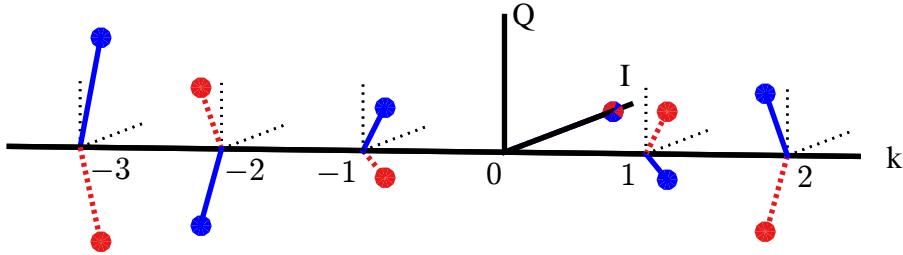


Figure 3.11: Conjugate operation of a matched filter aligns all spectral components by canceling the phase. The solid blue line is the signal template and dotted red line is the matched filter

I and Q Components

From the magnitude and phase expressions above in Eq (3.12) and Eq (3.13), respectively, and incorporating $\angle A$ in \pm form, $Z[k]$ can be written in terms of IQ as

$$Z_I[k] = \pm A \cdot |P[k]|^2 \cos 2\pi \frac{k}{N} L$$

$$Z_Q[k] = \mp A \cdot |P[k]|^2 \sin 2\pi \frac{k}{N} L$$

where the difference of \pm and \mp is due to the identities $\cos(-B) = \cos B$ and $\sin(-B) = -\sin B$.

To find the output in time domain $z(nT_S)$, iDFT of $Z[k]$ needs to be taken.

$$I \rightarrow z_I(nT_S) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[Z_I[k] \cos 2\pi \frac{k}{N} n - Z_Q[k] \sin 2\pi \frac{k}{N} n \right]$$

$$Q \uparrow z_Q(nT_S) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[Z_Q[k] \cos 2\pi \frac{k}{N} n + Z_I[k] \sin 2\pi \frac{k}{N} n \right]$$

Plugging $Z_I[k]$ and $Z_Q[k]$ in iDFT above and using identities $\cos A \cos B + \sin A \sin B = \cos(A - B)$ and $\sin A \cos B - \cos A \sin B = \sin(A - B)$, we get

$$I \rightarrow z_I(nT_S) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \pm A \cdot |P[k]|^2 \cos \left[2\pi \frac{k}{N} (n - L) \right] \quad (3.14)$$

$$Q \uparrow z_Q(nT_S) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \pm A \cdot |P[k]|^2 \sin \left[2\pi \frac{k}{N} (n - L) \right]$$

Sampling this matched filter output at the instant $n = T_M/T_S = L$ yields

$$I \rightarrow \quad z_I(nT_S) \Big|_{n=T_M/T_S} = z_I(T_M) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \pm A \cdot |P[k]|^2$$

$$Q \uparrow \quad z_Q(nT_S) \Big|_{n=T_M/T_S} = z_Q(T_M) = 0$$

Figure 3.12 verifies the above finding by illustrating the frequency terms on the right hand side of the above equation. Notice that there is no Q component and all frequency domain samples are perfectly aligned to contribute towards I term.

Matched filter output at correct sampling instant

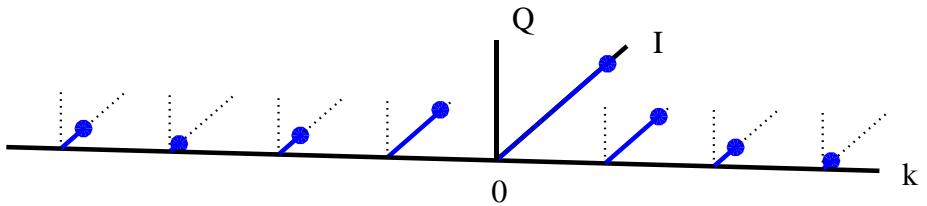


Figure 3.12: At correct sampling instant, the spectral components have all matching phase and contribute maximally towards the sampled output

Using Parseval's relation from Section 2.9 that relates the signal energy in time domain to that in frequency domain,

$$E_s = \sum_{n=0}^{N-1} |s[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |S[k]|^2$$

the matched filter output in time domain can be written as

$$I \rightarrow \quad z_I(T_M) = \pm A \sum_{n=0}^{N-1} |p[n]|^2 = \pm A \cdot E_p$$

$$Q \uparrow \quad z_Q(T_M) = 0$$

Owing to the unit energy pulse $E_p = 1$ from Eq (3.3),

$$I \rightarrow \quad z_I(T_M) = \pm A$$

$$Q \uparrow \quad z_Q(T_M) = 0$$

and hence the modulated information is retrieved through the amplitude of the matched filter output.

In fact, this is the purpose of auto-correlation at lag 0: finding the energy of the signal which is *a real number* with zero Q part.

$$\text{corr}[0] = \sum_{m=-\infty}^{\infty} s[m]s^*[m] = \sum_{m=-\infty}^{\infty} |s[m]|^2 = E_s$$

That is how the SNR is maximized by the matched filter output sampled at $n = L$.

For the case where an all 1 sequence is transmitted (no $-A$ in the received sequence) and signal in Eq (3.14) is sampled at a different time instant instead of $n = L$, say at $n = L - 1$, we have from Eq (3.14),

$$I \rightarrow z_I(nT_S) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} |P[k]|^2 \cos 2\pi \frac{k}{N}$$

$$Q \uparrow z_Q(nT_S) = -\frac{1}{N} \sum_{k=-N/2}^{N/2-1} |P[k]|^2 \sin 2\pi \frac{k}{N}$$

which is nothing but a linear phase shift arising in all frequency bins due to sampling the output one unit of time earlier. Recall by going back to Figure 1.56 that a phase shift of $2\pi(k/N)n_0$, as a function of k , is actually a frequency domain complex sinusoid that modulates the spectrum.

Now, the Q component of the output $z_Q(nT_S)$ is not zero. We can say that a part of the actual signal energy – that should have appeared in I branch – has leaked into the Q arm. The I arm with reduced energy is not optimal anymore for symbol detection. This is drawn in Figure 3.13. Notice that the spectral components have been rotated by symmetrical phase $-2\pi k/N$ due to a time difference of 1 sample. This non-zero Q

Matched filter output at incorrect sampling instant

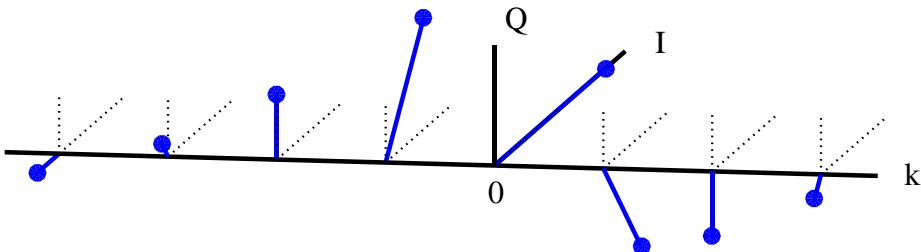


Figure 3.13: At incorrect sampling instant, the spectral components are rotated symmetrically which reduces their contribution towards I term as well as generating correlated noise samples through the filtering effect

sample also reveals the following.

- In time domain, all the samples of the correlation result are less in magnitude than the peak due to a partial overlap instead of the maximum overlap.
- Going into frequency domain, this disappeared energy can be found in Q arm associated with phase rotations. The same frequency response is very similar for all time shifts – the only difference being the scattered phases for $n \neq L$ resulting in misaligned summation.

Also remember that in an actual receiver, **noise** is also added to the transmitted signal and has to pass through the same filter (not matched for noise). For noise

$w(nT_S)$, the filtered noise samples $w'(nT_S)$ are given by

$$w'(nT_S) = \sum_i w(iT_S)h(nT_S - iT_S)$$

Therefore, this filtering on the noise generates correlation in noise samples at its output that is directly proportional to the auto-correlation of the pulse shape (such as the triangular auto-correlation of a rectangular pulse).

$$r_{w'}(nT_S) \propto h(nT_S) \diamond h^*(nT_S) \quad (3.15)$$

For this reason, this correlation among noise samples is zero only when the underlying pulse auto-correlation is zero, which occurs at a spacing of L samples either side of the peak, as is evident from Figure 3.8.

Here, we concentrated on the matched filter in relation to the underlying pulse shape due to the modulation being linear. This pulse shape is real and symmetric as well. In the field of signal processing, the matched filter has a much broader meaning that works well for non-linear modulations, distinct communication and radar waveforms and other specialized areas, where the matched filter is designed in relation to the expected signal itself. The fundamental concept is still the same: *utilize the knowledge of what is expected* and suitably invert, delay and take its complex conjugate.

So far, we have learned how to map a symbol to a signal and then a signal back to a symbol. All of this was focused within each symbol duration T_M . Next, we discuss the practical case of a bit stream and corresponding processes of modulation, waveform generation and detection.

3.5 Pulse Amplitude Modulation (PAM)

In Section 3.3, we said that the Pulse Amplitude Modulation (PAM) is an amplitude scaling of the pulse $p(nT_S)$ according to the symbol value. What happens when this process of scaling the pulse amplitude by symbols is repeated for every symbol during each interval T_M ? Clearly, a series of bits \underline{b} (1010 in our initial example) can be transmitted by choosing a rectangular pulse and scaling it with appropriate symbols.

$$\begin{array}{ll} m = 0 \rightarrow b = 1 & \rightarrow a[0] = +A \\ m = 1 \rightarrow b = 0 & \rightarrow a[1] = -A \\ m = 2 \rightarrow b = 1 & \rightarrow a[2] = +A \\ m = 3 \rightarrow b = 0 & \rightarrow a[3] = -A \end{array}$$

Our next step is forming a cumulative waveform from these individual symbol-scaled pulses. Remember that a signal $p(nT_S)$ delayed by an amount T_M or L samples is given as $p(nT_S - LT_S) = p(nT_S - T_M)$, where T_M is the symbol duration and L is samples/symbol defined as T_M/T_S . Since the same pulse is scaled by the symbol value during each T_M , we observe the following.

- At time instant $0T_M$, the output is $a[0]p(nT_S - 0T_M)$.
- At time instant $1T_M$, the output is $a[1]p(nT_S - 1T_M)$.
- At time instant $2T_M$, the output is $a[2]p(nT_S - 2T_M)$.

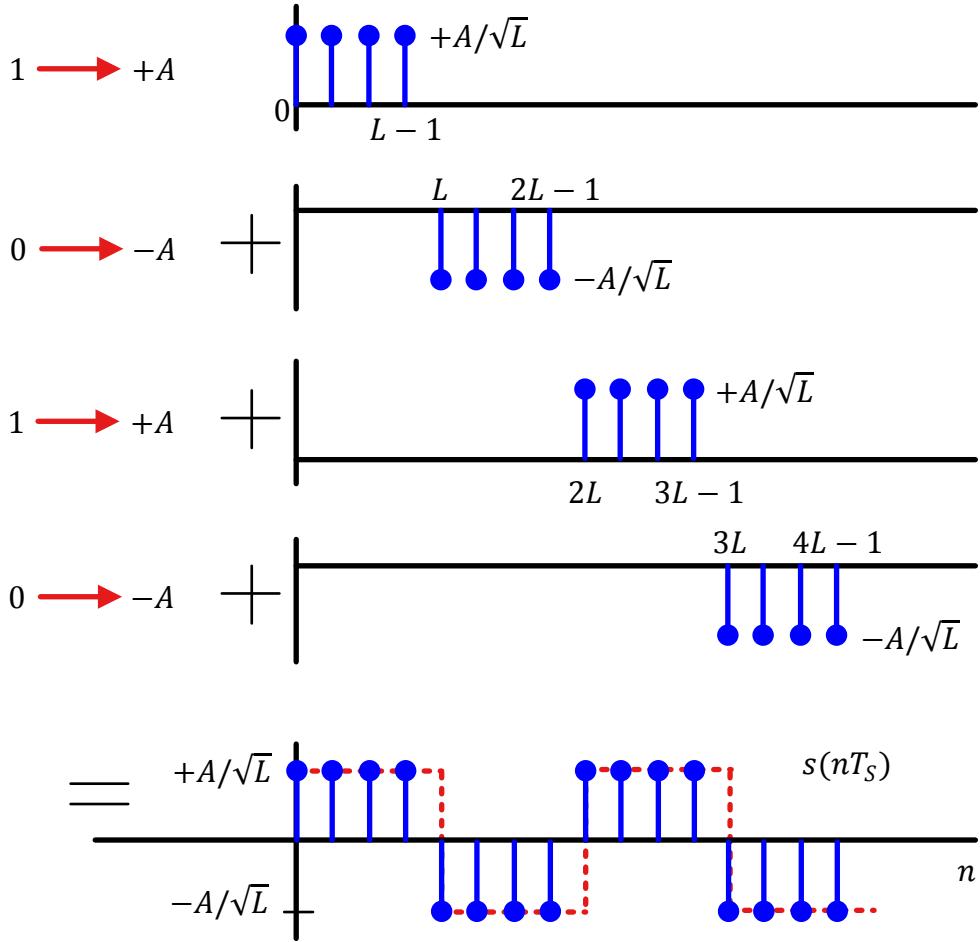


Figure 3.14: A 2-PAM waveform

■ At time instant $3T_M$, the output is $a[3]p(nT_S - 3T_M)$.

And so on. Finally, their addition gives the expression for a general PAM waveform:

$$\begin{aligned}
 s(nT_S) &= a[0]p(nT_S - 0T_M) + a[1]p(nT_S - 1T_M) + \\
 &\quad a[2]p(nT_S - 2T_M) + a[3]p(nT_S - 3T_M) + \dots \\
 &= \sum_m a[m]p(nT_S - mT_M)
 \end{aligned} \tag{3.16}$$

After Digital to Analog Conversion (DAC), the continuous-time signal $s(t)$ can be expressed as

$$s(t) = \sum_m a[m]p(t - mT_M) \tag{3.17}$$

As an example, a 2-PAM waveform is illustrated in Figure 3.14 with red dashed curve being the underlying continuous-time signal.

In a similar manner to 2-PAM, a 4-PAM waveform based on 4 symbols can be constructed by scaling the pulse amplitude by different symbol values during each T_M , as illustrated in Figure 3.15.

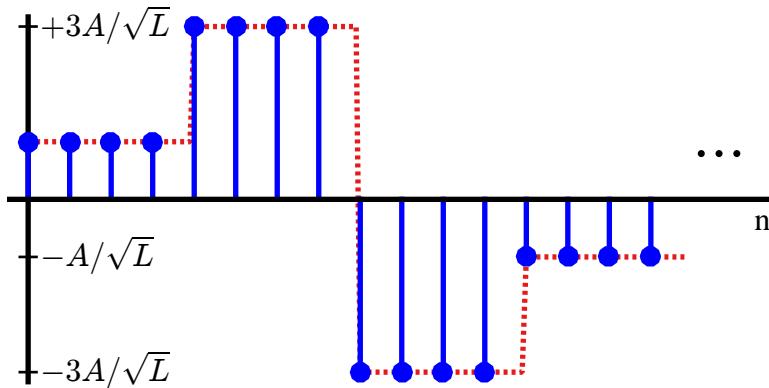


Figure 3.15: A 4-PAM waveform

Note 3.5 Orthogonality in frequency domain

There is something interesting about the PAM data generation above. The symbols are orthogonal in time due to the time shift mT_M of a basic pulse shape. This time difference of a symbol time T_M between two symbols implies that no symbol interferes with the other. How is the corresponding orthogonality maintained in frequency domain?

Recall from Eq (1.60) that for a time shift mT_M of the basic pulse shape, the DFT of this pulse shape is multiplied with a frequency domain complex sinusoid with inverse period mT_M . All data symbols in frequency domain are riding on the pulse shape DFT multiplied with the orthogonal complex sinusoids (overlapping within the pulse bandwidth) with increasingly large inverse periods as m increases. This is interesting since just a passing of time is giving rise to spiraling sinusoids in frequency domain. Only if it could solve the world's energy problems!

Constellation Diagram

Just like a constellation of stars, a *constellation diagram* shows the actual symbol values representing a set of $\log_2 M$ bits. We have already encountered constellation diagrams before (e.g., in Figure 3.1 for $M = 2$ and in Figure 3.3 for $M = 4$). Some example symbol sets are as follows.

$$2\text{-PAM Constellation} = \{-A, +A\}$$

$$4\text{-PAM Constellation} = \{-3A, -A, +A, +3A\}$$

$$M\text{-PAM Constellation} = \{-(M-1)A, \dots, -3A, -A, +A, +3A, \dots, +(M-1)A\}$$

A general constellation diagram for M -PAM is shown in Figure 3.16.

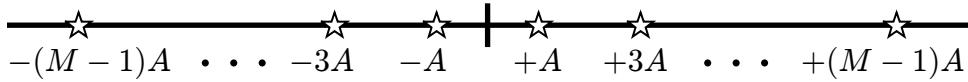


Figure 3.16: General constellation diagram for M -PAM

Average Symbol Energy

The *average symbol energy* in a constellation is given by the average of all individual symbol energies. For $M = 2$,

$$E_{M-\text{PAM}} = \frac{A^2}{2} \left\{ (+1)^2 + (-1)^2 \right\} = A^2$$

And for $M = 4$,

$$E_{M-\text{PAM}} = \frac{A^2}{4} \left\{ (-3)^2 + (-1)^2 + (+1)^2 + (+3)^2 \right\} = 5A^2$$

For a general M ,

$$\begin{aligned} E_{M-\text{PAM}} &= \frac{A^2}{M} \left\{ (-M+1)^2 + \cdots + (-3)^2 + (-1)^2 + \right. \\ &\quad \left. (+1)^2 + (+3)^2 + \cdots + (M-1)^2 \right\} \\ &= 2 \frac{A^2}{M} \left\{ 1^2 + 3^2 + \cdots + (M-1)^2 \right\} \end{aligned}$$

The term in the brackets is the sum of squares of first $(M-1+1)/2 = M/2$ odd integers. Using the formula $k(2k-1)(2k+1)/3$ for the first k odd integers squared, we get

$$\begin{aligned} E_{M-\text{PAM}} &= 2 \frac{A^2}{M} \cdot \frac{M/2 \cdot (M-1) \cdot (M+1)}{3} \\ &= \frac{M^2 - 1}{3} A^2 \end{aligned} \tag{3.18}$$

The main purpose of a communications system is to solve the following problem: which operations are necessary to perform at the Tx and Rx sides such that we can detect that symbol with the highest probability? To explore the answer, we start with a simple PAM modulator and detector.

PAM Modulator

At this stage, we are ready to build a conceptual PAM modulator. The block diagram is drawn in Figure 3.17 in which Tx signal is generated in the following way.

- Every T_b seconds, a new bit arrives at the input forming a serial bit stream.
- A serial-to-parallel (S/P) converter collects $\log_2 M$ such bits every $T_M = \log_2 M \times T_b$ seconds that are used as an address to access a Look-Up Table that stores M symbol values specified by the constellation.

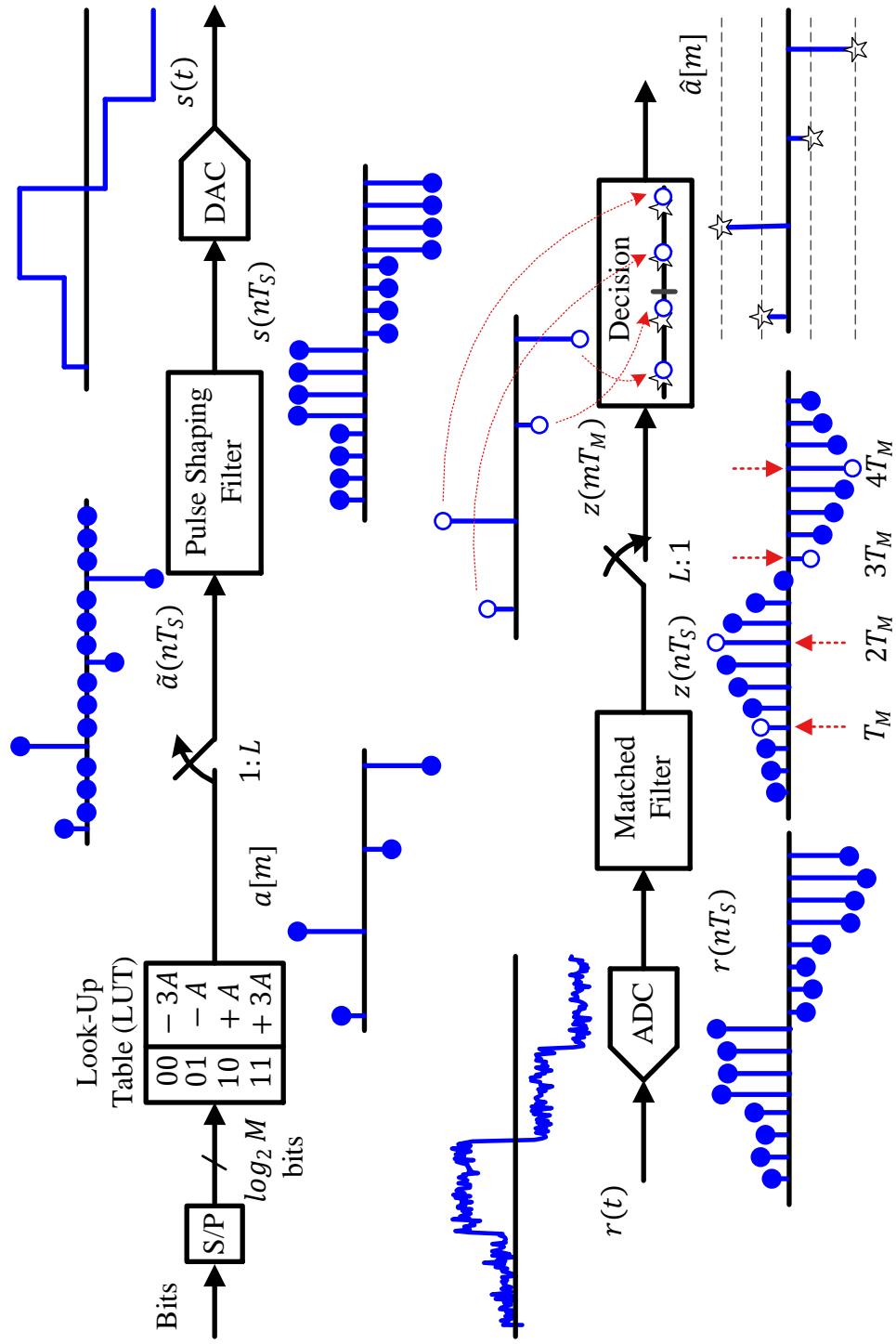


Figure 3.17: A general PAM modulator and detector

- To produce a PAM waveform, the symbol sequence $a[m]$ is converted to a discrete-time impulse train through upsampling by L where L is samples/symbol, i.e., the ratio of symbol time to sample time T_M/T_S , or equivalently, sample rate to symbol rate F_S/R_M . Let us denote such a sequence by $\tilde{a}(nT_S)$.
- The upsampling operation inserts $L - 1$ zeros between each set of two symbols after which the intermediate samples can be raised from dead with the help of a lowpass filter that suppresses all the spectral replicas except the primary one. We will see in the next section that a proper pulse shaping filter $p(nT_S)$ is also a lowpass filter and hence an extra lowpass filter is not actually required.

$$s(nT_S) = \tilde{a}(nT_S) * p(nT_S)$$

- The generated discrete-time signal $s(nT_S)$ is converted to a continuous-time signal $s(t)$ by a DAC.

PAM Detector

The received signal $r(t)$ is the same as the transmitted signal $s(t)$ but with the addition of additive white Gaussian noise (AWGN) $w(t)$. We will consider other distortions in the later chapters. The symbols are detected through the following steps illustrated in Figure 3.17.

- Through an Analog to Digital Converter (ADC), $r(t)$ is sampled at a rate of F_S samples/s to produce a sequence of T_S -spaced samples $r(nT_S)$.
- Next, $r(nT_S)$ is processed through a matched filter at the Rx side to generate $z(nT_S)$. As discussed earlier, the output of the matched filter $z(nT_S)$ is the correlation of the symbol-scaled pulse shape with an unscaled and time-reversed pulse shape $p(-nT_S)$.
- This output is downsampled by L at optimal sampling instants

$$n = mL = m \frac{T_M}{T_S} \quad (3.19)$$

to produce T_M -spaced numbers $z(mT_M)$ back from the signal. Take a special note of Eq (3.19). It will be employed over and over again.

- The minimum distance decision rule is employed to find the symbol estimates $\hat{a}[m]$.

Notice that a symbol is the basic building block of a digital communication system. Consequently, symbol time T_M is the basic unit of measurement along the time axis of such a system. While Figure 3.9 depicted sampling the output just once at the optimum instant 0 out of $-(L-1) \leq n \leq 0$, it is true for all integer multiples of symbol time in the case of a symbol sequence, i.e., the output is sampled just once for every symbol duration at optimum instants $T_M, 2T_M, \dots$.

Note 3.6 Key samples

Carefully examine the key samples at $T_M, 2T_M, \dots$. These are the samples we are looking for in the waveform for the detection purpose. Even when the waveform has suffered from all the distortions the real world has to offer, locating these samples and mapping them back to the constellation is a beautiful process, actual details of which we will encounter throughout this text.

Let us discuss the mathematical details of this process. The matched filter output is written as

$$z(nT_S) = r(nT_S) * h(nT_S) = \sum_l r(lT_S)h(nT_S - lT_S) \quad (3.20)$$

Now in a noiseless case, Rx signal $r(nT_S)$ is the same as $s(nT_S)$.

$$s(nT_S) = \sum_i a[i]p(nT_S - iT_M)$$

Moreover, the matched filter $h(nT_S)$ is the same as $p(-nT_S)$. The output of Eq (3.20) becomes

$$\begin{aligned} z(nT_S) &= \sum_l \left[\sum_i a[i]p(lT_S - iT_M) \right] p\{- (nT_S - lT_S)\} \\ &= \sum_i a[i] \cdot \sum_l p(lT_S - iT_M) p\{lT_S - iT_M - (nT_S - iT_M)\} \\ &= \sum_i a[i]r_p(nT_S - iT_M) \end{aligned} \quad (3.21)$$

where $r_p(nT_S)$ comes into play from the definition of auto-correlation function in Eq (3.6). To generate symbol decisions, T_M -spaced samples of the matched filter output are required at $n = mL = mT_M/T_S$. Downsampling the matched filter output generates

$$\begin{aligned} z(mT_M) &= z(nT_S) \Big|_{n=mL=mT_M/T_S} \\ &= \sum_i a[i]r_p(mT_M - iT_M) = \sum_i a[i]r_p\{(m-i)T_M\} \\ &= a[m] \end{aligned}$$

This is because for a rectangular pulse shape, the matched filter output is triangular with maximum occurring at $r_p[0]$, i.e., when

$$m - i = 0, \quad m = i$$

and zero at the next symbol location $r_p[L]$, see Figure 3.8.

Observe that the system shown in Figure 3.17 is a multirate system. In the PAM detector, for example, the ADC and the matched filter operate at the sample rate F_S . After the output of the matched filter is downsampled by L , the symbol decisions are made at the symbol rate $1/T_M$. Furthermore, there are some hidden assumptions in the PAM detector:

Resampling The ADC in general does not produce an integer number of samples per symbol, i.e., T_M/T_S is not an integer. A resampling system is required in the Rx chain that changes the sample rate from the ADC rate to a rate that is an integer multiple of the symbol rate.

Symbol Timing Synchronization The peak sample at the end of each symbol duration is not known in advance at the Rx and in fact does not necessarily coincide with a generated sample as well. This is because ADC just samples the incoming continuous waveform without any information about the symbol boundaries. This is a symbol timing synchronization problem which we will solve in Chapter 7.

Equalization: The above methodology works only in an AWGN channel. Multipath reflections from a wireless channel introduce ISI and distortion in the signal that need to be recovered through an equalizer. This is what we study in Chapter 8.

Matched Filter vs Correlator

We derived the concept of the matched filter in Section 3.4 starting with the definition of correlation. However, in communications jargon, a correlator is not the same as the matched filter and instead it is a block used in detectors with a slightly different functionality. Here, we elaborate the difference between a matched filter and a correlator in a communications Rx.

A correlator is a device that performs the correlation of a received signal with its template *within a given window of time*. In our context, that window of time is the symbol duration, T_M . So a correlator performs the following operations.

- It takes this sample-by-sample product and sums them together.
- Next, it samples the output of this accumulation at the optimal instant T_M .
- Finally, it *resets* itself at each sampling instant T_M , i.e., it starts computing the correlation again from zero for the next symbol. This is shown in Figure 3.18.

The matched filter, on the other hand, is just a filter that keeps convolving the input signal with the time reversed template while supplying the output at each multiple of T_M . Due to this reason, operating at L samples/symbol, both the matched filter and the correlator generate the same output at L^{th} sample but different outputs for the remaining $L - 1$ samples (that were going to be discarded anyway). This is plotted in Figure 3.18.

During the days when most of the signal processing was implemented in analog domain, the design of a correlator only required an analog mixer (multiplication) and an integrator (summation), the output of which could be sampled at multiples of T_M . On the other hand, the matched filter – while looking simpler on paper due to just a filtering operation – was more complicated due to the difficulty in designing an analog filter whose impulse response is some complicated pulse shape (we will see the examples of better pulse shapes than a rectangular one in Section 3.6) or a sequence. Hence, the correlator receiver was the preferred choice.

After the great advancement in computational power, most of the analog circuits capable of performing analog signal processing (e.g., by using resistors, capacitors, inductors, op amps and so on – *basically physics and devices*) have been replaced by powerful digital processors that can perform the required number crunching (*basically algorithms run by computer programs*) at a much better price vs performance point. Now, design and implementation of matched filters for any kind of applications is more convenient, requires less bookkeeping and combines efficiently with other receiver blocks such as synchronizers and equalizers[†].

[†]Sometimes I wonder how amazing our world looked like if a revolution like scaling down of a transistor size would have occurred in the mechanical domain as well.

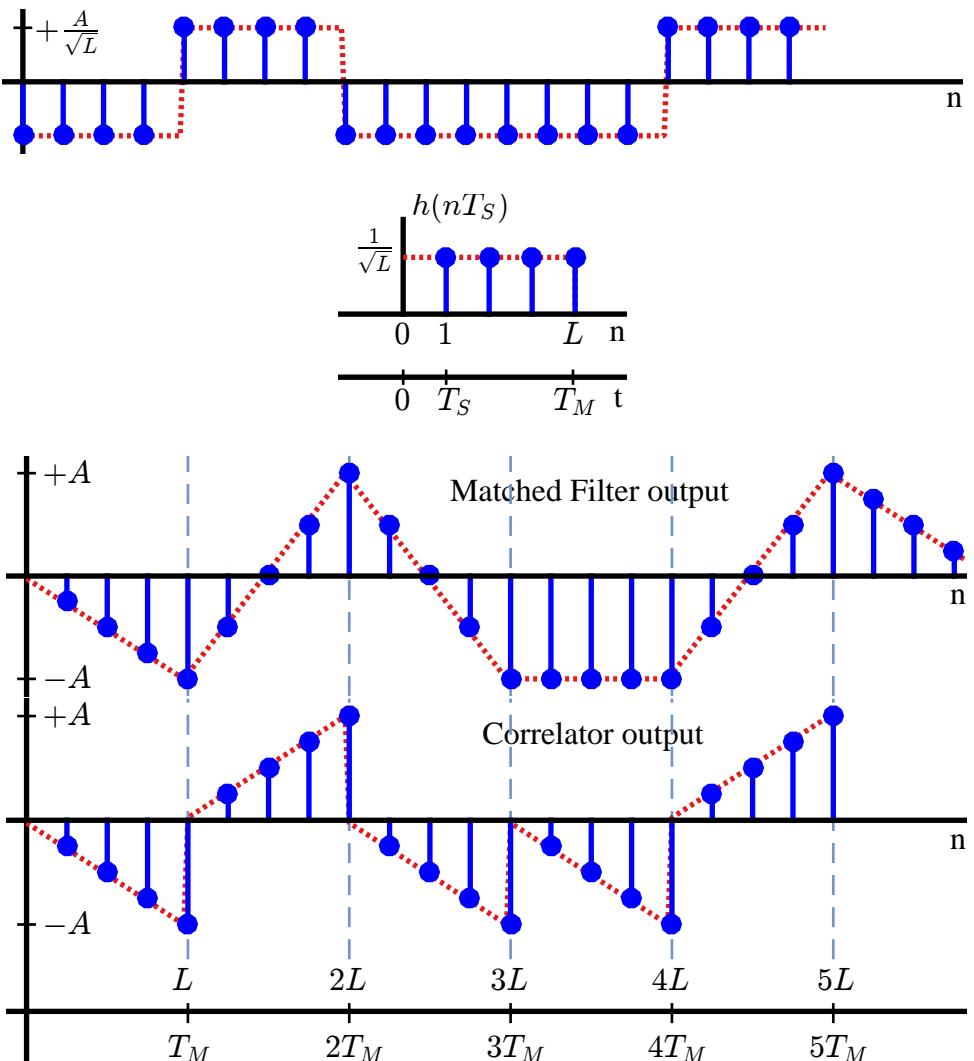


Figure 3.18: Matched filter vs correlator. The outputs are same at optimal sampling instants mT_M but different for the remaining symbol interval

To conclude, the matched filter computes *true correlation* of the received signal with the template signal for the duration of the whole symbol sequence, while the correlator resets itself to zero every symbol time.

Having understood the basic functionality of a PAM modulator and detector, we can discuss the important topic of a pulse shaping filter.

3.6 Pulse Shaping Filter

We have seen above how a Tx generates a signal from the numbers and the Rx recovers those numbers from that signal. In time domain, everything looks nice and perfect.

Let us investigate the system characteristics in frequency domain. As is clear from the block diagram of a PAM system, the main component that defines the spectral contents of the signal is the pulse shape $p(nT_S)$ at the Tx. We start with our attention towards the rectangular pulse shape used so far.

Note 3.7 Frequency domain view

It is straightforward to find the DFT of many signals from definition and examples in Chapter 1, or a software routine. The discrete frequency plot as a function of k , however, can only show a limited amount of frequency content due to distinct frequency domain samples. In the spirit of simplicity we have adopted in this text, we will interpolate many of the future graphs for a finer resolution. All we have to do is choose a large value of N when plotting.

As discussed in Section 1.8, for a reasonably band-limited signal, we can approximate the continuous frequency F by discrete frequency k/N when scaled by F_S . Then, does a larger N imply that we can cover a larger frequency range in reality? No, recall from Section 1.7 that $F = F_S \cdot k/N$ and

$$-0.5 \leq \frac{k}{N} = \frac{F}{F_S} < +0.5 \quad (3.22)$$

A large N just makes the resolution k/N finer and finer. The range k/N stays within -0.5 to $+0.5$ and so does F between $-0.5F_S$ and $+0.5F_S$, until the sample rate F_S itself changes. When we illustrate the frequency domain beyond this range, then we want to look into aliasing replicas of the spectrum as well.

Spectrum of a Rectangular Pulse

In time domain, a rectangular pulse has a clear benefit that Tx symbols do not naturally invade into each other's territory. In case there are no reflections or dispersions in the channel, there are no adjacent symbols interfering into each other at the Rx as well – a phenomenon called *Inter-Symbol Interference (ISI)*. ISI simplifies the detector in Rx design because everything related to a particular m^{th} symbol remains within the symbol interval $(m-1)T_M \leq t \leq mT_M$. Or in other words, *what happens in a symbol duration stays within that symbol duration*.

As for the frequency domain, we derived an analytical expression for the DFT of a rectangular sequence in Eq (1.67) of Section 1.10.1 as

$$|P_{\text{rect}}[k]| = \frac{\sin \pi L k / N}{\sin \pi k / N}$$

In the graph of this sinc function, nulls should occur wherever the numerator is zero, except at $k/N = 0$ where the peak exists. Now $\sin(\cdot) = 0$ when its argument is an integer multiple of π . Thus, equating the argument of the numerator to π returns the first null position.

$$\pi L \left(\frac{k}{N} \right)_{\text{null}} = \pi \quad \Rightarrow \quad \left(\frac{k}{N} \right)_{\text{null}} = \frac{1}{L}$$

Thus, the positions of nulls are $\pm 1/L, \pm 2/L, \pm 3/L, \dots$ with respect to discrete frequency k/N . In Hz, with $F_S = L/T_M$, the null positions are $\pm 1/T_M, \pm 2/T_M, \pm 3/T_M, \dots$ because

$$F_{\text{null}} = F_S \cdot \left(\frac{k}{N} \right)_{\text{null}} = F_S \cdot \frac{1}{L} = \frac{1}{T_M}$$

Finally, the range is

$$-0.5F_S = -0.5 \frac{L}{T_M} \leq F < +0.5 \frac{L}{T_M} = +0.5F_S$$

which for $L = 8$ samples/symbol produces

$$-\frac{4}{T_M} \leq F < +\frac{4}{T_M}$$

The spectrum of a rectangular pulse shape shows these deductions in Figure 3.19.

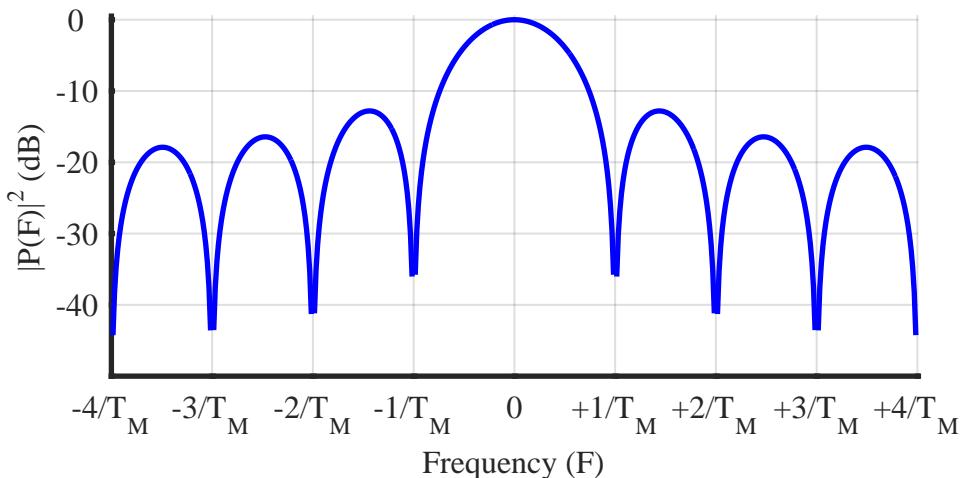


Figure 3.19: Spectrum of a rectangular pulse shape with $L = 8$ samples/symbol

A great disadvantage of a rectangular pulse shape is now visible: the sidelobe level is just 13 dB below the mainlobe, a very high value. Furthermore, the subsequent sidelobes are not decaying fast enough. If there is a communications link which occupies the drawn spectrum and the neighboring spectrum is allocated to someone else, then there will be a lot of interference between the two parties due to their sidelobe energies interfering with each other. In frequency domain, this is known as adjacent channel interference.

Just like real estate, radio spectrum is a very precious resource and must be utilized judiciously. This is why there are spectrum regulatory authorities in every country who impose strict restrictions to comply with a *spectral mask*. Even for wired channels, there is always a natural bandwidth of the medium (copper wire, coaxial cable, optical fiber) that imposes upper limits on its utilization.

Not only that a rectangular pulse shape is a poor choice due to its large spectral occupancy, another side effect of having a wider bandwidth pulse shape is that the matched filter at the Rx also has a similar bandwidth. Viewed in frequency domain, the larger the bandwidth of the Rx filter, the more noise and interference it allows into the system.

Note 3.8 Randomness of bit stream

It is important to remember that such a spectrum approximately appears at the output of the transmitter only if there is a sufficient randomness in the bit stream, i.e., bits 0 and 1 occur with equal probability and so there are enough transitions between them. Any pattern in the bit stream can alter the output spectrum significantly.

As an example, imagine a constant bit stream of 1's that is then modulated to a symbol level $+A$. After pulse scaling, the output is nothing but $+A$ times an all-ones sequence that has a Fourier Transform of an impulse at bin 0 (DC) and nothing else. In real applications, there is enough randomness in the modulated sequence and hence the spectrum will closely resemble the spectrum shown in Figure 3.19.

Reducing the bandwidth

Having known the disadvantages of a rectangular pulse shape, a superior pulse shaping filter needs to be looked for that should help in placing multiple channels adjacent to each other while minimizing inter-channel interference between them as well as noise bandwidth at the Rx. However, such advantages should not come at a price of introducing ISI in the system.

Nyquist No-ISI Criterion in Time Domain

To get that superior pulse shape, we trace the following sequence of steps through the help of Figure 3.20[†] that plots the auto-correlation $r_p(nT_S)$ of both a rectangular and a conceptual pulse shape. Auto-correlation of a pulse shape was defined in Eq (3.6) as

$$r_p(nT_S) = \sum_i p(iT_S)p(iT_S - nT_S)$$

and plots are shown in continuous-time to make the related steps below more understandable.

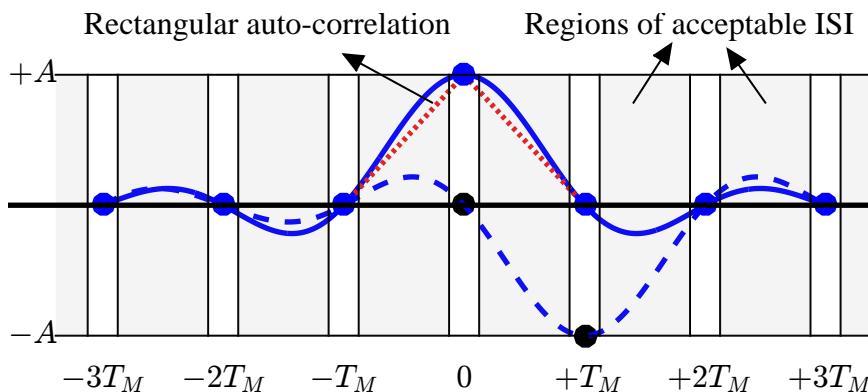


Figure 3.20: Auto-correlation functions $r_p(nT_S)$ of a rectangular pulse and a conceptual pulse extended over time

[†]Thanks to Eric Jacobsen for his wonderful article Ref. [8], Figure 4 of which provided the inspiration behind this figure.

- Recall that signals that are narrow in time domain are wide in frequency domain, and vice versa. Accordingly, a consequence of narrow time span of the pulse shape is that it gets expanded in frequency. That is why a rectangular pulse has a large bandwidth. To decrease the bandwidth, the length of the pulse shape needs to be extended in time domain – much more than a single symbol duration T_M .
- The effect of extending the length of the pulse shape is that every symbol now interferes with a number of symbols occurring both before and after that symbol. The dreaded ISI appears! Assuming a usual even shaped pulse, every symbol contributes towards ISI in symbols towards its left and symbols towards its right on the time axis.
- To strike peace with this ISI, we exploit a key property of digital communication waveforms explained before: the critical samples at the output of matched filter occur at the end of integer multiples of symbol duration T_M . For symbol detection, only these T_M -spaced samples are needed and the rest can be discarded. This is what we saw at the output of downampler in Figure 3.17.

If all but one sample during each symbol interval T_M are discarded, they can be assigned any value without any effect on the detector output. These are the “don’t care” samples we give up for ISI to play with in the expectation that we get to shape our desired spectrum in return, and with the result that our desired T_M -spaced samples remain intact. Keep in mind that *we are talking about the output of the matched filter – and hence auto-correlation of the pulse shape* – here. The reason is to elaborate the nature of signal that is eventually downsampled to yield symbol estimates. The actual pulse shape will be extracted from this auto-correlation, as we will shortly see.

- In the light of above discussion, three pulse shape auto-correlations are illustrated in Figure 3.20.
 1. A rectangular pulse shape has a duration of T_M seconds, or L samples. Hence, its auto-correlation extending over $2T_M$ seconds is shown as a dotted red line. Notice that it has a maximum value at the desired current sampling instant and zero for adjacent two symbols. The time span, however, is too short giving rise to an unreasonably large spectrum.
 2. A pulse shape auto-correlation that is expanded over many symbols is shown in blue curve, where it is scaled by a symbol $+A$. Note that at all the sampling instants (integer multiples of T_M), the effect of this particular symbol is zero. We intend to assign values to its “don’t care” samples in shaded regions to achieve a compact spectrum.
 3. An adjacent symbol $-A$ scales the same pulse shape at time T_M . Observe the same zero-ISI effect on the current symbol at time 0 and on all other symbols.

Not shown in the above figure are all other symbols shaped by pulses at $2T_M$, $3T_M$, and so on. But it is clear that sum total of ISI from all adjacent symbols is zero at sampling instant of every single symbol. For a concrete formulation of this zero-ISI criterion, we turn towards its mathematical foundation.

Just as in the case of rectangular pulse before, the input signal is

$$s(nT_S) = \sum_i a[i] p(nT_S - iT_M)$$

When this signal[†] is input to a matched filter $h(nT_S) = p(-nT_S)$, the output from Eq (3.21) is written as

$$\begin{aligned} z(nT_S) &= \left(\sum_i a[i] p(nT_S - iT_M) \right) * p(-nT_S) \\ &= \sum_i a[i] r_p(nT_S - iT_M) \end{aligned}$$

To generate symbol decisions, T_M -spaced samples of the matched filter output at $n = mL = mT_M/T_S$ are

$$\begin{aligned} z(mT_M) &= z(nT_S) \Big|_{n=mL=mT_M/T_S} \\ &= \sum_i a[i] r_p(mT_M - iT_M) = \sum_i a[i] r_p\{(m-i)T_M\} \\ &= \underbrace{a[m] r_p(0T_M)}_{\text{current symbol}} + \underbrace{\sum_{i \neq m} a[i] r_p\{(m-i)T_M\}}_{\text{ISI}} \end{aligned} \quad (3.23)$$

In the above equation, the first term is the currently desired symbol and the second term is Inter-Symbol Interference (ISI). It is obvious that ISI can be zero if the pulse auto-correlation satisfies the condition

$$r_p(mT_M) = \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases} \quad (3.24)$$

This is Nyquist no-ISI criterion in time domain – a mathematical expression of the same concept we explained above: to obtain zero ISI, the pulse auto-correlation should pass through zero for all integer multiples of T_M before and after the current symbol.

[†]Although this relation was derived in Eq (3.17) for a pulse with duration T_M , it is also true for longer pulses because

$$\begin{aligned} \sum_i a[i] p(nT_S - iT_M) &= a[0] p(nT_S) + a[1] p(nT_S - T_M) + \\ &\quad a[2] p(nT_S - 2T_M) + \dots \\ &= (a[0] \delta[nT_S] + a[1] \delta[nT_S - T_M] + \\ &\quad a[2] \delta[nT_S - 2T_M] + \dots) * p(nT_S) \end{aligned}$$

This is because convolution between any signal and a unit impulse results in the same signal. Therefore, even for longer pulses, scaling each individual symbol with its own pulse shape is equivalent to convolution of upsampled symbol stream with a pulse shaping filter.

Note 3.9 Nyquist filter

A filter with impulse response coefficients satisfying Nyquist no-ISI criterion is called a **Nyquist filter**. This is also usually called a Nyquist pulse but the Nyquist filter is a better term, as this is actually not the pulse shape but the auto-correlation of the underlying pulse shape. The coefficients of the pulse shape itself will be derived later.

When Nyquist no-ISI criterion in time domain is satisfied, plugging Eq (3.24) into Eq (3.23) gives

$$z(mT_M) = a[m]$$

i.e., the downsampled matched filter output maps back to the Tx symbol in the absence of noise. This is the crux of digital communications theory.

Nyquist No-ISI Criterion in Frequency Domain

Throughout the text, we have been avoiding complicated mathematical derivations. We will do the same here and instead of proving Nyquist no-ISI theorem which connects time and frequency domain properties of pulse auto-correlation, we again take the intuitive route.

Remember that the point of this whole exercise was to limit the enormous bandwidth occupied by a time domain rectangular pulse. In other words, the quest is to design a pulse shape such that it complies with the spectral mask within a channel bandwidth B . Nyquist no-ISI criterion in frequency domain helps here.

As long as we operate at a sample rate F_S , the spectral replicas are spaced at $F_S = L/T_M$ apart. But since we downsample $r_p(nT_S)$ by $L = T_M/T_S$ to get $r_p(mT_M)$, our sample rate changes from F_S at the output of matched filter to $1/T_M$ after the downampler. To see what happens in frequency domain, observe the following.

- In discrete domain, $r_p(mT_M)$ in Eq (3.24) is nothing but a single impulse at time 0. This can be seen as the blue dots in Figure 3.21a. Hence, its DFT is an all-ones rectangular sequence in frequency domain as

$$R_p[k] = 1$$

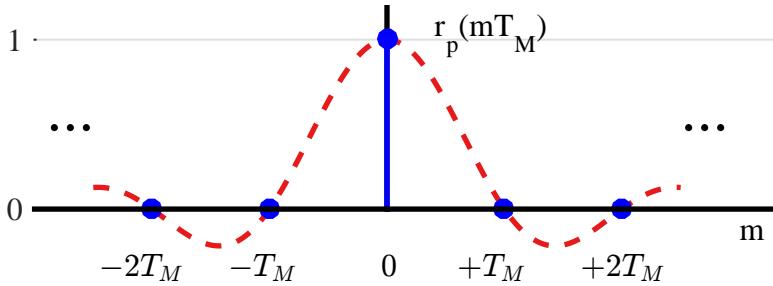
In terms of continuous frequency, we say that it is equal to a constant 1.

- Moreover, we know from Section 2.7.1 that a consequence of downsampling by L is appearance of new spectral aliases $F_S/L = 1/T_M$ apart from each other.

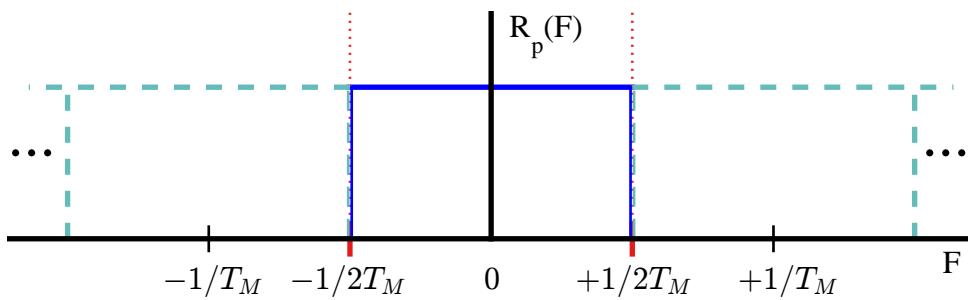
Combining the above two facts produces Figure 3.21b illustrating this relationship between symbol-spaced pulse auto-correlation and its Fourier Transform.

This is exactly what Nyquist in Ref. [9] showed: a necessary and sufficient condition for zero-ISI condition of Eq (3.24) is that the Fourier transform $R_p(F)$ of the pulse auto-correlation satisfies the condition

$$\begin{aligned} \sum_{i=-\infty}^{\infty} R_p(F + \frac{i}{T_M}) &= \cdots + R_p(F + \frac{2}{T_M}) + R_p(F + \frac{1}{T_M}) + \\ &\quad R_p(F) + R_p(F - \frac{1}{T_M}) + R_p(F - \frac{2}{T_M}) + \cdots \\ &= T_M \end{aligned} \tag{3.25}$$



(a) Time domain pulse auto-correlation downsampled by L



(b) DFT of pulse auto-correlation downsampled by L . Observe the spectrum replicas centered at $1/T_M$ Hz

Figure 3.21: Time and frequency domain interpretations of Nyquist no-ISI criterion

where the factor T_M is reminiscent of the factor N in iDFT definition. Since Figure 3.21b depicts that *the spectrum of the pulse auto-correlation should be a constant T_M* , we perform the following steps.

- Draw the primary spectrum $R_p(F)$ whose range is $-0.5F_S \leq F < +0.5F_S$.
- Draw its shifted replicas at frequencies $\pm 1/T_M$. Repeat the same for all integer multiples of $\pm 1/T_M$ from $-\infty$ to $+\infty$.
- Add all these replicas together.
- If this sum results in a constant T_M , only then the time domain pulse auto-correlation will satisfy the no-ISI condition (3.24).

Figure 3.22 shows two spectra, one of which satisfies the Nyquist no-ISI criterion while the other does not. The spectrum on the left is constant because the bandwidth is as wide as allowed by the sampling theorem (which is $B_{\max} = 1/2T_M$ for sample rate $1/T_M$). This not only avoids aliasing but a flat spectrum within $-0.5/T_M \leq F < +0.5/T_M$ produces a flat spectrum in the range $-1.5/T_M \leq F < -0.5/T_M$, $+0.5/T_M \leq F < +1.5/T_M$, and so on. Consequently, the cumulative spectrum satisfies Eq (3.25) as a constant function of frequency from $-\infty$ to $+\infty$.

We can also see that the signal bandwidth B in this particular case is

$$B = \frac{1}{2T_M}$$

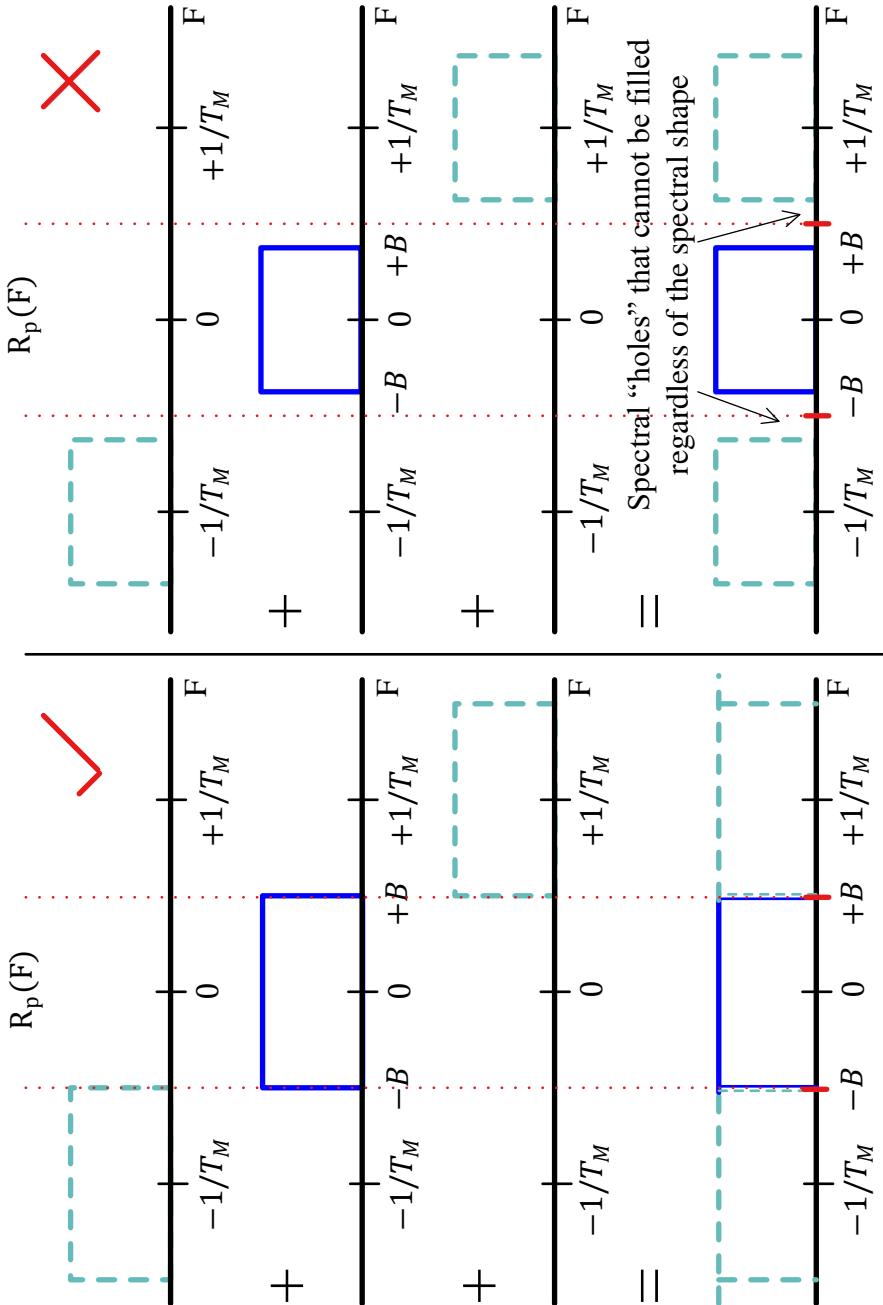


Figure 3.22: The spectrum on the left satisfies the Nyquist no-ISI criterion in frequency domain, while the one on the right does not

A word of caution: Never confuse a rectangular pulse shape in time domain with a pulse auto-correlation that is a rectangle in frequency domain. The above expression is the maximum bandwidth after which aliasing will occur from spectral replicas. This also sets the fundamental limit between the bandwidth B and symbol rate $1/T_M$ as

$$\left\{ \frac{1}{T_M} \right\}_{\max} \leq 2B \quad (3.26)$$

Hence, with zero ISI, a maximum symbol rate $1/T_M$ equal to $2B$ symbols/second can be supported within a bandwidth B^{\dagger} .

Interestingly, *this maximum bandwidth $1/2T_M$* allowed by sampling theorem is also the *minimum bandwidth* allowed by Nyquist no-ISI criterion in frequency domain through Eq (3.25). Any smaller than this, and it becomes impossible to obtain an overall flat spectrum. That is shown on the right of Figure 3.22 where there are spectral "holes" that cannot be filled regardless of the spectral shape.

Coefficients of an Ideal Pulse Auto-correlation

Above, we said that a filter whose impulse response coefficients satisfy Nyquist no-ISI criterion is a Nyquist filter, which is actually the auto-correlation $r_p(nT_S)$ of the underlying pulse shape. From Figure 3.22 and the discussion in the last section, *we find our ideal Nyquist filter*: a pulse auto-correlation $r_p(nT_S)$ whose spectrum $R_p(F)$ is a rectangle from $-1/2T_M$ to $+1/2T_M$. This filter is shown in Figure 3.21 and left of Figure 3.22.

To derive the time domain representation of such a filter, remember that a rectangle in frequency domain is a sinc in time domain due to time frequency duality. This rectangle covers the entire continuous frequency range from $F = -1/2T_M$ to $+1/2T_M$ (or discrete frequency range $k/N = -0.5$ to $+0.5$) yielding $L = N$ in Eq (1.67). Note that this L is not samples/symbol, just the sequence length in that equation. There is a scaling factor of $1/N$, furthermore, from the definition of iDFT. Combining these facts,

$$s[n] = \frac{1}{N} \frac{\sin \pi n}{\sin \pi n/N}$$

For our purpose with a sample rate of $1/T_M$ so far,

$$r_p[m] = \frac{1}{N} \frac{\sin \pi m}{\sin \pi m/N}$$

It satisfies Nyquist no-ISI criterion in Eq (3.24) because it is equal to 1 at $m = 0$ and zero for $m \neq 0$ and shown as blue dots in Figure 3.21.

Using the identity $\sin \theta \approx \theta$ for small θ , the term in the denominator is quite small as compared to numerator due to division by N . Then, the above equation can be written as

$$r_p[m] \approx \frac{1}{N} \frac{\sin \pi m}{\pi m/N} = \frac{\sin \pi m}{\pi m}$$

[†]An extension of such a framework is Faster Than Nyquist (FTN) signaling in which this fundamental limit is broken by design to send more symbols within the same bandwidth. The cost of such a setup is a loss of orthogonality, i.e., occurrence of ISI due to closer packing of symbols than T_M which is removed at the Rx through more computationally complex algorithms, usually by employing an iterative error correcting code.

This is the time domain waveform sampled at rate $1/T_M$. As we are interested in time domain coefficients $r_p[n]$ of this ideal pulse auto-correlation, we sample the waveform at rate $F_S = L/T_M$ instead of just $1/T_M$, an increase by a factor of L . Correspondingly, a decrease in sample time implies we plug $m = n/L$ in the above equation.

$$r_p[n] = \frac{\sin \pi n/L}{\pi n/L} \quad (3.27)$$

These are the coefficients of our ideal pulse auto-correlation, which extends from $-\infty$ to $+\infty$ in time domain as shown in Figure 3.23.

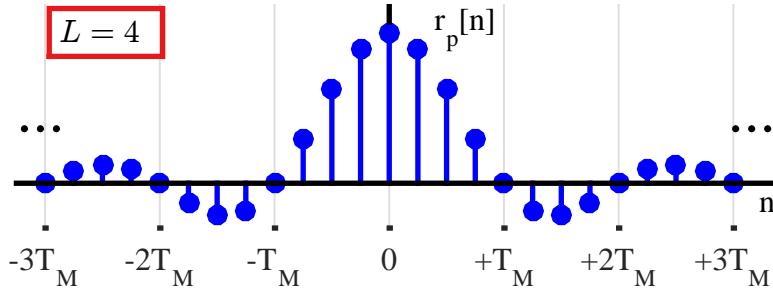


Figure 3.23: Coefficients of an ideal pulse auto-correlation with $L = 4$ samples/symbol

There are two main problems with this ideal pulse auto-correlation, *both arising due to a very slow rate of decay for the tail*.

Truncation Error Any practical system cannot use a filter with an infinite number of coefficients like the one shown in Figure 3.23. Hence, the filter impulse response must be truncated at a few symbols to the left and a few symbols to the right of the current symbol. To ensure that this truncated filter closely approximates the ideal impulse response, only very small values can be ignored. That generates a large filter length N .

Timing Errors We know that an optimal timing instant coincides with symbol boundaries. Timing synchronization block in a Rx is responsible for extracting this symbol-aligned clock. However, even a small error in timing synchronization output causes a significant increase in ISI due to large value of samples at the tail interacting with neighbouring modulated pulses. This is demonstrated later in Section 7.1. We want a quicker tail decay as it leads to less errors arising from the timing jitter when sampling adjacent pulses.

To address these issues, we take the reverse route now. We started with the aim of reducing the bandwidth of a rectangular pulse shape. We found an ideal pulse auto-correlation with a rectangular spectrum that satisfies Nyquist no-ISI criterion. To overcome the limitations in time domain associated with it, we seek to trade-off some bandwidth with a desirable time domain behaviour.

Raised Cosine (RC) Filter

So the target is to (slightly) increase the bandwidth now. Remember that we observed regions of acceptable ISI in Figure 3.20 and deduced that these “don’t care” samples

can be given up for ISI to play with in the expectation that we get to shape our desired spectrum in return, while our T_M -spaced samples remain intact. *If there is a relaxation and a restriction in time domain, then intuitively there should be a corresponding relaxation and restriction in frequency domain as well.* We have found the restriction to be a flat spectrum across the whole frequency range. The relaxation is that how this spectrum becomes flat is a matter of no concern. The actual spectral shape is a “don’t care” case, as long as the sum of all shifted replicas is flat.

It is clear that we cannot reduce the bandwidth below the fundamental limit $1/2T_M$, see Figure 3.22. The only way to alter the spectral shape is by expanding the bandwidth beyond $1/2T_M$ in such a way that the sum total of the replicas again becomes flat. For this purpose, an odd symmetry around the point $1/2T_M$ is required. The advantage of such odd symmetry is that the spectral magnitudes before and after $1/2T_M$ are 180° rotated versions of each other. In other words, the spectral copies at $\pm 1/T_M$ fold around $\pm 1/2T_M$ into the original bandwidth. That supplies the additional amplitude required to bring the spectrum in a flat shape, as illustrated in Figure 3.24.

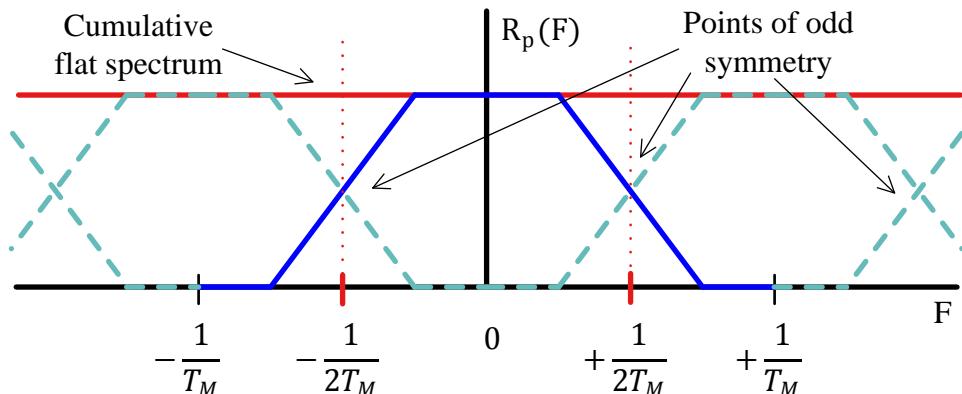


Figure 3.24: Odd symmetry around frequency points $\pm 1/2T_M$ adding up to a flat spectrum

Higher frequency components of a signal arise from abrupt changes in time domain, which was the reason of large spectral occupancy of a rectangular pulse shape. Similarly, long tails in a time domain signal arise from abrupt transition in the flat spectrum as in Figure 3.21b. Also recall that the length of an FIR filter depends on its transition bandwidth.

Having located the abrupt transition bandwidth around $\pm 1/2T_M$ as the root cause, we can extend the bandwidth of the pulse auto-correlation in any shape as long as it has odd symmetry around the points $\pm 1/2T_M$. The purpose is to smooth out its spectrum so that it has a short time domain support.

The smoothest spectral shape one can imagine is a sinusoid. If such a spectral taper is convolved with the ideal rectangular spectrum, the discontinuity in the spectrum can be removed. Since a half-cosine is an even symmetric shape, it is shown to be convolved with the rectangular spectrum in Figure 3.25. The width of the half-cosine is α/T_M where $0 \leq \alpha \leq 1$ which forms the transition bandwidth of the resultant spectrum and its even symmetric shape preserves the odd symmetry around $\pm 1/2T_M$. As a consequence of this odd symmetry, this is also a Nyquist filter. The resultant spectrum out of this convolution is discussed soon.

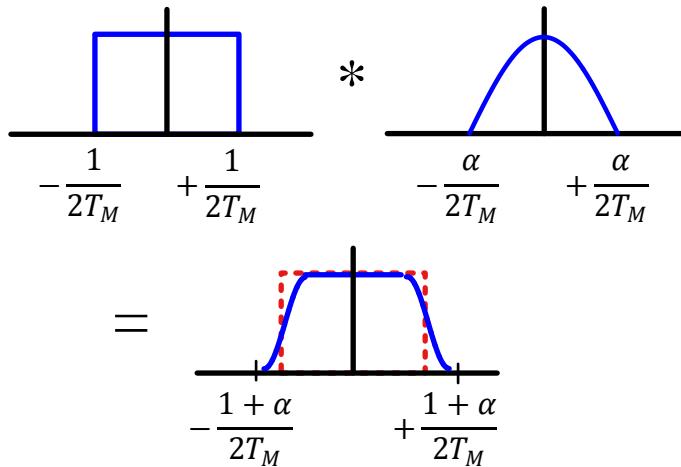


Figure 3.25: Spectral convolution of ideal Nyquist spectrum with even symmetric half-cosine

It can be observed that the effect of spectral convolution is an increase in bandwidth from $1/2T_M$ to $(1 + \alpha)/2T_M$ (convolution length in continuous domain is the sum of lengths of the two signals). Since α specifies the bandwidth beyond the minimum bandwidth $1/2T_M$ (Nyquist frequency or folding frequency), it is called **excess bandwidth** or **roll-off factor**. For example, when $\alpha = 0.25$, the total bandwidth is 25% more than the minimum and when $\alpha = 1$, the total bandwidth is exactly 100% more than, or twice, the minimum. Typical values of α range from 0.1 to 0.3.

In time domain, this is equivalent to product of a sinc signal with an even signal of time[†]. The sinc signal guarantees the zero crossings as they cannot be moved by multiplication, and the even signal dampens the long tails in time. The higher the bandwidth, the narrower this signal in time and hence faster the decay. The exact diagram of this process is drawn soon in Figure 3.27a.

First, consider $\alpha = 1$, the widest bandwidth case. Here, a rectangle of spectral width $1/T_M$ gets convolved with a half-cosine of width $1/T_M$. The convolution of the resultant spectrum again generates a cosine shape from $F = -1/T_M$ to $+1/T_M$ for a total width of $2/T_M$. This is plotted in Figure 3.26, where smooth frequency transition can be seen with an odd symmetry around $\pm 1/2T_M$. It is commonly known as a **Raised Cosine (RC)** filter and has nothing to do with an RC circuit consisting of a resistance and a capacitor. The name raised cosine comes from the fact that *the transition bandwidth of the spectrum consists of a cosine raised by a constant to make it non-negative* as shown in Figure 3.26. Keep in mind that it is a raised cosine in frequency domain, not time domain.

Let us explore its mathematical expression. Due to the way it is usually written in literature, the formula looks more complicated than it actually is. Recall that a sinusoid in time domain is written as

$$s(t) = \cos(2\pi \underbrace{F}_{=1/T, \text{ inverse period}} \underbrace{t}_{\text{independent variable}})$$

[†]Since the spectrum is real and even, the time domain signal is also real and even.

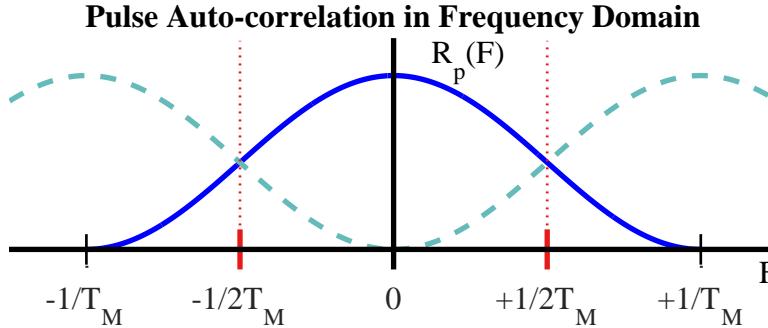


Figure 3.26: Raised Cosine spectrum with excess bandwidth $\alpha = 1$

Inverse period by definition is frequency; however, I devised this expression since we are going to use it into frequency domain. A frequency domain sinusoid will just interchange these roles, with inverse period and independent variable defined in terms of frequency. We change the above equation as follows.

- A cosine from $F = -1/T_M$ to $+1/T_M$ implies that the period in frequency is $2/T_M$ (the inverse of which is $T_M/2$).
- It is raised by a constant 1 to make it non-negative, as the smallest value of a cosine is -1 .
- Then, $1 + \cos(\cdot)$ will become 2 at $F = 0$, so it is scaled by $1/2$ to get a unity gain.
- Finally, a scaling factor of T_M comes from Nyquist no-ISI criterion in frequency domain, see Eq (3.25).

Thus, the expression for this cosine spectrum on the positive half is given as

$$\begin{aligned} R_p(F) &= \frac{T_M}{2} \left[1 + \cos(2\pi \underbrace{1/(2/T_M)}_{\text{inverse period}} \underbrace{F}_{\text{independent variable}}) \right] \\ &= \frac{T_M}{2} \left[1 + \cos(2\pi \cdot \frac{T_M}{2} \cdot F) \right] \quad 0 \leq F \leq +\frac{1}{T_M} \end{aligned}$$

The overall spectral shape from $-1/T_M$ to $+1/T_M$ can be given as

$$R_p(F) = \frac{T_M}{2} \left[1 + \cos \left(2\pi \cdot \frac{T_M}{2} \cdot |F| \right) \right] \quad -\frac{1}{T_M} \leq F \leq +\frac{1}{T_M} \quad (3.28)$$

where the negative half of the spectrum is the same as the positive half, and hence the term $|F|$.

We obtain its time domain expression through the following steps:

- In frequency domain, the constant term and cosine in the above expression do not have a frequency support from $-\infty$ to ∞ , but only from $-1/T_M$ to $+1/T_M$.
- In frequency domain, that is equivalent to multiplication with a rectangular window of the same width.

- Multiplication in frequency domain is convolution in time domain.
- In time domain, the constant term translates to an *impulse* at time location 0, and $\cos(\cdot)$ results in *two impulses* with half that amplitude at time locations (inverse period) $\pm T_M/2$.
- The rectangular window of width $2/T_M$ that defines its support in frequency is a sinc signal in time domain with zero crossings at integer multiples of $\pm T_M/2$.
- Convolution of a signal with an impulse is the signal itself. When convolved with sinc arising from the window, this generates three sinc signals: one at time 0 (convolution of the above sinc with impulse from constant term) while two at time $\pm T_M/2$ (convolution of the above sinc with impulses from the cosine).

The above sequence of steps result in time domain signal of an extended bandwidth pulse auto-correlation shown in Figure 3.27. The reason of choosing a fre-

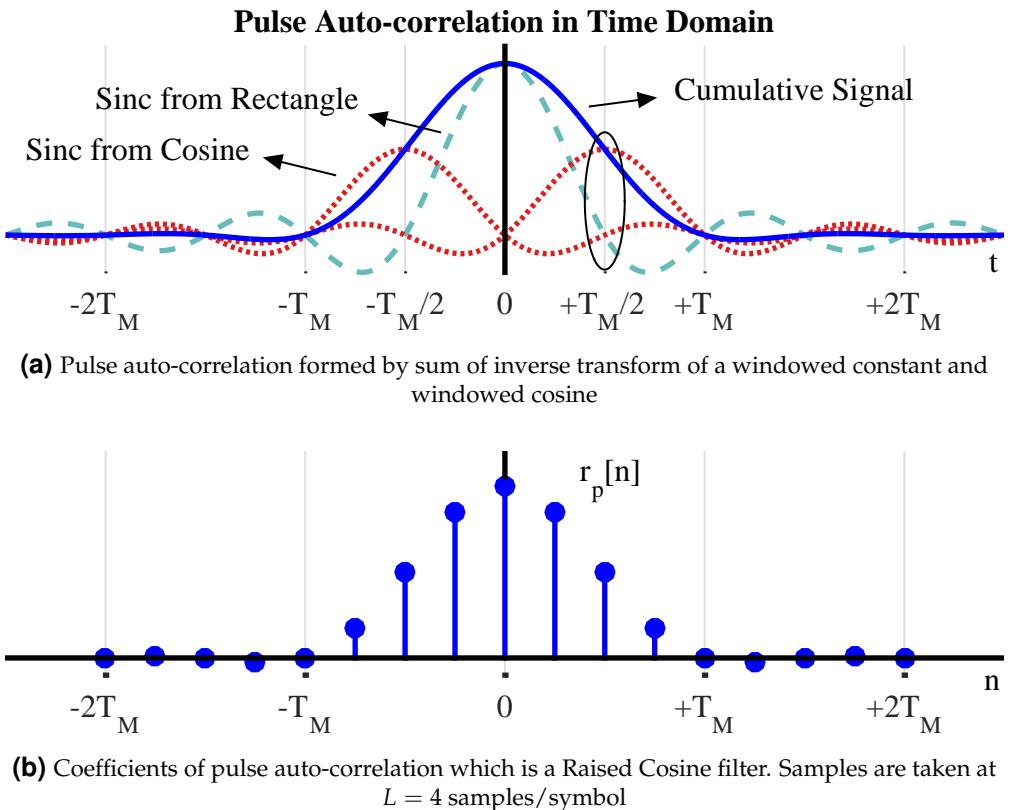


Figure 3.27: Time domain view of pulse auto-correlation of Figure 3.26 and its sampled coefficients

quency support $-1/T_M \leq F \leq +1/T_M$ is now clear. With support $B = 2/T_M$ in frequency, the sinc resulting from cosine adds out of phase at $t = T_M/2$ with the sinc from the constant term. The signs of their tails are opposite to each other and that is what brings down the levels of long tails of an ideal pulse auto-correlation. Furthermore, sincs from both constant term and cosine have zero crossings at integer

multiples of $\pm T_M/2$, as marked through the ellipse in the figure. Nevertheless, a time shift of $T_M/2$ for the sinc from cosine causes the cumulative signal to pass through zero at integer multiples of $\pm T_M$. This is how this pulse auto-correlation fulfills the Nyquist no-ISI criterion in time domain. Finally, sampling the pulse auto-correlation at $L = 4$ samples/symbol produces the discrete-time coefficients that can be used for pulse shaping in digital domain.

Compare this discrete-time signal with the ideal Nyquist filter in Figure 3.23. Now we can truncate the filter to a small length without much loss in accuracy. In addition, a sampling time misalignment at the Rx will have a relatively less influence on the current symbol. Remember that the cost of this relief is the wider bandwidth $1/T_M$. Observing this trend, we can think of striking a trade-off between the filter length and bandwidth expansion. The excess bandwidth α comes into play here.

For this purpose, the excess bandwidth α can be reduced from $\alpha = 1$ to a lower value. By reducing the bandwidth, the following sequence occurs.

- When $\alpha < 1$, the period in frequency of the half-cosine (on each side of zero) in Figure 3.26 decreases, while still maintaining odd symmetry around $\pm 1/2T_M$.
- As shown in Figure 3.28, a decrease in frequency period results in three distinct regions of the spectrum: a half-cosine in the positive part of spectrum

$$+\frac{1-\alpha}{2T_M} \leq F \leq +\frac{1+\alpha}{2T_M},$$

a half-cosine in the negative portion of the spectrum

$$-\frac{1+\alpha}{2T_M} \leq F \leq -\frac{1-\alpha}{2T_M},$$

and a constant term

$$-\frac{1-\alpha}{2T_M} \leq F \leq +\frac{1-\alpha}{2T_M}.$$

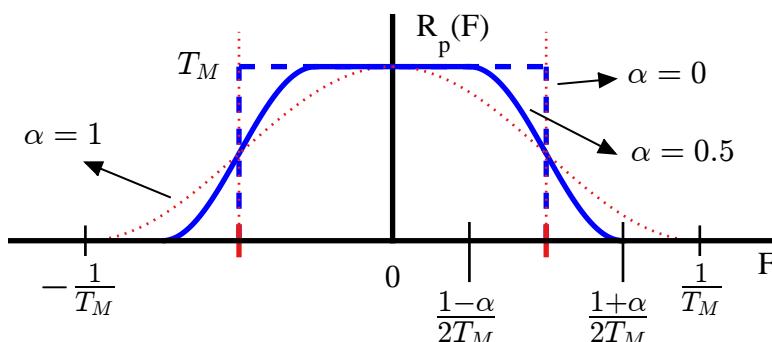


Figure 3.28: Raised Cosine spectrum with different excess bandwidths α . Note that $\alpha = 1$ is the maximum bandwidth case, while $\alpha = 0$ coincides with a rectangular spectrum of ideal Nyquist filter as in Figure 3.21 and Figure 3.22

Note that when $\alpha = 1$, the bandwidth was $1/T_M$ which can be written as $(1+1)/2T_M$. A decrease in α results in new bandwidth $(1+\alpha)/2T_M$ as drawn in

Figure 3.28. Also observe that $\alpha = 0$ case coincides with a rectangular spectrum of ideal Nyquist filter as in Figure 3.21 and Figure 3.22. This was the minimum bandwidth allowed by Nyquist no-ISI criterion.

- A decrease in bandwidth results in multiple windowed cosines and a constant. To avoid complication, its detailed analysis is not necessary. We just note that as α gets smaller and smaller, the sincs in time domain resulting from windowed cosine in frequency move away from each other, thus causing less tail suppression as compared to $\alpha = 1$ case. In conclusion, the excess bandwidth α gives the system designer a trade-off between reduced bandwidth and time domain tail suppression.

Taking clue from Eq (3.28), the general expression for a Raised Cosine can be written as

$$R_p(F) = \begin{cases} T_M & 0 \leq |F| \leq \frac{1-\alpha}{2T_M} \\ \frac{T_M}{2} \left[1 + \cos \left\{ 2\pi \cdot \frac{T_M}{2\alpha} \left(|F| - \frac{1-\alpha}{2T_M} \right) \right\} \right] & \frac{1-\alpha}{2T_M} \leq |F| \leq \frac{1+\alpha}{2T_M} \\ 0 & |F| \geq \frac{1+\alpha}{2T_M} \end{cases} \quad (3.29)$$

The above equation looks intimidating but it is not. There are only three minor differences from Eq (3.28).

1. The first term is a constant occupying the bandwidth left behind by the half-cosine. It is equal to T_M due to plugging $F = 0$ in the expression.
2. $T_M/2\alpha$ in the middle term shows a reduction in bandwidth and a spectral shift of cosine center from $F = 0$ to $F = (1 - \alpha)/2T_M$. This is the edge of the passband now and $(1 - \alpha)/2T_M$ is called the **passband frequency**.
3. The final term shows no bandwidth occupied from $(1 + \alpha)/2T_M$ onwards. This is the start of the stopband and $(1 + \alpha)/2T_M$ is called the **stopband frequency**.

Now it is clear why an oversampling factor of L samples/symbol – a sampling rate of $L \cdot 1/T_M$ – is used in digital communication systems. Applying sampling theorem to the stopband frequency $(1 + \alpha)/2T_M$, the sampling theorem sets the minimum sampling rate at $(1 + \alpha)/T_M$. So at least $L = 2$ (the closest integer to any $1 + \alpha$) or more samples are required per symbol.

To obtain its time domain expression, recall Figure 3.25 where we found that the resultant waveform in time domain is equivalent to product of a sinc signal with an even signal of time. Without exploring the expression for that even signal, we just state the final form that can be derived from the inverse transform of Eq (3.29).

$$r_p[n] = \frac{\sin(\pi n/L)}{\pi n/L} \cdot \frac{\cos(\pi \alpha n/L)}{1 - (2\alpha n/L)^2} \quad (3.30)$$

This is drawn in Figure 3.29 for different values of $\alpha = 0, 0.5$ and 1 . Note the simultaneous zero crossings of all waveforms at integer multiples of T_M . Also, plugging

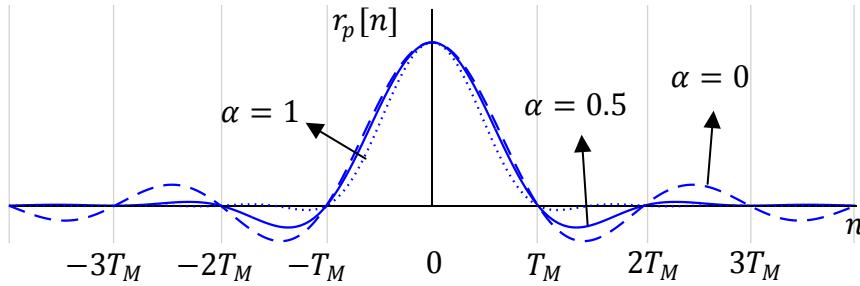


Figure 3.29: Raised Cosine waveform in time domain with different excess bandwidths α with $L = 64$ samples/symbol. Notice the simultaneous zero crossings of all waveforms at integer multiples of T_M

$\alpha = 0$ above produces the coefficients of a sinc signal, the same ideal Nyquist filter in Eq (3.27).

The above equation becomes indeterminate for $n = 0$ and $n = \pm L/(2\alpha)$. It can be shown that

$$r_p[0] = 1$$

$$r_p\left[\pm \frac{L}{2\alpha}\right] = \frac{\alpha}{2} \sin \frac{\pi}{2\alpha}$$

This is done by using a mathematical technique called L'Hôpital's rule in which the derivative of both the numerator and the denominator is taken before plugging in those values for n .

Note 3.10 A Summary

Figure 3.30 summarizes our sequence of thoughts so far. A rectangular pulse has large spectral sidelobes due to which we constrain the spectrum into a brickwall shape but that raises long tails in time domain. Consequently, introducing an excess bandwidth α is a compromise between the two extremes.

Resulting Pulse Shape

Until now in this section, everything we have discussed so far was about pulse auto-correlation that satisfies Nyquist no-ISI criteria and is also called a Nyquist filter. However, *how to derive the actual pulse shape is still not known*. This is what we intend to find out next.

We start with a simple question: where should a Raised Cosine filter be placed in the system? There are only three possible choices.

Receiver The most straightforward way to incorporate it is to have it at the Rx. However, then a significant disadvantage is that there would be no mechanism to control the spectral sidelobes at the Tx. The spectral shaping required to minimize the out-of-band energy cannot be avoided.

Transmitter If the spectrum is fully shaped at the Tx side, then any additional filtering to eliminate noise and interference at the Rx will not be “matched” to the incoming signal causing ISI. In the imaginary case of no filtering at all at the Rx, the

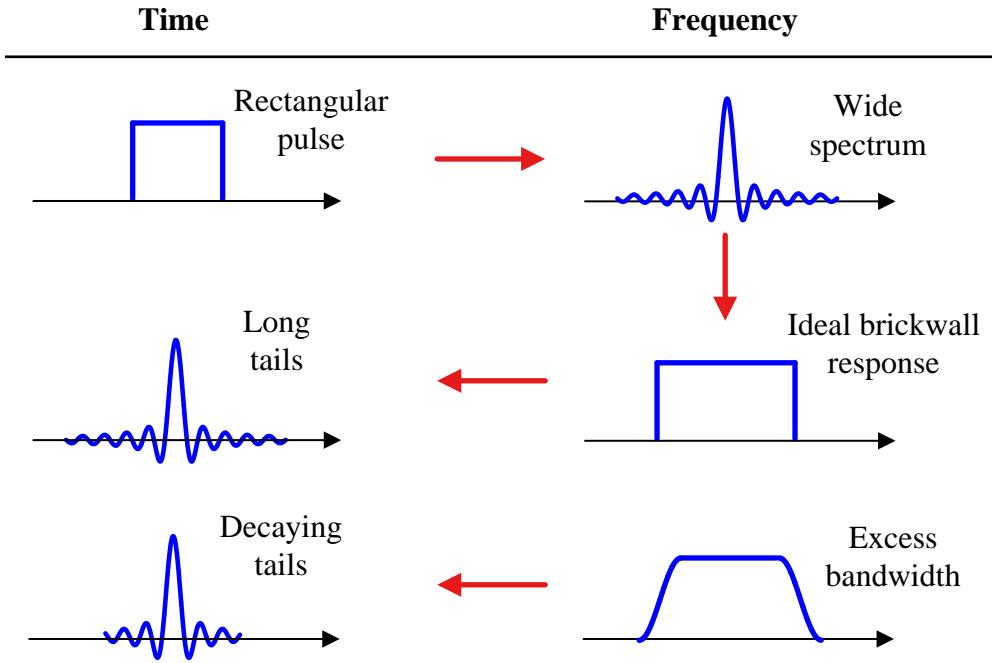


Figure 3.30: A sequence of thoughts for pulse shape design procedure

noise and energy from adjacent channels will enter in the Rx in addition to unavoidable in-band noise. This significantly reduces the SNR as well as causing other issues such as increasing the interference sensitivity and dynamic range (adjacent channel energy can be quite higher than the desired band). So the Rx filter must be as compact as possible around the Tx spectrum so that maximal amounts of noise and adjacent channel interference can be eliminated.

Both The solution then is to split the Raised Cosine spectrum into two parts, one at the Tx to control the spectrum and the other at the Rx to reject noise and adjacent channels but still have zero-ISI cumulative response. Remember that the pulse auto-correlation can be given by convolution formula as

$$r_p(nT_S) = p(nT_S) * p^*(-nT_S)$$

The pulse shape $p(nT_S)$ can be placed at the Tx and its matched filter $p^*(-nT_S)$ at the Rx (the same as $p(nT_S)$ as the pulse shape is real). Above, the response of the two filters in cascade is the convolution of their impulse responses, which implies multiplication of their frequency responses.

$$R_p(F) = P(F) \cdot P^*(F) = |P(F)|^2$$

From above equation, the frequency response of the actual pulse shape can be derived as

$$P(F) = \sqrt{R_p(F)} \quad (3.31)$$

A significant advantage of this arrangement is that the matched filter simultaneously maximizes the Rx SNR.

In the case of Raised Cosine pulse auto-correlation, the underlying pulse shape is called **Square-Root Raised Cosine (SRRC)** pulse or filter, where the square-root is in frequency domain. Referring to Eq (3.29), taking the square-root does not affect 1 and 0 values. Using the identity $0.5(1 + \cos 2\theta) = \cos^2 \theta$ and taking the square-root on both sides adjusts the middle term as

$$P(F) = \begin{cases} \sqrt{T_M} & 0 \leq |F| \leq \frac{1-\alpha}{2T_M} \\ \sqrt{T_M} \cos \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(|F| - \frac{1-\alpha}{2T_M} \right) \right\} & \frac{1-\alpha}{2T_M} \leq |F| \leq \frac{1+\alpha}{2T_M} \\ 0 & |F| \geq \frac{1+\alpha}{2T_M} \end{cases} \quad (3.32)$$

For various values of excess bandwidth α , this is shown in Figure 3.31. For $\alpha = 0$, there is no difference between a Raised Cosine and a Square-Root Raised Cosine filter due to a rectangular spectrum. Also notice from Eq (3.32) that *the transition band of a Square-Root Raised Cosine pulse is a quarter cycle of a cosine as compared to a half-cycle of a Raised Cosine filter*, which can be seen as follows. From the equation, the inverse period of the cosine is $T_M/4\alpha$ and hence the period in frequency domain is $4\alpha/T_M$. However, it exists in the region

$$\frac{1+\alpha}{2T_M} - \frac{1-\alpha}{2T_M} = \frac{\alpha}{T_M}$$

Compared to $4\alpha/T_M$, the transition band is a quarter cycle of a cosine. It has implications which we shortly see.

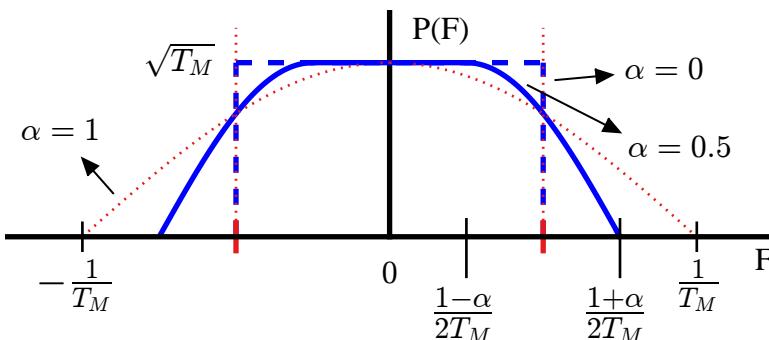


Figure 3.31: Square-Root Raised Cosine (SRRC) spectrum with different excess bandwidths α .

Without mathematical derivation, the time domain waveform is given from inverse transform of Eq (3.32) as

$$p[n] = \frac{1}{\sqrt{L}} \cdot \frac{\sin [\pi(1-\alpha)n/L] + 4\alpha n/L \cdot \cos [\pi(1+\alpha)n/L]}{\pi n/L [1 - (4\alpha n/L)^2]} \quad (3.33)$$

Figure 3.32 plots these time domain waveforms of Square-Root Raised Cosine for different values of α . Observe again that plugging $\alpha = 0$ produces a sinc signal, the same ideal Nyquist filter in Eq (3.27). Through the red ellipse in the figure, the

zero crossings are seen at integer multiples of T_M only for $\alpha = 0$. For all other α , the Square-Root Raised Cosine pulse does not satisfy Nyquist no-ISI criterion. This is not surprising because it has to fulfill that criterion only after matched filtering with another Square-Root Raised Cosine at the Rx to form the cumulative Raised Cosine shape.

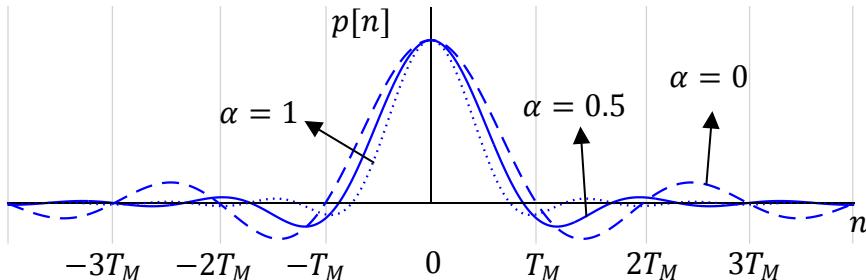


Figure 3.32: Square-root Raised Cosine (SRRC) waveform in time domain with different excess bandwidths α for $L = 64$ samples/symbol. Observe that zero crossings do not necessarily coincide with integer multiples of T_M

In a software routine, the above equation can be used to find the coefficients of the Square-Root Raised Cosine pulse shape used for Tx filtering. Just like in Raised Cosine case, the denominator in this equation becomes zero for $n = 0$ and $n = \pm L/(4\alpha)$. Pulse coefficients at these values can be given as

$$p[0] = \frac{1}{\sqrt{L}} \cdot \left(1 - \alpha + 4\frac{\alpha}{\pi} \right) \quad (3.34)$$

$$p[\pm \frac{L}{4\alpha}] = \frac{\alpha}{\sqrt{2L}} \left\{ \left(1 + \frac{2}{\pi} \right) \sin \frac{\pi}{4\alpha} + \left(1 - \frac{2}{\pi} \right) \cos \frac{\pi}{4\alpha} \right\} \quad (3.35)$$

Again, this is found by using L'Hôpital's rule in which the derivative of both the numerator and the denominator is taken before plugging in those values for n .

Being a band-limited signal, the Square-Root Raised Cosine is infinitely long in time that cannot be implemented in real systems. Therefore, it must be truncated to G symbols to the left and G symbols to the right. This is equivalent to $-LG \leq n \leq LG$ in samples, thus resulting in total filter length $N = 2LG + 1$. This time span of G symbols or LG samples is called **group delay**. The concept of group delay was discussed in detail in Section 2.5.

Exercise 3.1

In GNU Radio, an SRRC filter can be generated using the ‘Root Raised Cosine Filter’ block which is a wrapper for an `firdes` taps generating function. This function `firdes.root_raised_cosine()` can also be employed to assign the generated taps to a variable. The required parameters are as follows.

Gain: Different filter gains are set according to different requirements such as maintaining a similar amplitude or energy. See Table 2.3 in Section 2.7.2 for a detailed discussion.

Sample Rate: Sample rate F_S of the system.

Symbol Rate: Symbol rate $1/T_M$ is defined through the number of samples/symbol L , i.e., F_S/L .

Alpha: Excess bandwidth α .

Num Taps: Total number of taps is given by choosing the group delay G first as in the above discussion. Then, the filter length becomes $2LG + 1$.

In Matlab, the corresponding function is [rcosdesign\(\)](#) that takes the input parameters α , $2G$, L and an argument 'sqrt'. Replacing 'sqrt' with 'normal' generates the taps for a Raised Cosine pulse instead.

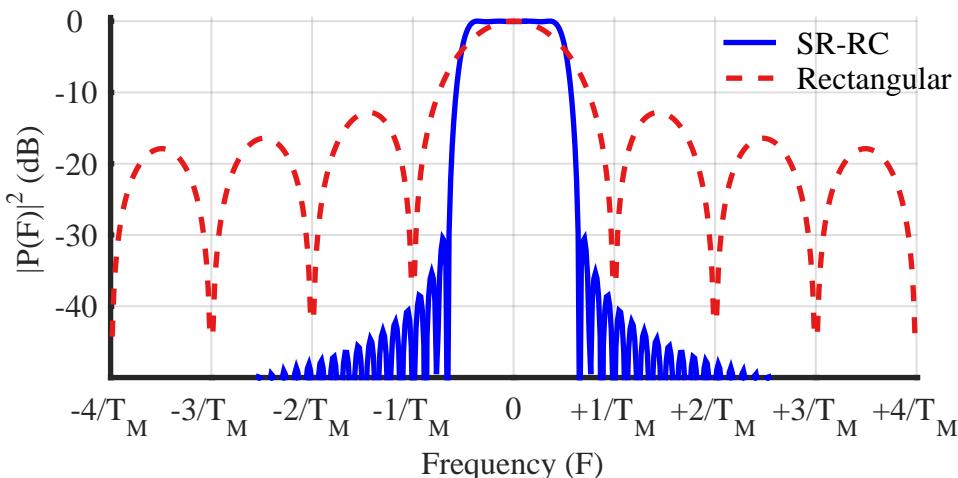


Figure 3.33: Spectrum of a Square-Root Raised Cosine pulse shape compared with that of a rectangular pulse. The pulse was generated for excess bandwidth $\alpha = 0.25$, $L = 8$ samples/symbol and length $N = 2LG + 1 = 65$

Figure 3.33 compares the spectra of a Square-Root Raised Cosine pulse with that of a rectangular pulse (rectangular in time, not frequency). The Square-Root Raised Cosine pulse is generated using $\alpha = 0.25$, $L = 8$ samples/symbol and $G = 4$ for a total filter length of $N = 2LG + 1 = 65$ samples. The duration of rectangular pulse is obviously 8 samples. A huge improvement in sidelobe suppression is fairly visible.

Example 3.1

As an example, Wideband Code-Division Multiple-Access (WCDMA) - the main technology behind 3rd-generation (3G) cellular systems - implements a Square-Root Raised Cosine pulse shape with excess bandwidth $\alpha = 0.22$, which translates the signaling rate of 3.84 MHz to a bandwidth of $0.5 \times (1 + \alpha) \times 2 \times 3.84 = 4.68$ MHz (the factor 2 arises for the RF bandwidth - we will discuss that in Section 3.7). Accounting for the guard bands to minimize interference between neighboring channels, the signal bandwidth in WCDMA systems is 5 MHz.

PAM Revisited

Let us now replace the Square-Root Raised Cosine filter above into our basic PAM system of Figure 3.17 in place of a rectangular pulse. For a 2-PAM modulation system with symbols $\pm A$, the process unfolds as follows. Keep comparing the signals thus generated with those from rectangular pulse shape in Figure 3.17.

- A source generates Tx bits to be sent to a destination. These bits are input to a Look-Up Table that maps bits to symbols according to a chosen modulation scheme. Next, the Tx symbols $a[m]$ are upsampled by $L = 2$ samples/symbol. An example created of 10 bits is illustrated in Figure 3.34.

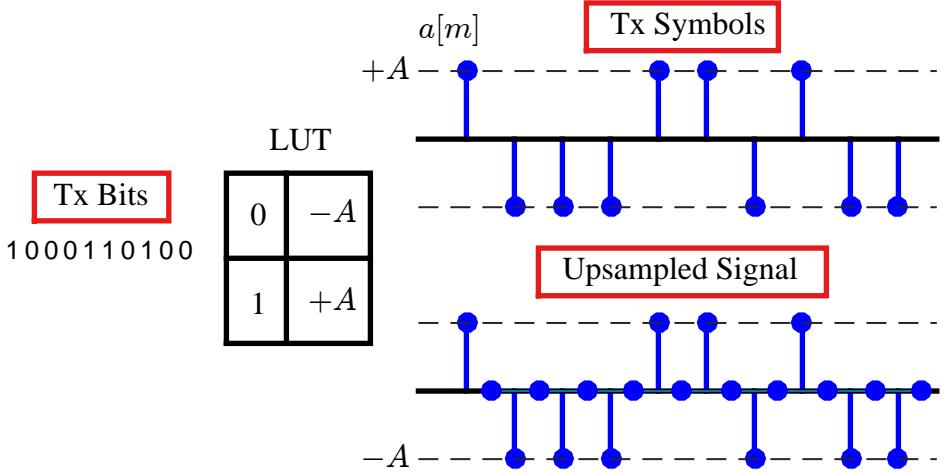


Figure 3.34: Tx bits, Look-Up Table and Tx symbols for 2-PAM modulation. The bottom plot shows the Tx symbols upsampled by $L = 2$ samples/symbol

- The pulse shaping block with $L = 2$ samples/symbol, excess bandwidth $\alpha = 0.5$ and group delay $G = 5$ symbols filters the waveform to output a smooth waveform $s(nT_S)$ which then is passed to the analog portion of the Tx. A DAC produces the continuous output $s(t)$ illustrated as dashed red line in Figure 3.35. The bit information behind the waveform is also shown.

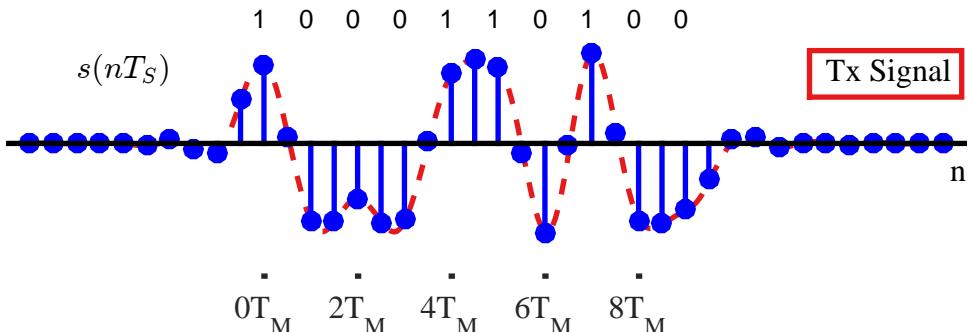


Figure 3.35: Upsampled signal shaped by a Square-Root Raised Cosine filter with $L = 2$ samples/symbol, excess bandwidth $\alpha = 0.5$ and group delay $G = 5$ symbols

- In the best case scenario, i.e., a channel that only adds AWGN to the Tx signal, the Rx signal $r(t)$ is drawn as in Figure 3.36.
- The Rx signal $r(t)$ is sampled by an ADC to produce $r(nT_S)$ and then input to a matched filter which is the same pulse shaping filter as at the Tx. The

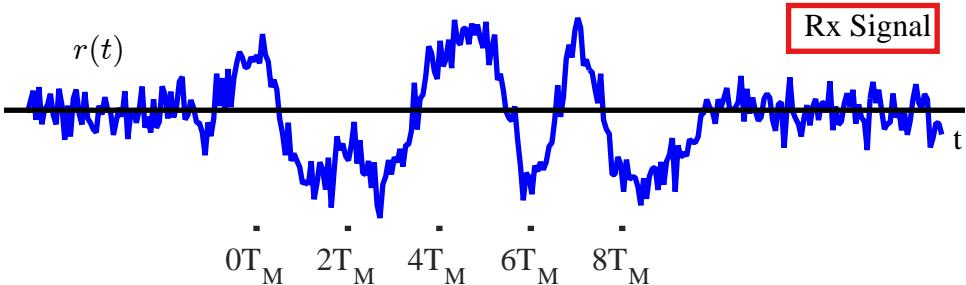


Figure 3.36: Rx signal in an AWGN channel

matched filter output $z(nT_S)$ at $L = 2$ samples/symbol is shown in Figure 3.37. As long as the combination of Tx and Rx filters (i.e., overall Raised Cosine filter) obeys Nyquist criterion, there is no ISI and one out of every L samples at $n = mL = mT_M/T_S$ can be preserved to form a symbol estimate while the remaining $L - 1$ samples are thrown away. If there was zero noise, these downsampled values directly map to the transmitted symbols without any error, from which the sequence of Rx bits can be found with the help of an LUT.

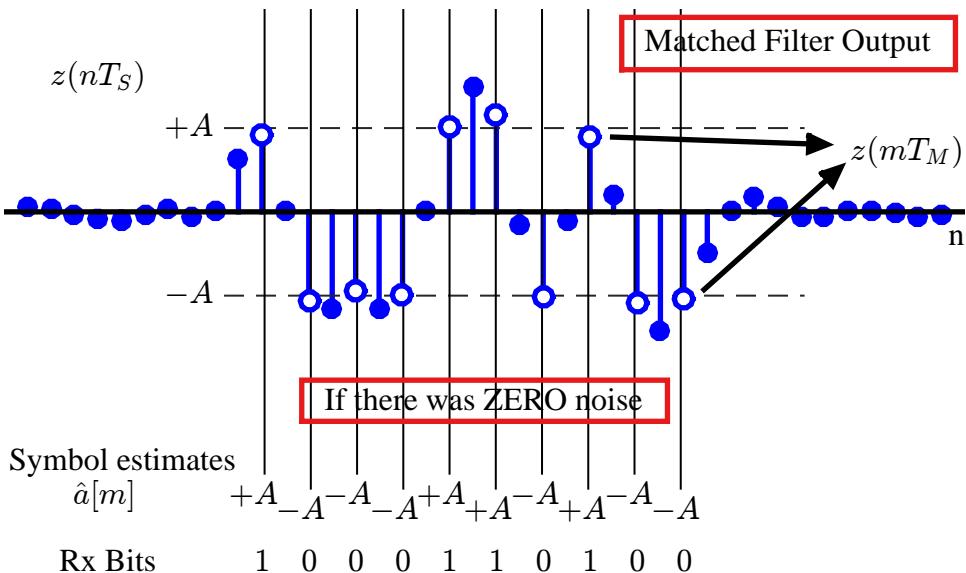


Figure 3.37: Matched filter output. White samples yield the symbol estimates

To decide which one sample to keep out of every L samples is the job of timing synchronization subsystem. We discuss symbol timing synchronization in Chapter 7.

- When the noise is present which is always the case, the minimum distance rule is employed to produce symbol estimates $\hat{a}[m]$ according to its shortest distance from a constellation point, as illustrated in Figure 3.38. Depending on numerous factors in system design, a proportion of bits can eventually end up in error, i.e.,

In presence of noise: minimum distance rule

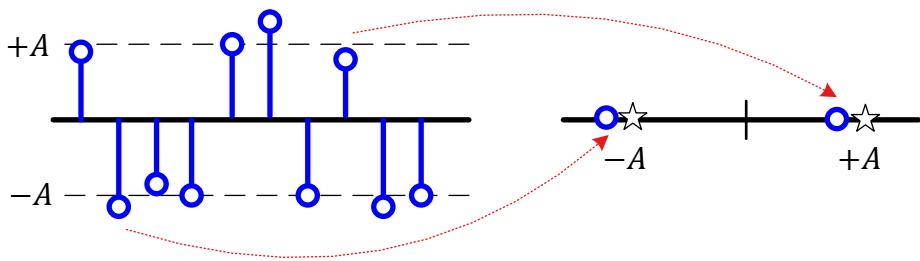


Figure 3.38: Downsampled matched filter output is mapped back to the constellation for decisions

they can be different than Tx bits. The ratio of number of Rx bits in error to the total number of bits is called *Bit Error Rate (BER)*.

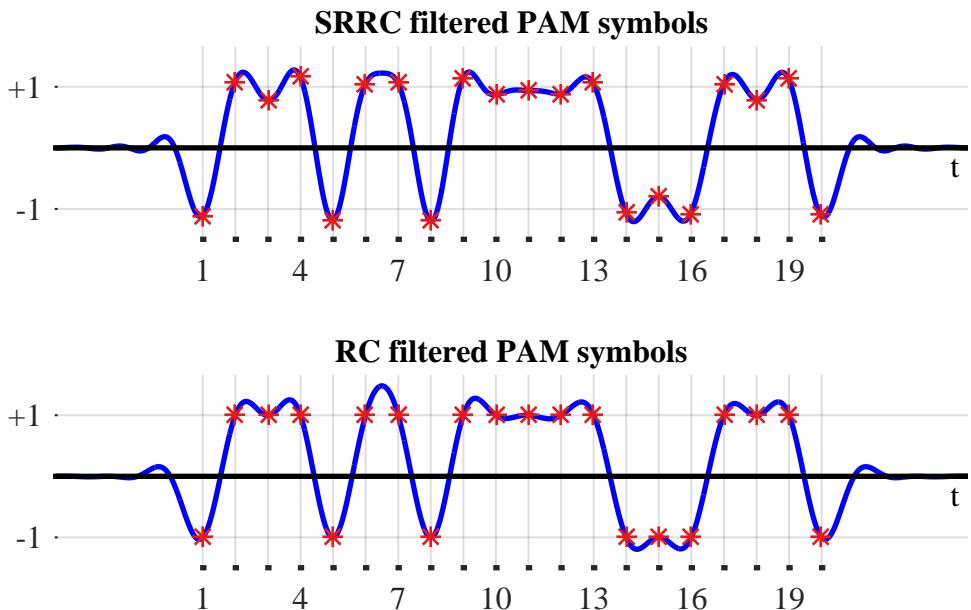


Figure 3.39: 20 binary PAM symbols filtered by a Square-Root Raised Cosine and a Raised Cosine filter with excess bandwidth $\alpha = 0.5$. Observe that values at optimum sampling locations exhibit ISI in the former but coincide with symbol values in the latter

In summary, when the source information bits are filtered by a square-root Nyquist filter, the sharp edges visible for a rectangular pulse are smoothed out by a considerable margin, as shown in Figure 3.39 for 20 2-PAM symbols and excess bandwidth

$\alpha = 0.5$. This smoothness in time domain actually limits the bandwidth. Also observe that in the absence of noise, the values at optimum sampling locations are not all the same and exhibit ISI in a square-root Nyquist case. This is because the zero crossings of a square-root Nyquist shape are not necessarily at integer multiples of symbol times, see Figure 3.32. However, after complete Nyquist filtering and no noise, all values coincide with the same optimum symbol amplitudes shown as red asterisks in RC filtered PAM symbols of Figure 3.39.

Coming back to our linear modulation framework where the pulse shape is scaled by a symbol and matched filter by the Rx, we have the following expression similar to Eq (3.17) for the matched filter output.

$$z(t) = \sum_m a[m] r_p(t - mT_M)$$

We can break this relation down to get two equivalent representations.

$$\begin{aligned} \sum_i a[i] r_p(t - iT_M) &= a[0]r_p(t) + a[1]r_p(t - T_M) + a[2]r_p(t - 2T_M) + \dots \\ &= (a[0]\delta(t) + a[1]\delta(t - T_M) + \\ &\quad a[2]\delta(t - 2T_M) + \dots) * r_p(nT_S) \end{aligned}$$

This is because convolution between any signal and a unit impulse results in the same signal. So far, we utilized the second expression involving the convolution in the above description. Now we have a look at the equivalent first relation for a deeper understanding. The unmodulated Nyquist pulse shapes (not square-root Nyquist) are drawn in Figure 3.40a. Notice the zero crossings of the 0th pulse passing through

$$T_M, 2T_M, 3T_M, \dots$$

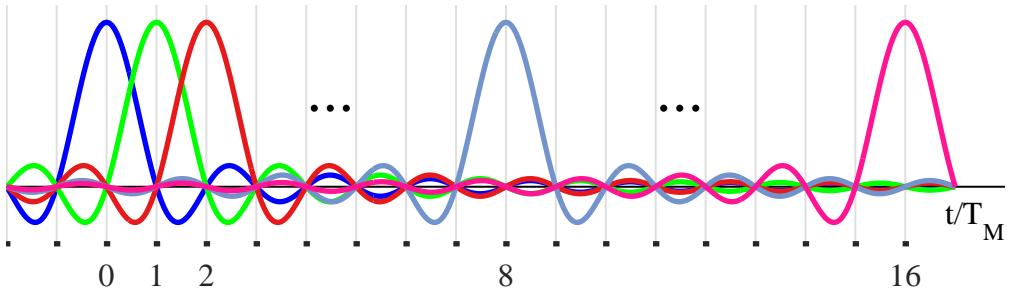
and so on, right at the peaks of the other pulse locations $t = mT_M$.

When a linear modulation scheme, such as BPSK, is utilized, the Nyquist pulse amplitudes are $a[m] = +1$ or $a[m] = -1$. The clock at the Rx then needs to sample at exactly the same instants mT_M , exhibiting zero ISI – maximum contribution comes from the desired pulse and zero contribution from all the rest. This is plotted in Figure 3.40b in the form of grid lines. Later in Section 7.1, we will see how the presence of a timing offset causes the sampling in time domain at the wrong instants thus introducing ISI in the system.

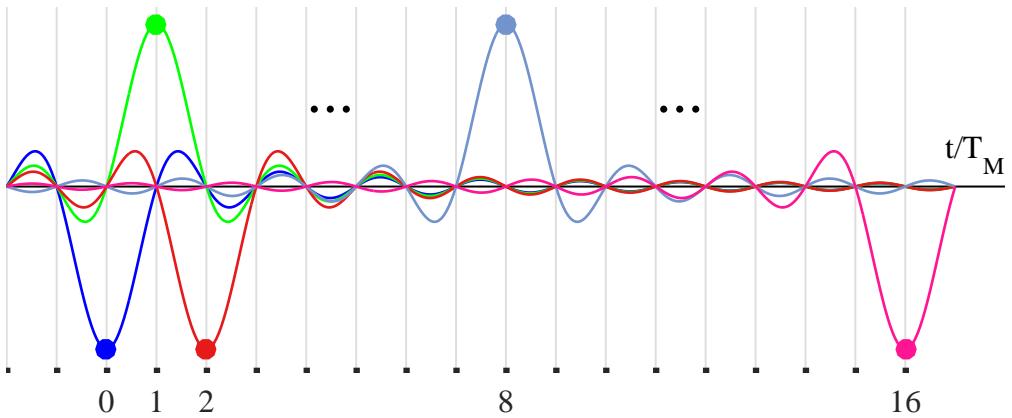
Drawbacks of Square-Root Raised Cosine Pulse

As discussed above, Square-Root Raised Cosine pulse is much better than a rectangular pulse in shaping the spectrum but it has two major drawbacks.

Insufficient Sidelobe Attenuation There is a limit to the sidelobe suppression that a Square-Root Raised Cosine pulse can achieve. The sidelobe levels are reasonably higher than realistic spectral mask requirements imposed by regulatory authorities of attenuating out-of-band energy up to 60 to 80 dB. Looking back, recall that the transition band of a Raised Cosine pulse is half cycle of a cosine. Therefore, *the transition band of a Square-Root Raised Cosine is a quarter cycle of a cosine*, see Eq (3.32) and Figure 3.31. Its abrupt termination at the stopband results in a discontinuity causing a relatively poor sidelobe response.



(a) Unmodulated Nyquist pulses in time domain. Notice the zero crossings of each pulse passing through other pulse locations $t = mT_M$



(b) Modulated Nyquist pulses in time domain. The Rx utilizes exactly the same instants $t = mT_M$ to sample the signal in time domain at ideal ISI-free instants

Figure 3.40: Unmodulated and modulated Nyquist pulses in time domain spaced T_M apart from each other

Increase in ISI Looking at Figure 3.33, an important question arises at this stage: The spectrum of Square-Root Raised Cosine should be precisely 0 beyond $(1 + \alpha)/2T_M$ Hz but why is it not? This is because as a consequence of truncation, the pulse is no more absolutely band-limited within $(1 + \alpha)/2T_M$ and assumes infinite support in frequency in the form of sidelobes. Remember that truncation means multiplication by a rectangular window. This multiplication between Square-Root Raised Cosine pulse and rectangular window in time domain is convolution between Square-Root Raised Cosine spectrum and a sinc signal in frequency domain[†].

$$p_{\text{Truncated}}[n] = p[n] \cdot w_{\text{Rect}}[n]$$

$$P_{\text{Truncated}}(F) = P(F) * \text{sinc}(F)$$

As a result of this truncation in time domain and subsequent convolution in frequency domain, the half amplitude values are moved away from the odd sym-

[†]The sidelobes and in-band ripple are inherited from that oscillating sinc signal and are a function of excess bandwidth α and the pulse extension G in symbols.

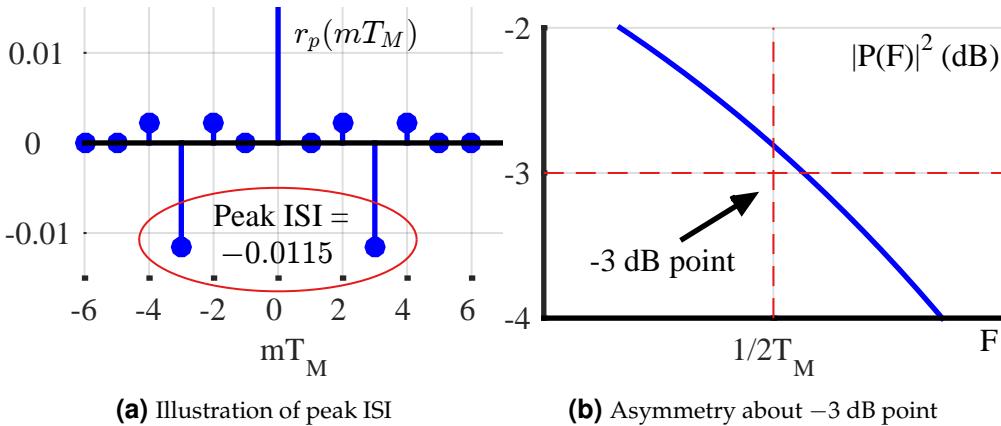


Figure 3.41: Raised Cosine generated by convolving two truncated Square-Root Raised Cosines with excess bandwidth $\alpha = 0.5$ and group delay $G = 3$ and downsampling by L samples/symbol. Due to asymmetry, spectral replicas do not add up to a constant value thus violating Nyquist no-ISI criterion

metry points of half symbol rate, or $F = \pm 1/2T_M$ as illustrated in Figure 3.41b where the filter's 1/2 (or -3 dB) point has moved away from the nominal band edge. Compare this to the case in Figure 3.24 where the pulse shape is passing through 1/2 value at exactly half the symbol rate. For this reason, the spectral replicas now do not add up to exactly a constant value. In other words, Nyquist no-ISI criterion is only approximately satisfied by the truncated pulse shape, thus giving rise to increased ISI. *The maximum magnitude of $r_p(mT_M)$ for $m \neq 0$ is called peak ISI*, shown in Figure 3.41a to be equal to -0.0115 for excess bandwidth $\alpha = 0.5$ and group delay $G = 3$.

If we attempt to control the sidelobes by applying a second window to the already rectangular windowed prototype, these ISI levels rise even more significantly. The reader is encouraged to plot the spectra for different values of α and G to appreciate their role in spectral shaping.

Square-Root Raised Cosine and Raised Cosine are good starting points for pulse shape design and capture the involved details fairly well. Moreover, they have closed-form mathematical expressions that are good for analytical purposes. Nevertheless, the quarter cycle cosine of the Square-Root Raised Cosine results in high in-band ripples and insufficient out-of-band attenuation levels. There are other pulse shape design procedures that produce a Nyquist filter with high sidelobe attenuation and preserve Nyquist no-ISI criteria as well.

A Frequency Domain Window Based Pulse

A spectral shape is a discrete-time signal and hence just a sequence of numbers. This sequence of numbers in time domain can be generated through iDFT of a carefully designed discrete-time frequency response. Recall that a Raised Cosine filter was generated in frequency domain through convolution of an ideal rectangular spectrum with a half-cosine taper, see Figure 3.25. Here, we replace the half-cosine with an

alternative taper that is an improved spectral window. The span of this spectral window is determined by the excess bandwidth α and the iDFT length.

The criteria for this spectral window design are the following.

1. Narrow width of the mainlobe to affect a smoother transition band in resulting pulse spectrum (the width of the transition band should remain unchanged).
2. Small sidelobe levels that get inherited in the pulse spectrum.

Based on a technique devised by fred harris in Ref. [10], one such candidate is a Kaiser window in frequency domain which has a minimum mainlobe spectral width for specified sidelobe levels. For a Kaiser window of a particular length, the sidelobe height is controlled by a parameter β . In Matlab, a Kaiser window of length N with parameter β can be generated through the command `kaiser(N,beta)`. Figure 3.42 compares this taper with a cosine taper with the same transition bandwidth $\alpha = 0.25$ and highlights the difference of smoothness between the two.

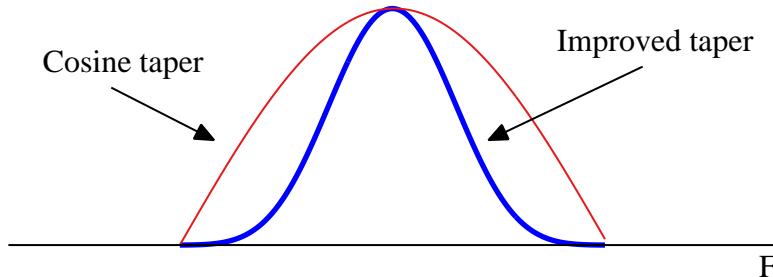
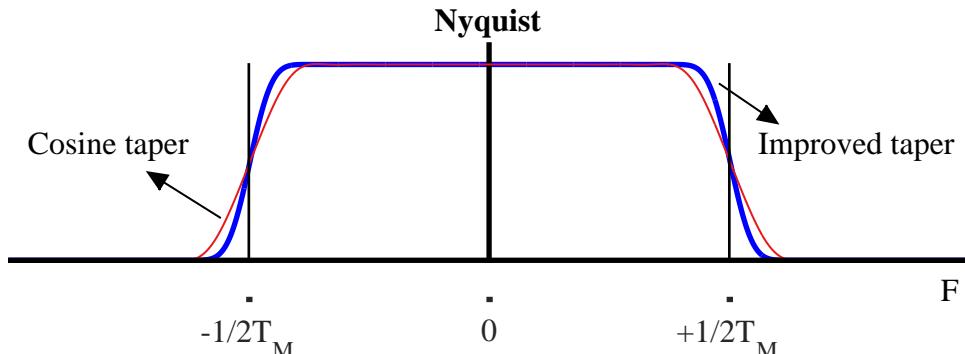


Figure 3.42: Comparison of cosine and improved spectral tapers for transition bandwidth $\alpha = 0.25$

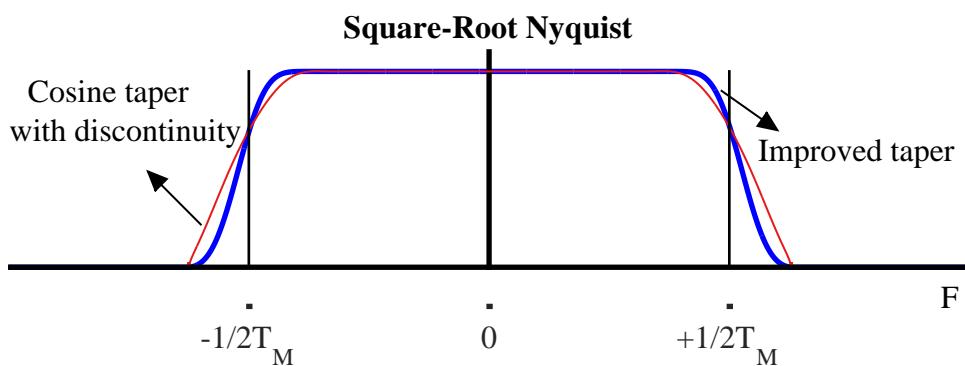
In frequency domain, the convolution of this Kaiser window taper with an ideal rectangular spectrum generates the desired pulse shape. This frequency domain result is shown in Figure 3.43 for excess bandwidth $\alpha = 0.25$ where it is compared with a Square-Root Raised Cosine pulse with the same excess bandwidth. Figure 3.43a depicts both the improved Nyquist filter and a Raised Cosine prior to square-root operation.

Notice that a narrower mainlobe width of the improved taper has produced a smoother transition band in the improved Nyquist filter as compared to a Raised Cosine. This smoothness gets inherited by square-root Nyquist filter and it does not exhibit a similar kind of discontinuity as a Square-Root Raised Cosine, both of which are shown in Figure 3.43b. This discontinuity gives rise to high levels of time response even after many symbol intervals. When such an impulse response is truncated, it generates high sidelobes as well as in-band ripple in the spectrum of a Square-Root Raised Cosine. This is evident from Figure 3.43c where an order of magnitude improvement in sidelobe suppression is visible. For example, an attenuation of 60 dB is obtained through this procedure as compared to 40 dB through Square-Root Raised Cosine.

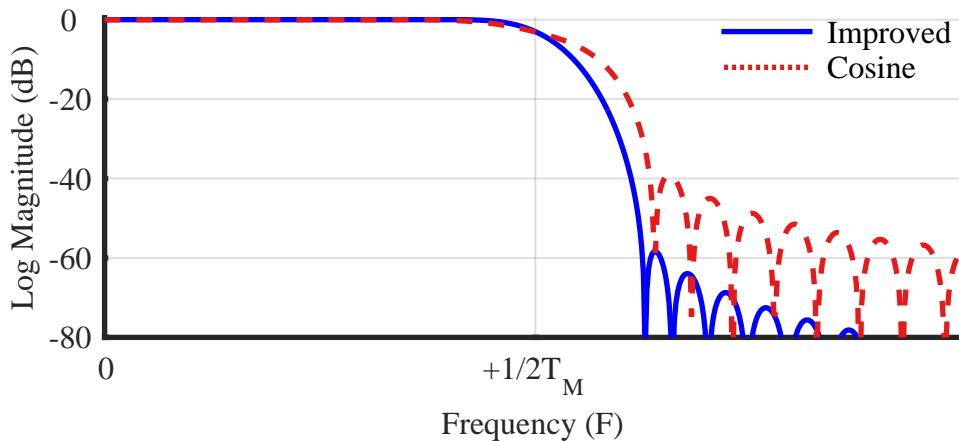
Not shown here is the fact that low levels of in-band ripple accompany the low levels of sidelobes because they are always equal in a window based design (arising from convolution of the same spectral taper sliding through both stopband and



(a) Nyquist filter with smoother transition band



(b) Square-root Nyquist filter without discontinuity at band edge



(c) Resulting better sidelobe suppression

Figure 3.43: A comparison of improved pulse with Square-Root Raised Cosine

passband). For reasons that are beyond the scope of this text, the low in-band ripple generates less ISI after matched filtering of the square-root Nyquist filter. Therefore,

the additional advantage of sidelobe suppression comes with a benefit, not at a cost.

Finally, there are other pulse shape design procedures as well that employ iterative techniques to convert an initial lowpass filter to a Nyquist filter with high sidelobe attenuation while preserving Nyquist no-ISI criteria as well. For the purpose of this text, we continue using the Square-Root Raised Cosine pulse and Raised Cosine pulse auto-correlation to keep things simple and assume that a reader who goes on to implement the system will use the better alternatives discussed above[†].

Until now, we saw only pulse amplitude modulation, a 1-dimensional modulation scheme. A wireless system achieves higher spectral efficiency when it implements a modulation scheme with 2 dimensions, namely I and Q . One example of such a scheme is Quadrature Amplitude Modulation Scheme (QAM) which we cover next.

3.7 Quadrature Amplitude Modulation (QAM)

We discussed earlier that Pulse Amplitude Modulation (PAM) transmits information through amplitude scaling of the pulse $p(nT_S)$ according to the symbol value. To understand QAM, two routes need to be traversed.

We start the first route with differentiating between baseband and passband signals. A *baseband signal* has a spectral magnitude that is non-zero only for frequencies around origin ($F = 0$) and negligible elsewhere. An example spectral plot for a PAM waveform (e.g., in Figure 3.35) is shown in Figure 3.44 for 200 2-PAM symbols shaped by a Square-Root Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ and samples/symbol $L = 8$. The PAM signal has its spectral contents around zero and hence it is a baseband signal.

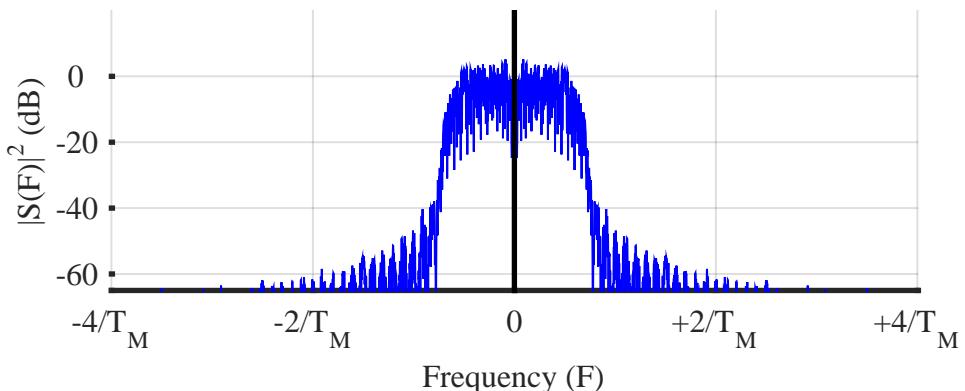


Figure 3.44: Spectrum of a 2-PAM modulated signal consisting of 200 symbols shaped by a Square-Root Raised Cosine pulse of excess bandwidth $\alpha = 0.5$ and samples/symbol $L = 8$

For wireless transmission, this information must be transmitted through space as an electromagnetic wave. Also observe that once this spectrum is occupied, no other entity in the surrounding region can share the wireless channel for the purpose of communications. For these reasons (and a few others as well), a wireless system is

[†]For non-linear modulations, a Gaussian pulse shape is a popular choice, e.g., in 2G (GSM) systems. A Gaussian pulse shape is not Nyquist and hence introduces ISI in neighboring symbols. This is a cost paid in exchange for excellent spectral properties of the pulse.

allocated a certain bandwidth around a higher carrier frequency and the generated baseband signal is shifted to that specific portion of the spectrum. Such signals are called *passband signals*. That is why wireless signals in everyday use such FM radio, WiFi, cellular like 4G, 5G, and Bluetooth are all passband signals which execute their transmissions within their respective frequency bands sharing the same physical medium.

The easiest method to shift the spectrum to a designated carrier frequency is by multiplying, or mixing, it with a sinusoidal waveform due to the following reason.

- In time domain, a PAM waveform can be multiplied with a carrier sinusoid at frequency F_C .
- Time domain multiplication between two signals is frequency domain convolution between their spectra.
- The spectrum of a pure sinusoid at any frequency is composed of two impulses at that frequency (one positive and one negative, both with half the amplitude).
- Hence in frequency domain, the convolution of a signal with a unit impulse results in the same signal “parked” at its allocated slot.

This is drawn in Figure 3.45.

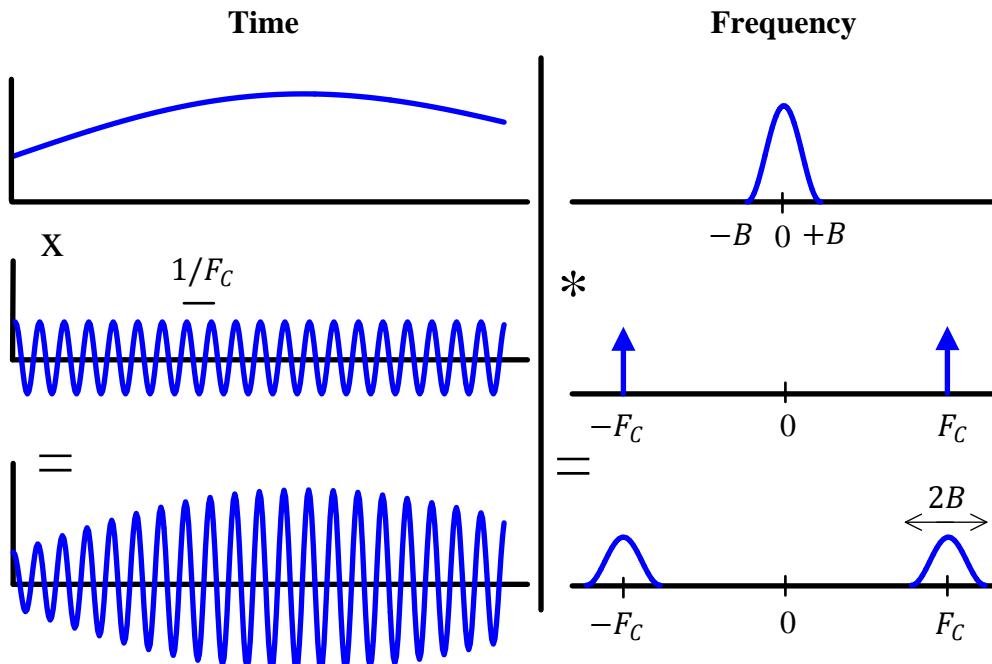


Figure 3.45: Multiplication in time domain of a signal with a sinusoid of frequency F_C shifts the spectrum at F_C due to its convolution with two impulses in frequency domain

After this *spectral upconversion*, both positive and negative portions of the baseband spectrum appear at $+F_C$ and $-F_C$. The bandwidth – positive portion of the

spectrum – hence becomes double. However, a relief comes from the fact that both $\cos(\cdot)$ and $\sin(\cdot)$ can be used to carry independent waveforms due to their orthogonality (having phases 90° apart) to each other over a complete period, i.e.,

$$\sum_{n=0}^{N-1} \cos 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n = 0$$

Obviously, this is true over a complete period. However, this is approximately true for very high carrier frequencies (typical case in wireless communications) as well because $2 \cos A \sin A = \sin 2A$ that doubles the frequency. Due to positive and negative halves of a sine, the area under the curve (sum of samples) is then zero for all integer number of cycles. This area under any fraction of a cycle outside the last integer cycle then remains negligible at a high carrier frequency.

The second route starts with amplitude modulation, where all the amplitudes still extend on the same axis of the PAM constellation diagram. On the other hand, for a phase modulated system, one axis is not enough because an actual angle is formed by *two lines*, not one. This is shown in Figure 3.46. In other words, an amplitude modulation scheme has a 1-dimensional constellation diagram while a phase modulation scheme requires a 2-dimensional constellation diagram.

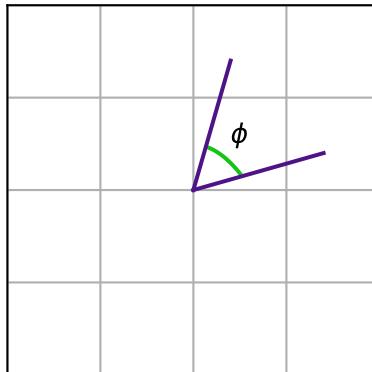


Figure 3.46: An angle is formed by two lines, not one

If amplitude modulation sends the information on discrete amplitudes in one dimension and phase modulation chooses a set of discrete phases in two dimensions but on the same circle, both of these ideas can be combined together in the form of **Quadrature Amplitude Modulation (QAM)**. For a practical implementation of this idea, we start with introducing *both* amplitude and phase modulation in a carrier waveform.

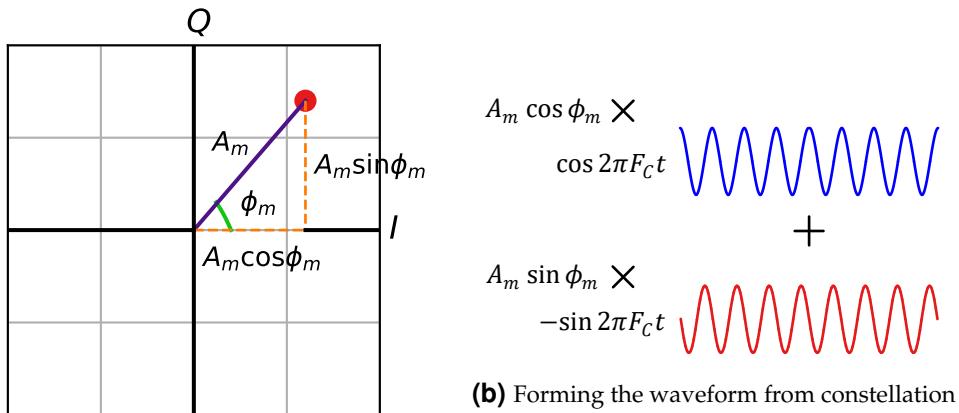
$$s(t) = A_m \cdot \cos(2\pi F t + \phi_m) \quad (3.36)$$

Using the identity $\cos(A + B) = \cos A \cos B - \sin A \sin B$, we open this up as

$$\begin{aligned} s(t) &= A_m \cdot \cos(2\pi F t) \cdot \cos \phi_m - A_m \cdot \sin 2\pi F t \cdot \sin \phi_m \\ &= \underbrace{(A_m \cos \phi_m)}_{I \text{ part}} \cdot \cos(2\pi F t) - \underbrace{(A_m \sin \phi_m)}_{Q \text{ part}} \cdot \sin 2\pi F t \end{aligned} \quad (3.37)$$

Here, the amplitude $A_m \cos \phi_m$ is the projection of the modulation point on x axis, while the amplitude $A_m \sin \phi_m$ is the projection of the modulation point on y axis. This

is plotted in Figure 3.47a. These amplitudes are then modulated on two *quadrature* carriers, namely $\cos(2\pi F_t)$ and $-\sin(2\pi F_t)$ according to Eq (3.37) and summed together to be sent over the air as plotted in Figure 3.47b. Why are there two waveforms as opposed to one (before the summation)? Because constructing or measuring an angle requires two lines, and hence constructing or measuring a phase requires two waveforms, one for each line.



(a) I/Q plane and modulation parameters A_m and ϕ_m

Figure 3.47: QAM waveform generation

To explain *I* and *Q* notations further, recall that each phase, like each angle, needs a reference. For an angle, this reference is provided by the horizontal line connecting point $(0,0)$ to point $(1,0)$. For a phase, a cosine wave with phase 0° , i.e., $\cos(2\pi F_t)$, is taken as the reference (an arbitrary choice) that is the first waveform in Eq (3.37). Acting as an amplitude of this waveform $\cos(2\pi F_t)$, the *symbol level* $A_m \cos \phi_m$ on the horizontal axis is known as the *In-Phase* component (in phase with a cosine), from which the notation *I* is extracted.

To understand the *Q* part, we need another waveform to represent the amplitude on the vertical axis. If the horizontal axis links angle 0° on a 2-D plane to a cosine wave $\cos(2\pi F_t)$ with the starting phase 0° , then the vertical axis at an angle 90° on a 2-D plane should be linked to which waveform? Clearly, this should be another cosine wave with the starting phase of 90° . A cosine wave with a starting phase of 90° is found by

$$\cos(2\pi F_t + 90^\circ) = \cos 2\pi F_t \cdot \underbrace{\cos 90^\circ}_0 - \sin 2\pi F_t \cdot \underbrace{\sin 90^\circ}_1 = -\sin 2\pi F_t$$

This second cosine wave comes out to be a negative sine wave!

That is the birth of *Quadrature Amplitude Modulation (QAM)*, in which two independent PAM waveforms are communicated through mixing one with a cosine and the other with a sine. Just like constellation, the term quadrature also comes from astronomy to describe the position of two objects 90° apart.

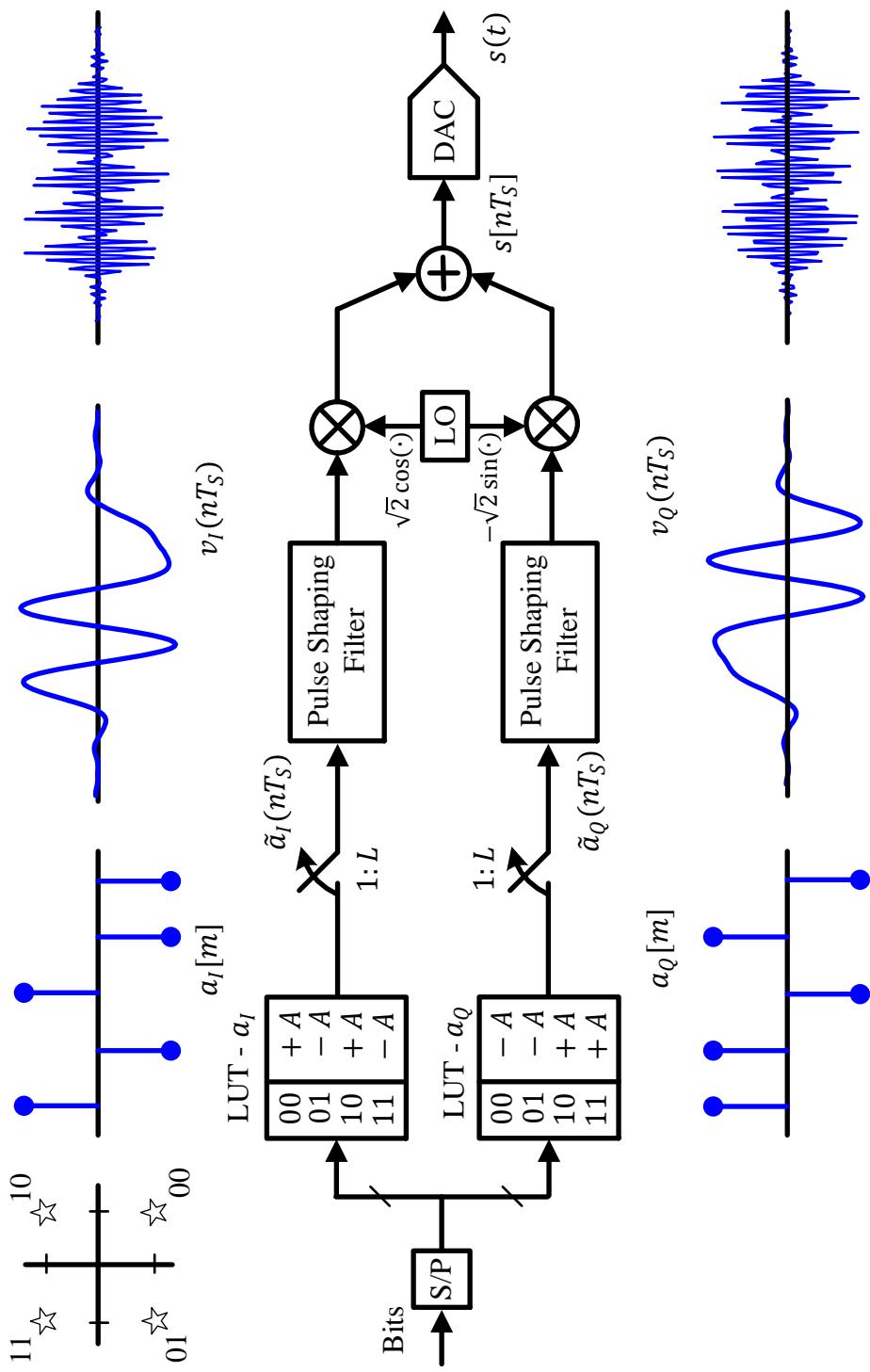


Figure 3.48: A general QAM modulator

QAM Modulator

To build a conceptual QAM modulator, we follow similar steps as in a PAM modulator in Section 3.5. The block diagram for a 4-QAM modulator is drawn in Figure 3.48.

- Every T_b seconds, a new bit arrives at the input forming a serial bit stream.
- A serial-to-parallel (S/P) converter collects $\log_2 M$ such bits every $T_M = \log_2 M \times T_b$ seconds that are used as an address to access *two Look-Up Tables (LUT)*. One LUT stores \sqrt{M} symbol values a_I while the other stores \sqrt{M} symbol values a_Q specified by the constellation. The choice of assigning a particular symbol set (e.g., $\{+A, -A\}$) to a particular bit set (e.g. 00) is completely arbitrary, as long as the Rx agrees on the same assignment.

For example, the two bits 00 form the address 0 in the Look-Up Tables which selects $a_I = +A$ and $a_Q = -A$. This comes from the assignment on the left hand side $\{+A, -A\}$ shown in Figure 3.49.

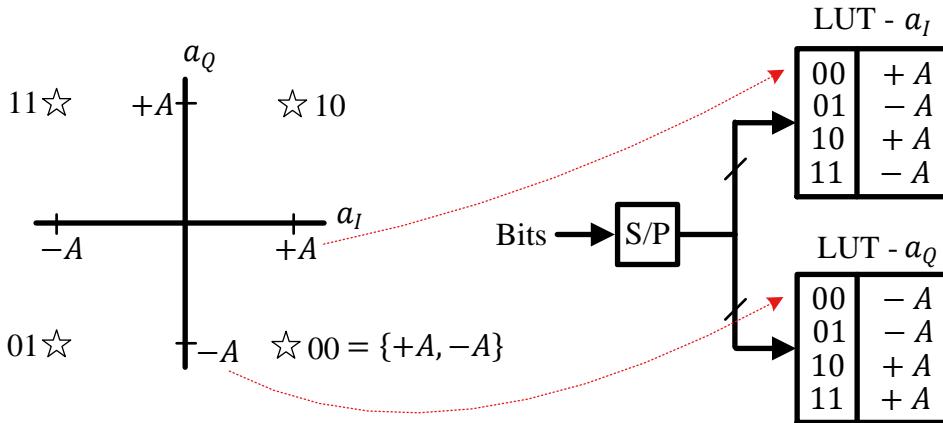


Figure 3.49: An example of selecting a_I and a_Q from a pair of Look-Up Tables. The choice of assigning a bit sequence to a particular symbol set is arbitrary

- To produce a QAM waveform, the symbol sequences $a_I[m]$ and $a_Q[m]$ are converted to discrete-time impulse trains in separate arms (one I and the other Q) through upsampling by L , where L is the number of samples per symbol (the ratio of symbol time to sample time T_M/T_S , or equivalently sample rate to symbol rate F_S/R_M). Let us denote these sequences by $\tilde{a}_I(nT_S)$ and $\tilde{a}_Q(nT_S)$, respectively.
- As explained in Section 2.7, upsampling inserts $L - 1$ zeros between each symbol after which the interpolated intermediate samples can be raised from dead with the help of a pulse shaping filter in each arm that – in addition to shaping the spectrum – suppresses all the spectral replicas arising from the upsampling process except the desired one. These are the two I and Q PAM waveforms.

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} v_I(nT_S) &= \tilde{a}_I(nT_S) * p(nT_S) \\ v_Q(nT_S) &= \tilde{a}_Q(nT_S) * p(nT_S) \end{aligned}$$

- Next, I PAM waveform $v_I(nT_S)$ is upconverted by mixing with the inphase carrier $\sqrt{2} \cos 2\pi F_C n T_S$ while the Q PAM waveform $v_Q(nT_S)$ with quadrature carrier $-\sqrt{2} \sin 2\pi F_C n T_S$, which are then summed to form the QAM signal $s(nT_S)$. The reason for choosing the scaling factor $\sqrt{2}$ as well as a negative sign with $\sin(\cdot)$ will shortly become clear. Note that this carrier is generated through a Local Oscillator (LO) at the Tx.
- This discrete-time signal $s(nT_S)$ is converted to a continuous-time signal $s(t)$ by a DAC.

For mathematical expression for QAM, first consider the above two independent PAM waveforms with symbol streams $a_I[m]$ and $a_Q[m]$, respectively.

$$\begin{array}{ll} I & \rightarrow \\ & v_I(nT_S) = \sum_m a_I[m] p(nT_S - mT_M) \\ Q & \uparrow \\ & v_Q(nT_S) = \sum_m a_Q[m] p(nT_S - mT_M) \end{array} \quad (3.38)$$

Next, we multiply the resulting complex signal with a complex sinusoid at a frequency k_C/N , where k_C/N corresponds to the carrier frequency F_C . Here, we assume a conceptual upconversion in discrete domain but in practice there are efficient techniques to accomplish this task that depend on the Tx architecture. Now from the multiplication rule of complex signals $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$,

$$\begin{array}{ll} I & \rightarrow \\ & v_I(nT_S) \sqrt{2} \cos 2\pi \frac{k_C}{N} n - v_Q(nT_S) \sqrt{2} \sin 2\pi \frac{k_C}{N} n \\ Q & \uparrow \\ & v_Q(nT_S) \sqrt{2} \cos 2\pi \frac{k_C}{N} n + v_I(nT_S) \sqrt{2} \sin 2\pi \frac{k_C}{N} n \end{array}$$

Now the signals sent over the air or any medium are always real, so we can choose any one of the above arms, I or Q . Note that both arms contain all the information regarding $v_I(nT_S)$ and $v_Q(nT_S)$. As a convention, we choose the I arm for this purpose. Consequently, a general QAM waveform can be written as

$$\begin{aligned} s(nT_S) &= v_I(nT_S) \sqrt{2} \cos 2\pi \frac{k_C}{N} n - v_Q(nT_S) \sqrt{2} \sin 2\pi \frac{k_C}{N} n \\ &= v_I(nT_S) \sqrt{2} \cos 2\pi \frac{F_C}{F_S} n - v_Q(nT_S) \sqrt{2} \sin 2\pi \frac{F_C}{F_S} n \end{aligned} \quad (3.39)$$

where we have used the fundamental relation between continuous and discrete frequencies $k/N = F/F_S$. Observe that $s(nT_S)$ is a real signal with no Q component. And in a block diagram, it is obtained by multiplying v_I arm with $\sqrt{2} \cos 2\pi(k_C/N)n$ and v_Q arm with $-\sqrt{2} \sin 2\pi(k_C/N)n$. Plugging in the definitions of $v_I(nT_S)$ and $v_Q(nT_S)$,

$$\begin{aligned} s(nT_S) &= \sum_m a_I[m] p(nT_S - mT_M) \sqrt{2} \cos 2\pi F_C n T_S - \\ &\quad \sum_m a_Q[m] p(nT_S - mT_M) \sqrt{2} \sin 2\pi F_C n T_S \end{aligned}$$

After Digital to Analog Conversion (DAC), the continuous-time signal $s(t)$ can be ex-

pressed as

$$\begin{aligned}s(t) &= v_I(t)\sqrt{2}\cos 2\pi F_C t - v_Q(t)\sqrt{2}\sin 2\pi F_C t \\ &= \sum_m a_I[m]p(t - mT_M)\sqrt{2}\cos 2\pi F_C t - \\ &\quad \sum_m a_Q[m]p(t - mT_M)\sqrt{2}\sin 2\pi F_C t\end{aligned}\tag{3.40}$$

In Eq (3.40), symbols $a_I[m]$ determine the I signal of $v(t)$ while $a_Q[m]$ control the Q part of $v(t)$. Such representation of the QAM waveform as a sum of amplitude scaled and pulse shaped sinusoids is known as the *rectangular form*. Sometimes it is difficult to imagine how two different data streams can be combined into a single signal and sent over the air. For the ease of illustration, we have drawn zoomed in versions of I and Q arms in a QAM modulator in Figure 3.50.

We can convert it into a polar form by using the relation

$$A \cos 2\pi F_C t - B \sin 2\pi F_C t = \sqrt{A^2 + B^2} \cos \left(2\pi F_C t + \tan^{-1} \frac{B}{A} \right)$$

While the actual derivation is longer, you can use the trigonometric identity $\cos(A + B) = \cos A \cos B - \sin A \sin B$ on the right hand side along with $\tan A = \sin A / \cos A$ and the Pythagorean theorem. Applying this to the QAM waveform above, its polar form can be written as

$$\begin{aligned}s(t) &= \sum_m \sqrt{a_I^2[m] + a_Q^2[m]} \cdot p(t - mT_M) \cdot \sqrt{2} \cos \left(2\pi F_C t + \tan^{-1} \frac{a_Q[m]}{a_I[m]} \right) \\ &= \sum_m X[m] \cdot p(t - mT_M) \cdot \sqrt{2} \cos (2\pi F_C t + \phi[m])\end{aligned}\tag{3.41}$$

where

$$\begin{aligned}X[m] &= \sqrt{a_I^2[m] + a_Q^2[m]} \\ \phi[m] &= \tan^{-1} \frac{a_Q[m]}{a_I[m]}\end{aligned}$$

In this *polar form*, we have a single sinusoid whose amplitude and phase are determined by the above combination of symbols $a_I[m]$ and $a_Q[m]$ during every symbol time. This is illustrated in Figure 3.51 for a slowly varying carrier wave.

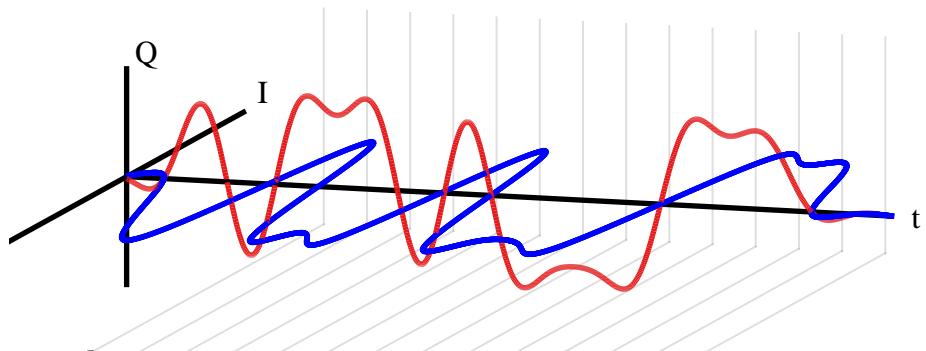
For example, the constellation points in BPSK are 0 and π . In QPSK, the constellation points are selected by $X[m]$ and $\phi[m]$ as

$$X[m] = 1, \quad \phi[m] = +\frac{\pi}{4}, +3\frac{\pi}{4}, -3\frac{\pi}{4}, -\frac{\pi}{4}$$

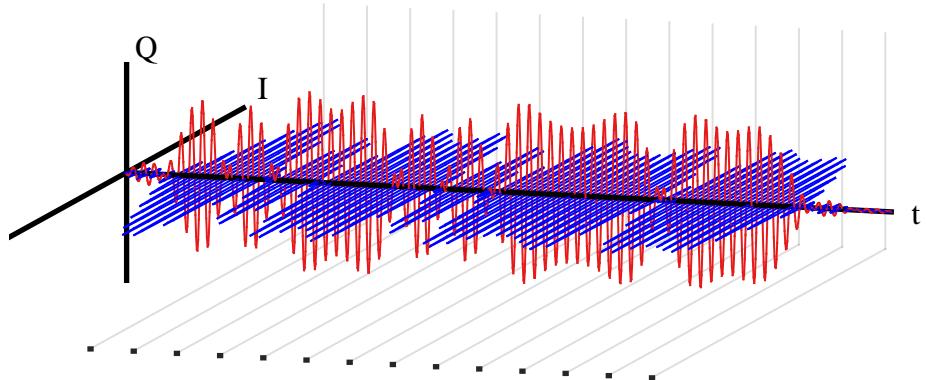
For a higher-order modulation scheme, the amplitude $X[m]$ also changes values according to the constellation points.

Constellation Diagram

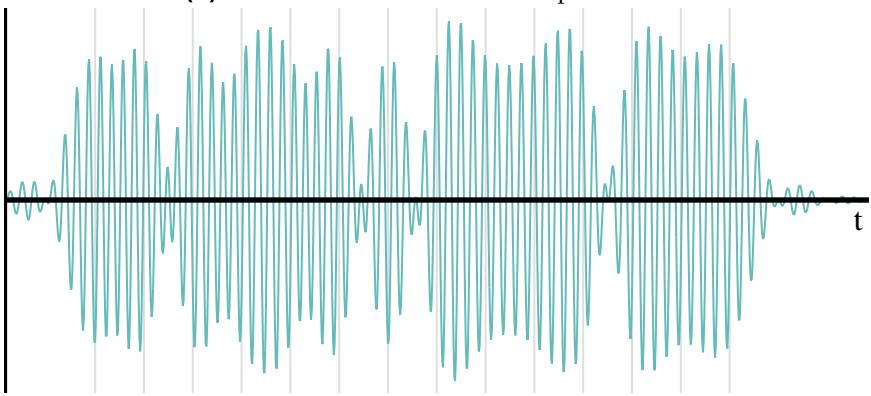
Some examples of QAM constellations are discussed now. For an even power of 2, *square QAM* is a constellation whose points are spaced on a grid in the form of a



(a) Two baseband PAM waveforms in I and Q arms



(b) Same waveforms after carrier upconversion



(c) Summation of the upconverted I and Q waveforms

Figure 3.50: QAM signal generation process

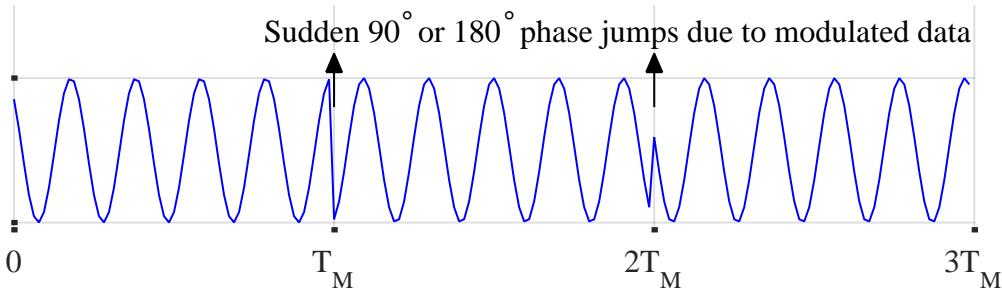


Figure 3.51: In polar form, a QPSK modulation can be seen to manifest 0° , $\pm 90^\circ$ or 180° phase jumps at each symbol time from the modulating data

square. It is formed by a product of two \sqrt{M} -PAM constellations, one on I -axis and the other on Q axis. For example, a 16-QAM constellation is formed by two $\sqrt{16} = 4$ PAM constellations as drawn in Figure 3.52, while some other square QAM constellations are also shown in Figure 3.53.

From Figure 3.53, notice how constellation points in higher-order QAM are closer to each other compared to lower-order QAM. A relatively lower noise power is then enough to cause a decision error by moving the received symbol over the decision boundary. This is the cost of increasing data rate by packing more bits in the same symbol. We will have more to say about it when we discuss Bit Error Rates (BER) for each constellation in Section 3.10.

For $M = 4$, the average symbol energy in square QAM is derived using Pythagoras theorem as

$$E_{4\text{-QAM}} = \frac{1}{4} \left\{ 4 \left(A^2 + A^2 \right) \right\} = 2A^2$$

For general M , a similar derivation as in the case of PAM studied earlier yields

$$E_{M\text{-QAM}} = \frac{2}{3} (M - 1) A^2 \quad (3.42)$$

A special case of M -QAM is M -PSK, which stands for Phase Shift Keying. As the name PSK implies, the amplitude remains constant in this configuration while the

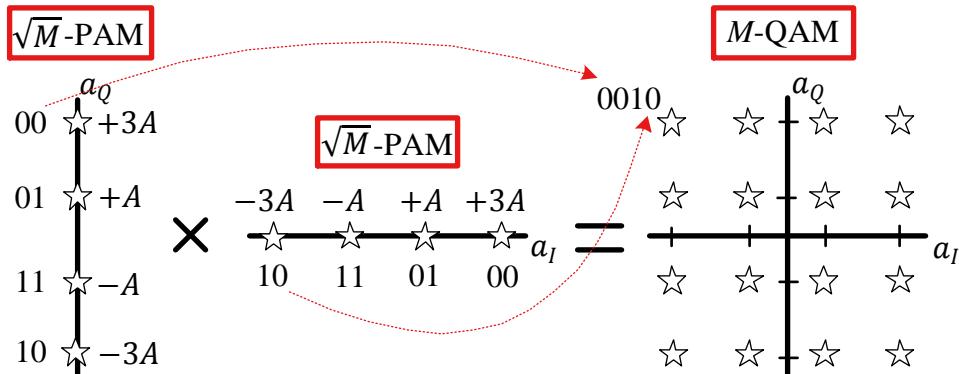


Figure 3.52: A 16-QAM constellation is formed by two $\sqrt{16} = 4$ PAM constellations

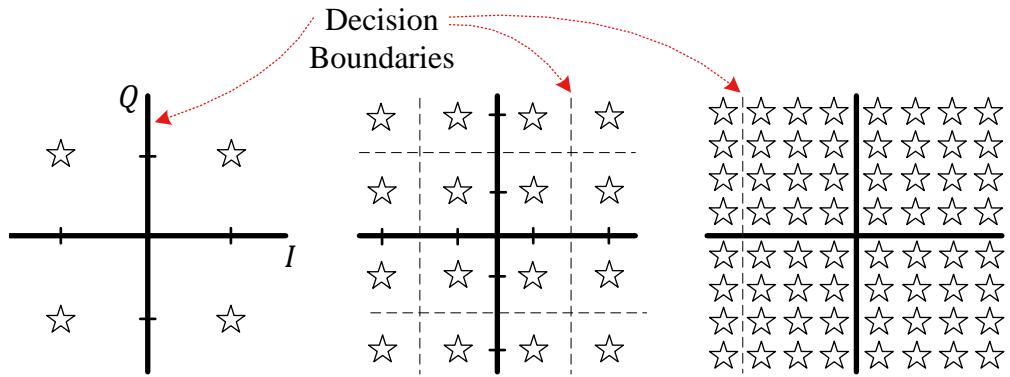


Figure 3.53: QAM constellation diagrams for $M = 4$, $M = 16$ and $M = 64$. Observe how points get closer for higher-order QAM and become prone to noise errors

information is conveyed by different phases. Examples of 2-PSK, 4-PSK and 8-PSK are drawn in Figure 3.54. Notice that

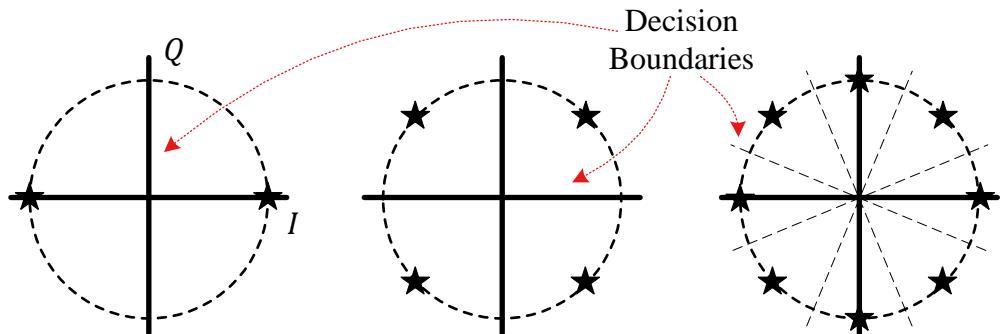


Figure 3.54: PSK constellation diagrams for $M = 2$, $M = 4$ and $M = 8$. Observe that all points lie on a circle. Also, they get closer for higher-order PSK and become prone to noise errors

- 2-PSK constellation (also known as [BPSK](#)) looks similar to 2-PAM. However, the difference is that there is no carrier upconversion in PAM systems.
- 4-PSK constellation (also known as [QPSK](#)) is exactly the same as square 4-QAM.
- For $M > 4$, square QAM packs the constellation points more efficiently than PSK and hence the modulation of choice in many wireless standards. These constellation points come relatively closer in PSK and the closer the points, the larger the probability of receiving a symbol in error due to a jump across the decision boundary. On the other hand, QAM heavily depends on overall system linearity due to information conveyed in the signal amplitude. A constant envelope of the modulated signal is much more suitable for transmission over nonlinear channels where PSK is the preferred choice. In such situations, the performance of PSK is quite insensitive to nonlinear distortion and heavy filtering, see Ref. [11].

Note 3.11 London Eye

Here is a little quiz for you: which PSK constellation is the London Eye shown in Figure 3.55? The answer is 32-PSK as the Eye has 32 pods but they are numbered from 1 to 33. The number 13 is skipped to avoid bad luck.



Figure 3.55: London Eye is a 32-PSK constellation

Since any constellation that uses both amplitude and phase modulation falls under the general category of QAM, there can be many other rearrangements of QAM constellation points. However, we focus on square QAM and PSK in this text which are widely used in wireless systems. Non-regular constellations can yield a marginal performance advantage at a cost of more complicated demodulation, some of which are utilized in wired media such as DSL and cable modems. Nevertheless, the more the points are in the constellation, the lesser is the difference.

QAM Detector

Let us consider the scenario where the received signal $r(t)$ is the same as the transmitted signal $s(t)$ but with the addition of additive white Gaussian noise (AWGN) $w(t)$. The symbols are detected through the following steps illustrated in Figure 3.56.

- Out of the infinite spectrum, the desired signal $r(t)$ is selected with the help of a Bandpass Filter (BPF).
- Through an Analog to Digital Converter (ADC), this signal is sampled at a rate of F_S samples/s to produce a sequence of T_S -spaced samples $r(nT_S)$.
- Next, *a complex signal $x(nT_S)$* is produced when $r(nT_S)$ is downconverted by mixing with the inphase and quadrature carriers which are generated through a Local Oscillator (LO) at the Rx. Just like upconversion, we assume a conceptual downconversion in discrete domain but in practice there are different techniques to accomplish this task that depend on the Rx architecture.

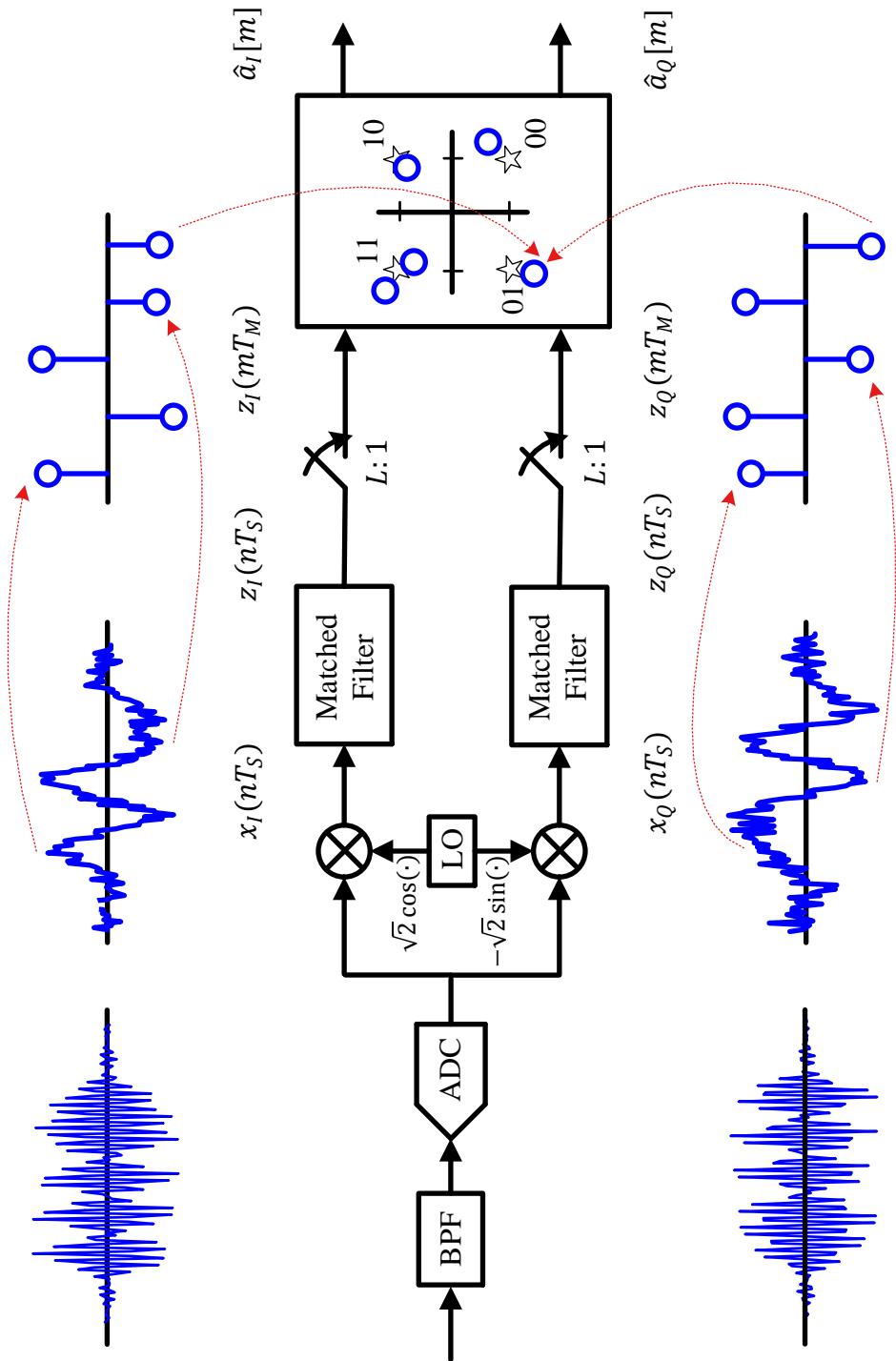


Figure 3.56: A general QAM detector

- The resulting complex waveform $x(nT_S)$ in I and Q arms is processed through two matched filters thus generating $z_I(nT_S)$ and $z_Q(nT_S)$. As discussed earlier, the outputs of the matched filter $z_I(nT_S)$ and $z_Q(nT_S)$ are the correlations of the symbol-scaled pulse shape with an unscaled and time-reversed pulse shape in their respective arms.

- These I and Q outputs are downsampled by L at optimal sampling instants

$$n = mL = m \frac{T_M}{T_S}$$

to produce T_M -spaced numbers $z_I(mT_M)$ and $z_Q(mT_M)$ back from the signal.

- The minimum distance decision rule is employed in IQ -plane to find the symbol estimates $\hat{a}_I[m]$ and $\hat{a}_Q[m]$ to decide on the final constellation point.

Let us discuss the mathematical details of this process for a noiseless Rx signal as in Eq (3.40).

$$r(t) = s(t) = v_I(t)\sqrt{2}\cos 2\pi F_C t - v_Q(t)\sqrt{2}\sin 2\pi F_C t \quad (3.43)$$

where F_C is the carrier frequency and $v_I(t)$ and $v_Q(t)$ are defined in Eq (3.38). After bandlimiting the incoming signal through a bandpass filter, it is sampled by the ADC operating at F_S samples/second to produce

$$\begin{aligned} r(nT_S) &= v_I(nT_S)\sqrt{2}\cos 2\pi F_C nT_S - v_Q(nT_S)\sqrt{2}\sin 2\pi F_C nT_S \\ &= v_I(nT_S)\sqrt{2}\cos 2\pi \frac{k_C}{N}n - v_Q(nT_S)\sqrt{2}\sin 2\pi \frac{k_C}{N}n \end{aligned}$$

where the relation $F/F_S = k/N$ is used. To produce a complex baseband signal from the Rx signal, the samples of this waveform are input to a mixer which multiplies them with discrete-time quadrature sinusoids $\sqrt{2}\cos 2\pi(k_C/N)n$ and $-\sqrt{2}\sin 2\pi(k_C/N)n$ yielding

$$\begin{aligned} I \rightarrow x_I(nT_S) &= r(nT_S) \cdot \sqrt{2}\cos 2\pi \frac{k_C}{N}n \\ &= 2v_I(nT_S)\cos^2 2\pi \frac{k_C}{N}n - 2v_Q(nT_S)\sin 2\pi \frac{k_C}{N}n \cos 2\pi \frac{k_C}{N}n \\ Q \uparrow x_Q(nT_S) &= r(nT_S) \cdot -\sqrt{2}\sin 2\pi \frac{k_C}{N}n \\ &= 2v_Q(nT_S)\sin^2 2\pi \frac{k_C}{N}n - 2v_I(nT_S)\cos 2\pi \frac{k_C}{N}n \sin 2\pi \frac{k_C}{N}n \end{aligned}$$

Using the identities $2\cos^2 A = 1 + \cos 2A$, $2\sin^2 A = 1 - \cos 2A$ and $2\sin A \cos A = \sin 2A$,

$$\begin{aligned} I \rightarrow x_I(nT_S) &= v_I(nT_S) + \underbrace{v_I(nT_S)\cos 2\pi \frac{2k_C}{N}n - v_Q(nT_S)\sin 2\pi \frac{2k_C}{N}n}_{\text{Double frequency terms}} \\ Q \uparrow x_Q(nT_S) &= v_Q(nT_S) - \underbrace{v_Q(nT_S)\cos 2\pi \frac{2k_C}{N}n - v_I(nT_S)\sin 2\pi \frac{2k_C}{N}n}_{\text{Double frequency terms}} \end{aligned}$$

Now we can observe why the two factors, a $\sqrt{2}$ with both sinusoids and a negative sign with $\sin 2\pi(k/N)n$, were inserted.

- A $\sqrt{2}$ at the modulator and later another $\sqrt{2}$ in the detector result in a gain of 2, which cancels the halving of sinusoid amplitudes in above trigonometric multiplications.
- When a complex signal is multiplied with a complex sinusoid, a negative sign appears with $v_Q(nT_S)$ in the I section of cumulative waveform. This is the real part of the complex product which is transmitted through the channel. Another negative sign with $\sin 2\pi(k/N)n$ at the Rx delivers a positive $v_Q(nT_S)$ at the output.

The matched filter output is written as

$$\begin{aligned}
 z_I(nT_S) &= x_I(nT_S) * p(-nT_S) \\
 &= (v_I(nT_S) + \text{Double freq terms}) * p(-nT_S) \\
 I \rightarrow &= \left(\sum_i a_I[i] p(nT_S - iT_M) + \frac{2k_C}{N} \text{ terms} \right) * p(-nT_S) \\
 &= \sum_i a_I[i] r_p(nT_S - iT_M)
 \end{aligned}$$

$$\begin{aligned}
 z_Q(nT_S) &= x_Q(nT_S) * p(-nT_S) \\
 &= (v_Q(nT_S) + \text{Double freq terms}) * p(-nT_S) \\
 Q \uparrow &= \left(\sum_i a_Q[i] p(nT_S - iT_M) + \frac{2k_C}{N} \text{ terms} \right) * p(-nT_S) \\
 &= \sum_i a_Q[i] r_p(nT_S - iT_M)
 \end{aligned}$$

where $r_p(nT_S)$ comes from applying the definition of the auto-correlation function in Eq (3.21). The double frequency terms in the above equation are filtered out by the matched filter $h(nT_S) = p(-nT_S)$, which also acts as a lowpass filter due to its spectrum limitation in the range $-1/2T_M \leq F \leq +1/2T_M$. To generate symbol decisions, T_M -spaced samples of the matched filter output are required at $n = mL = mT_M/T_S$. Downsampling the matched filter output generates

$$\begin{aligned}
 z_I(mT_M) &= z_I(nT_S) \Big|_{n=mL=mT_M/T_S} \\
 I \rightarrow &= \sum_i a_I[i] r_p(mT_M - iT_M) = \sum_i a_I[i] r_p\{(m-i)T_M\} \\
 &= \underbrace{a_I[m] r_p(0T_M)}_{\text{current symbol}} + \underbrace{\sum_{i \neq m} a_I[i] r_p\{(m-i)T_M\}}_{\text{ISI}}
 \end{aligned}$$

and

$$\begin{aligned}
 z_Q(mT_M) &= z_Q(nT_S) \Big|_{n=mL=mT_M/T_S} \\
 Q \uparrow &= \sum_i a_Q[i] r_p(mT_M - iT_M) = \sum_i a_Q[i] r_p\{(m-i)T_M\} \\
 &= \underbrace{a_Q[m] r_p(0T_M)}_{\text{current symbol}} + \underbrace{\sum_{i \neq m} a_Q[i] r_p\{(m-i)T_M\}}_{\text{ISI}}
 \end{aligned}$$

A square-root Nyquist pulse $p(nT_S)$ has an auto-correlation $r_p(nT_S)$ that satisfies no-ISI criterion of Eq (3.24).

$$r_p(mT_M) = \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases}$$

Thus,

$$\begin{aligned}
 I \rightarrow & z_I(mT_M) = a_I[m] \\
 Q \uparrow & z_Q(mT_M) = a_Q[m]
 \end{aligned}$$

In conclusion, the downsampled matched filter outputs map back to the Tx symbols in the absence of noise. If the world was simple, that would have been an end to it! But the world is complicated, and there are layers of issues that happen between the Tx information generation and Rx decision making. Anything we do after this chapter will be to combat a subset of signal distortions and towards recovering the original information.

Observe that the system shown in Figure 3.56 is a multirate system. In the QAM detector, for example, the ADC and the matched filters operate at the sample rate F_S . After the outputs of the matched filters are downsampled by L , the symbol decisions are made at the symbol rate $1/T_M$. Furthermore, there are some hidden assumptions in the QAM detector.

Symbol timing synchronization: The peak samples at the end of symbol durations in both I and Q arms are not known in advance at the Rx and in fact do not necessarily coincide with the generated samples as well. This is because ADC just samples the incoming continuous waveform without any information about the symbol boundaries. This is a symbol timing synchronization problem which we will study in Chapter 7.

Resampling: The ADC in general does not produce an integer number of samples per symbol, i.e., T_M/T_S is not an integer. As we will see later, a resampling system is required in the Rx chain that changes the sample rate from the ADC rate to a rate that is an integer multiple of the symbol rate.

Carrier frequency synchronization: The carrier frequency of the oscillator at the Tx and that at the Rx are not exactly the same. Instead, if the oscillator frequency at the Tx is denoted as F_C , then at the Rx, we have $F_C + F_\Delta$, where F_Δ is the difference between the two and can be either positive or negative. Any movement by the Tx, the Rx or within the environment between them also causes a shift in frequency (known as Doppler shift) that needs to be compensated. We will discuss carrier frequency synchronization in Chapter 6.

Carrier phase synchronization: Furthermore, the carrier phase at the Rx oscillators is not known beforehand and needs to be estimated which will be explained in Chapter 5.

Equalization: Only an AWGN channel has been assumed so far. Multipath reflections from a wireless channel introduce ISI and distortion in the signal that need to be recovered through an equalizer. This is what we cover in Chapter 8.

3.8 Stethoscopes for a Communication System

We are now in a position to devise some tools that help us diagnose problems with the communication system under study. I like to call them the *stethoscopes for a communication system* due to the crucial functionality they provide regarding the health of the communication system being analyzed. We discuss three such tools, namely an eye diagram, a transition diagram and a scatter plot below.

Eye Diagram

An eye diagram is an excellent summary of the signal behaviour in time domain, very similar to a spectrum in frequency domain. Imagine the samples of the matched filter

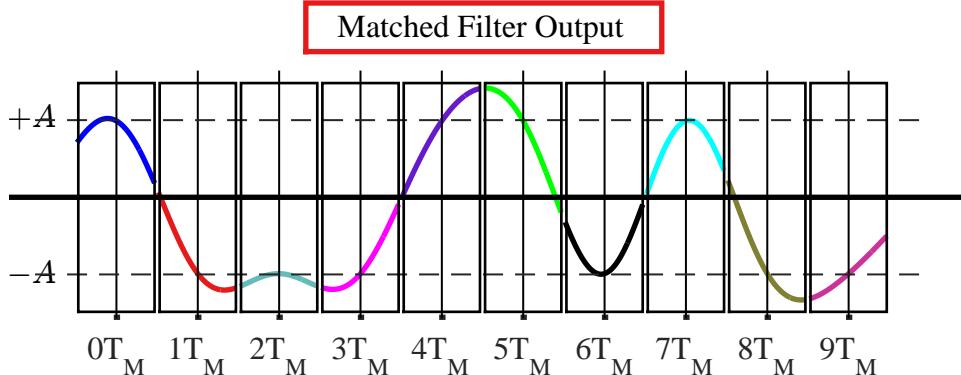


Figure 3.57: Matched filter output of Figure 3.37 as a continuous waveform

output taken at a much higher rate, say $L = 64$ samples/symbol, instead of $L = 2$ samples/symbol so that the underlying plot looks continuous as in Figure 3.57.

Next, the waveform is divided into black boxes of width T_M seconds such that each signal portion within a box starts half a symbol duration $T_M/2$ before the ideal sampling time iT_M , where i is an integer. Similarly, each signal portion within a box ends at half a symbol duration $T_M/2$ after the ideal sampling time iT_M .

Assume that such a stream of PAM symbols is printed on a paper and all black boxes are cut into separate pieces precisely at symbol boundaries. My daughter Varda just did that when I gave her a printed PAM sequence, as shown in Figure 3.58.

Now if they are placed on top of one another, we get a diagram drawn in Figure 3.59a. This is *eye diagram*, which is a modulo- T_M plot of matched filter output against time. Compare the color patterns between both figures and observe that time

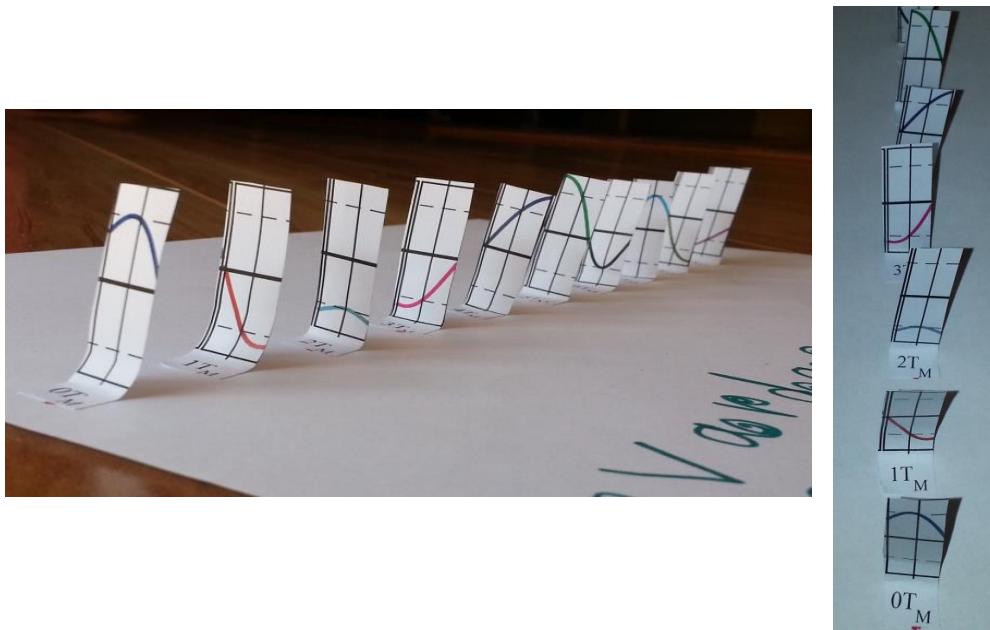


Figure 3.58: PAM symbol stream printed on a paper and cut into pieces at symbol boundaries

base in Figure 3.59a is shifted such that optimum sampling instant occurs in the middle of the plot. *This shift in time base can sometimes cause confusion* and hence it should be remembered that although the ideal sampling instants actually occur at the end of a symbol interval, an eye diagram shows that instant in the middle.

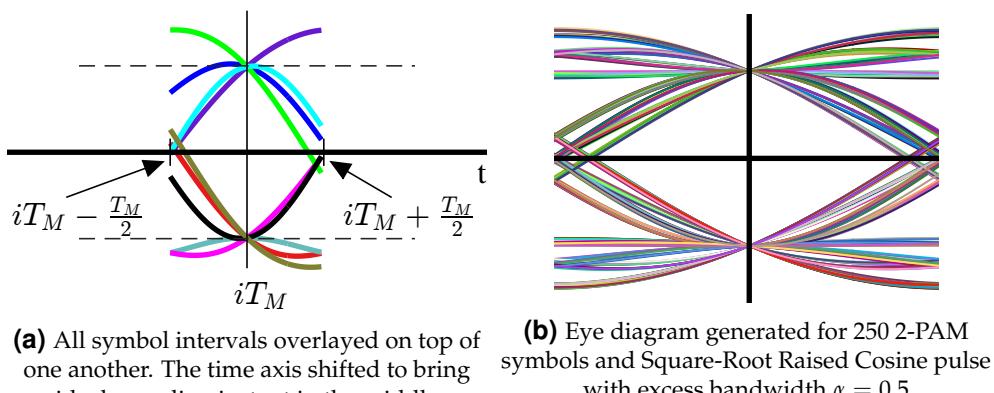


Figure 3.59: Construction of an eye diagram

Therefore, it can be concluded in general that an eye diagram draws an overlapping symbol waveform within an interval

$$-0.5T_M \leq t \leq +0.5T_M$$

Also note that the eventual pattern strongly depends on the underlying pulse auto-correlation from which the pulse shape for transmitting data is derived. Now remember that we are only using an example of a 2-PAM modulation with 10 bits, and hence 10 symbols, only. When a large number of symbols are generated for such a plot and overlaid on each other, all possible trajectories of the pulse auto-correlation dictated by the symbol sequence come into play. A diagram for Square-Root Raised Cosine pulse shape (and hence Raised Cosine pulse auto-correlation) generated for 250 symbols and excess bandwidth $\alpha = 0.5$ is shown in Figure 3.59b. *Notice that this plot resembles a human eye, hence the name eye diagram.*

Example 3.2

Interestingly, tracing a single transition in an eye diagram gives information about 3 symbols. As an example, look at the trace around $5T_M$ in Figure 3.59a:

NOW Observe that the trace goes through $+A$ at the current sampling instant.

Past Also, it starts from a high voltage level. That is an indicator that in a clean system, its previous symbol would have been $+A$. Now compare it with Figure 3.57 and it is verified that its previous symbol is indeed $+A$.

Future Finally, its level is falling below zero and towards $-A$ which indicates that the next symbol should be $-A$. The fact that the next trace in Figure 3.57 is $-A$ verifies this observation.

This will help us when we discuss symbol timing synchronization in Chapter 7.

Eye diagram is a diagnostic tool that helps in evaluation of the effects of channel noise and Inter-Symbol Interference (ISI) on the performance of a communication system. For this purpose, an eye diagram for a real transmission system can be generated through an oscilloscope. The horizontal time base of the oscilloscope is set equal to a symbol interval T_M and the matched filtered sequence is connected to the vertical axis. This superimposes the symbol intervals into a family of traces, all displayed within the same duration. The persistence of oscilloscope display makes it look like an eye.

To see how it helps, consider Figure 3.60 that consists of two symbol durations. Relevant information that can be extracted from an eye diagram is detailed below.

Best Sampling Instant: For Nyquist pulses, no ISI occurs at the end of a symbol duration T_M – which falls in the center of the eye. This gives the maximum possible SNR on average because the sample value is farthest from the decision threshold at this point. A wrong decision will only happen if the noise is sufficient to move the sample towards the other side of this threshold.

Timing Error: A timing error occurs when the eye is not sampled at maximum average opening. For a data waveform, it means that the ideal sample is not at the peak of the auto-correlation and hence the timing instant is either early or late.

Noise Margin: If a timing error occurs, the signal is sampled closer to the decision boundary. A relatively smaller amount of noise can cause a decision error. Or in other words, noise margin gets reduced[†].

[†]For some trajectories, a timing error actually improves the noise margin, while for many others, it

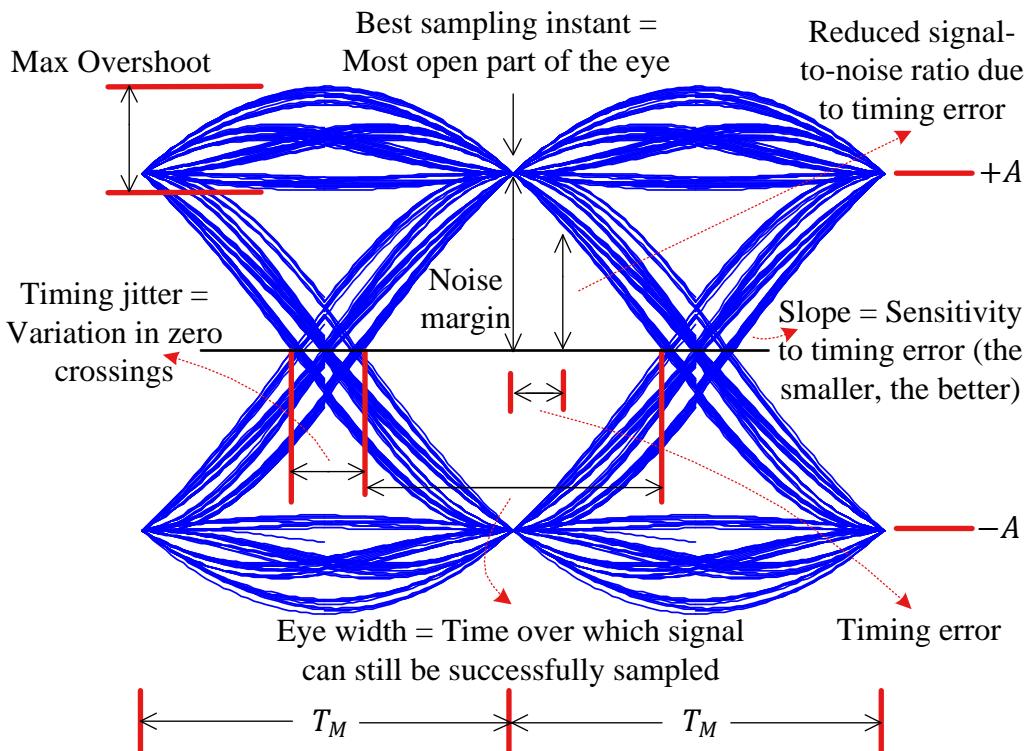


Figure 3.60: Eye diagram as a system diagnostic tool in a noiseless case

Timing Jitter: Timing jitter is a measure of average deviation around the mean zero crossings. In the old days, it was not much of a problem due to low symbol rates. Modern high speed communication systems have an increasingly shorter symbol time T_M . Since the timing jitter originates from the actual device circuitry, it becomes increasingly larger as a percentage of this symbol period.

Eye Width: The wider the eye, the cleaner the channel. Channel distortion decreases the eye width and also decreases the eye opening at the best sampling instant. An open eye pattern corresponds to minimal signal distortion. Many wireless channels cause the eye to *close*: there remains no sampling instant where a best symbol estimate can be obtained.

In fact, the eye in Figure 3.60 looks so symmetric because it is drawn for a noiseless case. Even for an AWGN channel with a low SNR, the eye can close but will remain symmetric in general as illustrated in Figure 3.61. Depending on the SNR, there is distortion at the best sampling instant as opposed to the previous case. Moreover, noise margin gets reduced, as a consequence of which decision errors can occur for relatively smaller noise power.

Slope: Slope of the eye determines its sensitivity to timing errors. A large slope implies that even a little deviation in the timing instant can cause the sample

reduces that margin. When the average of all trajectories is taken into account, the noise margin decreases proportionally to the timing error.

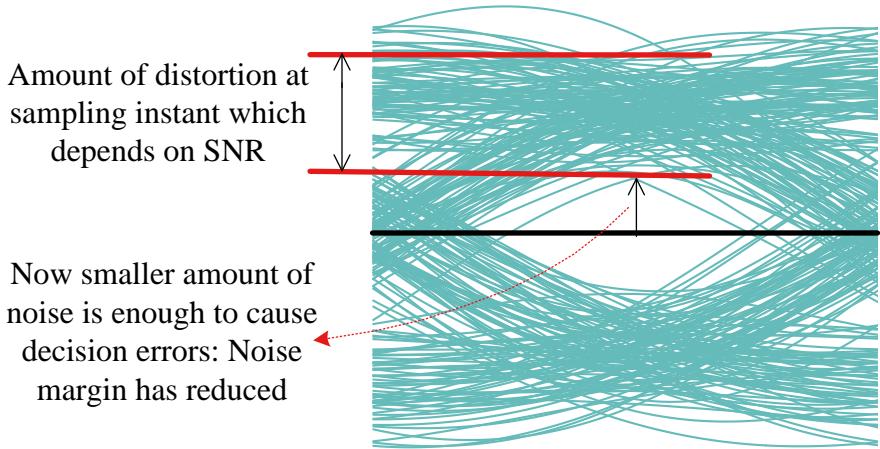


Figure 3.61: Eye diagram in an AWGN channel

value – and subsequently noise margin – to reduce significantly. Therefore, a smaller slope lessens the effect of this dependence.

Eye diagrams can also be drawn for modulation schemes packing multiple bits in one symbol. For the case $M = 4$, due to the presence of more than 2 symbols, the next symbol after every particular symbol can be $+3A$, $+A$, $-A$ or $-3A$ resulting into many different trajectories. Therefore, an eye diagram for modulation order $M > 2$ displays multiple eyes, an example of which is illustrated in Figure 3.62. Also notice that the larger the excess bandwidth α , the lesser the interference among adjacent symbols in time and the eye is more open.

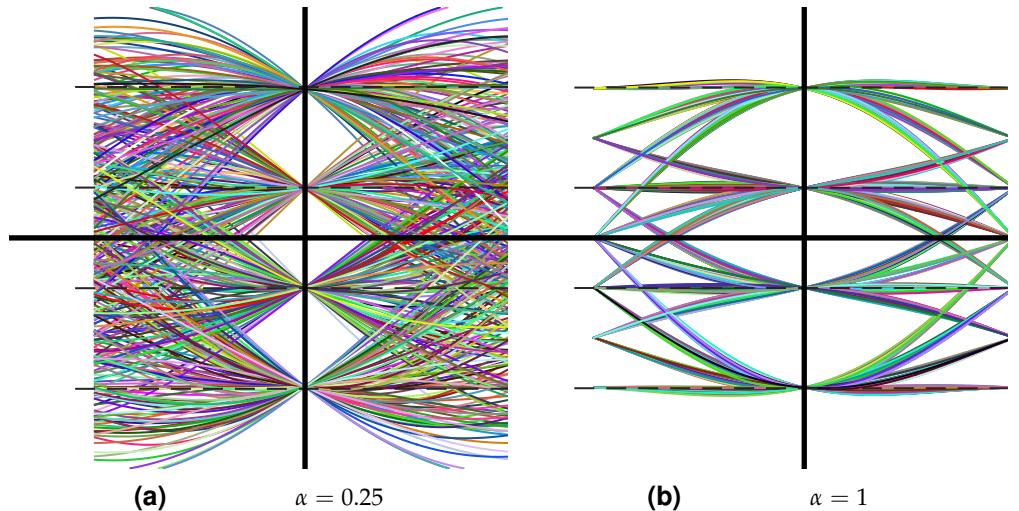


Figure 3.62: Eye diagrams generated for 400 4-PAM symbols and Square-Root Raised Cosine pulse. Higher modulations display multiple eyes. Moreover, larger excess bandwidth α results in more open eye

The eye diagrams studied before consist of a 1-dimensional modulation scheme, namely PAM. On the other hand, a QAM signal consists of two PAM signals riding on orthogonal carriers. At baseband, these two PAM signals appear as I and Q components of a complex signal at the Tx and Rx. Therefore, there are two eye diagrams for a QAM modulated signal, one for I and the other for Q and both of them are exactly the same as PAM.

Transition Diagram (Phase Trajectory)

We discussed above that in the case of QAM transmission, the matched filter output is complex valued and hence two eye diagrams are needed, one is the modulo- T_M plot of the inphase component and the other is the modulo- T_M plot of the quadrature component. Instead of drawing the I and Q parts in separate diagrams, another alternative that displays the relationship between Q versus I is a *transition diagram* or a *phase trajectory* plot. A transition diagram is just a 2D diagram depicting the continuous-time Q vs I signal in the complex plane itself *without any time axis*.

Some examples of the transition diagrams for a 4-QAM and a 16-QAM modulation schemes are drawn in Figure 3.63 for a Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ and $\alpha = 1$ for zero noise. Some observations are as follows.

- The angle of the plot with respect to I -axis is the instantaneous phase of the QAM modulated signal (without any carrier).
- There is no explicit time axis in this plot but it can be traced from the plot moving from one constellation point to another.
- We can easily identify the underlying constellation points from the phase trajectory.
- A higher excess bandwidth, e.g., $\alpha = 1$, prevents any substantial overshoots from appearing when the signal transitions from one constellation point to another. This makes sense because in a time domain plot, a higher excess bandwidth gives rise to smooth transitions between symbols (very similar to an unmodulated sinusoid).

Scatter Plot

In general, *a scatter plot is a graph that shows the relationship between two sets of data*. More specifically, a scatter plot of a complex signal is a plot of its Q samples drawn versus its I samples. For the purpose of digital communications, a scatter plot can be explained as follows.

Remember that we used a continuous version of the matched filter output to understand the eye diagrams and the transition diagrams. For scatter plot, we will use the matched filter output downsampled at 1 sample/symbol as a reference. After downsampling the matched filter output to symbol rate, the samples thus obtained are mapped back to the constellation as illustrated in Figure 3.38. Before the symbol decisions are made, those samples form a cloud around the ideal constellation points as shown in Figure 3.64a. This cloud of symbol-spaced samples mapped on the original constellation diagram is called a *scatter plot*.

For a QAM modulation, this cloud of samples around the constellation points is now 2-dimensional (for PAM, there was no Q component) and can also be understood

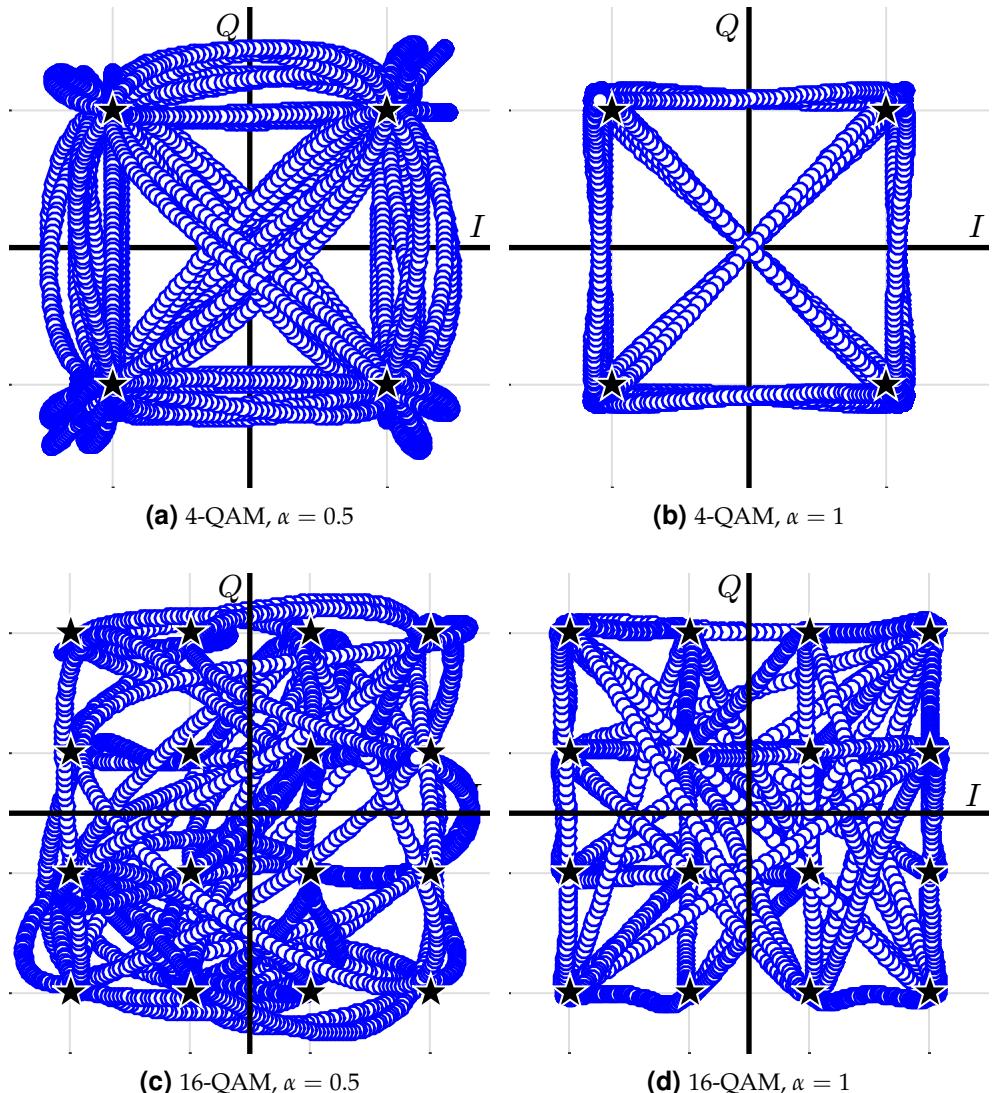
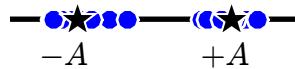


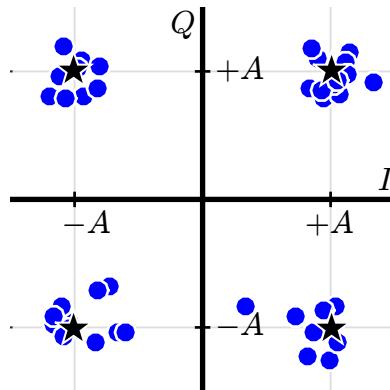
Figure 3.63: Transition diagram or phase trajectory for 4-QAM and 16-QAM for a Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ and $\alpha = 1$ and zero noise

as a plot of Q versus I for each mapped value, i.e., a downsampled version of the transition diagram. This is illustrated in Figure 3.64b.

Looking at the scatter plot, one can readily deduce a lot of performance measures for the particular transmission system. One such measure widely used in practice is the [Error Vector Magnitude \(EVM\)](#) that computes the vector sum of all the Rx points from the ideal location. At this stage, however, it is enough to observe that the diameter of these clouds is a rough measure of the noise power corrupting the signal. For the ideal case of no noise, this diameter is zero and all the optimally timed samples coincide with their respective constellation points.



(a) 2-PAM modulation in the presence of AWGN



(b) 4-QAM modulation in the presence of AWGN

Figure 3.64: Scatter plots for PAM and QAM

3.9 Modulation Bandwidths

After covering linear modulations and pulse shaping, we can correctly determine the occupied bandwidth for each modulation scheme. Figure 3.31 shows the bandwidth of a Square-Root Raised Cosine pulse shape as $(1 + \alpha)/2T_M$. We have also seen that the spectrum approximately remains the same, provided that there is enough randomness in the bit stream and the resulting symbols are equally likely and independent from each other. Therefore, the bandwidth for a PAM modulated signal can be given as

$$BW_{\text{PAM}} = \frac{1 + \alpha}{2T_M} \quad (3.44)$$

QAM is basically a similar modulation scheme except that it is modulated on a carrier. After the spectral upconversion, both positive and negative portions of the baseband spectrum appear at $+F_C$ and $-F_C$. The bandwidth – positive portion of the spectrum – hence becomes double.

$$BW_{\text{QAM}} = 2 \frac{1 + \alpha}{2T_M} = \frac{1 + \alpha}{T_M} \quad (3.45)$$

The same equation holds for PSK modulation. In case some other pulse shape is used with a bandwidth B , the factor $0.5(1 + \alpha)$ can be replaced with an appropriate factor in the above equations.

3.10 Computing Error Rates

Having built a simple digital communication system, it is necessary to know how to measure its performance. As the names say, *Symbol Error Rate (SER)* and *Bit Error Rate (BER)* are the probabilities of receiving a symbol and bit in error, respectively. SER and BER can be approximated through simulating a complete digital communication system involving a large number of bits and comparing the ratio of symbols or bits received in error to the total number of bits. Hence,

$$SER = \frac{\text{No. of symbols in error}}{\text{Total no. of transmitted symbols}} \quad (3.46)$$

and

$$BER = \frac{\text{No. of bits in error}}{\text{Total no. of transmitted bits}} \quad (3.47)$$

Till now, the only imperfection between Tx and Rx entities we have covered is AWGN. It then makes sense that more distortions should corrupt the signal further and result in more errors. Hence, SER or BER plotted against some measure of signal-to-noise ratio (SNR) is the benchmark against which the performance of a system can be measured. Next, we discuss how we choose a specific measure of SNR against which SER and BER should be plotted.

Figure of Merit: E_b / N_0

The *SNR* in a digital communication system is defined as a ratio of signal power in a symbol P_M to noise power P_w as

$$SNR = \frac{P_M}{P_w} \quad (3.48)$$

where the signal power P_M is the average symbol energy of the modulation per unit symbol time.

$$P_M = \frac{E_M}{T_M} \quad (3.49)$$

For PAM and QAM modulation schemes, average E_M was defined in Eq (3.18) and Eq (3.42). Moreover, how to determine the noise power P_w was detailed in Section 2.8, Eq (2.47).

When specified in dB units and using a factor of 10 for power conversions, it is

$$\begin{aligned} SNR_{dB} &= P_{M,dB} - P_{w,dB} \\ &= 10 \log_{10} P_M - 10 \log_{10} P_w \end{aligned} \quad (3.50)$$

A visual measure of finding SNR_{dB} is shown in Figure 3.65.

Using Eq (3.49) and Eq (2.47), notice that the SNR can also be written as

$$P_M \cdot \frac{1}{P_w} = \frac{E_M}{T_M} \cdot \frac{1}{N_0 B}$$

The above equation can be rearranged as

$$\frac{E_M}{N_0} = \frac{P_M}{P_w} \cdot BT_M$$

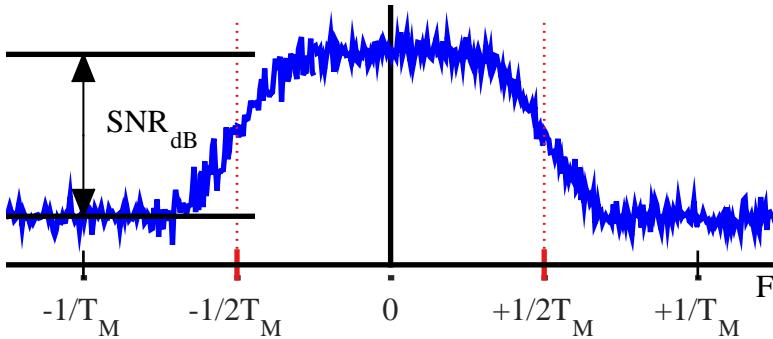


Figure 3.65: A visual measure of SNR_{dB}

Recall that there are $\log_2 M$ bits in one symbol, and hence

$$E_b = \frac{1}{\log_2 M} \cdot E_M \quad (3.51)$$

where E_b is energy per bit. Similarly, the bit rate is

$$R_b = \frac{\log_2 M}{T_M} \quad (3.52)$$

Plugging the values for E_b and R_b ,

$$\frac{E_b}{N_0} = \frac{P_M}{P_w} \cdot \frac{B}{R_b} = \text{SNR} \cdot \frac{B}{R_b} \quad (3.53)$$

So we can say that E_b/N_0 is equal to the SNR scaled by the ratio of bandwidth to bit rate.

Now the question is: why is E_b/N_0 *a natural figure of merit* in digital communication systems instead of SNR.

- Intuitively, the ratio of signal power P_M to noise power P_w in the system (i.e., SNR) should be a good criterion for measuring performance of different communication systems. And indeed it is true for analog communication systems where the signal continuity transcends the need to partition the signal into some units of time. On the other hand, a symbol is the basic building block of a digital communication system and signal energy within each symbol turns out to be a more useful parameter.
- As we saw in Section 3.2, many bits ($\log_2 M$ to be precise) can be packed into every symbol for an M -symbol modulation. Consequently, within every T_M seconds, either 1 bit is travelling through the air (BPSK), or 2 bits (QPSK, 4-PAM), or 4 bits (16-QAM, 16-PSK), and so on. As shown in Eq (3.53), using E_b/N_0 instead of SNR automatically incorporates this bits per symbol factor through R_b .
- Again from Eq (3.53), bandwidth is also taken into account. A system using less bandwidth than the other will have this factor reflected in its E_b/N_0 values.

In summary, E_b / N_0 is a normalized SNR measure, which proves really useful when comparing the SER/BER performance of digital modulation schemes with different bits per symbol and bandwidths.

The ratio E_b / N_0 is dimensionless as

$$\frac{E_b}{N_0} = \frac{\text{Joules}}{\text{Watts/Hz}} = \frac{\text{Watts} \cdot \text{seconds}}{\text{Watts} \cdot \text{seconds}}$$

It is frequently expressed in decibels as $10 \log_{10} E_b / N_0$.

Scaling Factors

In most of the figures in this text, you would have noticed a very clear labeling on the x-axis. On the y-axis though, the emphasis has been on the shape rather than the exact values. Now when the SER/BER measurements are to be taken, scaling factors on y-axis become equally important.

Constellation

Starting with the symbol constellations, we scale the values such that the average energy per symbol is 1. This can be achieved by dividing the constellation values by the square root of average symbol energy (sum of squares of the constellation points). For example, for a PAM modulation with $A = 1$, E_M was shown in Eq (3.18) to be equal to

$$E_{M-\text{PAM}} = \frac{M^2 - 1}{3}$$

which yields

$$E_{2-\text{PAM}} = 1$$

$$E_{4-\text{PAM}} = 5$$

$$E_{8-\text{PAM}} = 21$$

⋮

When the constellation points $\pm 1, \pm 3, \pm 5, \dots$, are divided by respective $\sqrt{E_M}$, various M -PAM constellations become normalized to have average symbol energy equal to 1, as illustrated in Figure 3.66. As a verification,

$$E_{2-\text{PAM},\text{norm}} = \frac{1}{2} (1^2 + 1^2) = 1$$

$$E_{4-\text{PAM},\text{norm}} = \frac{1}{4} \cdot \frac{1}{5} (-3^2 + -1^2 + 1^2 + 3^2) = \frac{1}{20} \cdot 20 = 1$$

$$E_{8-\text{PAM},\text{norm}} = 1$$

where the subscript ‘norm’ stands for normalized.

On a similar note, the scaling factor for QAM with $A = 1$ was given in Eq (3.42) as

$$E_{M-\text{QAM}} = \frac{2}{3} (M - 1)$$

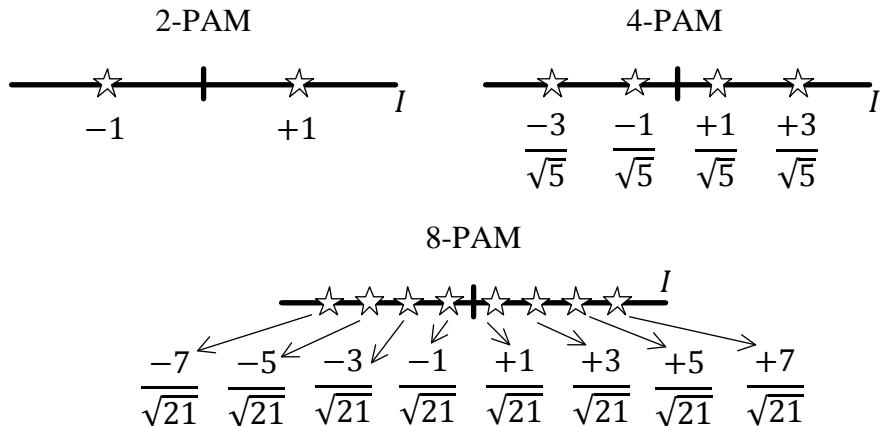


Figure 3.66: Normalized M -PAM constellations with average energy per symbol
= 1

which yields

$$E_{4\text{-QAM}} = 2$$

$$E_{16\text{-QAM}} = 10$$

$$E_{64\text{-QAM}} = 42$$

\vdots

For example, when scaled with square-root of this normalization factor, a normalized 4-QAM constellation is shown in Figure 3.67. Here, using Pythagoras theorem and realizing that all 4 symbols have equal energies,

$$E_{4\text{-QAM, norm}} = \frac{1}{4} \cdot 4 \cdot \left(\frac{1}{2} + \frac{1}{2} \right) = 1$$

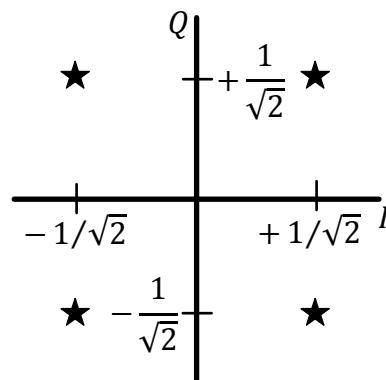


Figure 3.67: Normalized 4-QAM constellation with average energy per symbol = 1

Note 3.12 System Design Tradeoffs

Observe that the constellation points come closer to each other for higher-order modulation and normalized average symbol energy. This emphasizes the fact that a higher bit rate R_b , or more bits per symbol, does not come for free. The major tradeoffs involved are the following.

- *Power* needs to be increased for a fixed bandwidth, because more energy within the same symbol time is needed for maintaining a similar distance between constellation points.
- *Bandwidth* needs to be expanded to accommodate a higher symbol rate for fixed power, because same energy then becomes available to spend during a shorter amount of time.
- An increased *error rate* needs to be accepted for fixed power and bandwidth.

Tx Pulse

To understand the scaling factor for Tx pulse shape, also known as Tx filter, consider the block diagram of a PAM modulator in Figure 3.17 or a QAM modulator in Figure 3.48. The next step after converting bits to symbols is upsampling the symbol train before it is filtered by a square-root Nyquist filter. This square-root Nyquist filter performs two operations:

1. It lowpass filters the spectral replicas arising as a result of increasing the sample rate, see Section 2.7.2 for details.
2. It shapes the symbol sequence so that it is bandlimited in the frequency domain according to given specifications.

We also discussed in Section 2.7.2 that to maintain the same peak time domain amplitude in the upsampled signal as before, the filter should be designed with a gain equal to the upsampling factor, L in this case. The three available choices, time domain amplitudes and energies are presented in Table 2.3. When we choose this option, the symbol energy gets scaled by L as well, as was derived in Eq (2.45). Therefore, denoting energy/symbol in the upsampled and filtered case with E'_M ,

$$E'_M = L \cdot E_{M,\text{norm}} = L \quad (3.54)$$

Remember from Eq (3.34) that the peak value in a Square-Root Raised Cosine filter is $1 - \alpha + 4\alpha/\pi$. In an actual implementation of the system, the pulse coefficients should be scaled by this value as well so that the peak value becomes unity and the precision with which the coefficients are represented in a fixed-point arithmetic processor can be maintained. However, here the purpose is only to simulate the BER due to which we ignore this factor.

Noise Power

In Section 2.8, the noise power P_w within a bandwidth B was derived as

$$P_w = N_0 B \quad (3.55)$$

For a discrete signal with sampling rate F_S , it is understood that the bandwidth has already been constrained by an analog filter within the range $\pm 0.5F_S$ to avoid aliasing. Plugging $B = 0.5F_S$ in the above equation, the noise power in a sampled bandlimited system is given as

$$P_w = N_0 \cdot \frac{F_S}{2}$$

A digital modulation signal is sampled at a rate $F_S = L/T_M$ where L is the number of samples/symbol. Bit rate R_b can change with different modulation schemes and sample rate F_S can be increased or decreased during the processing chain. On the other hand, the symbol rate $R_M = 1/T_M$ is a standard parameter which is directly proportional to the signal bandwidth as shown in Section 3.9, and hence it remains constant throughout the system implementation. For this reason, $1/T_M$ in most simulation systems is chosen to be equal to 1 and the other parameters can be scaled accordingly. Then,

$$P_w = N_0 \cdot \frac{L}{2} \quad (3.56)$$

From here, the relationship between noise power P_w and E_b/N_0 can be written as

$$P_w = \frac{E_b}{E_b/N_0} \cdot \frac{L}{2}$$

which can be modified using $\log_2 M$ bits per symbol,

$$P_w = \frac{E_M}{\log_2 M \cdot E_b/N_0} \cdot \frac{L}{2} \quad (3.57)$$

where E_M is the symbol energy. In case a normalized energy is used for symbol constellations, it can be replaced with 1.

Notice that noise power increases by a factor of L due to oversampling the signal. On the other hand, the signal energy increases by a factor of L due to scaling shown in Eq (3.54). The overall result is that both factors cancel out in the numerator and denominator of SNR expression in Eq (3.48). This is illustrated in Figure 3.68.

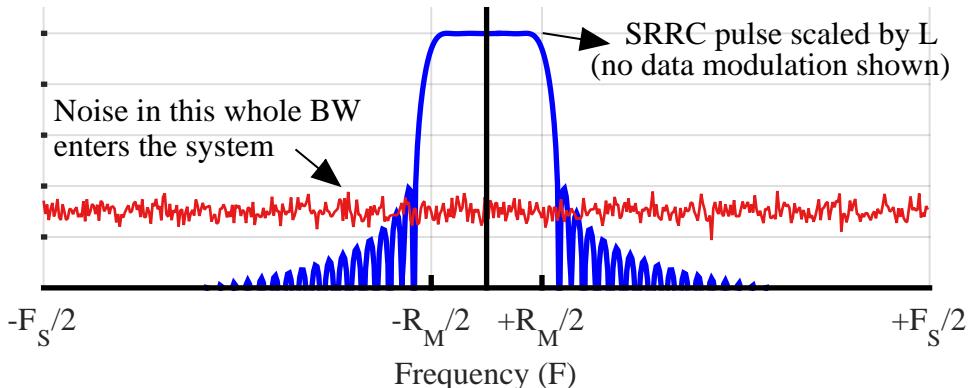


Figure 3.68: Signal amplitude scaling by L and noise bandwidth scaling by L keep the SNR unaffected

An interesting question at this stage is as follows. We did all the noise power calculations based on a rectangular filter to justify the relation

$$P_w = N_0 \cdot \frac{F_S}{2}$$

which clearly assumes a brickwall filter at $F_S/2$ such that the noise power is given by the width and height of the rectangle. How are these calculations valid for an overall Nyquist filter (e.g., Raised Cosine) that is not a brickwall and in fact has a sloping transition bandwidth?

This is justified due to the odd symmetry of the Nyquist filter around $1/2T_M$. The curve (as the term symmetry implies) moves away from the frequency $1/2T_M$ equally in both directions, i.e., the area under the curve in which it expands is the same as the area it leaves behind. This is shown in Figure 3.69. Due to these equal areas, the noise calculations for a rectangular filter are valid for a Nyquist filter as well.

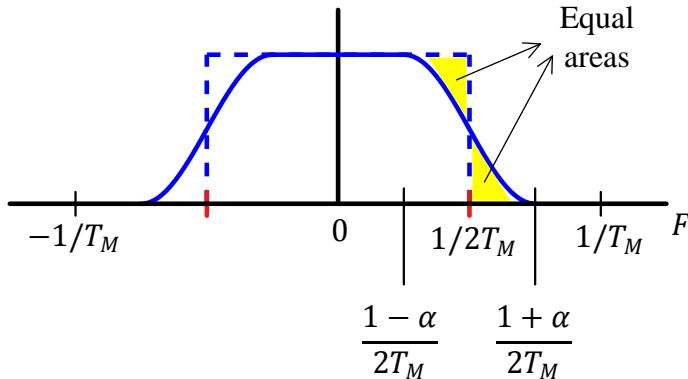


Figure 3.69: Noise calculations for a brickwall and a Nyquist filter are equivalent

Rx Pulse

Both the signal and noise enter the system before filtered by the Rx pulse. Therefore, any scaling has no effect on the SNR and it only changes the time domain amplitudes. Since the Raised Cosine pulse is a convolution of Tx and Rx Square-Root Raised Cosine pulses defined above, it has a peak amplitude of 1 and hence maps the symbols exactly on the constellation.

Finally, remember that the group delay is now given by the contribution from both Tx and Rx square-root Nyquist filters. For that reason, we remove $LG/2 + LG/2 = LG$ samples at the start and LG samples at the end of the received sequence.

BER Plots

Now we are at a stage where we can plot the symbol and bit error rates of all three modulation schemes discussed above, namely PAM, QAM and PSK. To avoid any confusion, we leave SER plots and just concentrate on BER plots. SER plots are an obvious byproduct of the simulation setup discussed above.

The BER plots for 2, 4 and 8-PAM modulation are shown in Figure 3.70. Note that for a fixed E_b/N_0 , the BER increases with M because the same symbol energy has to be divided among $\log_2 M$ bits. Similarly, for a fixed BER, more energy is required as M increases to maintain similar distances among constellation points. As an example, for a target BER of 10^{-4} ,

- E_b/N_0 needed by 2-PAM is 8.4 dB,
- E_b/N_0 needed by 4-PAM is 12.2 dB,
- E_b/N_0 needed by 8-PAM is 16.5 dB,

A *gray code* is a method of assigning bits to symbols in a manner that adjacent symbols differ in only one bit. For example, symbols in a 4-PAM constellation can be assigned bits 00, 01, 11 and 10 (as opposed to 00, 01, 10 and 11) such that all symbols have neighbours with only one bit difference. Since Gaussian noise has a higher probability around the mean value, the most likely symbol errors end up in adjacent symbols. As a result, these most likely symbol errors produce one erroneous bit and remaining $M - 1$ correct bits. Hence, the relation between BER and SER can be defined as

$$\text{BER} = \frac{1}{\log_2 M} \text{SER}$$

which signifies that one symbol error produces one bit error only.

Moving on to QAM, the BER plots for 4, 16 and 64-QAM modulation are shown in Figure 3.71. Similar conclusions hold for BER vs E_b/N_0 as in the case of PAM.

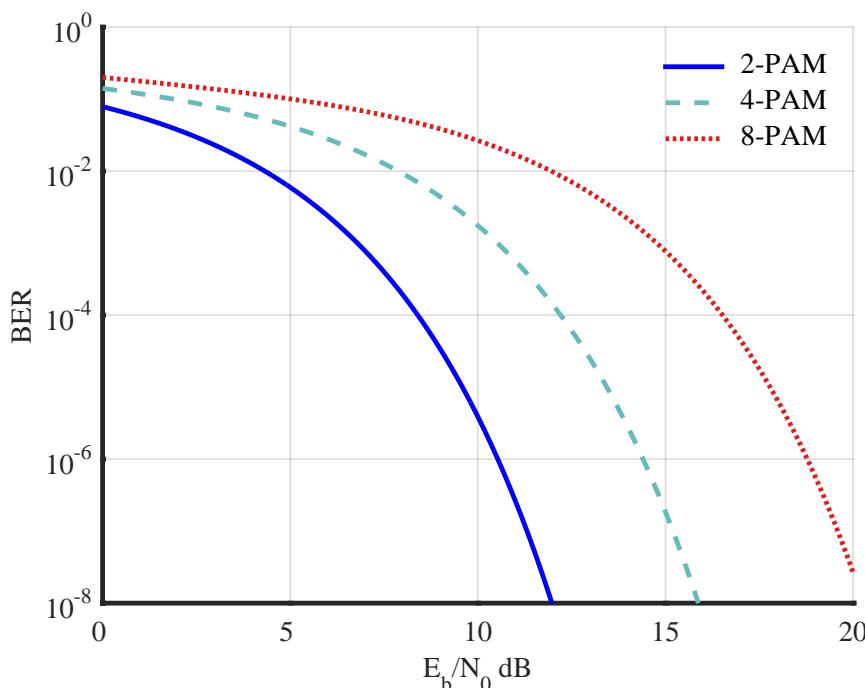


Figure 3.70: BER plots for 2, 4 and 8-PAM modulations

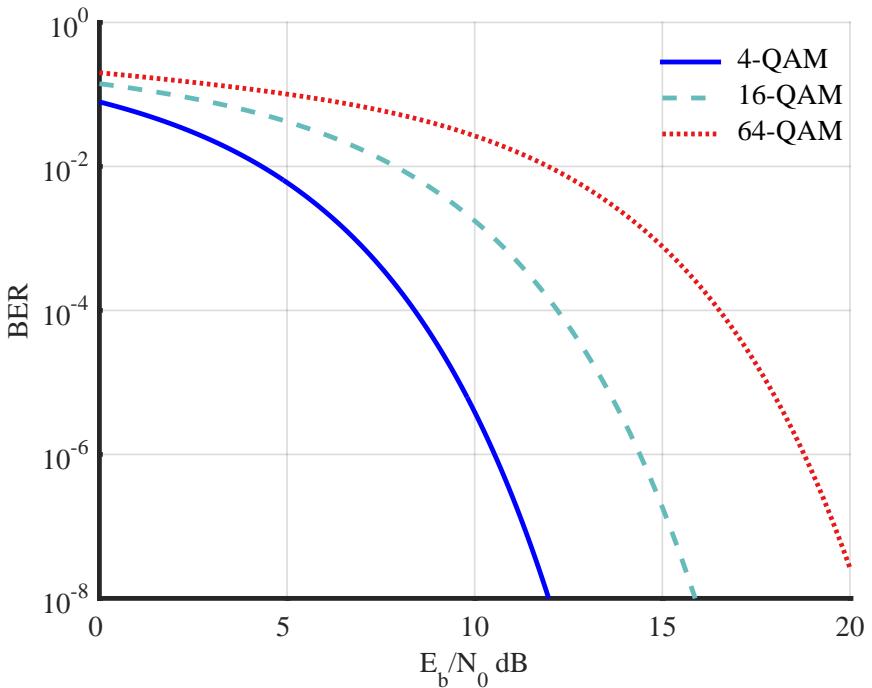


Figure 3.71: BER plots for 4, 16 and 64-QAM modulations

Finally, the BER plots for 2, 4, 8 and 16-PSK modulation are shown in Figure 3.72. Notice that the plots for BPSK and QPSK overlap with each other because QPSK can be regarded as a pair of orthogonal BPSK systems. We will have more to say about this point in the next section.

3.11 Spectral Efficiency

In BER vs E_b/N_0 plots above, we noticed that for a target BER of 10^{-4} , E_b/N_0 required by 2-PAM, 4-PAM and 8-PAM are 8.4 dB, 12.2 dB and 16.5 dB, respectively. Does this mean that 8-PAM is 8 dB worse than 2-PAM modulation scheme? The answer is no, because the bandwidth has not been taken into consideration in this comparison. For the same symbol rate (which means same bandwidth), 8-PAM transmits 3 bits of information as compared to just 1 bit of information by 2-PAM.

For that reason, bandwidth must also be considered to get a complete picture of how different modulations compare to one another. Thus, *spectral efficiency* is defined as the ratio of bit rate to the bandwidth and its units are bits/s/Hz.

$$SE = \frac{R_b}{B} \quad \text{bits/s/Hz} \quad (3.58)$$

In essence, spectral efficiency is a way to figure out how fast the system is per unit of bandwidth. For QAM and PSK constellations with a square-root Nyquist filter of excess bandwidth α , we saw in Section 3.9 that the required bandwidth is $(1 + \alpha)/T_M$.

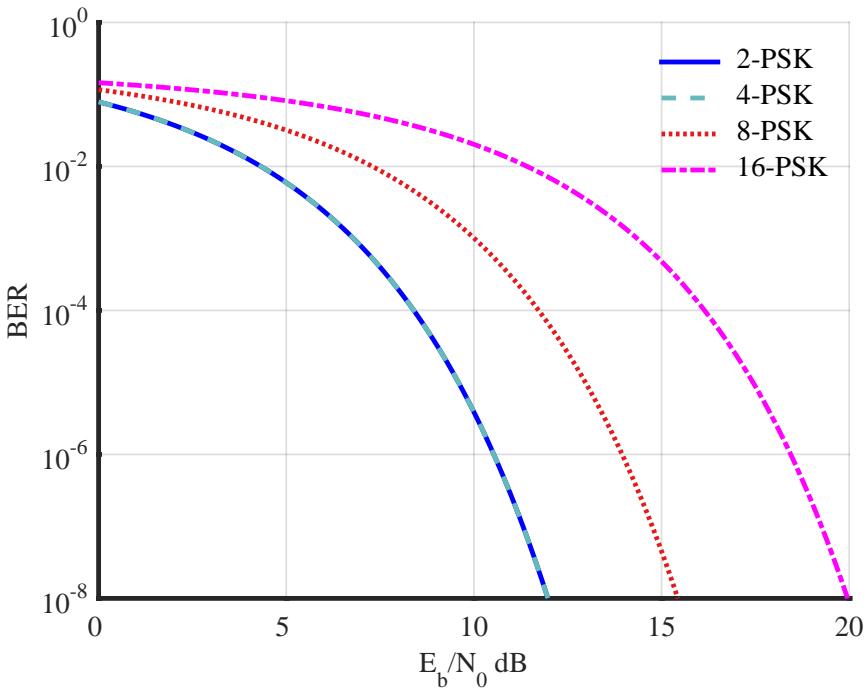


Figure 3.72: BER plots for 2, 4, 8 and 16-PSK modulations

In terms of bit rate,

$$B_{\text{QAM}} = (1 + \alpha) \frac{R_b}{\log_2 M}$$

from which spectral efficiency can be determined as

$$SE_{\text{QAM}} = \frac{\log_2 M}{1 + \alpha} \text{ bits/s/Hz} \quad (3.59)$$

Note 3.13 BER for BPSK and QPSK

Going back to our discussion on similar BER for BPSK and QPSK, note that the BPSK system conveys one bit per symbol interval T_M , whereas the QPSK system conveys two bits per symbol interval. For a given *symbol rate* $1/T_M$ (and hence the same bandwidth) and a given signal power, QPSK has to divide the energy into two bits resulting in smaller E_b/N_0 ratio and higher BER. However, its bit rate R_b is twice as compared to BPSK.

On the other hand, for a given *bit rate* R_b and a given signal power, QPSK symbol interval T_M is twice as long as that of the BPSK, similar energy is available per unit time and hence both BPSK and QPSK have the same BER. However, its symbol rate $1/T_M$ is half and so it uses half the bandwidth as compared to BPSK.

Turning towards PAM, although the bandwidth required is half as compared to QAM, it transmits half the number of bits as well due to using I channel only. Therefore, its spectral efficiency is similar to QAM.

Different constellations with different number of points can be compared by taking all the parameters into account, namely bits/symbol, bandwidth and SNR performance. This can be achieved by plotting the spectral efficiency against the value of E_b/N_0 required to obtain a specific target BER. Figure 3.73 shows spectral efficiency versus E_b/N_0 for QAM and PSK constellations for $\text{BER} = 10^{-5}$ and excess bandwidth $\alpha = 0.5$, where E_b/N_0 values can be taken from the x-axis of BER curves in the previous section. Note that QAM provides better spectral efficiency than PSK because it packs the constellation points in a more efficient manner. In general, better modulations operate at high spectral efficiency and low E_b/N_0 , i.e., to the top left of this curve.

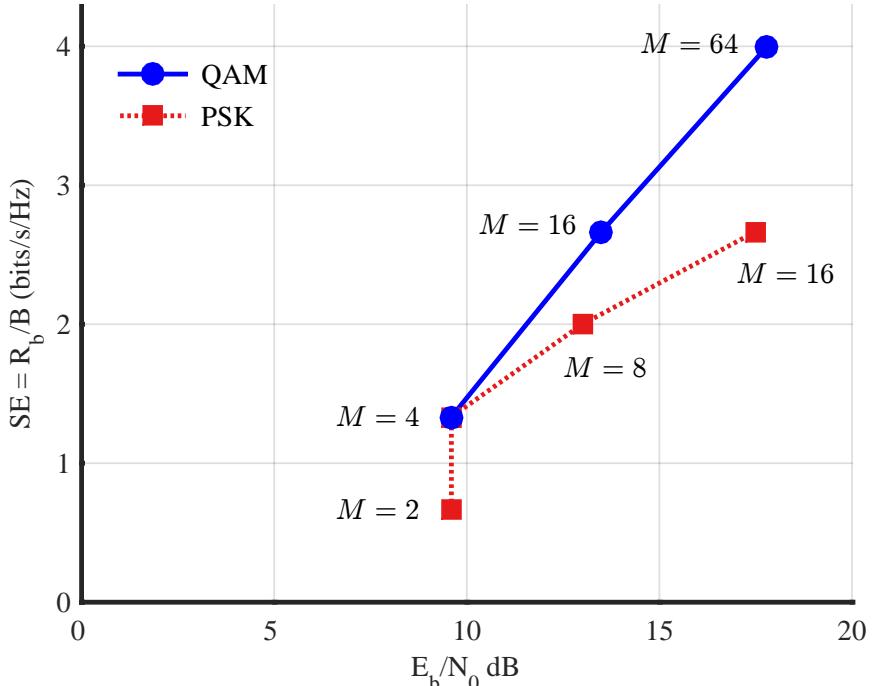


Figure 3.73: Spectral efficiency R_b/B versus E_b/N_0 determines the overall merit of a modulation scheme. This figure is drawn for $\text{BER} = 10^{-5}$ and a Square-Root Raised Cosine pulse with 50% excess bandwidth

On a final note, we rewrite Eq (3.53) here.

$$\frac{E_b}{N_0} = \frac{P_M}{P_w} \cdot \frac{B}{R_b} = \text{SNR} \cdot \frac{B}{R_b}$$

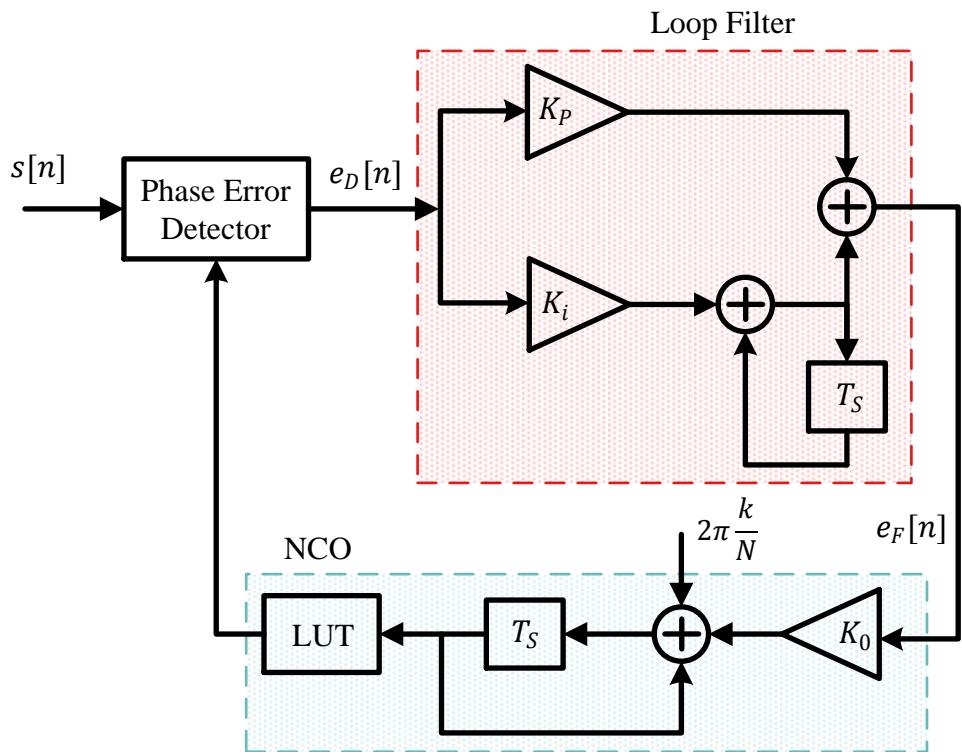
The above equation can be written as

$$\text{SE} = \frac{R_b}{B} = \frac{\text{SNR}}{E_b/N_0}$$

As a result, SE can also be seen as the ratio of SNR to E_b/N_0 .

Chapter 4

Phase Locked Loop



“Sync is both strange and beautiful. It is strange because it seems to defy the laws of physics It is beautiful because it results in a kind of cosmic ballet.”

Steven Strogatz

IBM Watson and Google DeepMind are the most complex computers that, some believe, will try to run the world in a distant future. A Phase Locked Loop (PLL) on the other hand is the simplest computer that actually runs much of the world as a fundamental component of intelligent electronic circuits. The PLL was invented by the French engineer Henri de Bellescize in 1932 when he published his first implementation in the French journal *L’Onde Electrique*, see Ref. [12]. Some initial ideas on PLL even started appearing as early as 1919.

Before we start, however, we cover some background information on what synchronization is and where a PLL is required in this picture.

4.1 What is Synchronization?

Until now, the only impairment that affected the transmitted signals was Additive White Gaussian Noise (AWGN). In Chapter 3, the effects of this noise on the transmitted signal in terms of eye diagrams and scatter plots was explained while Bit Error Rates (BER) of various linear modulation schemes were plotted in its presence. Such a study revealed that if AWGN was the only problem with communicating through signals in the real world, the design of digital and wireless communication systems would have been a simple project.

The real world, however, is different and brings with it numerous other distortions that a wireless communications designer needs to deal with. Maintaining synchronization between a Tx and a Rx is one such significant challenge.

The word *Synchronization* is derived from *syn* (the same) and *Chronos* (the Greek god of time). Synchronizing is to make one or more things coordinate to occur in unison. The most common example is a march of soldiers where they move their arms and legs at exactly the same time and with the same speed. In an internal combustion engine, a timing belt synchronizes the rotation of the crankshaft with the engine valves thus allowing them to open and close at the correct times.

Here, I want to write a few delightful lines from the book by Steven Strogatz [13] in regards to the place of synchronization in our universe and founding the basis of our signals in terms of frequency or cycles/second.

“At the heart of the universe is a steady, insistent beat: the sound of cycles in sync. It pervades nature at every scale from the nucleus to the cosmos. Every night along the tidal rivers of Malaysia, thousands of fireflies congregate in the mangroves and flash in unison, without any leader or cue from the environment. Trillions of electrons march in lockstep in a superconductor, enabling electricity to flow through it with zero resistance. In the solar system, gravitational synchrony can eject huge boulders out of the asteroid belt and toward earth; the cataclysmic impact of one such meteor is thought to have killed the dinosaurs. Even our bodies are symphonies of rhythm, kept alive by the relentless, coordinated firing of thousands of pacemaker cells in our hearts. In every case, these feats of synchrony occur spontaneously, almost as if nature has an eerie yearning for order.”

For successful operation of a digital receiver, it has to be synchronized with the incoming waveform. In the context of digital communication systems, it implies *carrier synchronization and timing synchronization*.

Carrier synchronization requires the phase and frequency alignment of the local oscillator at the Rx with the carrier of the incoming signal. In the context of this text, it is a problem to be solved in coherent QAM and PSK systems (but not PAM as it is not a passband modulation). Non-coherent modulation schemes, on the other hand, do not require a carrier estimate. This text is focused on coherent modulation schemes only which exhibit superior performance as compared to non-coherent techniques.

Timing synchronization is concerned with sampling the Rx waveform at the proper instants of symbol boundaries. This is a problem to be solved in all digital communication systems, including PAM, QAM and PSK.

In every digital communication system, the Tx has the easier role of signal generation while the Rx has the tougher job of figuring out the intended message just like solving a puzzle. Estimating and compensating for the frequency, phase and timing offsets between Tx and Rx oscillators is one such challenge. Keep in mind that in the subsequent discussions, *acquisition* refers to the initial capture of the synchronization parameters while *tracking* implies following little deviations in those parameters during the later steady state operation. In other words, the variables are in close vicinity of their steady state values during tracking while acquisition refers to bringing them from an initial state to the tracking mode.

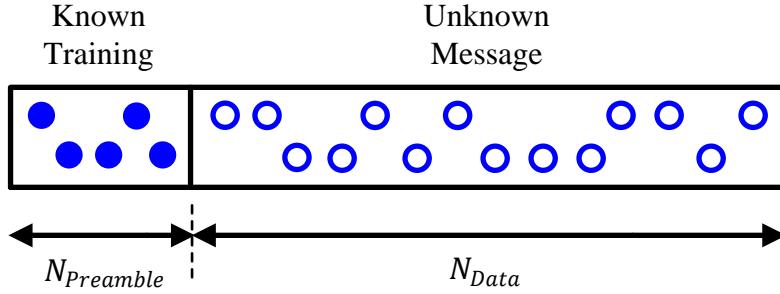
Keep in mind that most of the algorithms described in this text are for QPSK modulated signals. However, similar concepts apply for higher-order QAM schemes with slight modifications that take advantage of the arrangements of points in a particular constellation. Moreover, we study in Chapter 8 how a wireless channel distorts the modulated symbol waveform. Even in this fading channel scenario, the fundamental principles presented here stay the same and little modifications in synchronization strategies are required, mostly in the form of pre and post processing stages.

Availability of Known Data

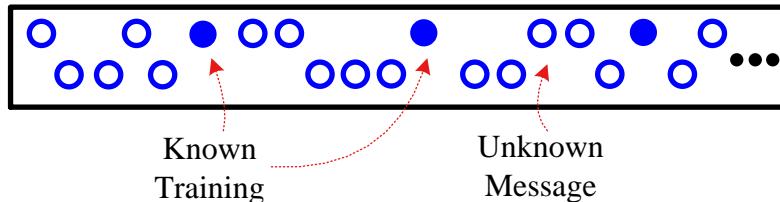
Depending on the availability of known data, synchronization in digital systems is largely based on the following three approaches.

Data-aided: To help the Rx in many systems, the Tx inserts symbols already agreed upon with the Rx within the message such that the Rx can acquire the unknown parameters through the knowledge of this 'data'. This is shown in Figure 4.1. Performing synchronization using this training is known as *data-aided synchronization*. Most widespread wireless communication systems in today's world such as LTE and WiFi implement algorithms based upon this approach. As an example, the preamble in Bluetooth Low Energy is 8 bits long value 0xAA, i.e., 10101010. This kind of alternating sequence helps in fast synchronization at the Rx.

There are two common ways to embed the known symbols within the Tx sequence. One is to prepend a training sequence before the actual data symbols. This is a common configuration in systems where the parameters such as carrier phase and frequency or the channel impulse response are not expected to



(a) Known training sequence, or a preamble, is prepended



(b) Training can also be inserted periodically within the message

Figure 4.1: In many systems, the incoming signal contains known data in order to help the Rx estimate and correct various disturbances in the Rx signal

change much for the duration of the block or packet. The other option is to periodically inject the known symbols, called pilots, within the Tx sequence so that the Rx has access to known symbols until the very end of the message. With their help, it can track the unknown parameters – even when they are slowly changing – to successfully decode the packet.

One problem with data-aided synchronization strategy is the waste of resources. The power and time spent on transmitting the training could have been used for sending more data: the spectral efficiency of the system is reduced by a non-negligible factor. Assuming a training length of N_p and data length of N_d , the spectral efficiency decreases by a factor of

$$\frac{N_p}{N_d + N_p} \quad (4.1)$$

Another way to look at how the presence of known training results in increased bandwidth is as follows. Assume that a data rate of R_b bits/second needs to be supported on a link which translates to a symbol rate of $R_M = 1/T_M$ symbols/second for a total of N_d symbols. However, an addition of N_p training symbols – and the requirement to send the same amount of bits within the same duration – implies that the new symbol rate $R_{M,\text{new}}$ becomes

$$R_{M,\text{new}} = R_M \left(1 + \frac{N_p}{N_d} \right) \text{ symbols/second}$$

Since the bandwidth is directly proportional to the symbol rate, the system needs a larger bandwidth.

Decision-directed: To avoid this penalty on spectral efficiency, alternative techniques need to be adopted. Extending the above idea, once the Rx starts demodulating the signal and making decisions, it can use those decisions as known data in order to successfully track the changes in nuisance parameters, such as a slowly changing carrier phase offset. This technique is known as *decision-directed synchronization*. It is evident that decision-directed approach can work well only when the detector decisions are correct such as in a high SNR case. Otherwise, a wrong decision leads to a poor estimate first, then a poor estimate leads to a wrong decision in the next cycle, and the chain continues in the form of error propagation.

Non-data-aided or Blind: In many wireless systems, neither a preamble nor the decisions are used and non-data-aided techniques are required for synchronization purpose. Blind techniques also prove useful in mobile communication systems where the receiver can lose the signal due to channel fading or blockage and a fast acquisition is required after the signal returns. Here, some particular characteristics of the incoming signal can be employed to estimate the unknown parameters. This is known as *non-data-aided or blind synchronization* technique. Adopting a non-data-aided synchronization approach has the advantage of maintaining the spectral efficiency. The drawback is its slow convergence because a large amount of data needs to be processed to average out the effects of noise as well as data randomness and find a reliable estimate.

Table 4.1: Synchronization approaches

	Benefits	Drawbacks
Data-aided	Accuracy, speed	Low spectral efficiency
Decision-directed	Accuracy	Error propagation
Non-data-aided	High spectral efficiency	Slow convergence

The benefits, drawbacks and conditions for these synchronization approaches are summarized in Table 4.1.

Feedback or Feedforward

Irrespective of the data knowledge, synchronization blocks can be implemented in one of the following two manners:

Feedforward, Open Loop, or Batch Processing: In data communication systems, there are many applications where the transmission occurs in a start and stop manner, i.e., a continuous transmission is not needed. This is known as *burst mode communication*. Here, fast acquisition is required, so a buffer is first filled with the samples of the received signal which are then processed to establish a direct one-shot estimate of the target parameter through batch processing. Signal processing to establish the expression for the estimate is based on an

algorithm derived from the mathematical structure of the Rx signal. Once this parameter is determined, it is corrected from the Rx signal without feedback to any previous block. In case of phase synchronization for example, the phase estimate can be used to de-rotate all data in that burst. In burst mode, it is common for the Tx to form a complete packet by inserting a sequence of known symbols – called a preamble or a training sequence – before the actual message symbols as shown in Figure 4.1a.

Feedback, Closed Loop or Iterative: Many other communication links work in *continuous mode* where the signal is transmitted either at all times or for a long duration. Here, the system can work in small steps by operating on each sample as it arrives, gradually converging towards the final solution. A fast acquisition is not as important (though still needed) and the objective is to lock onto the target parameter within a reasonable time after the arrival of the received signal. So an estimate of the error signal (for example, the phase error) is derived which forms the basis of a corrective signal that is fed back to a compensation unit. A Phase Locked Loop (PLL) can be employed for this purpose with some modifications discussed later. Feedback acquisition can work blindly, in a decision-directed manner or can also take help from training inserted periodically within the message as shown in Figure 4.1b. This category of processing has an inherent ability to automatically track slowly varying parameter changes.

In summary, there are $3 \times 2 = 6$ possible ways to implement a synchronizer depending on the knowledge of data and the loop being closed or open. Different algorithms can be designed for some of these topologies but not all, examples of which we will see throughout this text. For instance, a feedforward decision-directed estimate makes little sense.

4.2 Basic Components of a PLL

A Phase Locked Loop (PLL) is a device used to synchronize a periodic waveform with a reference periodic waveform. In essence, it is an automatic control system, an example of which is a cruise control in a car that maintains a constant speed around a given threshold. Although a PLL can be used for a variety of applications, it is enough for our purpose to treat it as a device that tracks the phase and frequency of an incoming sinusoid.

In a PLL, a control mechanism adjusts input signal to an oscillator according to a derived phase error such that the eventual phase error converges towards zero. We say that the phase of the output signal is *locked* to the phase of the input reference signal and hence it is called a *Phase Locked Loop*. In this text, we will focus on a discrete-time PLL which is a discrete-time system operating on a sampled input.

Note 4.1 PLL design and analysis

From a functional perspective, a PLL is the most important block in a digital communication system and hence it requires careful mathematical understanding and design. Usually this is done through application of Laplace Transform in continuous-time domain and z-Transform in discrete-time domain. However, for the sake of simplicity, this text treats just one transform, namely the Discrete Fourier Transform (DFT).

Therefore, in regard to PLL design and analysis, we will take some key results from

the literature without deriving them. This is due to our limitation of not covering the Laplace and z-Transforms. It should also be remembered that the design and analysis of a PLL does become mathematically intractable beyond an initial assumption of linearity and extensive computer simulations are needed in any case for its implementation in a particular application.

Let us start with a block diagram of a basic PLL shown in Figure 4.2. Assume that the discrete-time sinusoidal input to the PLL is given as

$$\text{input} = A \cos \left(2\pi \frac{k}{N} n + \theta[n] \right)$$

The PLL is designed in a way that the output is

$$\text{output} = \cos \left(2\pi \frac{k}{N} n + \hat{\theta}[n] \right)$$

where $\hat{\theta}[n]$ should be as close to $\theta[n]$ as possible after acquisition. This phase difference $\theta[n] - \hat{\theta}[n]$ is called **phase error** and is denoted by $\theta_e[n]$.

$$\theta_e[n] = \theta[n] - \hat{\theta}[n]$$

The phase error $\theta_e[n]$ computed at a certain time n is drawn in Figure 4.3 for a continuous-time signal.

The role played by each block in the PLL, as shown in Figure 4.2, is as follows.

Phase Error Detector (PED): A phase error detector determines the phase difference between a reference input waveform and a locally generated output waveform. In proportion to this phase difference, it generates an error signal denoted as $e_D[n]$.

Loop Filter: A loop filter sets the dynamic performance limits of a PLL and can be designed to successfully track only a constant phase or a change in input frequency as well. Moreover, it helps filter out noise and irrelevant frequency components generated in the phase error detector. The loop filter output signal is denoted as $e_F[n]$.

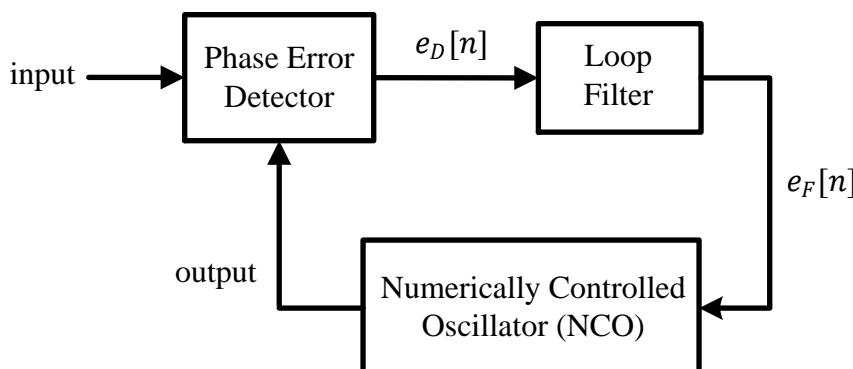


Figure 4.2: Basic structure of a PLL

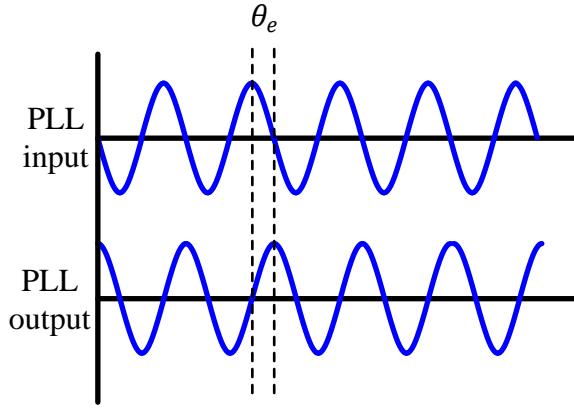


Figure 4.3: Phase difference between the PLL input and output is shown for a continuous-time signal for clarity

Numerically Controlled Oscillator (NCO): An NCO generates a local discrete-time discrete-valued waveform with a phase as close to the phase of the reference signal as possible. The amount of the phase adjustment during each step is determined by the loop filter output.

To avoid confusion among various phase error related terms discussed above, Table 4.2 lists these quantities along with their context.

Table 4.2: A summary of phase error related terms

$\theta_e[n]$	$\theta[n] - \hat{\theta}[n]$
$e_D[n]$	Phase error detector output
$e_F[n]$	Loop filter output

As a first step towards our understanding, assume that $\theta[n]$ in Figure 4.2 is zero and hence the frequency of the input signal is exactly $2\pi k/N$.

- After the acquisition, the following steps occur. The NCO operates at the same frequency as well and the phase error $\theta_e[n]$ is zero. Therefore, the phase error detector output $e_D[n]$ must ideally be zero. This leads to loop filter output $e_F[n]$ to be zero. With zero input, the NCO keeps operating at frequency $2\pi k/N$.
- Now assume that $\theta_e[n]$ was not zero at the start. The phase error detector would develop a non-zero output signal $e_D[n]$ which would rise or fall depending on $\theta_e[n]$. Subsequently, the loop filter would generate a finite signal $e_F[n]$ that would cause the NCO to change its phase in such a way as to turn $\theta_e[n]$ towards zero again.

Let us find out how this control mechanism adapts favorably in opposite direction to the input phase changes.

4.3 Phase Error Detector

The phase error detector is a device which outputs some function $f\{\cdot\}$ of the difference between the phase $\theta[n]$ of the PLL input and the phase $\hat{\theta}[n]$ of the NCO output. So the phase error detector output is written as

$$e_D[n] = f\{\theta[n] - \hat{\theta}[n]\} = f\{\theta_e[n]\} \quad (4.2)$$

The function $f\{\cdot\}$ is in general non-linear due to the fact that the phase $\theta[n]$ is embedded within the incoming sinusoid and is not directly accessible. A *phase equivalent representation* of such a PLL can be drawn by taking into account the phases of all sinusoids and tracking the operations on those phases through the loop. This is illustrated in Figure 4.4.

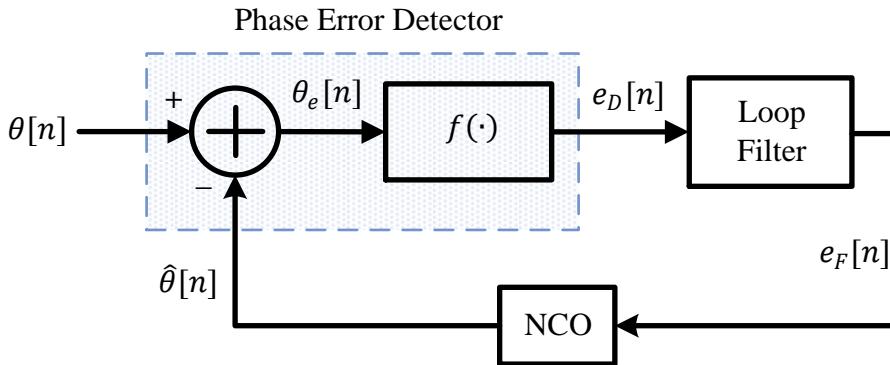


Figure 4.4: Phase equivalent PLL corresponding to the actual PLL in Figure 4.2

As mentioned before, the phase error detector output is in general a non-linear function $f(\cdot)$ of the phase difference between the input and output sinusoids. However, a vast majority of the PLLs in locked condition can be approximated as linear due to the following reason.

In equilibrium, the loop has to keep adjusting the control signal $e_F[n]$ such that the output $\hat{\theta}[n]$ of the NCO is almost equal to the input phase $\theta[n]$. So during a proper operation, the phase error $\theta_e[n]$ should go to zero.

$$\theta_e[n] = \theta[n] - \hat{\theta}[n] \rightarrow 0$$

To make this happen, *what should be the shape of the curve at the phase error detector output $e_D[n] = f(\theta_e[n])$?*

For finding the answer, first assume that $\theta_e[n]$ is positive and see what can make it go to zero.

$$\begin{aligned}
 \theta_e[n] > 0 &\implies \theta[n] - \hat{\theta}[n] > 0 \\
 &\implies \theta[n] > \hat{\theta}[n] \\
 \therefore \hat{\theta}[n] &\text{ should increase} \\
 &\implies e_F[n] > 0 \\
 &\implies e_D[n] > 0 \\
 &\implies f(\theta_e[n]) > 0
 \end{aligned} \tag{4.3}$$

Similarly, when $\theta_e[n] < 0$, it can be concluded that $e_D[n] = f(\theta_e[n])$ should be negative as well. From here, the phase error detector input/output relationship relationship usually turns out to be similar to what is symbolically drawn in Figure 4.5 for the *phase error θ_e* and *average phase error detector output Mean{ $e_D[n]$ }* $\overline{e_D}$. This kind of relationship is called an *S-curve* due to its shape resembling the English letter "S". In essence, it is a plot that shows how $\overline{e_D}$ depends on the actual phase difference. In Chapter 5, we will learn more about this shape and name.

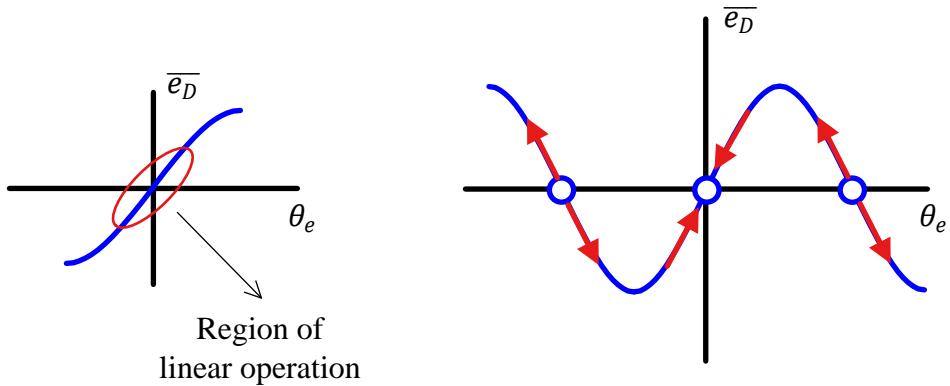


Figure 4.5: S-curve of the average phase error detector output $\text{Mean}\{e_D[n]\} \equiv \overline{e_D}$ (observe its shape like the letter "S") that gives rise to a stable locking point

Under steady state conditions, $\theta_e[n]$ hovers around the origin and hence $e_D[n] = f(\theta_e[n])$ also stays within the region indicated with the red ellipse in Figure 4.5. An extended typical S-curve is also drawn where one can observe that the PLL has the ability to pull back even a larger error $\theta_e[n]$. However, the steering force depends on the magnitude of $e_D[n]$ and the trend is not linear everywhere. It can be deduced that in the linear region of operation (a straight line relationship),

- a *positive slope* around zero produces a stable lock point, and
- a *negative slope* around zero does not generate a stable lock point, follow the logic in Eq (4.3) to see this point.

Within the small linear operating range, the PLL can be analyzed using linear system techniques. Around this region for small θ_e , the non-linear operation $f(\cdot)$ can be approximated as

$$f(\theta_e) \approx K_D \cdot \theta_e$$

where K_D is the slope of the line known as the gain of the phase error detector.

The phase equivalent loop for this linear model is shown in Figure 4.6, where the phase error detector now consists of just an adder and a multiplier: the difference between the input phase and the output phase is simply scaled by the gain K_D .

As we will find in later chapters, the phase error detector is the most versatile block in a PLL that leads to an extremely wide range of PLL designs. On the other hand, depending on a certain application, there are set rules for choosing a loop filter and an NCO which simplify the process to some extent. In this text, our main purpose to employ PLLs is to build phase and timing synchronization modules. Therefore, we will

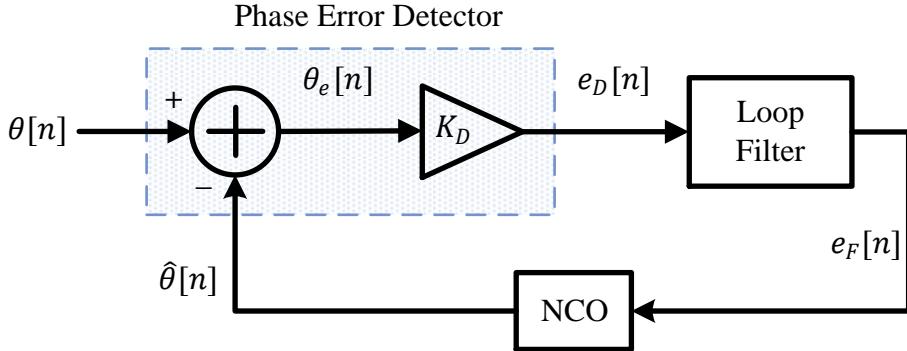


Figure 4.6: For small phase error $\theta_e[n]$, the phase error detector output $e_D[n]$ is a linear function of $\theta_e[n]$

devise several different kinds of phase error detectors while using the same loop filter and NCO for each PLL.

4.4 Proportional + Integrator (PI) Loop Filter

The loop filter in a PLL performs two main tasks.

1. The *primary task* of a loop filter that delivers a suitable control signal to the NCO is to establish the dynamic performance of the loop. Most PLL applications require a loop filter that is capable of driving not only a phase offset between the input and output sinusoids to zero, but also tracking frequency offsets within a certain range. This is known as fine frequency tracking.
2. A *secondary task* is to suppress the noise and high frequency signal components.

For this purpose, a *Proportional + Integrator (PI)* loop filter is most commonly used in PLL design. As the name suggests, a PI filter has a proportional and an integrator component, for which we refer to Figure 4.7. See Ref. [2] for the relevant derivation which employs the theory of Laplace Transform.

Proportional: The proportional term is a simple gain of K_P . To the filter output, it contributes a signal that is proportional to the filter input as

$$e_{F,1}[n] = K_P \cdot e_D[n]$$

Integrator: The integrator term is an ideal integrator with a gain of K_i . To the filter output, it contributes a signal that is proportional to the integral of the input signal. Or in discrete-time,

$$e_{F,2}[n] = e_{F,2}[n - 1] + K_i \cdot e_D[n]$$

Compared with Eq (2.34) in Section 2.6 on discrete-time integrators, it can be deduced that it executes forward difference integration to accumulate its input. The accumulation component is necessary to drive the steady-state error at the PLL output to zero in the presence of a fine frequency offset.

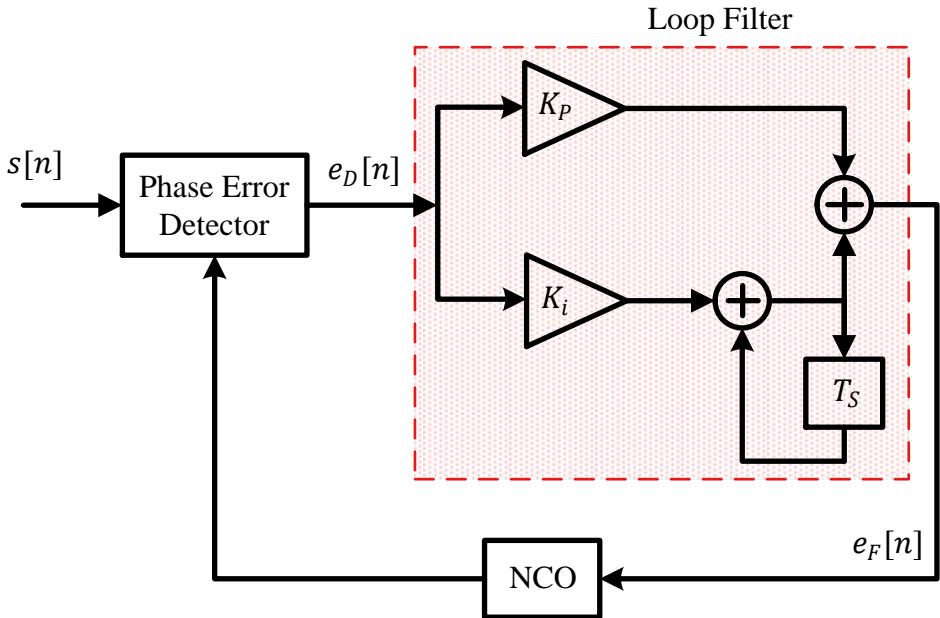


Figure 4.7: A discrete-time PLL with a Proportional + Integrator (PI) loop filter

Combining the proportional and integrator components leads to the loop filter output $e_F[n]$.

$$e_F[n] = e_{F,1}[n] + e_{F,2}[n]$$

When a PI filter is incorporated in the linear PLL model, we get the discrete-time PLL block diagram drawn in Figure 4.7. The notation T_S represents a delay of one sample time.

For the sake of completion, it is important to know that a PLL without a loop filter (understandably known as a first order PLL) is also used in some applications where noise is not a primary concern (Ref. [14]). Furthermore, a higher order loop filter can suppress spurs but increasing the order also increases the phase shift of such filters, thus making them prone to become unstable.

4.5 Numerically Controlled Oscillator (NCO)

The signal $e_F[n]$ forms the input as a control signal to set the phase of an oscillator. The name controlled oscillator arises from the fact that its phase depends on the amplitude of the input control signal. Some examples of controlled oscillators are Voltage Controlled Oscillator (VCO) and a Numerically Controlled Oscillator (NCO).

The oscillation frequency of a Voltage Controlled Oscillator (VCO) is controlled by its voltage input and hence it is an integral part of analog PLLs. As more and more functionality of the transceiver shifts towards the digital domain, the analog PLLs are seldom employed for waveform synchronization.

A Numerically Controlled Oscillator (NCO) creates a discrete-time waveform whose phase is steered by digital representation of a number at its input. In wireless commu-

nication devices, an NCO plays a central role in creating a digital version of a PLL for synchronization purpose. Shown in Figure 4.8, it has two main components.

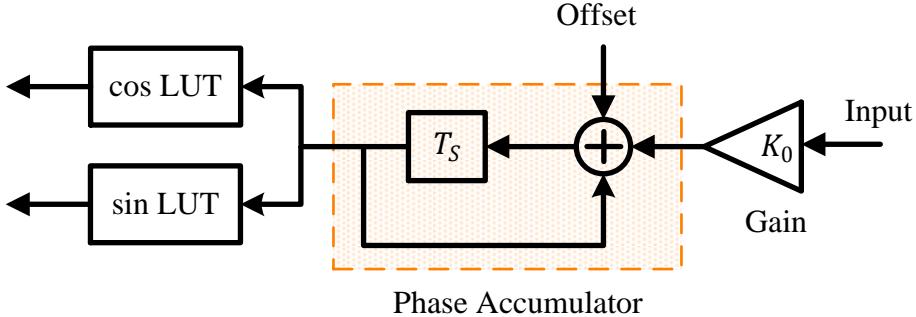


Figure 4.8: A block diagram of the NCO consisting of a phase accumulator and a Look-Up Table (LUT)

Phase Accumulator: The NCO adjusts its output phase $\hat{\theta}[n]$ based on its input signal $e_F[n]$ as

$$\hat{\theta}[n] = K_0 \sum_{i=-\infty}^{n-1} e_F[i]$$

where K_0 is a constant of proportionality known as the oscillator gain and summation to $n - 1$ comes from the output used by the phase error detector in a PLL. From this expression, we can see that an NCO acts as a phase accumulator. Note that the output can also be modified as

$$\begin{aligned} \hat{\theta}[n] &= K_0 \sum_{i=-\infty}^{n-1} e_F[i] = K_0 \sum_{i=-\infty}^{n-2} e_F[i] + K_0 \cdot e_F[n-1] \\ &= \hat{\theta}[n-1] + K_0 \cdot e_F[n-1] \mod 2\pi \end{aligned} \quad (4.4)$$

Compared with Eq (2.35) in Section 2.6 on discrete-time integrators, it can be deduced that an NCO executes backward difference integration to accumulate its input. Instead of using the frequency control word and phase control word here, we stick to the simpler input and offset terminologies. Unlike the analog VCO, the gain K_0 of the phase accumulator can be easily set to a fixed value, say 1.

Look-Up Table (LUT): In embedded wireless devices, the phase update $\hat{\theta}[n]$ from the integrator serves as an index into a Look-Up Table which stores the numeric values of a desired sampled waveform (such as a sine and a cosine). So the output can be computed as

$$\begin{array}{ll} I \rightarrow & s_I[n] = \cos \hat{\theta}[n] \\ Q \uparrow & s_Q[n] = \sin \hat{\theta}[n] \end{array}$$

Naturally, the size of the Look-Up Table determines the memory requirements as well as the amount of quantization on $\hat{\theta}[n]$, hence leading to a tradeoff between memory consumption and waveform approximation error. In most applications, a finer estimate is required to reduce this phase error noise which can

be generated through interpolation between the stored samples and a change in the LUT size is not needed.

With the inner workings of NCO available, a complete block diagram of a phase equivalent model of a PLL is now drawn in Figure 4.9. The notation T_S represents a delay of one sample time.

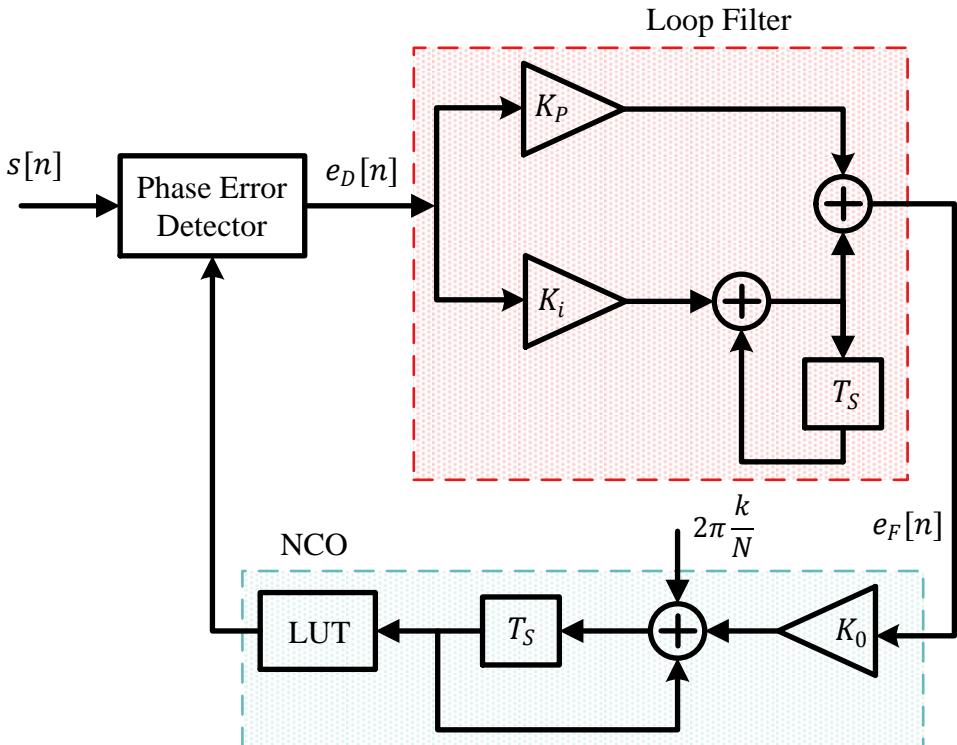


Figure 4.9: A discrete-time PLL with a PI loop filter and an NCO consisting of a phase accumulator and a Look-Up Table (LUT)

As stated earlier, it is easier to establish the kind of loop filter and NCO according to the desired PLL performance and then search for a suitable phase error detector. For the purpose of carrier phase synchronization here and in Chapter 5, we will continue to use a PI loop filter and an NCO for all different phase error detectors. For symbol timing synchronization in Chapter 7, the loop filter will remain the same while an interpolator along with an interpolator controller will be employed instead of an NCO due to the nature of the underlying problem.

Note 4.2 Build-measure-learn feedback loop

While the mathematical details are heavy, the concept of PLL correlates easily with real world problems. In his book *The Lean Startup*, Eric Ries introduced the build-measure-learn feedback loop as a guide for modern startups.

Build: Develop a minimum viable product.

Measure: Evaluate actionable metrics that reflect true cause and effect.

Learn: Decide which aspects of the product need to change and how much.

As you can see from Figure 4.9, this is exactly what a PLL accomplishes through the corresponding blocks of NCO (building a waveform), phase error detector (measuring the phase difference) and loop filter (learning the change). This is shown in Figure 4.10. Since electronics engineers and scientists in many other fields were already implementing such loops for several decades, his actual innovation was introducing this methodology to the world of software entrepreneurs. The main challenge from this perspective is to design a proper loop filter.

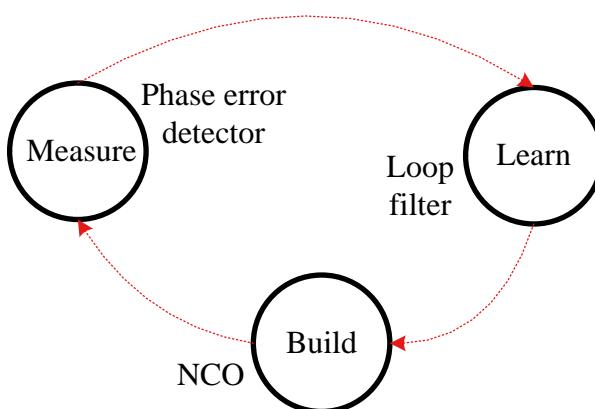


Figure 4.10: PLL seen as a build-measure-learn feedback loop

4.6 Designing a PLL

For the synchronization setup, the PLL response is determined by two parameters: damping factor ζ and natural frequency ω_n , which are taken from the standard control system terminology for a second order system. A description of ζ and ω_n is as follows.

Damping factor ζ : Imagine dropping a tennis ball on the ground. After hitting the ground, the ball bounces up to a distance, and repeats damped oscillations before finally settling in equilibrium. Similarly, a PLL phase acquisition process exhibits an oscillatory behavior at the start before settling towards a steady state. This oscillatory response can be controlled by a damping factor where a high damping factor corresponds to less bouncing and vice versa.

For a given input signal, a PLL behaves differently for different values of ζ . This is illustrated in Figure 4.11 for a unit step phase input (when the input is a unit impulse, the output is an impulse response and when the input is a unit step, the output is known as a step response).

- When $\zeta < 1$, the loop response exhibits damped oscillations in the form of overshoots and undershoots and the system is termed as *underdamped*.
- When $\zeta > 1$, the loop response is the sum of decaying exponentials, oscillatory behaviour disappears with large ζ and the system is *overdamped*.

- Finally, when $\zeta = 1$, the response is somewhere between damped oscillations and decaying exponentials and the PLL is termed as *critically damped*.

PLL output as a response to a step input

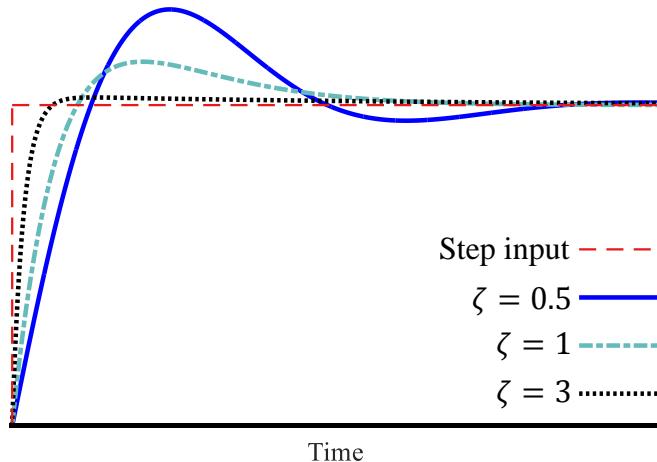


Figure 4.11: Step response of the PLL with a PI loop filter for different values of ζ

Natural frequency ω_n : We will shortly see that the PLL in tracking mode – which is the steady state achieved after settling into equilibrium – acts as a lowpass filter. In this role, the natural frequency ω_n can be considered as a coarse measure of the loop bandwidth.

PLL as a Lowpass Filter

The purpose of employing a PLL in a communications receiver is to track an incoming waveform in phase and frequency. This input signal is inherently corrupted by additive noise. In such a setup, a receiver locked in phase should reproduce this original signal adequately while removing as much noise as possible. The Rx uses a VCO or an NCO with a frequency close to that expected in the signal for this purpose. Through the loop filter, the PLL averages the phase error detector output over a length of time and keeps tuning its oscillator based on this average.

If the input signal has a stable frequency, this long term average produces very accurate phase tracking, thus eliminating a significant amount of noise. In such a scenario, the input to the PLL is a noisy signal while the output is a clean version of the input. We can say that when operating as a linear tracking system, *a PLL is a filter that passes signal and rejects noise*.

Note 4.3 Passband of a PLL

Having established the filtering operation of a PLL, we need to find out what kind of a filter a PLL is. For this purpose, consider the fact that within the linear region of

operation, the PLL output phase closely follows the input phase for small and slow phase deviations. On the other hand, it loses lock for large and quick input variations, thus necessitating a *lowpass frequency response*.

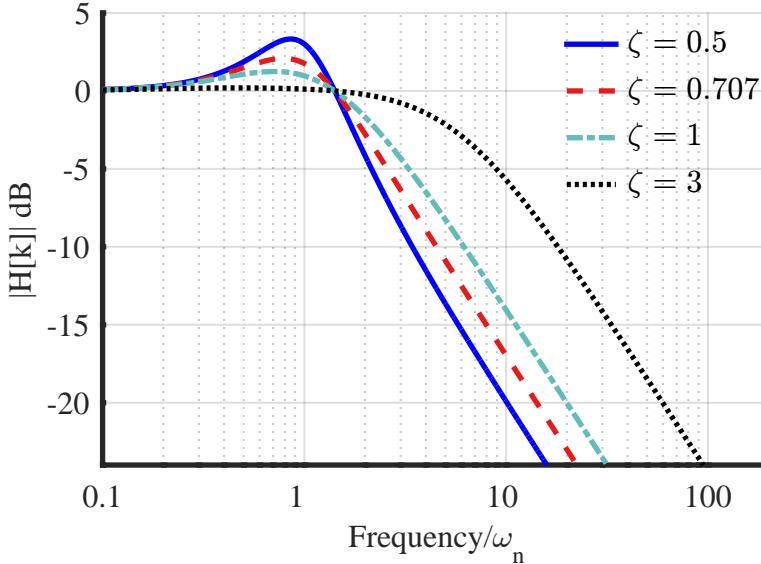


Figure 4.12: Frequency response of the PLL with a PI loop filter shows that it essentially acts as a lowpass filter

Figure 4.12 shows the frequency response of a PLL with PI loop filter: *it is indeed a lowpass filter*. Before we think of it having a sharp transition bandwidth, remember that the frequency axis is also drawn on a logarithmic scale. Moreover, the frequency scale is normalized to natural frequency ω_n which makes the curve valid for all second order PLLs.

The figure also reveals that the spectrum of this PLL as a lowpass filter is approximately flat between zero and ω_n . This implies that the PLL should be able to track phase and frequency variations in the reference signal as long as these variations remain roughly below ω_n .

By the same token, the bandwidth of this lowpass system varies with ω_n . However, a better definition of the bandwidth is needed because as shown in Figure 4.12, the loop frequency response strongly depends on ζ for the same ω_n . Therefore, a bandwidth measure known as the *equivalent noise bandwidth B_n* is used. Equivalent noise bandwidth B_n is defined as the bandwidth of an ideal brickwall filter which results in the same amount of total noise power (i.e., area under the curve) as that of the actual filter. This is drawn in Figure 4.13 where it is clear that a different value of ζ will produce a different B_n even for the same ω_n .

For a PLL acting as a lowpass filter with a PI loop filter, Ref. [14] relates the equivalent noise bandwidth B_n to natural frequency ω_n and damping factor ζ as

$$B_n = \frac{\omega_n}{2} \left(\zeta + \frac{1}{4\zeta} \right) \quad (4.5)$$

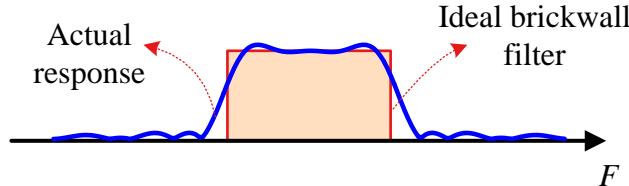


Figure 4.13: Defining the loop bandwidth

This is commonly known as the '*loop bandwidth*'.

Computing the Loop Constants

Designing a PLL in a software defined radio starts with defining the damping coefficient ζ and noise bandwidth B_n .

Damping factor ζ : As we have seen above, a large ζ results in no overshoots but long convergence time while a small ζ exhibits relatively fast convergence but damped oscillations. A good balance between the two is achieved with $1/\sqrt{2} = 0.707$ as a frequently used value. Typical values of ζ range from 0.5 to 2 in practical applications.

Loop bandwidth B_n : As we will see in Example 4.2, there is a tradeoff involved between choosing

- a small noise bandwidth that filters out most of the noise (and by extension the frequencies falling away from the passband), and
- a large noise bandwidth that can track fast phase variations, i.e., higher frequencies (and by extension allowing more noise to enter through the loop).

Both of the above objectives cannot be achieved simultaneously. However, a software radio based approach allows some relaxation as explained later in this chapter. For most cases of communication receivers, a B_n value around 1% of the input rate serves as a good starting point. This is why the recommended normalized loop bandwidth in GNU Radio documentation is around $2\pi/100$ that includes a normalization factor of 2π .

Next, having already chosen a PI loop filter, there are four constants that need to be determined: K_0 , K_D , K_P and K_i .

K_0 : In a discrete-time system, the NCO gain K_0 can be easily fixed to a suitable value, say 1.

K_D : The phase error detector gain K_D is computed according to the structure and resulting expression of the phase error detector, some examples of which we will see later in the following examples. Due to this dependence on the nature of phase error detector, K_D can be treated as a given parameter around which the rest of the loop is designed.

K_P, K_i : With K_0 and K_D established, the PI filter coefficients K_P and K_i remain the two unknowns in this framework and hence can easily be computed. Denoting the normalized natural frequency as

$$\theta_n = \omega_n \frac{T_S}{2}$$

we get from Eq (4.5)

$$\theta_n = \frac{B_n T_S}{\zeta + \frac{1}{4\zeta}}, \quad (4.6)$$

Next, the control systems theory sets the following relationships between the loop constants and loop parameters, see Ref. [2] for a complete derivation.

$$K_P = \frac{1}{K_D K_0} \cdot \frac{4\zeta\theta_n}{1 + 2\zeta\theta_n + \theta_n^2} \quad (4.7)$$

$$K_i = \frac{1}{K_D K_0} \cdot \frac{4\theta_n^2}{1 + 2\zeta\theta_n + \theta_n^2}$$

Notice that the PLL noise bandwidth is specified according to the sample rate $1/T_S$. However, symbol rate $1/T_M$ is a more appropriate parameter in digital communication systems for timing and carrier phase synchronization and hence noise bandwidth B_n can be specified in relation to $1/T_M$. Then, from $T_M = T_S/L$, where L is the number of samples/symbol, all the θ_n terms in Eq (4.7) need to be divided by L .

These are the equations we will use for computing the values for loop constants in specific PLL applications. In summary, a software PLL in an embedded device can be designed through the procedure outlined in Figure 4.14. We will now see how to set the values in a GNU Radio flowgraph.

Software PLL Design Procedure for a PI filter

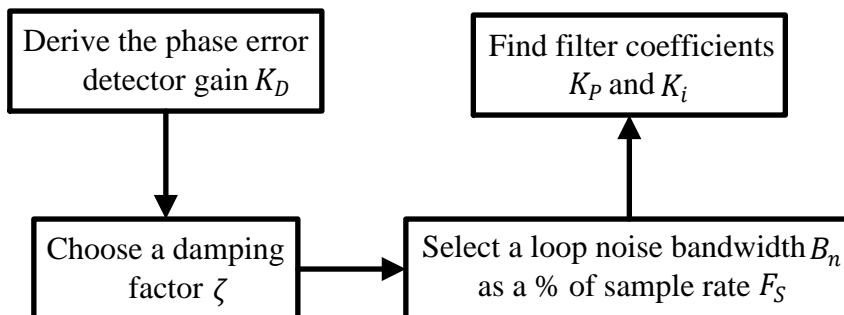


Figure 4.14: Design procedure for a software PLL with a PI loop filter

Exercise 4.1

While the code and documentation in an open source software project like GNU Radio is always evolving, I relate it to the above PLL parameters according to the current documentation. Most of what follows is based on my understanding from Tom Rondeau in Ref. [15].

- In most of the synchronization blocks, only the loop bandwidth is required. Therefore, the damping factor ζ is fixed to a certain value. The documentation of the block ‘Polyphase Clock Sync’ says that it is a critically damped system which implies that $\zeta = 1$.
- The NCO gain K_0 and detector gain K_D can be assumed to be 1 if the blocks work on the normalized values.
- Since typical values of loop noise bandwidth are very small, i.e., $B_n T_S \ll 1$, the expression $2\zeta\theta_n + \theta_n^2$ in the denominator of Eq (4.7) can be ignored, see Eq (4.6). With $\zeta = K_0 = K_D = 1$, we get from Eq (4.7)

$$\begin{aligned}K_P &= 4\theta_n \\K_i &= 4\theta_n^2\end{aligned}\tag{4.8}$$

From here, we can write

$$K_i = \frac{K_P^2}{4}$$

- In sync blocks, the parameter ‘alpha’ is the proportional gain K_P and the parameter ‘beta’ is the integral gain K_i of Eq (4.7). In some documentation (e.g., in the block ‘Polyphase Clock Sync’), the term ‘alpha’ is simply set as ‘gain’. With the above settings in place, we get

$$\begin{aligned}\text{‘alpha’} &= \text{‘gain’} \\ \text{‘beta’} &= \frac{\text{‘gain’}^2}{4}\end{aligned}\tag{4.9}$$

Therefore, the PLL design procedure in Figure 4.14 is not required in GNU Radio as it is simplified for the users to the extent of choosing the loop bandwidth only.

There is a common question on choosing a loop bandwidth that guarantees a certain acquisition time. As mentioned before, the loop can be started anywhere around $2\pi/100$ and tuned for the desired performance. However, PLL being a non-linear device, there are no exact closed-form solutions and the derivations in PLL textbooks set the general rules of thumb for this purpose. Next, we cover a few examples to demonstrate the phase tracking capability of a PLL and how different parameters influence its performance.

Example 4.1

Assume that the a PLL has to be designed such that it locks to a real sinusoid with discrete frequency $k/N = 1/15$ cycles/sample. Thus, the incoming sinusoid can be

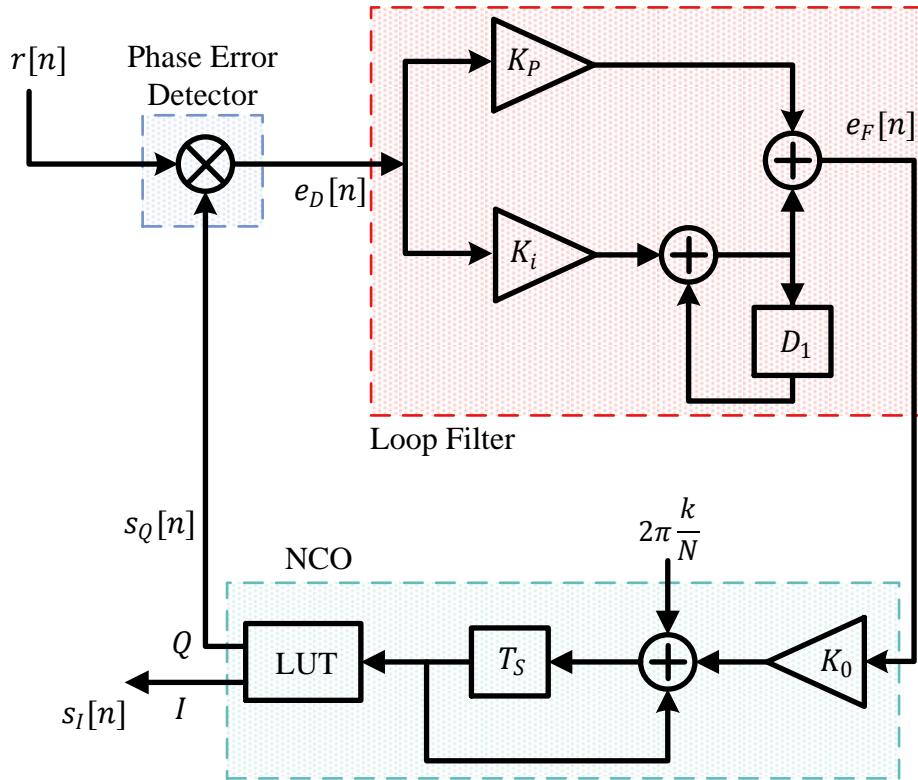


Figure 4.15: A discrete-time PLL with a phase error detector that computes the product between the input sinusoid and quadrature output of the NCO

written as

$$r[n] = A \cos \left(2\pi \frac{1}{15} n + \theta[n] \right)$$

where $\theta[n]$ can be a slowly changing phase. Here, we set a constant phase angle $\theta[n] = \pi$ that needs to be tracked. Such a large phase difference will enable us to clearly observe the PLL convergence process.

The block diagram for such a system is drawn in Figure 4.15. The NCO has one complex output, or two real outputs with an inphase and a quadrature component.

$$\begin{aligned} I &\rightarrow s_I[n] = \cos \left(2\pi \frac{k}{N} n + \hat{\theta}[n] \right) \\ Q &\uparrow s_Q[n] = -\sin \left(2\pi \frac{k}{N} n + \hat{\theta}[n] \right) \end{aligned}$$

Phase Error Detector

The phase error detector is a simple multiplier that forms the product between the input sinusoid and the quadrature component of the NCO output.

$$\begin{aligned} e_D[n] &= -\sin\left(2\pi \frac{k}{N}n + \hat{\theta}[n]\right) \cdot A \cos\left(2\pi \frac{k}{N}n + \theta[n]\right) \\ &= \frac{A}{2} \sin\left(\theta[n] - \hat{\theta}[n]\right) - \frac{A}{2} \sin\left(2\pi \frac{2 \cdot k}{N}n + \theta[n] + \hat{\theta}[n]\right) \\ &= \frac{A}{2} \sin\left(\theta_e[n]\right) + \text{double frequency term} \end{aligned}$$

where the identity $\cos(A)\sin(B) = 0.5 \{\sin(A+B)\} - \sin(A-B)\}$ has been used. The second term involving $2\pi(2 \cdot k/N)$ is the double frequency term that is filtered out by the loop filter. Therefore, the loop tracks the first term only, given by

$$e_D[n] \approx \frac{A}{2} \sin\left(\theta_e[n]\right)$$

The S-curve is the sine of θ_e which can be approximated using the identity $\sin A \approx A$ for small A . For this reason, the phase error detector output is approximately linear for steady state operation around the origin.

$$\begin{aligned} \overline{e_D} &= \frac{A}{2} \sin \theta_e \\ &\approx \frac{A}{2} \theta_e \quad \text{for small } \theta_e \end{aligned}$$

This S-curve is plotted in Figure 4.16.

Loop Constants

From the above equation, the phase error detector gain K_D is clearly seen to be

$$K_D = \frac{A}{2}$$

and hence is a function of sinusoid amplitude at the PLL input. Remember from Eq (4.7) that loop filter coefficients K_P and K_i involve K_D in their expressions. If the input signal level is different than what was expected and used in those calculations, or it keeps changing, the loop filter will have incorrect coefficients and the design will not perform according to the assumed noise bandwidth and damping factor. In a wireless receiver, this amplitude is maintained at a predetermined value by using an Automatic Gain Control (AGC).

For the purpose of this example, we assume that A is fixed to 1 and hence

$$K_D = 0.5$$

Next, we design a PLL with damping factor

$$\zeta = 1/\sqrt{2} = 0.707$$

and loop noise bandwidth $B_n = 5\%$ of the sample rate $1/T_S$, or

$$B_n T_S = 0.05$$

Thus, plugging these parameters in Eq (4.7) and ignoring the denominator, we get

$$K_P = 0.2667$$

$$K_i = 0.0178$$

After setting the rest of the parameters at the start of this example, the PLL can be simulated through any programming loop in a software routine that computes the sample-by-sample product of the loop input and quadrature loop output. Some results from an implementation according to Figure 4.15 are shown next.

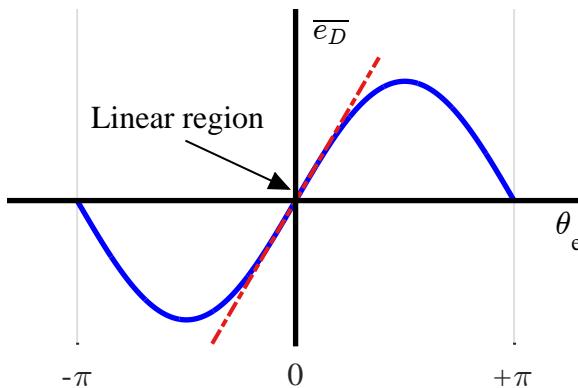


Figure 4.16: S-curve corresponding to the product phase error detector. The linear region of operation can be seen as red dashed line

Phase Error Detector Output $e_D[n]$

We start with the plot shown in Figure 4.17 displaying $e_D[n]$ that contains the following two components.

- A slowly varying average component hidden in $e_D[n]$ is shown in red. This can be thought of the true phase error to which the loop responds by converging to the incoming phase. Observe that this error stays positive for the first 27 samples, then goes negative before settling to zero at around 70 samples mark. Therefore, as we will see later, the NCO output should overcome the initial phase difference of π and track the input sinusoid after around 70 samples.
- A constant amplitude sinusoid with double frequency $2 \cdot 2\pi(k/N) = 2\pi(2/15)$ is shown in blue. Observe in Figure 4.17 that every 15 samples, there are 2 complete oscillations of this sinusoid riding on average error curve. This is more clearly visible towards the end of the curve where steady state is seen to be reached. Since $A/2 = 0.5$, the amplitude of this double frequency term approximately varies from around -0.5 to $+0.5$, resulting in a peak-to-peak amplitude of 1.

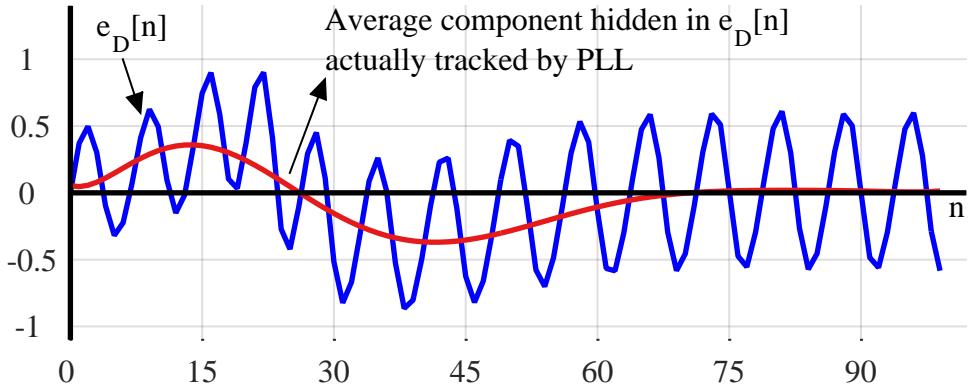


Figure 4.17: Phase error detector output $e_D[n]$ is composed of a slowly varying average component and a double frequency sinusoid due to product formation between input and output sinusoids

Loop Filter Output $e_F[n]$

Figure 4.18 displays the error signal $e_F[n]$ at the output of the loop filter. Similar kinds of trajectories are visible as in $e_D[n]$. However, the amplitude of double frequency term has been reduced from a peak-to-peak value of 1 in $e_D[n]$ to a peak-to-peak value of 0.3. This behaviour reinforces the lowpass characteristic of a PLL because the average curve is still the same while the double frequency term is attenuated as compared to Figure 4.17. Recall that the input sinusoid has a discrete frequency $k/N = 1/15$ which was chosen for the purpose of better visualization of error convergence. If we had chosen a higher frequency, the attenuation of double frequency term would have been different.

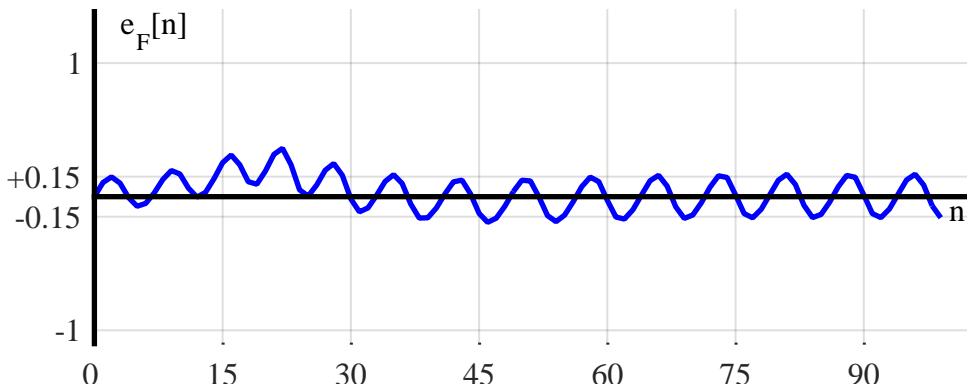


Figure 4.18: Output $e_F[n]$ at the output of loop filter. Similar trajectories are visible as in $e_D[n]$ but with attenuated amplitude

Phase Estimate $\hat{\theta}[n]$

The phase estimate $\hat{\theta}[n]$ is plotted in Figure 4.19. Just like $e_D[n]$ approaching zero after 70 samples, $\hat{\theta}[n]$ is seen to approach the initial phase difference π between the input and PLL output sinusoids at the same time. The oscillations due to double frequency term still remain.

Observe that the phase estimate does not directly converge to the actual value of π . Instead, its average value exhibits oscillatory behaviour by going beyond π , then returning and slowly oscillating around this value, which could have been clearer if the figure was extended to display more samples. This is due to the value chosen for the damping factor $\zeta = 0.707$.

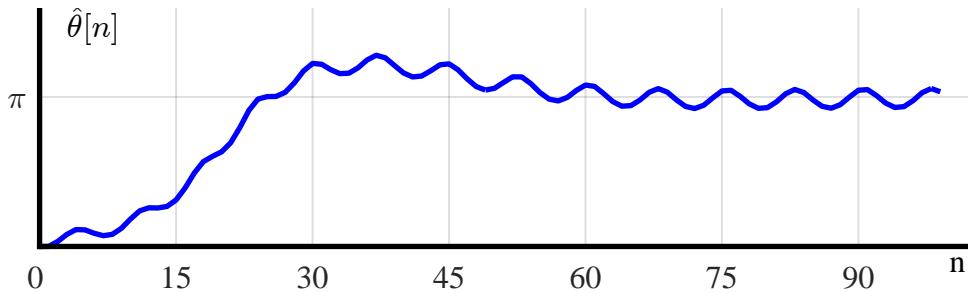


Figure 4.19: Phase estimate $\hat{\theta}[n]$ at the output of the NCO. It can be seen to slowly converging to π , which is the phase of the incoming sinusoid chosen in this example

PLL Output Sinusoid

Finally, the output inphase sinusoid $\cos(2\pi(k/N)n + \hat{\theta}[n])$ is shown along with the input sinusoid in Figure 4.20. Initially, there is a phase difference of π between them but gradually the PLL compensates for this difference and approaches the input sinusoid tracking successfully afterwards. This happens after 70 samples where $e_D[n]$ was seen to draw towards zero.

Interestingly, the little oscillatory behaviour shown by $\hat{\theta}[n]$ can be spotted here as well after convergence has reached, where the red dashed curve slightly leads and then slightly lags behind the input blue curve.

To understand this process, we can refer to the tennis ball analogy mentioned earlier. This oscillatory behaviour of a PLL and convergence towards the equilibrium point is similar to (but not exactly) a tennis ball dropped from a height. After hitting the ground, the ball bounces up to a distance, and repeats damped oscillations before finally settling in equilibrium, as shown in Figure 4.21. The height of the jump depends on the materials of the ball and the surface that drive its damping behaviour.

Similar to the above example, different PLLs can be designed for different phase detectors, damping factors ζ and loop noise bandwidths B_n and the results can be plotted to see how each parameter value affects the PLL behaviour. In the next example, we implement a PLL based on complex signal processing.

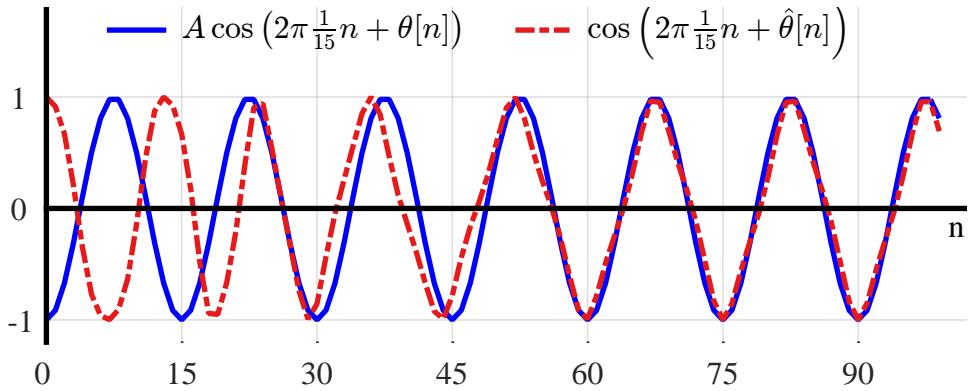


Figure 4.20: After 70 samples, the output inphase sinusoid $\cos(2\pi(k/N)n + \hat{\theta}[n])$ in red dashed line compensates for the initial phase difference and tracks the input sinusoid shown in blue

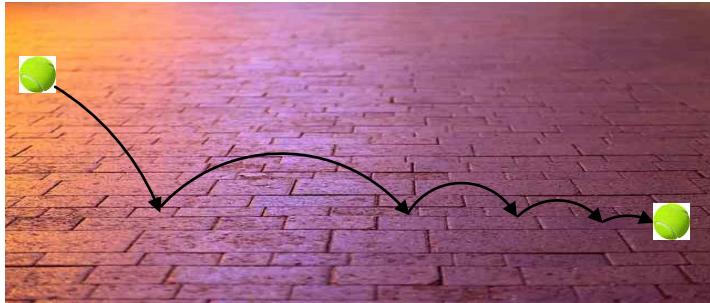


Figure 4.21: A tennis ball convergence towards equilibrium

Example 4.2

The PLL now has to be designed such that it locks to a complex sinusoid with discrete frequency $k/N = 1/15$ cycles/sample. Thus, the input sinusoid can be written as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} r_I[n] &= A \cos \left(2\pi \frac{k}{N} n + \theta[n] \right) \\ r_Q[n] &= A \sin \left(2\pi \frac{k}{N} n + \theta[n] \right) \end{aligned}$$

The block diagram for such a system is drawn in Figure 4.22. The NCO also has a complex output, or two real outputs with an inphase and a quadrature component, written as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} s_I[n] &= \cos \left(2\pi \frac{k}{N} n + \hat{\theta}[n] \right) \\ s_Q[n] &= -\sin \left(2\pi \frac{k}{N} n + \hat{\theta}[n] \right) \end{aligned}$$

Here, the phase detector first computes the product of the input and output complex sinusoids. Although the block diagram shows a simple product operator, remember that

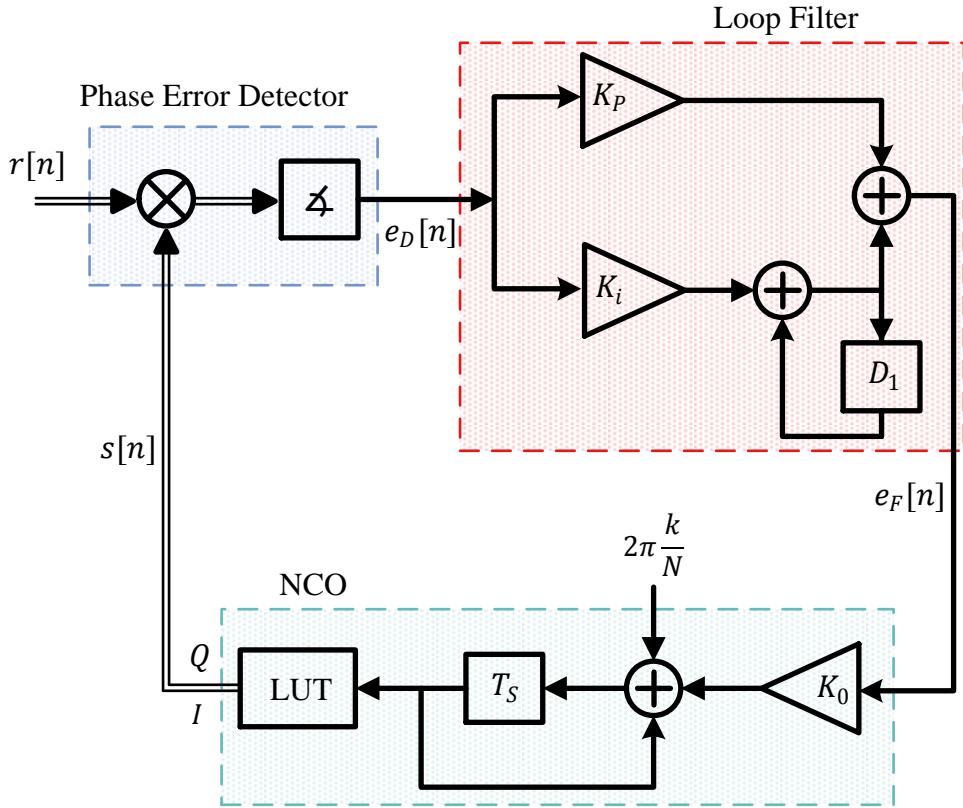


Figure 4.22: A discrete-time PLL with complex input and output

a complex product actually implements 4 real multiplications and 2 real additions. Using the trigonometric identities as before, the double frequency terms simply cancel out and the result of the product is

$$\begin{aligned} I &\rightarrow \{r[n] \cdot s[n]\}_I = \cos(\theta[n] - \hat{\theta}[n]) = \cos \theta_e[n] \\ Q &\uparrow \quad \quad \quad \{r[n] \cdot s[n]\}_Q = \sin(\theta[n] - \hat{\theta}[n]) = \sin \theta_e[n] \end{aligned}$$

Note the difference in complex signal processing: *the double frequency terms actually cancel out instead of being filtered out by the loop filter*. Furthermore, the phase of the above complex signal is precisely the error signal which needs to be extracted to form $e_D[n]$.

Accordingly, a four-quadrant inverse tangent as in Eq (1.9) is used to compute the phase of this complex signal at multiplier output^a. Therefore, the output of the phase detector is simply

$$e_D[n] = \angle(r[n] \cdot s[n]) = \theta_e[n]$$

for which the corresponding S-curve is drawn in Figure 4.23.

$$\overline{e_D} = \theta_e$$

Notice that the S-curve is linear over the whole range $-\pi \leq \theta_e \leq \pi$ and its expression implies that the phase detector gain $K_D = 1$.

^aSince the result of $\tan \angle = +Q/I$ is the same as $-Q/-I$ although the former angle is in quadrant I while the latter in quadrant III, individual signs of the arguments are needed to determine the correct quadrant of operation and accordingly correct angle. A similar argument holds to differentiate between $-Q/I$ and $+Q/-I$ to generate actual angles in quadrants IV and II, respectively. Eq (1.9) explains that adjustment.

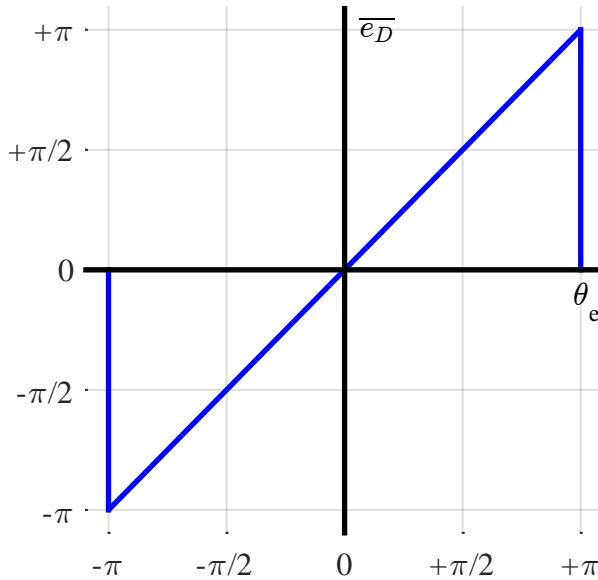


Figure 4.23: S-curve is linear in the whole range $-\pi \leq \theta_e \leq \pi$

Now we design three PLLs with different damping factors ζ and loop noise bandwidths normalized with sample rate $B_n T_S$ as follows.

Case 1 $\zeta = 1/2$ and $B_n T_S = 0.05$

Case 2 $\zeta = 3$ and $B_n T_S = 0.05$

Case 3 $\zeta = 3$ and $B_n T_S = 0.01$

Loop filter coefficients K_P and K_i can be found by plugging these values in Eq (4.7). Next, the PLL can be simulated as in the previous example and the phase detector output $e_D[n]$ as well as the phase estimate $\hat{\theta}[n]$ are plotted for these three PLLs in Figure 4.24 and Figure 4.25, respectively.

Here, some comments about acquisition and locking behaviour of a PLL are in order.

4.7 Comments on Locking and Acquisition

A complete study of a PLL design and performance involves an in-depth mathematical formulation including solutions to non-linear equations. Just like we took some key results for PLL design without deriving them, we next comment on some key parameters that govern its performance. We start with the parameters that specify the frequency

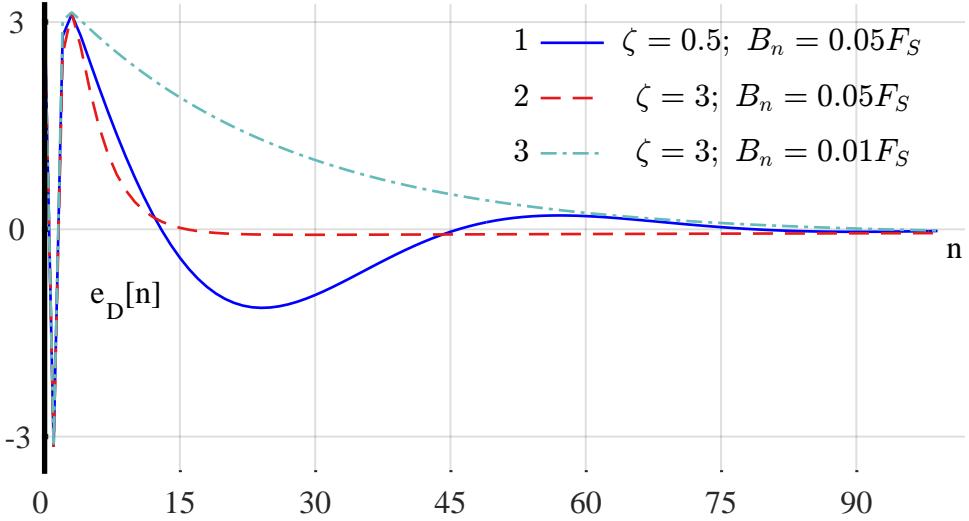


Figure 4.24: Phase detector output $e_D[n]$ for different damping factors ζ and loop noise bandwidths B_n . As opposed to an S-curve, the x-axis here is time n

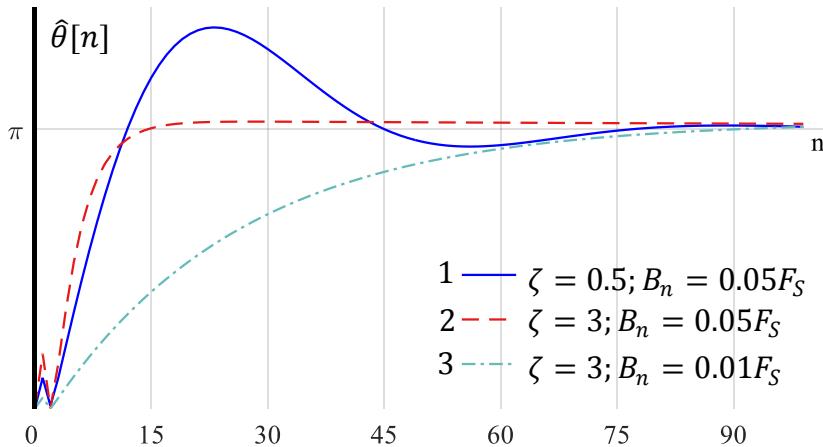


Figure 4.25: Phase estimate $\hat{\theta}[n]$ for different damping factors ζ and loop noise bandwidths B_n

range within which a PLL can be operated. The details of most of what follows are nicely explained in Ref. [14]. Keep in mind that other authors define these terms differently (e.g., lock range in place of hold range) and there is no set convention to name these ranges.

Hold range: Starting from a locked state, it is the frequency range in which the PLL is able to follow the input frequency variations. In other words, there is a critical value of the frequency offset between the input and output waveforms after which the slightest disturbance causes the PLL to lose phase tracking forever. This range, in which a PLL can statically maintain phase tracking, is known as **hold**

range F_H .

Pull-in range: When the frequency offset of the reference signal in an unlocked state reduces below another critical value, the buildup of the average phase error starts decelerating thus leading to an eventual system lock. This value is known as *pull-in range* F_P (and sometimes called capture range). The pull-in range is significantly smaller than the hold range. Even though the pull-in process itself is relatively slow, the PLL will always become locked for an offset within this range. In general, the pull-in range is proportional to the loop noise bandwidth.

$$F_P \propto B_n$$

For this reason, a PLL is able to not only track a phase offset in an input signal but a small frequency offset as well, an exact derivation of which is beyond the scope of this text.

Lock range: Obtaining a locked state within a short time duration is desirable in most applications. If the frequency offset reduces below another value – called the *lock range* F_L – the PLL becomes locked within one single-beat note between the reference and the output frequencies. The lock range is much smaller than the pull-in range; however on the up side, the lock-in process itself is much faster than the pull-in process.

Remember that the lock actually implies that for each cycle of the input, there is one and only one cycle of NCO output. Even with a phase lock, both steady phase errors and fluctuating phase errors can be present. In practical applications, the operating frequency range of a PLL is normally restricted to the lock range.

In summary, the hold range and the lock range are the largest and the smallest, respectively, while the pull-in range lies somewhere within the boundaries set by them. Thus, the following inequality holds.

$$F_H > F_P > F_L$$

Next, we describe two other important quantities which determine the suitability of a designed PLL for a given application.

Acquisition time: A PLL requires a finite amount of time to successfully adjust to the incoming signal and reduce the phase error to zero, which is called *acquisition time*. The acquisition time is given by the sum of the time to achieve frequency lock as well as that of the phase lock. It is inversely proportional to B_n , because a PLL with a large noise bandwidth lets a wider frequency range through its passband, consequently becoming able to track rapid variations and locking quicker than a PLL with narrower noise bandwidth. In Figure 4.24 and Figure 4.25, this can be noticed in a comparison between cases 2 and 3 with same $\zeta = 3$ but different loop noise bandwidths.

As a general rule, for a frequency offset F_Δ ,

$$T_{\text{acq}} \propto \frac{F_\Delta^2}{B_n^3} \quad (4.10)$$

while for a constant phase offset

$$T_{\text{acq}} \propto \frac{1}{B_n} \quad (4.11)$$

Tracking error: The performance of a PLL is also determined by the tracking error which is the power of the phase error signal. For a PLL in tracking mode (i.e., during linear operation), the noise power at the PLL input is given by AWGN with power spectral density N_0 [†] and the loop noise bandwidth B_n as

$$P_w = N_0 \cdot B_n$$

For a sinusoidal input with power P_s at the PLL input, the ratio P_s/P_w is the Signal-to-Noise Ratio (SNR). The expression for the tracking error ρ_{θ_e} is

$$\rho_{\theta_e} = \frac{N_0 B_n}{P_s} = \frac{P_w}{P_s} = \frac{1}{P_s/P_w} \quad (4.12)$$

Therefore, the tracking error in the presence of AWGN is inversely proportional to SNR and consequently directly proportional to B_n . It makes sense that a wider bandwidth allows a larger amount of noise at the PLL output, thus increasing the tracking error.

A Fast Lock Technique

From Eq (4.11) above, it is evident that selecting a large B_n in PLL design results in faster acquisition since the acquisition time is inversely proportional to a power of B_n . On the other hand, Eq (4.12) shows that a narrow B_n generates less tracking error since it is directly proportional to B_n . In conclusion, a good PLL design balances the conflicting criteria of fast acquisition time and reduced tracking error.

In the world of hardware radio, PLL designers had to balance these two performance criteria by finding an acceptable compromise, or switch in hardware from an acquisition stage to a locked stage. The realm of software radio offers a better solution due to our ability to change the code on the fly which is explained below.

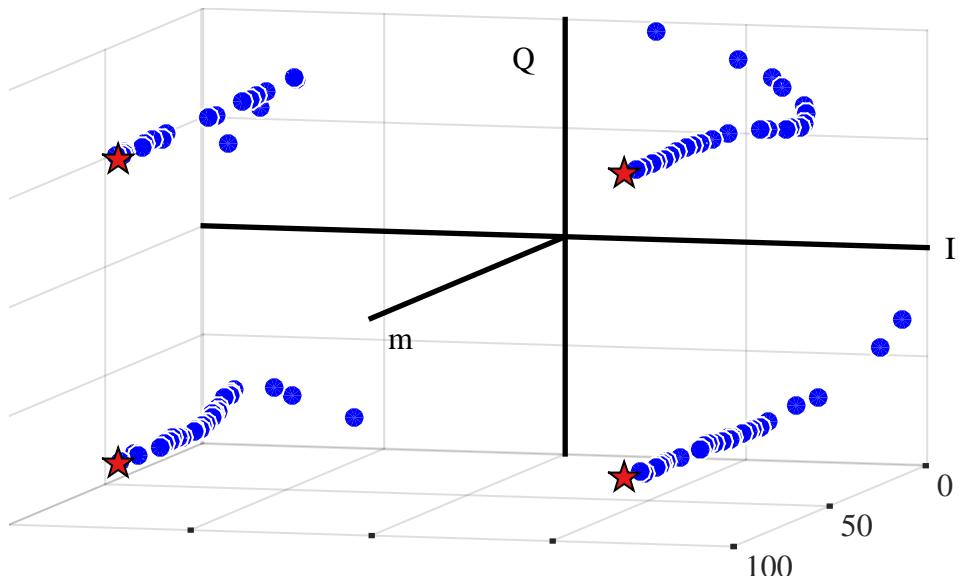
In an unlocked state, the PLL noise bandwidth B_n is made large so that it can achieve fast lock. In parallel, a certain algorithm known as a *lock detector* is run which generates a binary output depending on whether the PLL has acquired lock or not. When the lock detector raises a positive flag, the values of the loop constants are changed in steps such that the PLL bandwidth B_n is gradually reduced to a smaller value, since the loop constants are reconfigurable in this scenario.

In subsequent chapters, we discuss carrier and timing synchronization procedures in a communications receiver. These blocks incorporate a PLL as an integral component.

[†]Power spectral density of additive white Gaussian noise was defined in Section 2.8.

Chapter 5

Carrier Phase Synchronization



“A man with a watch knows what time it is. A man with two watches is never sure.”

Segal's Law

In Section 3.7, we discussed that a wireless communication system is allocated a certain bandwidth around a higher carrier frequency so that the generated baseband signal can be shifted to that specific portion of the spectrum. Such signals are called *passband signals*. For passband modulation schemes like QAM and PSK, we assumed in their detection process explained in Section 3.7 that the phase and frequency of the local oscillator at the Rx is in perfect frequency and phase alignment with the carrier of the incoming signal. This is not true in practice and specific subsystems are designed at the Rx to operate in frequency and phase synchronization with the received signal.

The first issue in this regard is the origin of this phase offset in the modulated signal. A phase offset here is introduced because the information at the Tx side is mapped onto the modulation symbols according to a certain time reference. This time reference establishes the phase reference of the Tx oscillator as well and is unknown to the Rx. Naturally, the Rx has its own time reference too which establishes the phase reference for its own oscillator used during the downconversion process. Furthermore, the overall channel impulse response – that includes all the analog and digital filters at the Tx and Rx with their magnitude and phase responses – induces a phase offset to the signal as well. Finally, the propagation delay from the Tx to the Rx brings one more term to this equation and the final phase offset rises from a combination of all the above factors.

A phase offset is the simplest of signal distortions which is why we discuss it first. The operation of *phase synchronization* is also known as *phase recovery*. In this chapter, we explain the process of phase synchronization employing Feedforward (FF) and Feedback (FB) techniques. Remember that a Phase Locked Loop (PLL), which is a feedback system, can cope with small (or fine) frequency offsets as well. Therefore, here we assume that most of the frequency offset has already been compensated for, while the discussion about acquiring the signal in the presence of large frequency offsets is kept for Chapter 6.

In addition, the Rx sample rate F_S should also be synchronous to the Tx symbol rate $1/T_M$. However, due to the local generation of Rx sample clock, these two rates are not commensurate to each other. Since timing synchronization is explained in detail in Chapter 7, it is assumed here that the Tx and Rx sampling clocks are already synchronized.

It is also worth noting that there are certain modulation schemes such as Differential Phase Shift Keying (DPSK) which do not require a phase reference for symbol demodulation. For example, the demodulators in such systems, known as *non-coherent demodulators*, compare the phase of a symbol with the previous symbol to cancel the channel induced phase offset. In this process, noise power gets doubled as well which is why coherent systems that employ a phase recovery system have a superior performance as compared to non-coherent systems and are more desirable to implement wherever possible.

Next, we discuss the effect of carrier phase mismatch between the received signal and the local oscillator on the detection mechanism.

5.1 Effect of Carrier Phase Mismatch

In case of QAM and other passband modulation schemes, a Rx has no information about carrier phase of the Tx oscillator. We start with Figure 5.1 to clarify the signal notations used at various stages of the transceiver.

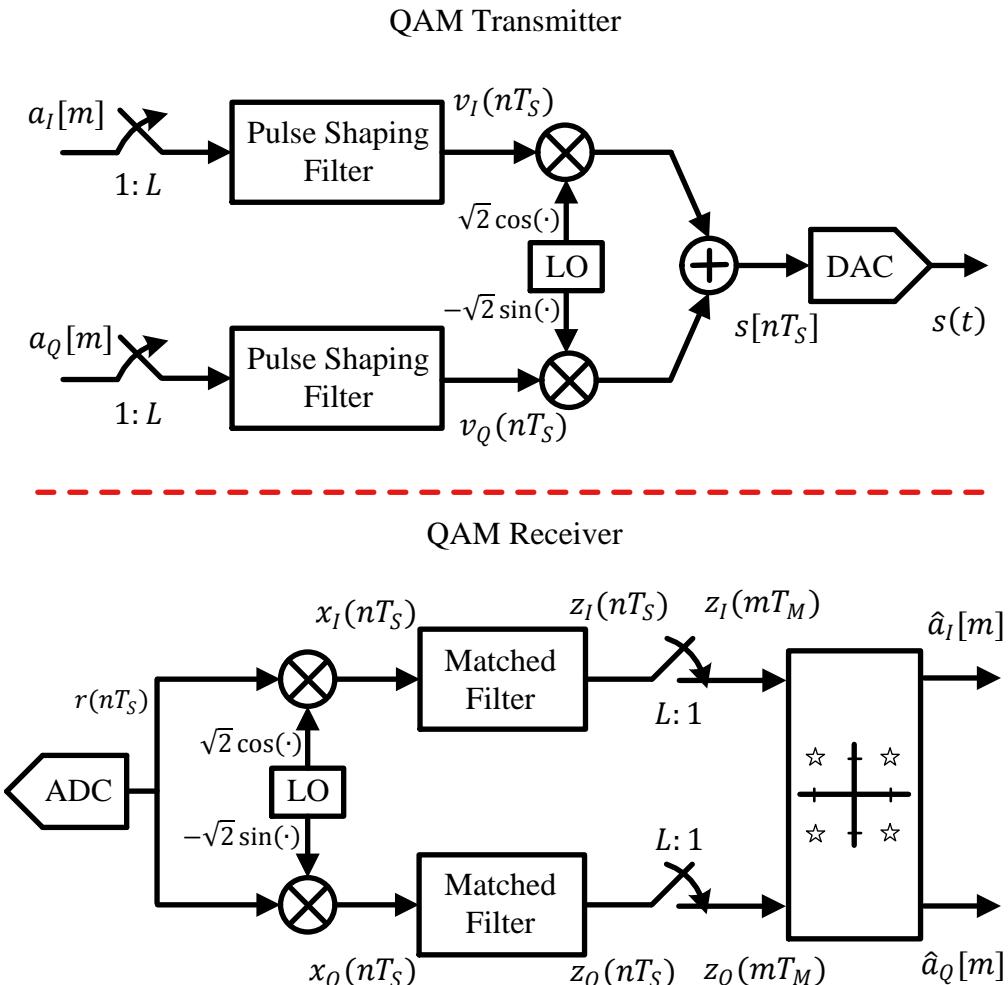


Figure 5.1: A block diagram of a QAM system to explain the signal notations at various blocks

To see the effect of this carrier phase offset, consider that a transmitted passband signal consists of two PAM waveforms in I and Q rails denoted by $v_I(t)$ and $v_Q(t)$ respectively. These are combined as

$$s(t) = v_I(t)\sqrt{2} \cos(2\pi F_C t + \theta_\Delta) - v_Q(t)\sqrt{2} \sin(2\pi F_C t + \theta_\Delta) \quad (5.1)$$

where the angle θ_Δ is the phase difference between the incoming carrier and the local oscillator at the Rx, F_C is the carrier frequency and $v_I(t)$ and $v_Q(t)$ are the continuous

versions of sampled signals $v_I(nT_S)$ and $v_Q(nT_S)$ given by

$$\begin{array}{ll} I \rightarrow & v_I(nT_S) = \sum_i a_I[i] p(nT_S - iT_M) \\ Q \uparrow & v_Q(nT_S) = \sum_i a_Q[i] p(nT_S - iT_M) \end{array} \quad (5.2)$$

In the above equation, $a_I[i]$ and $a_Q[i]$ are the inphase and quadrature components of the i^{th} symbol, $p(nT_S)$ are the samples of a square-root Nyquist pulse while T_S and T_M are the sample time and symbol time, respectively.

As derived in Appendix 5.9, the symbol-spaced matched filter output in the presence of phase offset θ_Δ is given as

$$\begin{array}{ll} I \rightarrow & z_I(mT_M) = a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \\ Q \uparrow & z_Q(mT_M) = a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \end{array} \quad (5.3)$$

From the anticlockwise phase rotation rule of complex numbers $I \cos \theta - Q \sin \theta$ and $Q \cos \theta + I \sin \theta$, we know that this expression is nothing but anticlockwise rotation of the matched filter output by an angle θ_Δ .

In conclusion, *a mismatch of θ_Δ between incoming carrier and Rx oscillator rotates the desired data symbols $a[m]$ on the constellation plane by an angle θ_Δ* . This is drawn in the constellation plot of Figure 5.2 for a 4-QAM modulation scheme.

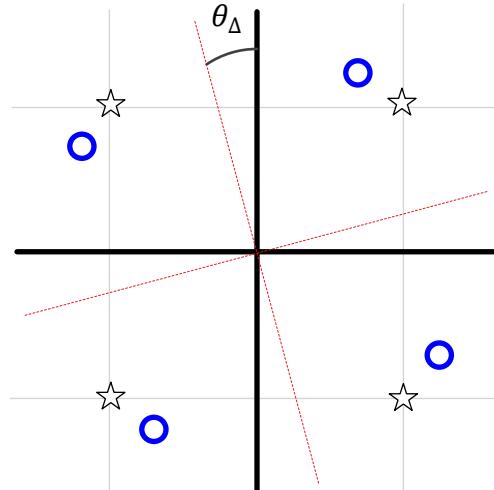


Figure 5.2: A 4-QAM constellation in the presence of carrier phase offset θ_Δ in a noiseless case

In polar form, the expression in Eq (5.3) can be written as

$$\begin{aligned} |z(mT_M)| &= \sqrt{a_I^2[m] + a_Q^2[m]} \\ \angle z(mT_M) &= \angle a[m] + \theta_\Delta \end{aligned} \quad (5.4)$$

where $a[m]$ is the m^{th} constellation symbol comprising of $a_I[m]$ and $a_Q[m]$ as inphase and quadrature components, respectively.

Note 5.1 Cross-talk

In Eq (5.3), start with $\theta_\Delta = 0$ and observe that the I and Q outputs are $a_I[m]$ and $a_Q[m]$, respectively. This implies that signals in I and Q arms are completely independent of each other. Gradually increasing θ_Δ has two effects:

1. Since $\cos \theta_\Delta < \cos 0 = 1$, amplitude of $a_I[m]$ in $z_I(mT_M)$ reduces. The same phenomenon happens with $a_Q[m]$ in $z_Q(mT_M)$.
2. Since $\sin \theta_\Delta > \sin 0 = 0$, interference of $a_Q[m]$ in $z_I(mT_M)$ increases as well as that of $a_I[m]$ in $z_Q(mT_M)$.

This interference between I and Q components is known as cross-talk. Cross-talk increases with θ_Δ until for a 90° difference, $a_I[m]$ appears at Q output and $-a_Q[m]$ at I output.

The effect of this cross-talk on a Raised Cosine shaped 4-QAM waveform with excess bandwidth $\alpha = 0.5$ is shown in Figure 5.3 for a phase difference $\theta_\Delta = 30^\circ$. Observe the first sample: from Figure 5.3a, it is $(-1, +1)$ in quadrant II. After phase rotation, I part moved towards left thus increasing its amplitude and Q moved downwards reducing its amplitude. This is evident through the first samples in Figure 5.3b. A similar argument holds for all other symbols.

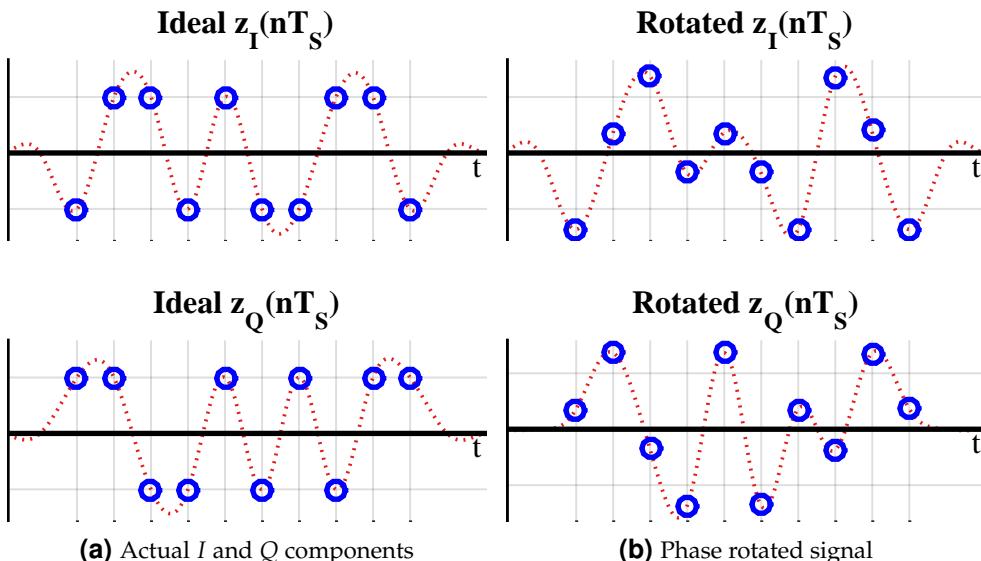


Figure 5.3: Effect of 30° phase rotation on a time domain 4-QAM waveform for a Raised Cosine filter with excess bandwidth $\alpha = 0.5$. Observe how the samples at optimal locations move away from the ideal amplitudes

Something interesting has happened in Figure 5.3. Notice that although the amplitude has decreased for some symbols, it has *risen* for some other symbols. This is the outcome of a circular rotation and the average effect is detrimental.

Scatter Plot with Phase Rotation

What was discussed above can be extended to the whole symbol stream. The cumulative effect of a phase offset is straightforward to see in a scatter plot, see Figure 5.2 for QPSK modulation, keeping in mind that now the blue circles are not one but several symbols mapped over one another due to a similar phase shift.

In the case of BPSK modulation, the received and sampled signal is expressed as

$$r(nT_S) = v_I(nT_S)\sqrt{2} \cos\left(2\pi\frac{k_C}{N}n + \theta_\Delta\right) + \text{noise}$$

where k_C corresponds to carrier frequency F_C and $v_I(nT_S)$ is the inphase waveform

$$I \rightarrow \quad v_I(nT_S) = \sum_i a_I[i] p(nT_S - iT_M)$$

Similar to the analysis in Section 5.1, a carrier phase mismatch θ_Δ distorts the matched filter output with $a_Q[m]$ being all zeros in this case. Following from Eq (5.3) with $a_Q[m] = 0$,

$I \rightarrow$	$z_I(mT_M) = a_I[m] \cos \theta_\Delta$	(5.5)
$Q \uparrow$	$z_Q(mT_M) = a_I[m] \sin \theta_\Delta$	

We can say that a phase error θ_Δ attempts to rotate the constellation in an anticlockwise manner. However, since there is no Q arm in the detector, *it only succeeds in projecting the inphase part of the result on the I arm*. The overall effect is that the downsampled matched filter output $z_I(mT_M)$ shifts on the I -axis by an amount equal to cosine of the phase difference times the inphase component. This is drawn in Figure 5.4 where a hypothetical QPSK constellation is also shown for comparison. For example, a simple phase error of 30° causes a signal loss of

$$20 \log_{10} \cos \theta_\Delta = 20 \log_{10} \cos 30^\circ \cdot \frac{\pi}{180} = 1.25 \text{ dB}$$

An increased amount of energy per bit needs to be provided to make up for such errors when a specific bit error rate performance needs to be delivered. We conclude that there is a loss of amplitude in BPSK when only the I branch is employed for detection purpose (recall that there is no magnitude reduction in QPSK for a phase error). For phase errors close to 90° , this can completely wipe out the signal.

Eye Diagram with Phase Rotation

Since the scatter plot is different than a raw time domain waveform, we employ the eye diagram to examine the effects of carrier phase offset (say, on an oscilloscope). First, start with a BPSK modulation scheme and remember that there is no Q channel in this case and consequently no cross-talk. However, the effect of phase rotation is a reduction in signal amplitude which was seen through Eq (5.5).

With a phase offset of 45° , the I branch loses half of its energy with the remaining half going in the Q arm. This is drawn in Figure 5.5. In fact for a 90° phase rotation, the I contribution actually reaches zero and all the energy of the signal appears across

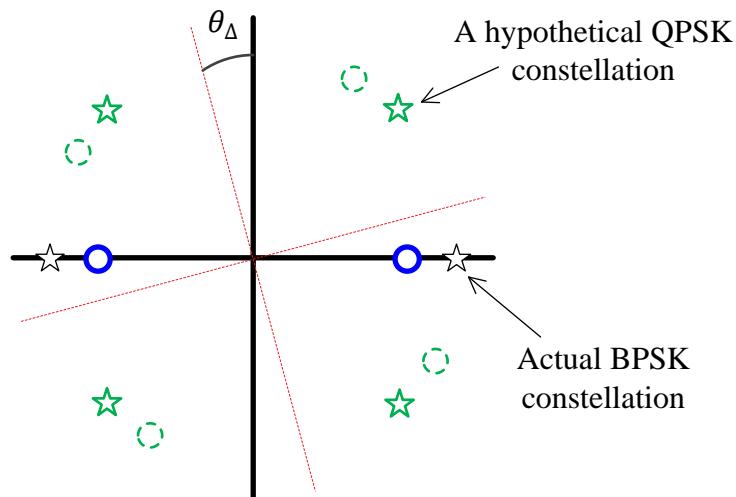


Figure 5.4: A BPSK constellation in the presence of carrier phase offset θ_Δ in a noiseless case. A hypothetical QPSK constellation is also shown for a comparison with QPSK

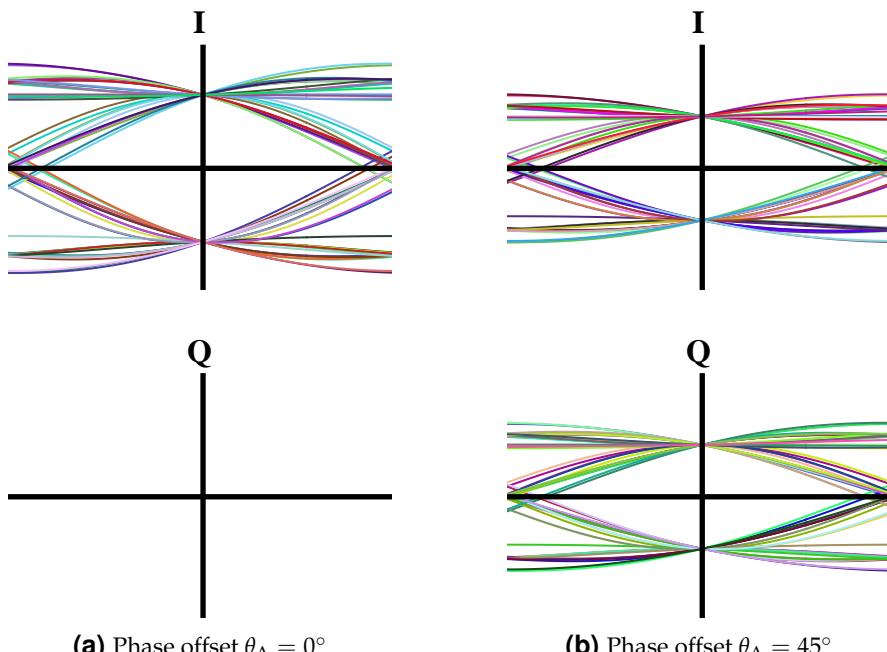


Figure 5.5: Eye diagrams of a BPSK signal for 0° and 45° phase rotations and a Raised Cosine filter with excess bandwidth $\alpha = 0.5$. Observe a reduction in I amplitude in proportion to the energy rising in the Q arm

the Q branch. Due to this reason, we will see later that the Q arm is still employed for BPSK signals – not for data detection but helping in the phase synchronization

procedure.

Next, we turn our focus towards QAM and observe the amplitude change and cross-talk between I and Q branches for three different phase offsets: 15° , 30° and 45° . Figure 5.6 illustrates the I channel for these phase offsets in a noiseless case and a 4-QAM signal. A similar diagram holds for Q arm as well and not drawn here. The optimal sampling instants are still visible due to zero noise but *the eye diagram looks more like a 4-PAM signal* than that of a single 4-QAM signal due to the cross-talk from Q arm. It is also evident that I and Q affect each other in equal proportions.

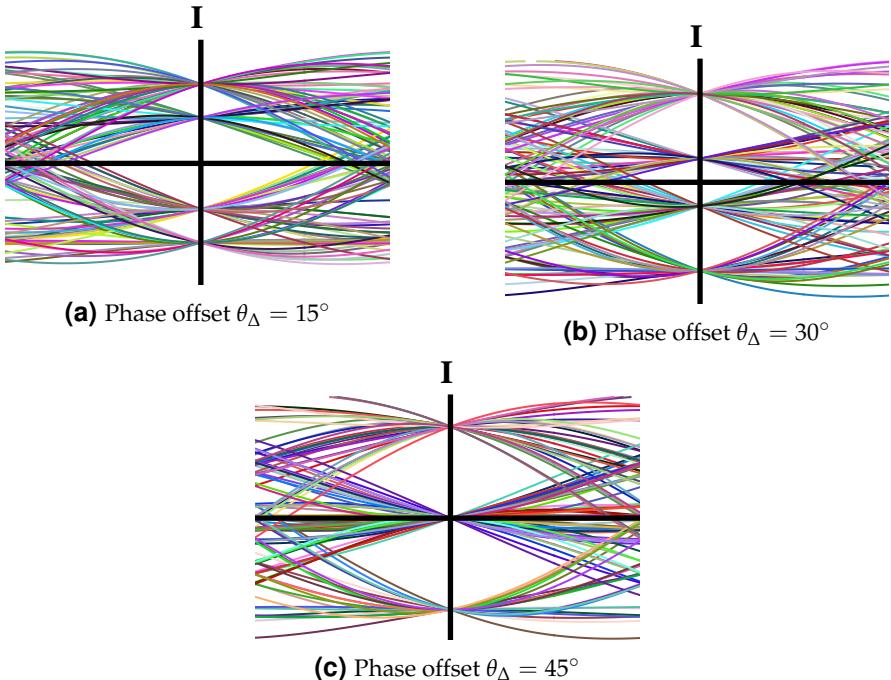


Figure 5.6: Eye diagrams for I arm of a 4-QAM signal for 15° , 30° and 45° phase offsets and a Raised Cosine filter with excess bandwidth $\alpha = 0.5$. A similar eye diagram exists for Q arm as well

The reason there are only two eyes for 45° phase offset is that $\cos \theta_\Delta$ and $\sin \theta_\Delta$ in Eq (5.3) become equal. From a scatter plot point of view, a rotation of 45° shifts the constellation points onto the inphase and quadrature axes. For the I plot shown here, the output at the sampling instant coincides only with a positive or negative symbol value while Q projection is zero in both cases.

In conclusion, as a result of phase rotation, the matched filter outputs do not map back perfectly onto the expected constellation, even in the absence of noise and any other distortion. Unless this rotation is small enough, it causes the symbol-spaced optimal samples to cross the decision boundary and fall in the wrong decision zone. And even for small rotations, relatively less amount of noise can cause decision errors in this case, i.e., noise margin is reduced. In fact, for higher-order modulation, the rotation seen in Figure 5.2 becomes even worse because the constellation points are

closely spaced with each other for the same Tx power.

5.2 The Fundamental Problem of Synchronization

Now we discuss the fundamental problem of synchronization whether it is carrier phase, frequency or timing. Subsequently, we develop a systematic methodology to overcome this problem that in this chapter leads to phase estimation algorithms.

The Problem

The task of a phase synchronization unit in a Rx is to estimate the phase offset and de-rotate[†] the matched filter outputs by this estimate either in a feedforward or a feedback manner. In such a way, the carrier at the Rx can be said to become synchronized with the carrier used at the Tx. For this purpose, we need to analyze the Rx phase as follows.

Note 5.2 Breakdown of the Rx phase

From Eq (5.4) reproduced below, the modulated signal arriving at the Rx consists of two different phase components.

$$\angle z(mT_M) = \angle a[m] + \theta_\Delta$$

1. Unknown phase shifts arising due to modulating data occurring at symbol rate $1/T_M$. For example, in BPSK modulation, phase changes by 0° or 180° *at the boundary of each symbol interval T_M* . For QPSK, there are four different phase shift possibilities, i.e., 45° , 135° , -135° , or -45° . A similar argument holds for QAM as well.
2. Unknown phase difference between Tx and Rx local oscillator, θ_Δ .

Imagine a QPSK signal input directly into a black box designed to estimate and compensate for the phase and frequency of a sinusoid (a PLL for example). Recall from Eq (3.41) that we can write a passband QAM waveform in polar form as

$$s(t) = \sum_m \sqrt{a_I^2[m] + a_Q^2[m]} \cdot p(t - mT_M) \cdot \sqrt{2} \cos \left(2\pi F_C t + \tan^{-1} \frac{a_Q[m]}{a_I[m]} \right)$$

In this polar form, we have a single sinusoid whose amplitude is fixed for PSK modulation and phase is determined by symbols $a_I[m]$ and $a_Q[m]$. Depending on its design parameters, the mechanism implemented in the black box (e.g., a PLL) will try to lock onto the incoming phase within a specific time duration. However, instead of being a regular sinusoid, *the signal phase is jumping around by 0° , $\pm 90^\circ$ or 180° at every symbol boundary* due to the modulating data, never allowing our mechanism to converge. This is the fundamental problem every synchronization system needs to address and is illustrated in Figure 5.7.

The Recovery Stage

Two remarks are in order now.

[†]Rotation of a complex number is defined with respect to an anticlockwise fashion. Thus, *de-rotation* implies rotating the input in a clockwise direction.

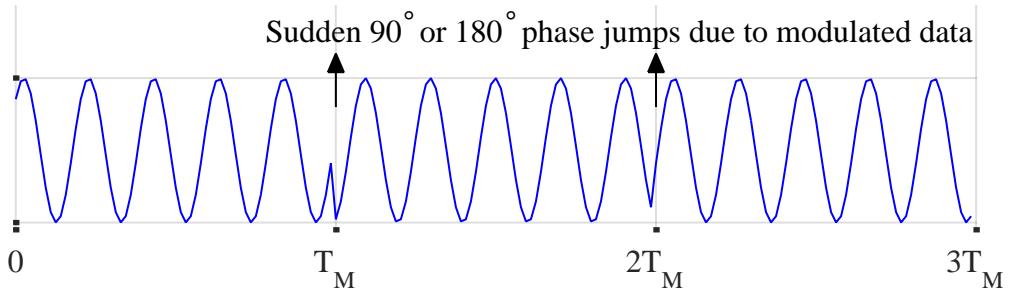


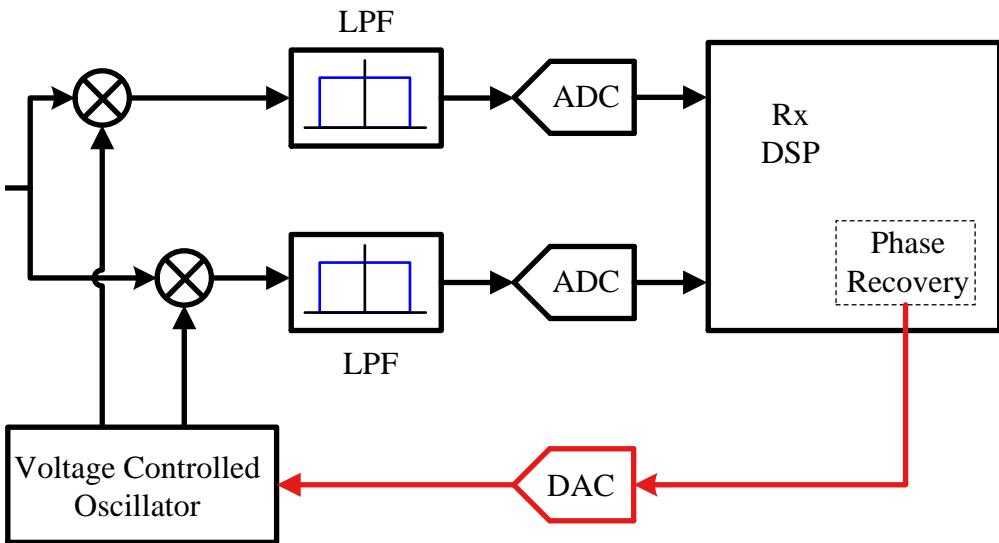
Figure 5.7: Fundamental problem of synchronization: 0° , $\pm 90^\circ$ or 180° phase jumps at each symbol time from the modulating data onto the carrier phase for a 4-QAM waveform

Phase estimation: The above example involving a sinusoidal carrier is for illustration purpose only because estimating the phase at the carrier level is not necessary. As we will shortly see, the phase information can be passed on unimpaired to further stages after mixing with the local oscillator. In this context, the fundamental problem of synchronization is to remove the (*preserved*) **phase jumps** from the symbol-spaced *IQ* samples arriving at the (much lower) symbol rate. We saw such phase rotated *IQ* samples in Figure 5.3 before. We will also have a look at such samples in Figure 5.9 coming soon.

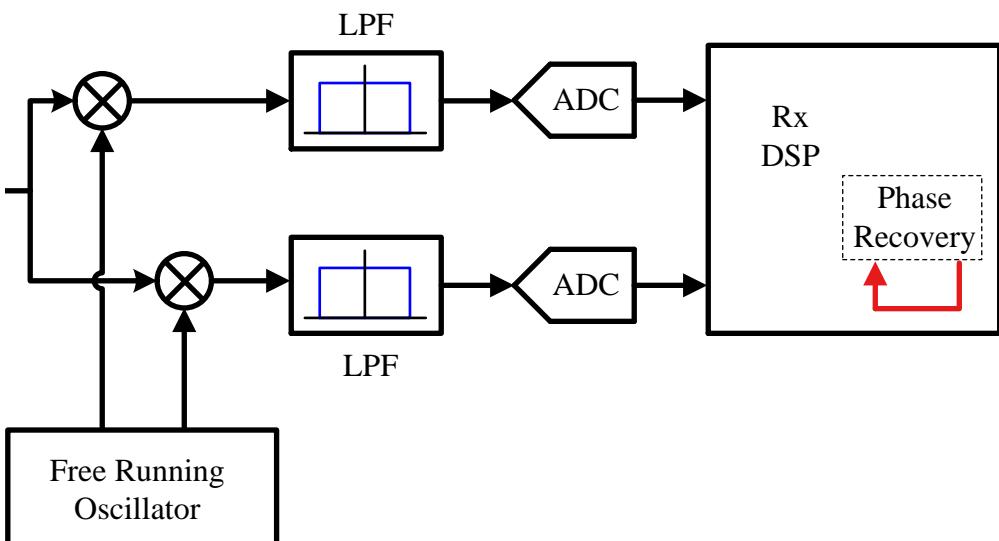
Phase compensation: The initial trend in digital modems was to emulate the analog signal processing through the digital hardware. Since the carrier phase recovery was done utilizing the detector decisions, a feedback signal from the digital domain to the analog side was supplied to adjust the carrier frequency and phase of a Voltage Controlled Oscillator (VCO). This is why the phase compensation in earlier digital modems was performed in passband as shown in Figure 5.8a. This complicated the system design due to the presence of additional digital to analog conversion and the related hardware as an interface between discrete-time and continuous-time worlds.

On the other hand, digital signal processing of complex signals is nothing more than storing and playing with twice the set of numbers, usually a small burden by virtue of Moore's law. Consequently, with the expansion of unique digital processing solutions and shrinking of the analog side in wireless communication systems, the signal is downconverted by a free running oscillator at a fixed frequency (leading to low phase jitter) and the phase offset can be compensated in the *IQ* modulated data right there and then. This is drawn in Figure 5.8b. Here, our objective is to highlight the compensation path and not the placement of the ADC, as it depends on the Rx architecture. We discuss the feasible Rx architectures in Chapter 10.

Obviously, the feedback loop is not necessary and both a feedforward (one-shot) estimator and a feedback mechanism (a PLL) can compensate for this unknown phase.



(a) A digital modem emulating an analog modem with a feedback path to a Voltage Controlled Oscillator (VCO)



(b) A digital modem with a free running oscillator and phase synchronization block residing in digital domain only

Figure 5.8: Two options for the feedback path from the phase recovery block to adjust the carrier phase offset

Note 5.3 Where is the carrier?

This discussion was needed here because many SDR beginners find it strange when

they encounter a carrier phase synchronization block of code (e.g., a PLL or a Costas loop in GNU Radio) in which there is no ‘carrier’ to be compensated for (except a slowly rotating phase due to carrier frequency offset)! Instead, they only find a phase de-rotation of the IQ samples going into the symbol detector. This is what we explain in the coming sections.

We now move towards the exact details of implementing this second strategy.

5.3 The Big Picture

Most timing synchronization algorithms operate independently of the phase information. Therefore, almost all the DSP based carrier phase synchronization algorithms are timing-aided which means that the symbol boundaries in the Rx sampled waveform are established before the phase recovery block. Knowing the exact symbol boundaries is the same as identifying the optimal sampling instants where the eye opening is maximum and Inter-Symbol Interference (ISI) from the neighbouring symbols is zero. An example is drawn in Figure 5.9.

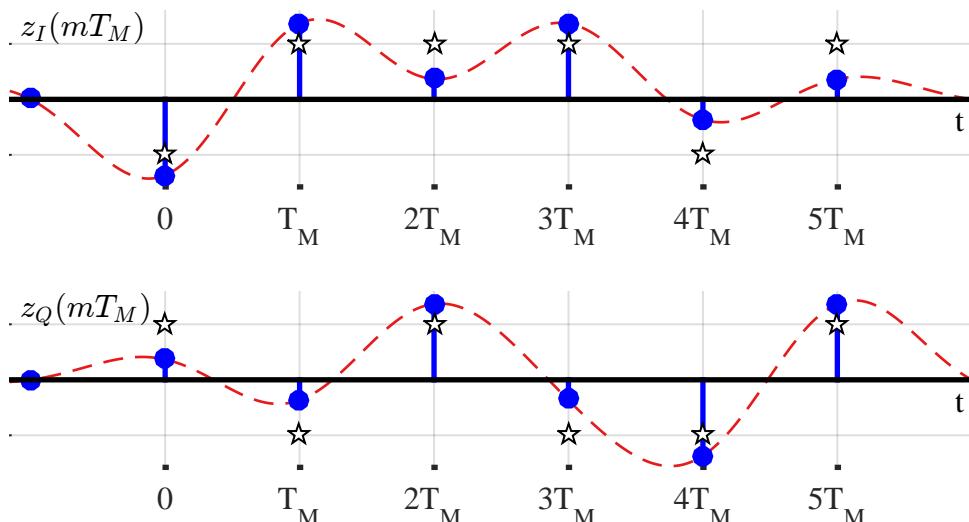


Figure 5.9: Timing-aided implies that the optimal sampling instants have already been figured out by the Rx and the imperfect mapping is due to the phase offset

In this figure, I and Q waveforms of a 4-QAM matched filtered sequence, $z_I(mT_M)$ and $z_Q(mT_M)$, rotated by a phase offset of 30° are shown that are sampled at ideal instants, i.e., integer multiples of the symbol time T_M . The sampling is ideal because the optimal sampling instants remain the same after the signal has undergone a phase shift and timing recovery can be achieved beforehand. However, the desired amplitudes are off from their actual values due to the phase offset. The actual values are illustrated as the constellation stars to demonstrate this effect. The job of the phase recovery is to compensate for this phase by de-rotating the I and Q waveforms.

Timing-aided in turn implies that while the matched filtering occurs at L samples/symbol, any processing faster than symbol rate $1/T_M$ to adjust the phase is un-

necessary. Therefore, the carrier phase synchronization block works at exactly the symbol rate $1/T_M$, or just 1 sample/symbol. Treating the phase adjustment problem at this stage results in minimal computational complexity without any degradation in performance.

Therefore, in this chapter, I do not classify the algorithms as timing-aided and non-timing-aided, as opposed to carrier frequency synchronization in Chapter 6. All schemes here are timing-aided (also known as clock-aided) only. Due to the availability of the symbol timing, the carrier phase synchronization algorithms only need to be classified as data-aided/decision-directed or non-data-aided as illustrated in Figure 5.10.

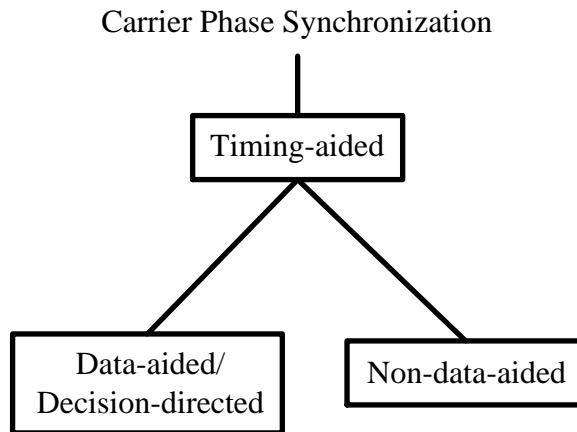


Figure 5.10: Classification of phase synchronization algorithms. Due to the nature of the digital processing, timing is acquired before the phase and hence almost all phase recovery algorithms operate with the help of available timing

Another virtue of known timing is that phase acquisition is performed through data-aided techniques when a preamble is available in the incoming data or in a decision-directed manner by utilizing the decisions fed back by the detector during tracking. Remember that the availability of data always implicitly indicates the availability of timing.

$$\text{no timing} \Rightarrow \text{no data}$$

We cover a simple example below.

Example 5.1

Consider the scatter plot in Figure 5.11. Suppose that the training symbols $a_I[m]$ and $a_Q[m]$ are known (a data-aided system). To find the phase offset $\hat{\theta}_\Delta[m]$ at symbol time m , their angle can be subtracted from the angle of the matched filter output at each symbol time m as

$$\hat{\theta}_\Delta[m] = \angle \frac{z_Q(mT_M)}{z_I(mT_M)} - \angle \frac{a_Q[m]}{a_I[m]} \quad (5.6)$$

The intuition behind this approach is simple: the phase difference between the received and the expected symbols is the remaining phase offset.

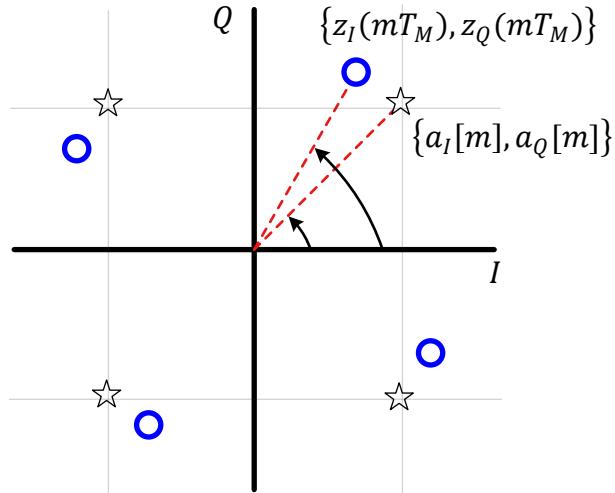


Figure 5.11: Phase difference between known symbols and downsampled matched filter outputs can be utilized to estimate θ_Δ

In the discussion that follows, we omit the Rx signal amplitude by normalizing it to 1. This amplitude scaling arises from the gains and losses when the signal travels through the wireless channel and frontend components at Tx and Rx such as antennas, filters, amplifiers, mixers and so on. In practice, an Automatic Gain Control (AGC) at the Rx side delivers a signal with a predetermined amplitude to the DSP blocks for further processing.

5.4 Enter the Correlation

For this problem, we have two options. We can either come up with some intuitive ad hoc phase estimators, or employ the correlation principle to this problem and see where it leads. This is because in Section 1.6, we said that *correlation is the Master Algorithm* for a digital communication system, not only for data detection but for parameter estimation of various kinds as well. We applied it to derive the matched filter for optimal detection. Here, we apply the principle of maximum correlation for solving our phase synchronization problem. Interestingly, it turns out that the derivation of phase estimators through the maximum correlation process actually leads to the same intuitively satisfying results[†].

The discussion in this context usually considers QPSK modulation but the extension to higher-order modulation schemes is a straightforward task. A similar framework is used for finding and compensating for other distortions as well such as carrier frequency offset and symbol timing offset, the examples of which we will see in later chapters.

[†]Maximum correlation is a result of a procedure termed as maximum likelihood estimation. Many ad hoc synchronization algorithms can be derived through different approximations of the maximum likelihood estimation.

Note 5.4 Correlation with what?

In order to implement correlation, one signal is obviously the Rx signal $r(nT_S)$. It is correlated with a clean version of what is expected at the Rx, the expected signal template. That is the most logical solution to detect the signal and estimate the desired parameters. For example, in case of a carrier phase offset, the Rx signal $r(nT_S)$ will be correlated with a perfect Tx signal but rotated in phase by an unknown offset $\hat{\theta}_\Delta$. This procedure then generates an algorithm to estimate that phase offset.

Now after the phase rotation by θ_Δ , the Tx signal is given by

$$\begin{aligned} s(nT_S) &= v_I(nT_S)\sqrt{2}\cos\left(2\pi\frac{k_C}{N}n + \theta_\Delta\right) - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left(2\pi\frac{k_C}{N}n + \theta_\Delta\right) \end{aligned} \quad (5.7)$$

where k_C corresponds to carrier frequency F_C and $v_I(nT_S)$ and $v_Q(nT_S)$ are the inphase and quadrature waveforms.

$$\begin{array}{ll} I & \rightarrow v_I(nT_S) = \sum_m a_I[m]p(nT_S - mT_M) \\ Q & \uparrow v_Q(nT_S) = \sum_m a_Q[m]p(nT_S - mT_M) \end{array}$$

In the presence of noise, the received and sampled signal from Eq (5.50) is

$$r(nT_S) = s(nT_S) + \text{noise} \quad (5.8)$$

The correlation between $r(nT_S)$ and its expected template $s(nT_S)$ is defined as

$$\begin{aligned} \text{corr}[j] &= r(nT_S) \heartsuit s(nT_S) \\ &= \sum_{n=-\infty}^{\infty} r(nT_S)s(nT_S - j) \end{aligned} \quad (5.9)$$

To implement the above relation, we need to consider the following.

Observation interval: The above sum is computed from $-\infty$ to $+\infty$, which implies that we could theoretically continue computing this correlation with every received sample $r(nT_S)$. However, the useful component of the received signal $r(nT_S)$ is given by a span of N_0 symbols while every sample outside this interval is noise only. The time duration of such N_0 symbols is

$$T_0 = N_0 T_M \quad (5.10)$$

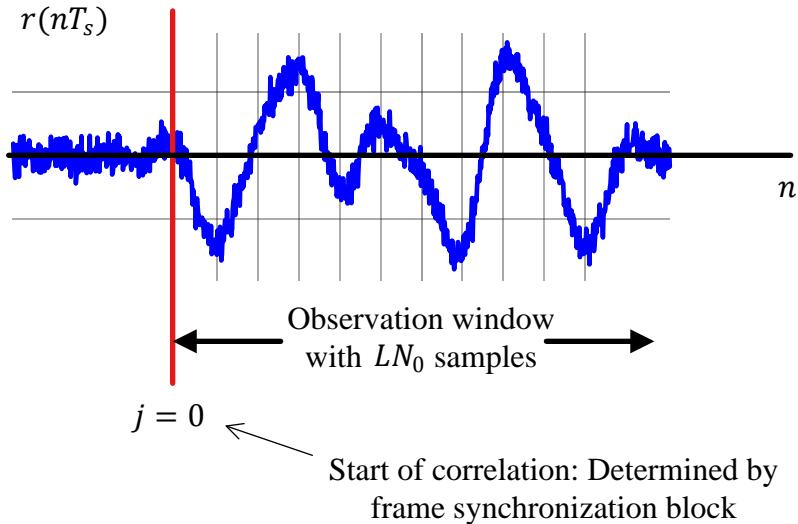
Thus, the finite observation interval is given by

$$0 \rightarrow \underbrace{T_0}_{\text{seconds}} = \underbrace{N_0}_{\text{symbols}} \cdot T_M = \underbrace{N_0 \cdot L}_{\text{samples}} \cdot T_S$$

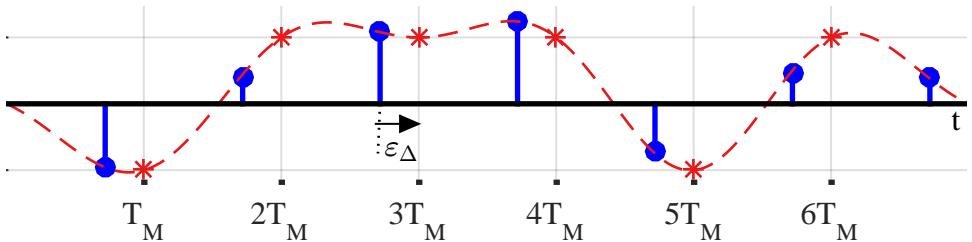
where L is samples/symbol, or T_M/T_S . Therefore, the observation window consists of $n = 0, 1, \dots, LN_0 - 1$ samples. It is understood in this calculation that the group delay arising from pulse shaping and matched filtering on either side of the transmission sequence has been taken care of by the Rx.

Start of frame: The second problem is to determine the starting point of the useful component in a sampled signal, as many of its initial samples are noise only. It is the job of a frame synchronization unit to determine the first symbol of the Rx signal which is shown in Figure 5.12a. This is the same as finding the integer component of the signal alignment.

On the other hand, the symbol timing synchronization block establishes the fractional sampling instant within a symbol and hence known as symbol timing offset. Figure 5.12b illustrates this problem for a visual clarification (the grid lines on the x-axis are important here). The meaning of symbol timing offset problem now will be explained in detail in Chapter 7.



(a) Observation window for determining a phase estimate for θ_Δ . Identifying the integer component of the alignment, i.e., the first symbol, is regulated by frame synchronization block



(b) Identifying the fractional component of the alignment, i.e., the exact fractional sampling instant within a symbol (hence known as symbol timing offset ε_Δ), is regulated by timing synchronization block

Figure 5.12: Alignment of the incoming signal at the Rx is handled by frame and timing synchronization blocks

Carrier Frequency: We also assume downconversion with perfect frequency recovery, i.e., Tx and Rx oscillators have the same frequency F_C , to emphasize on phase recovery problem. We will cover carrier frequency synchronization in Chapter 6.

Distortion at edges: Due to the pulse shape spreading beyond a single symbol and actually extending to G symbols in each direction, there is a distortion introduced at the edges of the observation interval T_0 in the correlation result. However, it can be ignored if T_0 is sufficiently long.

With these settings in place, we can proceed with the correlation of $r(nT_S)$ with $s(nT_S)$ that

contains the phase offset θ_Δ in its expression.

$$\text{corr}[j] = \sum_{n=0}^{LN_0-1} r(nT_S)s(nT_S - j)$$

Having determined $j = 0$ through frame synchronization block and using $s(nT_S)$ from Eq (5.7), we can write

$$\begin{aligned} \text{corr}[0] &= \sum_{n=0}^{LN_0-1} r(nT_S)s(nT_S) \\ &= \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ v_I(nT_S) \sqrt{2} \cos \left(2\pi \frac{k_C}{N} n + \hat{\theta}_\Delta \right) - \right. \\ &\quad \left. v_Q(nT_S) \sqrt{2} \sin \left(2\pi \frac{k_C}{N} n + \hat{\theta}_\Delta \right) \right\} \end{aligned}$$

where we replace θ_Δ with $\hat{\theta}_\Delta$ since this parameter needs to be estimated. We now open the sinusoidal terms using the identities $\cos(A + B) = \cos A \cos B - \sin A \sin B$ and $\sin(A + B) = \sin A \cos B + \cos A \sin B$ and collect the common terms.

$$\begin{aligned} \text{corr}[0] &= \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ v_I(nT_S) \cos \hat{\theta}_\Delta - v_Q(nT_S) \sin \hat{\theta}_\Delta \right\} \sqrt{2} \cos 2\pi \frac{k_C}{N} n - \\ &\quad \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ v_Q(nT_S) \cos \hat{\theta}_\Delta + v_I(nT_S) \sin \hat{\theta}_\Delta \right\} \sqrt{2} \sin 2\pi \frac{k_C}{N} n \end{aligned}$$

Substituting the expressions for two PAM symbol streams $v_I(nT_S)$ and $v_Q(nT_S)$ yields

$$\begin{aligned} \text{corr}[0] &= \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ \sum_m a_I[m] p(nT_S - mT_M) \cos \hat{\theta}_\Delta - \right. \\ &\quad \left. \sum_m a_Q[m] p(nT_S - mT_M) \sin \hat{\theta}_\Delta \right\} \sqrt{2} \cos 2\pi \frac{k_C}{N} n \\ &\quad - \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ \sum_m a_Q[m] p(nT_S - mT_M) \cos \hat{\theta}_\Delta + \right. \\ &\quad \left. \sum_m a_I[m] p(nT_S - mT_M) \sin \hat{\theta}_\Delta \right\} \sqrt{2} \sin 2\pi \frac{k_C}{N} n \end{aligned}$$

Recalling the fact that $p(nT_S)$ are the samples of a square-root Nyquist pulse with support $-LG \leq n \leq LG$ samples, we can define

$$\begin{aligned} I \rightarrow z_I(mT_M) &= \sum_{n=(m-G)L}^{(m+G)L} \underbrace{\left\{ r(nT_S) \sqrt{2} \cos 2\pi \frac{k_C}{N} n \right\}}_{x_I(nT_S)} p(nT_S - mT_M) \\ Q \uparrow z_Q(mT_M) &= -2 \sum_{n=(m-G)L}^{(m+G)L} \underbrace{\left\{ r(nT_S) \sqrt{2} \sin 2\pi \frac{k_C}{N} n \right\}}_{x_Q(nT_S)} p(nT_S - mT_M) \end{aligned} \tag{5.11}$$

This is because from the Rx block diagram in Figure 5.1, a product of $r(nT_S)$ with the sinusoid $\sqrt{2} \cos 2\pi(k_C/N)n$ produces $x_I(nT_S)$ that is the I component in the top branch while the product of $r(nT_S)$ with $-\sqrt{2} \sin 2\pi(k_C/N)n$ generates $x_Q(nT_S)$ that is the Q part in the bottom branch.

In the above definitions, the Rx signal $r(nT_S)$ is being correlated with the pulse shape $p(nT_S)$. Another way of implementing this correlation is through filtering the signal with its flipped version, $p(-nT_S)$ which is *matched filtering*. Clearly, after summing over the sample index n , we are left with a signal with symbol rate spaced samples at times mT_M .

Owing to the above description, we can identify these terms as downsampled matched filtered outputs, one in the inphase and other in the quadrature arm[†] and hence the notation $z_I(mT_M)$ and $z_Q(mT_M)$. Ignoring the distortion at the edges of the summation, we can plug them into the correlation expression above before rearranging.

$$\begin{aligned} \text{corr}[0] = & \sum_{m=0}^{N_0-1} a_I[m] \left\{ z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \right\} \\ & + \sum_{m=0}^{N_0-1} a_Q[m] \left\{ z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \right\} \end{aligned} \quad (5.12)$$

The above expression is important. As we proceed, this Eq (5.12) will form the basis of many types of phase estimators, both in feedforward and feedback schemes.

A question at this stage is the following: *starting from correlation, what have we achieved so far?* While not clear as of now, we are on our way to solve the fundamental problem of the synchronization process raised in Section 5.2. As a first step, we have removed the carrier dependence and are now operating at the baseband where we will perform the phase recovery process. Most of the discussion that follows is for QPSK modulation and holds true for other modulations with little modification.

5.5 Data-Aided/Decision-Directed Techniques

Recall that some data symbols $a[m]$ can be known beforehand for synchronization purpose. This happens for example in the case of data-aided synchronization where the Tx inserts symbols already agreed upon with the Rx within the message known as a *training sequence* such that the Rx can acquire unknown parameters through knowledge of this ‘data’. This leads to a data-aided strategy of synchronization as below.

Due to its simplicity, we treat the phase synchronization problem with the feedforward (FF) approach first and later shift to the feedback (FB) techniques.

5.5.1 Feedforward: Conjugate Product Estimator

In burst mode wireless communications, data is sent over a link in the form of separate packets. In most such applications with short packets, the phase offset θ_Δ remains constant throughout the duration of the packet and a single shot estimator is enough for its compensation. In a feedforward system, the primary task is to develop this closed-form expression for the phase estimator. Then, this estimate can be treated as the unknown original phase offset and message symbols can be phase corrected by de-rotating the matched filter output by this amount.

[†]Interestingly, we could also combine this phase rotation with the matched filter $p(-nT_S)$ to come up with a modified matched filter (square-root Nyquist pulse rotated by $\hat{\theta}_\Delta$) that would be a true matched filter.

Here, we treat the problem of phase synchronization using techniques based on the conjugate product of the matched filter outputs and the data symbols (or their decisions), which is a direct result of carrying forward the derivation of maximum correlation above. The details of this implementation are slightly different for a feedforward (FF) and a feedback (FB) implementation.

Assume that there are N_p symbols $a[m]$ in a preamble with $a_I[m]$ and $a_Q[m]$ as inphase and quadrature parts. To derive an estimator based on phase difference, rearrange the above correlation expression in Eq (5.12) as

$$\begin{aligned}\text{corr}[0] &= \sum_{m=0}^{N_p-1} \left\{ a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M) \right\} \cos \hat{\theta}_\Delta \\ &\quad + \sum_{m=0}^{N_p-1} \left\{ a_I[m]z_Q(mT_M) - a_Q[m]z_I(mT_M) \right\} \sin \hat{\theta}_\Delta\end{aligned}$$

Let $y(mT_M) = \sum_{m=0}^{N_p-1} a^*[m]z(mT_M)$. From the definition of complex conjugate and multiplication rule $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$,

$$\begin{aligned}I \rightarrow y_I &= \sum_{m=0}^{N_p-1} \left\{ a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M) \right\} \\ Q \uparrow y_Q &= \sum_{m=0}^{N_p-1} \left\{ a_I[m]z_Q(mT_M) - a_Q[m]z_I(mT_M) \right\}\end{aligned}$$

which leads to

$$\begin{aligned}\text{corr}[0] &= y_I(mT_M) \cos \hat{\theta}_\Delta + y_Q(mT_M) \sin \hat{\theta}_\Delta \\ &= |y(mT_M)| \left\{ \cos(\angle y(mT_M)) \cos \hat{\theta}_\Delta + \sin(\angle y(mT_M)) \sin \hat{\theta}_\Delta \right\} \\ &= |y(mT_M)| \cos(\angle y(mT_M) - \hat{\theta}_\Delta)\end{aligned}$$

where we have utilized the definition of inphase and quadrature components of $y(mT_M)$ as well as the identity $\cos A \cos B + \sin A \sin B = \cos(A - B)$, and \angle is the four-quadrant inverse tangent.

Clearly, the correlation is maximized when \cos of its phase equals 1. In other words, the angle $\angle y(mT_M) - \hat{\theta}_\Delta$ goes to zero which implies that

$$\hat{\theta}_\Delta = \angle y(mT_M)$$

Thus, the expression for the feedforward conjugate product phase estimator is given by

$$\hat{\theta}_\Delta = \angle \left\{ \sum_{m=0}^{N_p-1} a^*[m]z(mT_M) \right\} \quad (5.13)$$

A related estimator, except the usual difference between a four-quadrant inverse

tangent and a simple inverse tangent, is

$$\begin{aligned}
 \hat{\theta}_\Delta &= \tan^{-1} \frac{y_Q(mT_M)}{y_I(mT_M)} \\
 &= \tan^{-1} \frac{\sum_{m=0}^{N_p-1} \left\{ a_I[m]z_Q(mT_M) - a_Q[m]z_I(mT_M) \right\}}{\sum_{m=0}^{N_p-1} \left\{ a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M) \right\}}
 \end{aligned} \tag{5.14}$$

A block diagram for this conjugate product estimator implementation is drawn in the upper half of Figure 5.13, where the complex multiplication between conjugate of the known training and downsampled matched filter outputs is expanded into real operations. It is known as the maximum likelihood estimator because it is the solution of our search for finding the maximum correlation between the noisy Rx signal and its clean expected replica at the Tx.

Time Domain View

Now we can clearly see how the fundamental problem of synchronization indicated through Figure 5.7 has been solved.

Conjugate product Remember from Eq (5.4) that the phase of $z(mT_M)$ is a sum of phase shift arising from modulating data symbols that varies from symbol to symbol, and a constant phase shift arising from θ_Δ .

$$\angle z(mT_M) = \angle a[m] + \theta_\Delta$$

Looking at Eq (5.13), $a^*[m]z(mT_M)$ removes the effect of phase shifts arising from data symbols. This is because $a^*[m]$ is the conjugate of symbols $a[m]$ which results in the same magnitude but opposite phase.

$$\angle \{a^*[m]z(mT_M)\} = \angle a[m] + \theta_\Delta - \angle a[m] = \theta_\Delta$$

What is left after its multiplication with $z(mT_M)$ then is just the phase due to θ_Δ plus noise. By removing the phase jumps due to the modulating data at each symbol, the correlation process addresses the fundamental problem of synchronization.

Summation Since the samples are noisy and a single such sample will produce a wayward result, a summation over multiple samples is then applied to suppress the noise which coherently adds the desired signal while driving the noise input towards zero (due to Gaussian nature of noise acquiring both positive and negative values, the result of summing a large number of noise-only samples tends to zero).

To gain more insight into this process, assume that there is a factor of $1/N_p$ included before the summation which effectively makes this an averaging operation. Recall from Section 2.6 that averaging the samples within a certain

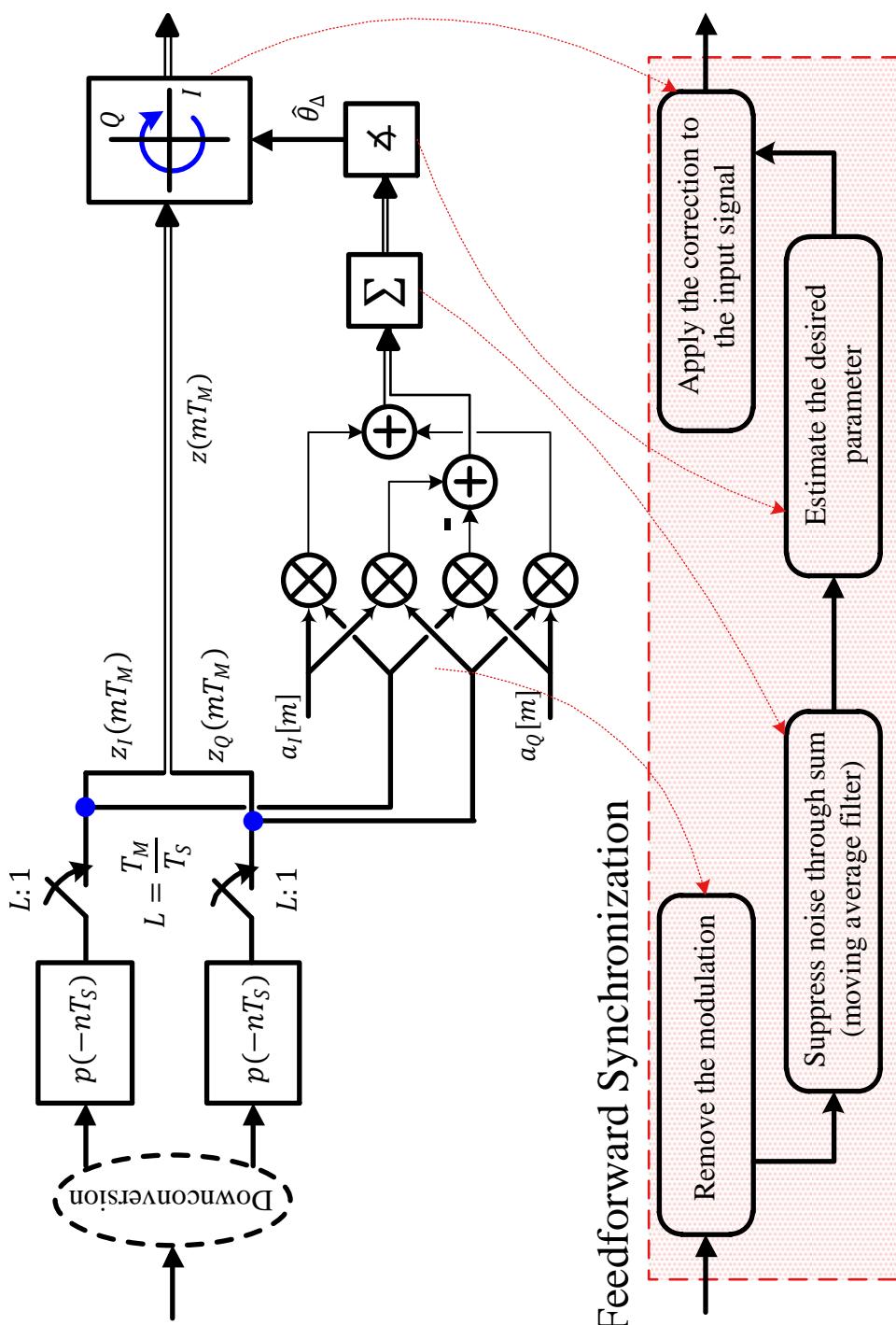


Figure 5.13: A block diagram for the implementation of the conjugate product estimator

segment of the input signal is equivalent to passing the signal through a *moving average filter* $h[m]$, a useful tool for noise reduction.

$$h[m] = \frac{1}{N_P} [1 \underbrace{1 \quad 1}_{N_P} \cdots 1]$$

In our current case, the input is a length- N_P sequence of a constant complex number while the filter is also a constant sequence of the same length. Therefore, the filtering operation generates a ramp (an increasing sequence) of which all values can be ignored except when the sample at $N_P - 1$ enters the system and the maximum output occurs. Since the scaling factor $1/N_P$ does not affect the phase output, it can be discarded altogether.

Four-quadrant inverse tangent Finally, θ_Δ is computed through the four-quadrant inverse tangent operation.

To visually comprehend the underlying signal level modifications, the Tx signal $s(t)$ in Figure 5.14 consists of fast variations due to the carrier and is downconverted at the Rx by a local oscillator having a phase offset θ_Δ with respect to the Tx. The downconverted, matched filtered and downsampled I and Q outputs $z_I(mT_M)$ and $z_Q(mT_M)$ are drawn next, where these are corrupted by additive noise at each stage after arriving at the Rx. *Since the only distortion present is a phase offset, removing the modulation reveals I and Q parts of a complex number whose phase is the sought after parameter.* The noise is reduced from this signal through applying a moving average filter.

Note 5.5 What did the correlation do?

From the description above, now we can identify how the correlation has helped accomplishing our goal of estimating the unknown parameter. First, rewrite Eq (5.12).

$$\begin{aligned} \text{corr}[0] &= \sum_{m=0}^{N_0-1} a_I[m] \left\{ z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \right\} \\ &\quad + \sum_{m=0}^{N_0-1} a_Q[m] \left\{ z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \right\} \end{aligned}$$

Clearly, all but one of the parameters in the expected signal template that is correlated with $r(nT_S)$ are known, i.e.,

$$\begin{aligned} \text{Known parameters} &\rightarrow a_I[m], a_Q[m], p(nT_S) \\ \text{Unknown parameters} &\rightarrow \theta_\Delta \end{aligned}$$

which is why it was replaced with $\hat{\theta}_\Delta$. Since the only unknown is the phase offset θ_Δ , correlation strips all the known parameters in its expression to reveal the only unknown parameter. This is illustrated in the 3rd row of Figure 5.14 where the phase offset θ_Δ is isolated alone. Hence, this process forms a relation of $r(nT_S)$ with the known parameters from which the unknown parameter can be estimated.

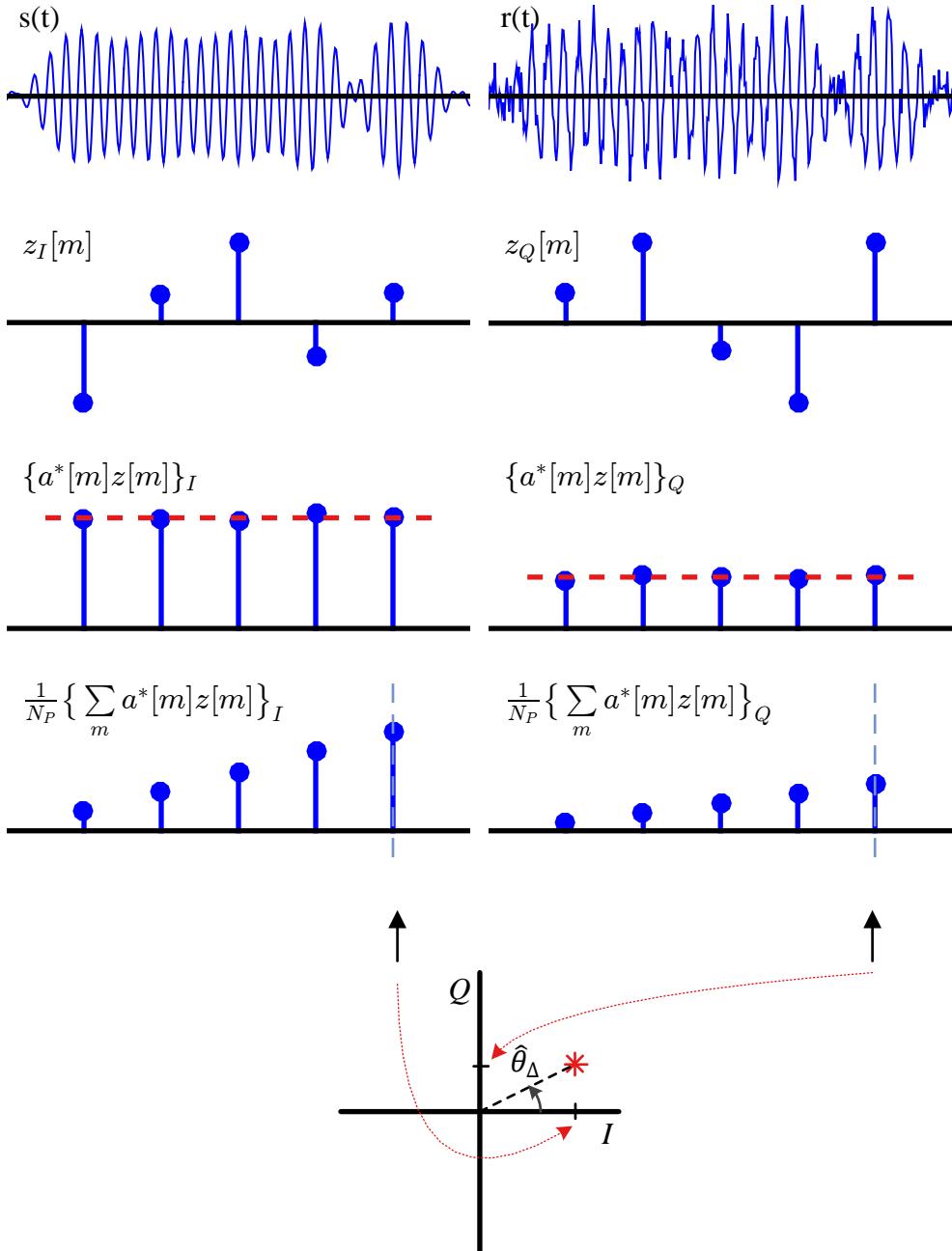


Figure 5.14: A signal level view of phase estimation process. The whole correlation process downconverts the signal, performs matched filtering and finally strips the modulation off this signal to reveal the underlying phase offset θ_Δ

Frequency Domain View

Another way to look at this result is in frequency domain as follows. Assume that there is only one data symbol $a[m]$ in which the I and Q components are from the set $\pm A$.

This is for visualization purpose only and the whole sequence of $a[m]$ for all m needs to be taken into account otherwise. Essentially the Tx signal becomes a combination of two pulse shaping filters $P[k]$ scaled by $\pm A$, one in I and the other in Q rail, that get rotated by θ_Δ .

It is presented to two matched filters, one in I and the other in Q rail, which have a frequency response $P^*[k]$. The pulse shape is real and even[†], thus $P[k]$ as well as $P^*[k]$ are also real and even. In frequency domain, the output of a matched filter is its frequency response multiplied with that of the incoming signal. From the multiplication rule $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$, the matched filter output in terms of the DFT of the pulse autocorrelation $R_p[k] = |P[k]|^2$ becomes

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} Z_I[k] &= a_I|P[k]|^2 \cos \theta_\Delta - a_Q|P[k]|^2 \sin \theta_\Delta \\ Z_Q[k] &= a_Q|P[k]|^2 \cos \theta_\Delta + a_I|P[k]|^2 \sin \theta_\Delta \end{aligned} \quad (5.15)$$

This signal is shown through red dashed line in Figure 5.15.

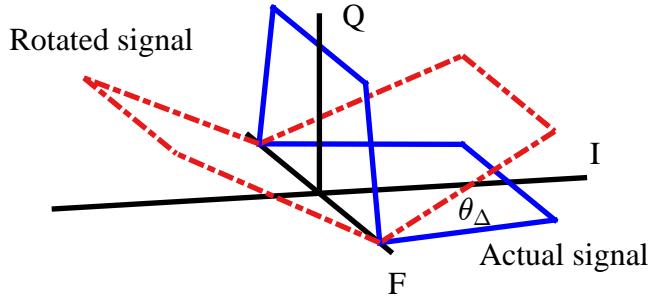


Figure 5.15: The rotated and matched filtered signal in I and Q arms

Notice that the angle of Q output with positive I -axis is $90^\circ + \theta_\Delta$ so its projection on I -axis (while zero before rotation) now becomes

$$a_Q|P[k]|^2 \cos(90^\circ + \theta_\Delta) = -a_Q|P[k]|^2 \sin \theta_\Delta$$

where we have used the identity $\cos(A + B) = \cos A \cos B - \sin A \sin B$. This adds to $a_I|P[k]|^2 \cos \theta_\Delta$ of the I output and geometrically justifies the I part of the rotation in Eq (5.15). A similar derivation holds for Q arm.

This example was drawn for both a_I and a_Q as $+A$. For an arbitrary modulation, e.g., $-A$ that further rotates it by 180° , this rotation of modulated data cancels out due to the conjugate operation and the correlation is naturally maximized when this Rx signal is exactly similar to the matched filter, i.e., rotated back by an angle $-\theta_\Delta$.

Finally, instead of a single symbol, when the whole sequence of symbols is present along with the noise, the moving average filter of time domain acts as a *lowpass filter* in frequency domain to suppresses the out-of-band noise.

[†]This is only true in most cases. For example, ATSC transmission system for digital television uses an 8-VSB modulation with a spectrally shifted Square-Root Raised Cosine pulse shape that has a complex impulse response.

Comments on Performance

To see what happens in the presence of noise, consider the following in regard to Eq (5.13).

$$\text{signal}^*(\text{signal} + \text{noise}) = \underbrace{\text{signal}^* \times \text{signal}}_{\text{term containing the phase } \theta_\Delta} + \text{signal}^* \times \text{noise} \quad (5.16)$$

It is clear that due to the Rx knowledge of the training sequence $a[m]$ sent with the data, there is a single noise term and hence no noise enhancement in the estimation process. As we see later, this is not the case with non-data-aided estimators. Therefore, the phase estimation through this method renders the best performance possible.

Tradeoffs

This benefit comes at a price of increased bandwidth: assume that a data rate of R_b bits/second needs to be supported on a link which translates to a symbol rate of $R_M = 1/T_M$ symbols/second for a total of N_d symbols. However, an addition of N_P training symbols – and the requirement to send the same amount of bits within the same time – implies that the new symbol rate $R_{M,\text{new}}$ becomes

$$R_{M,\text{new}} = R_M \left(1 + \frac{N_P}{N_d}\right) \text{symbols/second}$$

Since the bandwidth is directly proportional to the symbol rate

$$BW_{\text{QAM}} = (1 + \alpha)R_M$$

the system needs a larger bandwidth. In other words, the power and time spent on transmitting the training sequence could have been used for sending more data: the spectral efficiency of the system is reduced by a non-negligible factor.

5.5.2 General Approach to Feedforward Synchronization

From the bottom half of Figure 5.13, a general approach to feedforward phase synchronization can be constructed as follows.

- 1. Modulation removal** Remove the modulation from the input, i.e., the matched filter outputs.
- 2. Filtering** Filter the noise through summation, which in effect is a moving average filter.
- 3. Estimation** Estimate the phase offset θ_Δ .
- 4. Correction** Apply the correction to the input through rotating it by $-\hat{\theta}_\Delta$.

Note 5.6 Correlation makes sense

Recall from Section 5.2 where we said that ad hoc phase estimators can be built based on intuition, or correlation can be applied to this problem and see where it leads. As we found out, the derivation of phase estimators through the correlation process has actually lead to an intuitively systematic process. This is why many heuristic techniques devised for carrier and timing synchronization later turned out to be an approximation of the maximum likelihood procedure.

5.5.3 Feedback: Phase Difference Phase Error Detector

Now assume that a feedback (FB) system, like a Phase Locked Loop (PLL), is employed for synchronizing the Rx oscillator phase to the phase of the Tx signal. However, the PLL design in Chapter 4 revolved around synchronizing the output against a single sinusoid with a constant phase offset. The input signal in our current scenario is the matched filter output $z(mT_M)$, the phase of which constitutes both the phase offset θ_Δ and the phase due to the modulated data $a[m]$ that changes at symbol intervals.

$$\angle z(mT_M) = \angle a[m] + \theta_\Delta$$

To assist the Rx in signal acquisition, some communication systems transmit a reference sinusoid – called the *pilot tone* – along with the Tx signal. It is then a straightforward task for Rx PLL to lock onto this incoming sinusoid after passing it through a bandpass filter. However, this procedure costs additional bandwidth and power.

In most situations, the Rx PLL can be driven through the actual Tx signal itself and without any pilot tone if the phase shifts at symbol intervals arising from modulating data can be removed. To cater for input phase $\angle z(mT_M)$ rapidly changing at symbol rate $1/T_M$, *a new kind of phase detector needs to be devised* that can remove the impact of modulation symbols $\angle a[m]$ from $\angle z(mT_M)$ at each symbol interval, generating an output as a function of θ_Δ only. With this modification, the input to the loop filter becomes similar to a simple sinusoidal case and hence all the concepts about a PLL tracking a simple input sinusoid apply for the phase synchronization block as before. Next, we describe how a basic PLL can be modified for accomplishing such a task. This discussion on feedback phase error detectors is mostly based on Ref. [2].

To start, the maximum correlation approach yielded a conjugate product estimator of the phase offset θ_Δ through the expression in Eq (5.13) reproduced below.

$$\hat{\theta}_\Delta = \angle \left\{ \sum_{m=0}^{N_p-1} a^*[m] z(mT_M) \right\}$$

Since nothing in a simple PLL needs to be changed except the Phase Error Detector (PED), our initial idea to modify a simple PLL is as follows. Assume that we have an initial estimate of $\hat{\theta}_\Delta$ available (regardless of how far it is from the actual θ_Δ). At each symbol time mT_M , we can perform the following operations.

- De-rotate (clockwise rotation) by $\hat{\theta}_\Delta$ the matched filtered output $z(mT_M)$ through the output of a Numerically Controlled Oscillator (NCO). Recalling the phase

rotation rule in clockwise direction $I \cos \theta + Q \sin \theta$ and $Q \cos \theta - I \sin \theta$, we can define the de-rotated matched filtered signal $\hat{z}(mT_M)$ as

$$\begin{array}{ll} I & \rightarrow & \hat{z}_I(mT_M) = z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \\ Q & \uparrow & \hat{z}_Q(mT_M) = z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \end{array} \quad (5.17)$$

- Use *the phase of the conjugate product* as a phase error detector (and hence the name phase difference phase error detector) in the following way. Transform the feedforward conjugate product estimator $\hat{\theta}_\Delta$ of Eq (5.13) into a feedback phase error detector output $e_D[m]$ by

- ◊ removing the summation, and
- ◊ replacing the matched filter output $z(mT_M)$ by de-rotated matched filter output $\hat{z}(mT_M)$

which leads to

$$e_D[m] = \angle \left\{ a^*[m] \hat{z}(mT_M) \right\} \quad (5.18)$$

Here, the I and Q components of $a^*[m] \hat{z}(mT_M)$ are

$$\begin{array}{ll} I & \rightarrow & \left\{ a^*[m] \hat{z}(mT_M) \right\}_I = a_I[m] \hat{z}_I(mT_M) + a_Q[m] \hat{z}_Q(mT_M) \\ Q & \uparrow & \left\{ a^*[m] \hat{z}(mT_M) \right\}_Q = a_I[m] \hat{z}_Q(mT_M) - a_Q[m] \hat{z}_I(mT_M) \end{array} \quad (5.19)$$

Replacing the four-quadrant inverse tangent with a simple inverse tangent produces another form of the phase error detector.

$$\begin{aligned} e_D[m] &= \tan^{-1} \frac{\left\{ a^*[m] \hat{z}(mT_M) \right\}_Q}{\left\{ a^*[m] \hat{z}(mT_M) \right\}_I} \\ &= \tan^{-1} \frac{a_I[m] \hat{z}_Q(mT_M) - a_Q[m] \hat{z}_I(mT_M)}{a_I[m] \hat{z}_I(mT_M) + a_Q[m] \hat{z}_Q(mT_M)} \end{aligned} \quad (5.20)$$

As opposed to wiping off the carrier phase offset once and for all, this kind of implementation is essentially a strategy to converge towards the final solution in small incremental steps.

As explained before, the intuition behind the expression phase difference phase error detector is that $a^*[m]$ is the conjugate of known training sequence $a[m]$ which has the same magnitude but opposite phase, so $a^*[m] \hat{z}(mT_M)$ is actually a phase difference which removes the effect of phase shifts belonging to $a[m]$ from $\hat{z}(mT_M)$ and leaves the remaining phase offset as a residue. A block diagram of a carrier phase synchronization PLL for a QPSK modulation scheme in the presence of a known training sequence is drawn in Figure 5.16.

In this feedback process, at each symbol interval m ,

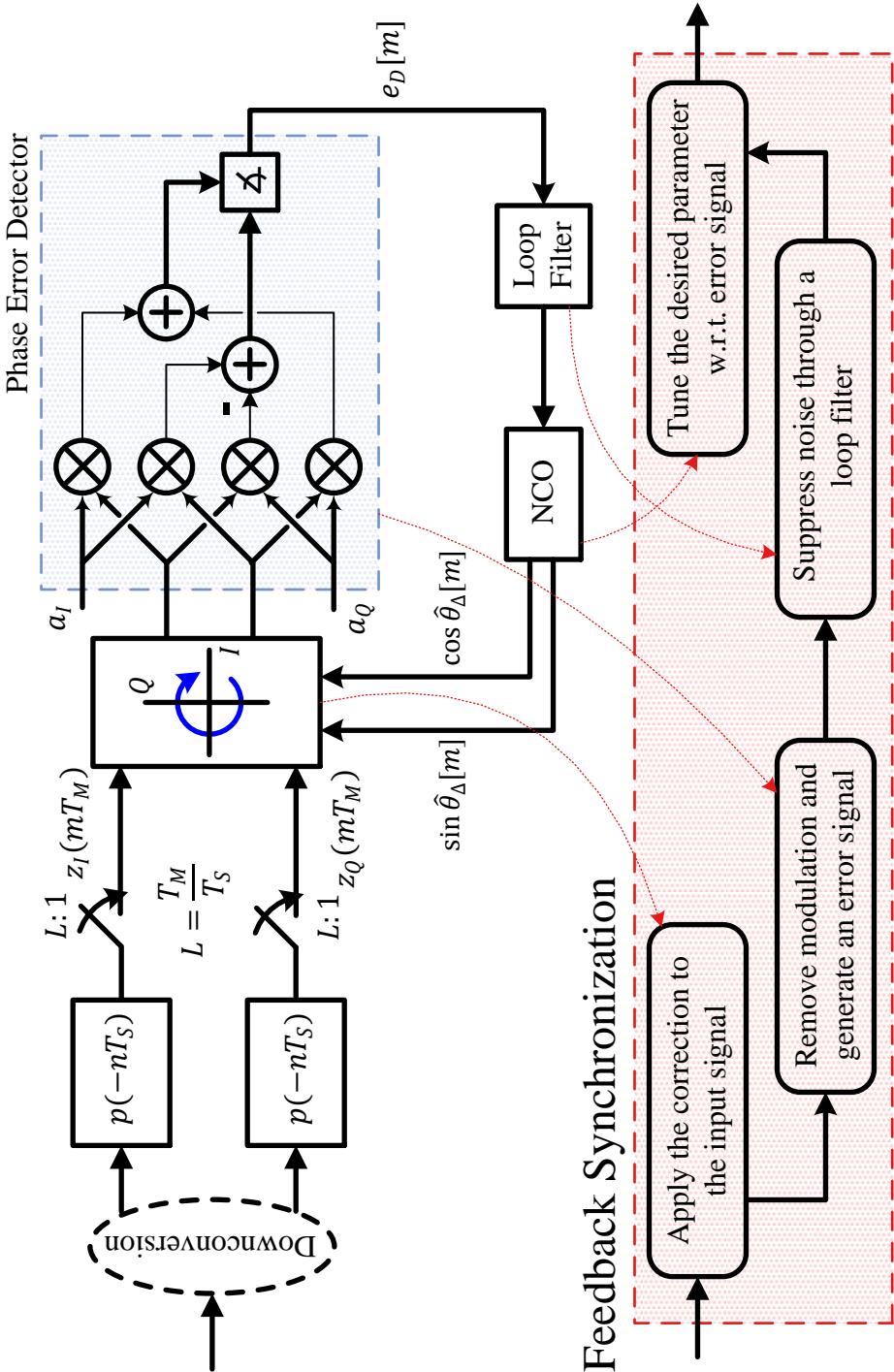


Figure 5.16: A block diagram for a QPSK carrier phase PLL employing the phase difference PED in data-aided mode

- the matched filter outputs are rotated clockwise by $\hat{\theta}_\Delta[m]$ – the current estimate of θ_Δ – to compensate for the original anticlockwise rotation performed by θ_Δ ,
- a phase detector output is formed by the relation in Eq (5.18),
- a loop filter suppresses the out-of-band noise and undesired frequency components, which is implemented in the form of a phase accumulator, and
- the NCO produces cosine and sine of the filter output which is input to the rotation block for the next symbol time $m + 1$.

Compare Figure 5.16 with a standard PLL in Chapter 4; *it is also a PLL but with a modified Phase Error Detector (PED)*. We started with applying correlation between the noisy Rx signal and its clean template and arrive through a different route at an iterative or feedback structure that is a PLL. We conclude that the solution to the fundamental problem of synchronization can be solved by employing an ordinary PLL with a modified phase error detector. The only difference between an ordinary PLL tracking a single sinusoid in phase and frequency and a PLL employed in data modulated signals is the Phase Error Detector (PED). Other than removing the phase shifts arising from symbol changes at each symbol interval T_M , the design principles and operational characteristics of the PLL remain the same.

Defining a Mean Curve for Phase Error Detector

A key tool to investigate the phase acquisition behaviour of any PLL is the average of the phase detector output $e_D[m]$ known as the *S-curve* (which we will call a mean curve).

$$\text{S-curve : } \text{Mean}\{e_D[m]\} \equiv \bar{e}_D \quad \text{for each } \theta_\Delta - \hat{\theta}_\Delta \equiv \theta_{\Delta:e} \quad (5.21)$$

The ultimate purpose is to chart the phase error detector output for each value of phase error $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$ and make conclusions about its operation. This mean curve is generated by the following steps:

- open the loop feedback,
- de-rotate the matched filter output by a test phase $\hat{\theta}_\Delta$,
- observe the phase error detector output by sufficiently averaging it, and
- repeat the procedure for a new $\hat{\theta}_\Delta$ such that $\theta_{\Delta:e}$ ultimately varies in the whole range $-\pi \leq \theta_{\Delta:e} \leq +\pi$ (for example, when $\hat{\theta}_\Delta = \theta_\Delta$, then $\theta_{\Delta:e} = 0$).

This procedure is drawn in Figure 5.17. Some of the advantages of deriving the mean curve of a phase error detector are

- finding the equilibrium region in which the phase error detector operates in a linear fashion,
- indicating the areas of phase ambiguity (as we shortly see in a decision-directed case), and

- computing the phase error detector gain K_D through the slope of the mean curve at the origin.

If it is linear around the origin,

$$\overline{e_D} \approx K_D \cdot \theta_{\Delta:e}$$

Computing the gain K_D is very important because this value is then used to derive the values of the loop constants K_P and K_i of the loop filter, as detailed in Eq (4.7) of Chapter 4.

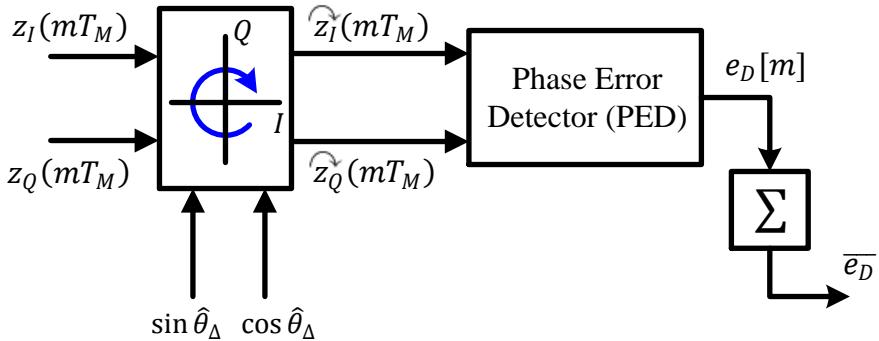


Figure 5.17: A block diagram for measuring the mean curve of a phase error detector. Here, $\hat{\theta}_\Delta$ is a test phase that is varied such that $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$ spans the whole range $-\pi \leq \theta_{\Delta:e} \leq +\pi$

Note 5.7 Quality metric of an error detector

A measure of performance for an error detector is the ratio of its squared mean value to the variance around the origin. From the shape of the S-curve, we can deduce that the larger this ratio is, the quicker is the convergence to the steady state value. For simplicity, we will only focus on the mean curve in this chapter.

Mean curve for Data-Aided PED

The mathematical expression for the mean curve of a data-aided phase difference PED is derived in Appendix 5.9 and is given by

$$\overline{e_D} = \theta_{\Delta:e} \quad (5.22)$$

There are a few observations in regards to this mean curve.

- Since the form of the phase difference PED involves a \angle operation, the amplitude of the Rx signal becomes irrelevant. So in this context, we have not included this amplitude in our expressions yet.
- From the above expression, the gain of the phase difference PED can easily be seen as

$$K_D = 1 \quad (5.23)$$

- The mean curve for data-aided phase difference PED is drawn in Figure 5.18 which is an ideal linear curve in the whole range $-\pi \leq \theta_{\Delta:e} \leq +\pi$. The positive slope around zero only occurs at $\theta_{\Delta:e} = 0$, so there is no ambiguity in the final locked phase (we will soon define phase ambiguity during our discussion on decision-directed PED next).
- This stable locking point is indicated by arrows converging towards zero at the positive slope. A negative slope, on the other hand, does not generate a stable locking point. This is also indicated by arrows converging away from the zero at the edges.

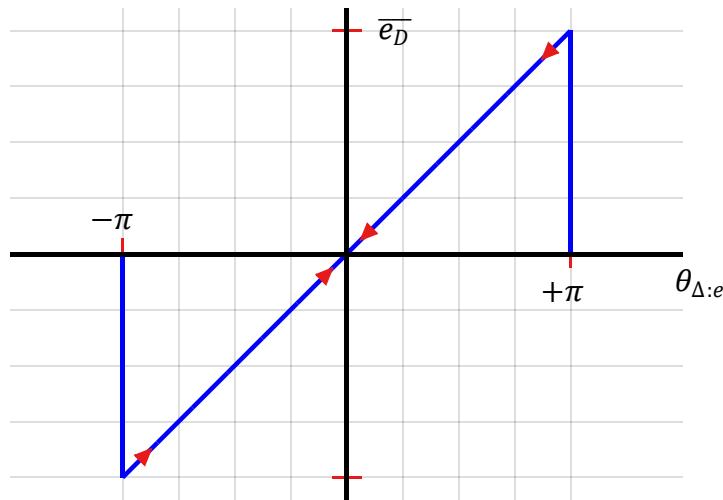


Figure 5.18: Mean curve for data-aided phase difference PED. Observe the positive slope occurring only at $\theta_{\Delta:e} = 0$ and linearity over the whole range $-\pi \leq \theta_{\Delta:e} \leq +\pi$

Mean Curve for Decision-Directed PED

Now what happens when the training sequence $a[m]$ is not available? For example, when the training is finished and data segment of the packet has started, or when the system is required to operate in a non-data-aided mode. One solution at high SNR is that the decisions at the output of the detector are reliable enough to be used in place of known data. This leads to a *decision-directed synchronizer*.

For a QPSK constellation, the decision device generates the output $\{\hat{a}_I[m], \hat{a}_Q[m]\}$ that has the minimum Euclidean distance from the de-rotated matched filter output $\{\hat{z}_I(mT_M), \hat{z}_Q(mT_M)\}$. This is equivalent to taking the sign of the de-rotated matched filter output scaled with the amplitude A , because QPSK is just a combination of two orthogonal BPSK streams for which a sign operation is enough to find the symbol estimate.

Mathematically, the QPSK symbol estimates can be written as

$$\begin{aligned}\hat{a}_I[m] &= A \times \text{sign} \left\{ \widehat{z}_I(mT_M) \right\} \\ \hat{a}_Q[m] &= A \times \text{sign} \left\{ \widehat{z}_Q(mT_M) \right\}\end{aligned}\quad (5.24)$$

This approach maps the de-rotated matched filter outputs to the nearest constellation points. The phase error detector is then given as

$$e_D[m] = \angle \left\{ \hat{a}^*[m] \widehat{z}(mT_M) \right\} \quad (5.25)$$

for which an inverse tangent version is

$$e_D[m] = \tan^{-1} \frac{\hat{a}_I[m] \widehat{z}_Q(mT_M) - \hat{a}_Q[m] \widehat{z}_I(mT_M)}{\hat{a}_I[m] \widehat{z}_I(mT_M) + \hat{a}_Q[m] \widehat{z}_Q(mT_M)} \quad (5.26)$$

A block diagram for implementation of a decision-directed PED and QPSK modulation is drawn in Figure 5.20. Here, the decision is the sign of the de-rotated matched filter output as evident from Eq (5.24), which is drawn just like the shape of a sign function.

Interestingly, if the phase offset θ_Δ is within the range $\pm\pi/4$ (and the SNR is high), the detector decisions are correct and both the data-aided and decision-directed approaches produce the same result. However, for θ_Δ outside this range, the detector maps its decisions to another closest constellation point, as illustrated in Figure 5.19.

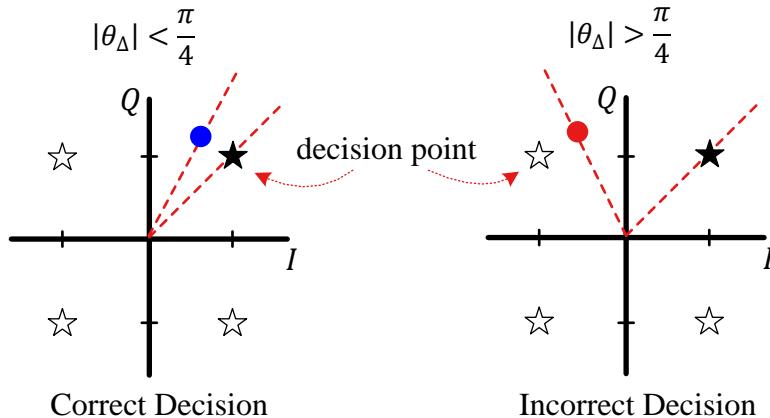


Figure 5.19: Minimum distance decision mapping for a QPSK constellation based on the sign of de-rotated matched filter output. On the left, $\theta_\Delta < \pi/4$, so the detector decision is correct. On the right, $\theta_\Delta > \pi/4$, so the detector decision is wrong

When the whole constellation is taken into account, then we can see different decision regions as well and how a phase offset affects the decisions. Observe from Figure 5.21 that when the minimum distance rule is employed to compute detector decisions, $\hat{\theta}_\Delta$ will always appear within $\{-\pi/4, +\pi/4\}$ regardless of the actual data symbol. Therefore, it is offset from actual θ_Δ by a multiple of $\pi/2$. Starting from the

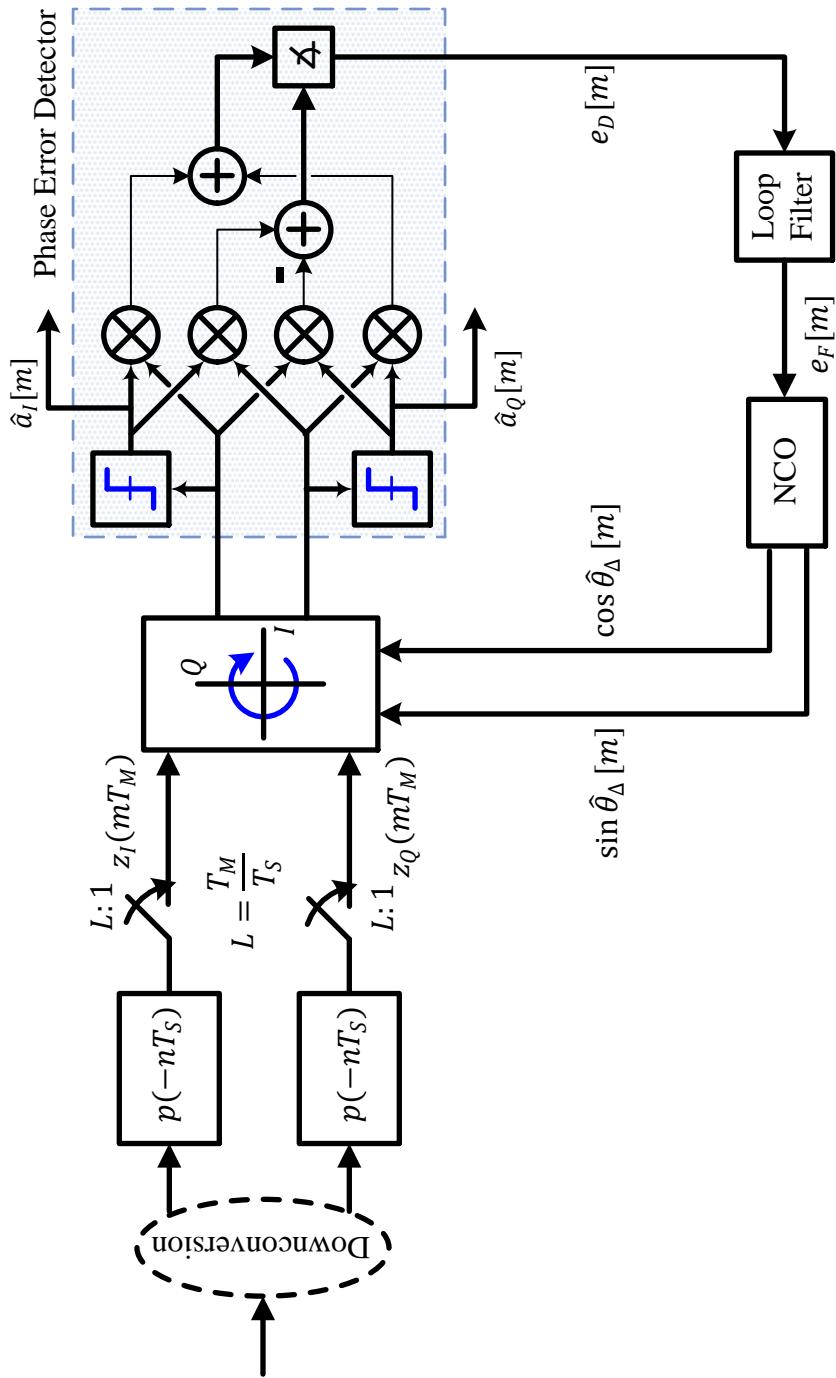


Figure 5.20: A block diagram for a QPSK carrier phase PLL employing the phase difference PED in decision-directed mode

reference symbol in Figure 5.21 and stepping anticlockwise,

$$\theta_{\Delta} = \begin{cases} 0 \cdot \frac{\pi}{2} + \hat{\theta}_{\Delta} & -\frac{\pi}{4} \leq \theta_{\Delta} \leq +\frac{\pi}{4} \\ +1 \cdot \frac{\pi}{2} + \hat{\theta}_{\Delta} & +\frac{\pi}{4} \leq \theta_{\Delta} \leq +\frac{3\pi}{4} \\ +2 \cdot \frac{\pi}{2} + \hat{\theta}_{\Delta} & +\frac{3\pi}{4} \leq \theta_{\Delta} \leq +\pi \\ -2 \cdot \frac{\pi}{2} + \hat{\theta}_{\Delta} & -\pi \leq \theta_{\Delta} \leq -\frac{3\pi}{4} \\ -1 \cdot \frac{\pi}{2} + \hat{\theta}_{\Delta} & -\frac{3\pi}{4} \leq \theta_{\Delta} \leq -\frac{\pi}{4} \end{cases}$$

Compare the decisions regions on the right hand side of the above equation with the blue arrows in Figure 5.21. With the information above at hand and recalling that the

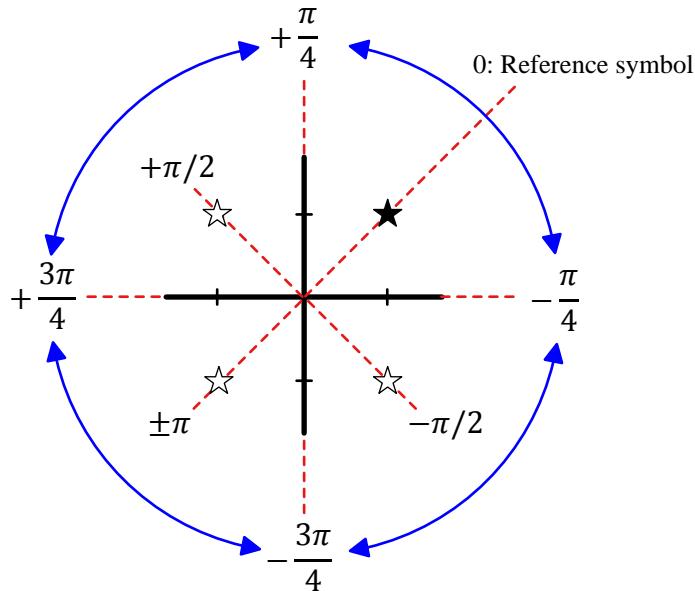


Figure 5.21: In decision-directed scenario, the actual phase offset θ_{Δ} is offset from $\hat{\theta}_{\Delta}$ by a multiple of $\pi/2$

mean curve expression for data-aided scenario is given by

$$\overline{e_D} = \theta_{\Delta:e},$$

as well as the definition of $\theta_{\Delta:e} = \theta_{\Delta} - \hat{\theta}_{\Delta}$, the mean curve for the decision-directed case is

$$\overline{e_D} = \begin{cases} \theta_{\Delta:e} + \pi & -\pi \leq \theta_{\Delta:e} \leq -\frac{3\pi}{4} \\ \theta_{\Delta:e} + \frac{\pi}{2} & -\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq -\frac{\pi}{4} \\ \theta_{\Delta:e} & -\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{\pi}{4} \\ \theta_{\Delta:e} - \frac{\pi}{2} & +\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{3\pi}{4} \\ \theta_{\Delta:e} - \pi & +\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq +\pi \end{cases} \quad (5.27)$$

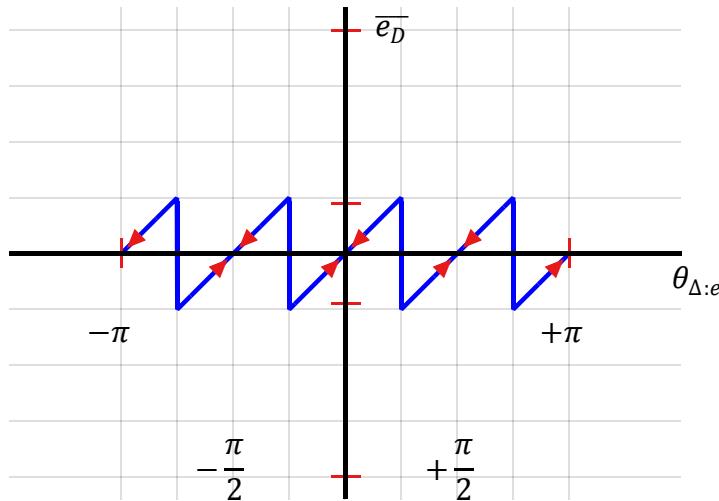


Figure 5.22: Mean curve for decision-directed phase difference PED. Observe the positive slope at $\theta_{\Delta:e} = 0, \pm\pi/2$ and π due to the $\pi/2$ rotational symmetry of the QPSK constellation. Also notice the linearity of the curve over the expected range

This is drawn in Figure 5.22. As opposed to the mean curve of data-aided phase difference PED, it crosses zero with a positive slope (and hence stable lock points) at

$$\theta_{\Delta:e} = -\frac{\pi}{2}, 0, +\frac{\pi}{2}, \pi$$

which leads to a $\pi/2$ phase ambiguity. This is indicated by arrows converging towards zeros at the positive slopes. Consequently, the QPSK carrier phase PLL employing a decision-directed phase difference PED can compensate for a carrier phase offset but it will be unknown which of the above four points it has locked to. Different techniques such as inserting a unique word or differential encoding/decoding are utilized for resolving this ambiguity.

Finally, from the expression in Eq (5.27), the gain of the phase difference PED can be expressed as

$$K_D = 1 \quad (5.28)$$

BPSK Modulation

The carrier phase synchronization for a BPSK modulation scheme can be developed in a very similar manner as with QPSK. The major difference is that while a Q arm is absent in BPSK as far as the data modulation is concerned, this Q branch is necessary to implement in the Rx for the phase synchronization purpose. The combination of both I and Q branches as a complex signal restores the loss of amplitude and enables simple phase synchronization.

In a similar manner as before, a phase difference PED for a BPSK modulation can be constructed by equating $a_Q[m] = 0$ in QPSK case, see Eq (5.18) and Eq (5.19). For a known preamble $a_I[m]$, the expression for a data-aided PED can then be simpli-

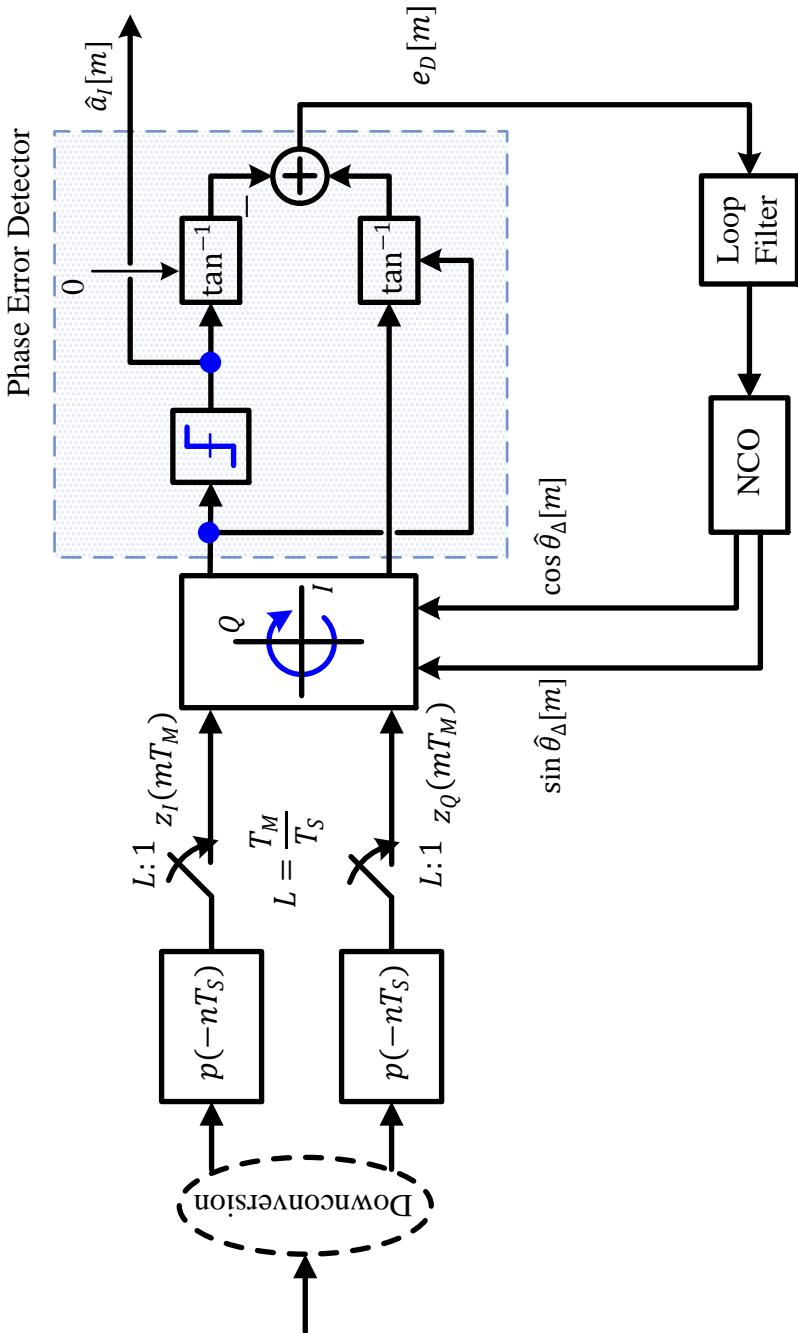


Figure 5.23: A block diagram for implementing the decision-directed phase difference PED in a BPSK Rx

fied as

$$\begin{aligned}
e_D[m] &= \angle \left\{ a^*[m] \tilde{z}(mT_M) \right\} \\
&= \tan^{-1} \frac{\tilde{z}_Q(mT_M)}{\tilde{z}_I(mT_M)} - \tan^{-1} \frac{0}{a_I[m]} \\
&= \tan^{-1} \frac{\tilde{z}_Q(mT_M)}{\tilde{z}_I(mT_M)} - \begin{cases} 0 & a_I[m] = +A \\ \pi & a_I[m] = -A \end{cases}
\end{aligned} \tag{5.29}$$

where we have utilized the definition of a four-quadrant inverse tangent. In the absence of any training, the expression for a decision-directed PED is given as

$$\begin{aligned}
e_D[m] &= \angle \left\{ \hat{a}^*[m] \tilde{z}(mT_M) \right\} \\
&= \tan^{-1} \frac{\tilde{z}_Q(mT_M)}{\tilde{z}_I(mT_M)} - \tan^{-1} \frac{0}{\hat{a}_I[m]} \\
&= \tan^{-1} \frac{\tilde{z}_Q(mT_M)}{\tilde{z}_I(mT_M)} - \begin{cases} 0 & \hat{a}_I[m] = +A \\ \pi & \hat{a}_I[m] = -A \end{cases} \\
&= \tan^{-1} \frac{\tilde{z}_Q(mT_M)}{\tilde{z}_I(mT_M)} - \begin{cases} 0 & \tilde{z}_I(mT_M) > 0 \\ \pi & \tilde{z}_I(mT_M) < 0 \end{cases}
\end{aligned} \tag{5.30}$$

where the third relation signifies the fact that a decision $\hat{a}_I[m]$ is taken according to the sign of $\tilde{z}_I(mT_M)$. A block diagram for its implementation is drawn in Figure 5.23.

On the same note, the mean curves can also be derived as in QPSK. For instance, the mean curve for a data-aided phase difference PED in BPSK case is again written as

$$\overline{e_D} = \theta_{\Delta:e}$$

Similarly, the mean curve for a decision-directed phase difference PED is

$$\overline{e_D} = \begin{cases} \theta_{\Delta:e} + \pi & -\pi \leq \theta_{\Delta:e} \leq -\frac{\pi}{2} \\ \theta_{\Delta:e} & -\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\frac{\pi}{2} \\ \theta_{\Delta:e} - \pi & +\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\pi \end{cases}$$

These curves are illustrated in Figure 5.24. While both are completely linear over the whole range $-\pi \leq \theta_{\Delta:e} \leq +\pi$, notice the opposite directions of arrows for $\theta_{\Delta:e} = \pm\pi$ for data-aided and decision-directed PEDs.

5.5.4 General Approach to Feedback Synchronization

From the bottom half of Figure 5.16 where a block diagram of the data-aided phase difference PED was drawn for a QPSK constellation, a general approach to feedback phase synchronization can be constructed as follows.

- 1. Correction** Apply the correction to the input through clockwise rotation by the updated estimate of the offset parameter.

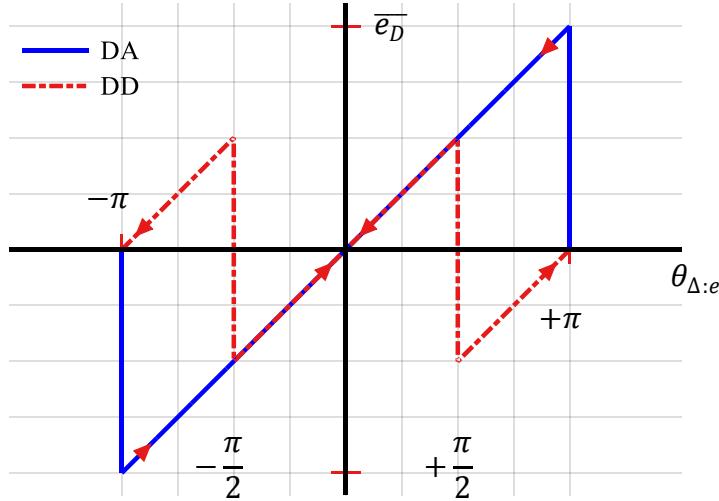


Figure 5.24: Mean curves for data-aided (DA) and decision-directed (DD) phase difference phase error detectors for BPSK

2. **Modulation removal** Remove the modulation from the matched filter outputs and generate an error signal proportional to the residual offset parameter.
3. **Filtering** Filter the noise through a loop filter (an integrator is required for tracking a small frequency offset).
4. **Adjustment** Tune the Numerically Controlled Oscillator (NCO) output according to the error signal at the filter output.

Compare these steps with those in Section 5.5.2 where a general approach to feedforward synchronization was studied, or equivalently, Figure 5.16 with Figure 5.13. It is evident that both the feedforward and the feedback methods are effectively implementing the same strategy with the following differences.

Order of the blocks To some extent, the order of the blocks is in reverse order. For example, the feedback technique applies the correction first and then figures out the remaining phase offset while the feedforward method finds the phase offset first and applies the correction at the last step.

Processing The feedback system works *iteratively* in small steps by operating on a sample-by-sample basis and gradually converging towards the final solution. On the other hand, the feedforward procedure waits for the buffer to be filled with a data set, then estimates and corrects the desired offset in one go, i.e., through *batch processing*.

5.5.5 Feedback: Cross Product Phase Error Detector

Now we describe another extremely useful phase error detector that is widely used in digital communication systems. Remember that starting from the correlation, we ar-

rived at the relation in Eq (5.12) reproduced below which formed the basis of both the conjugate product feedforward and phase difference feedback synchronization techniques.

$$\begin{aligned} \text{corr}[0] &= \sum_{m=0}^{N_0-1} a_I[m] \left\{ z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \right\} \\ &\quad + \sum_{m=0}^{N_0-1} a_Q[m] \left\{ z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \right\} \end{aligned}$$

For the phase difference PED, we simply maximized this correlation by equating the argument of the cosine to zero. Here, we exploit the other route to maximizing a function by taking its derivative and equate that to zero. If you do not know about differentiation, just remember that the derivative at a point is defined as the slope of the line tangent to the curve at that point. This tangent indicates the rate of change of the signal at that point as explained in Section 2.6. There, we also saw the derivatives of $\cos \theta_\Delta$ and $\sin \theta_\Delta$ and argued that the maximum of a function can be found by finding its rate of change and equating it to zero.

Differentiating $\text{corr}[0]$ with respect to θ_Δ ,

$$\begin{aligned} \frac{d \text{corr}[0]}{d \theta_\Delta} &= \sum_{m=0}^{N_0-1} a_I[m] \left\{ -z_I(mT_M) \sin \hat{\theta}_\Delta + z_Q(mT_M) \cos \hat{\theta}_\Delta \right\} \\ &\quad + \sum_{m=0}^{N_0-1} a_Q[m] \left\{ -z_Q(mT_M) \sin \hat{\theta}_\Delta - z_I(mT_M) \cos \hat{\theta}_\Delta \right\} \end{aligned} \quad (5.31)$$

Utilizing the clockwise rotation rule $I \cos \theta + Q \sin \theta$ and $Q \cos \theta - I \sin \theta$, the de-rotated matched filter outputs can be written as below. These relations were also defined in Eq (5.17) during the discussion on phase difference PED.

$$\begin{aligned} I \rightarrow \quad \widehat{z}_I(mT_M) &= z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \\ Q \uparrow \quad \widehat{z}_Q(mT_M) &= z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \end{aligned}$$

Plugging them back in Eq (5.31) produces

$$\frac{d \text{corr}[0]}{d \theta_\Delta} = \sum_{m=0}^{N_0-1} \left\{ a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M) \right\}$$

which can now be equated to zero to yield the maximum.

$$\sum_{m=0}^{N_p-1} a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M) = 0$$

In a feedback setting, rather than summing all the values and finding a single estimate, the objective is to steer the derivative in a sample-by-sample manner. As opposed to the feedforward scenario, this form of the equation leads to the feedback structure of the phase synchronization setup. Setting it to zero inherently invokes an iterative solution which is drawn in Figure 5.25. Consequently, a Cross Product Phase

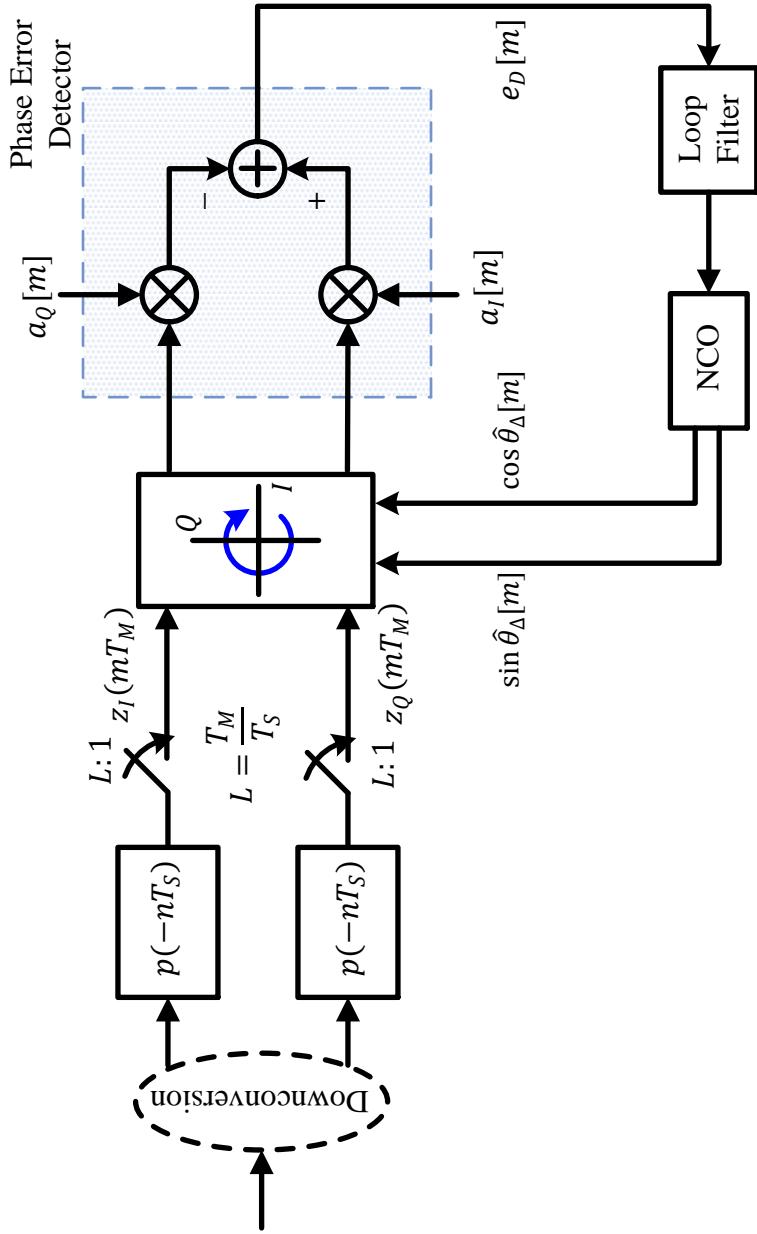


Figure 5.25: A block diagram for a QPSK carrier phase PLL employing the cross product PED in data-aided mode

Error Detector (PED) is defined as

$$e_D[m] = a_I[m]\hat{z}_Q(mT_M) - a_Q[m]\hat{z}_I(mT_M) \quad (5.32)$$

In summary, the correlation process is de-rotating the rotated matched filter outputs $z(nT_S)$ back by an angle θ_Δ in an iterative manner. Since $z(mT_M)$ ideally should have been mapped to the actual constellation but get rotated anticlockwise due to the presence of the phase offset θ_Δ , the need for this compensation in a clockwise direction makes sense.

We arrived at this relation through taking the derivative and equating it to zero. Since this was the solution to the correlation equation which comes from the maximum likelihood estimation, *the cross product PED is also known as maximum likelihood PED*.

To gain an intuitive understanding behind the term *cross product PED*, first recall the feedforward conjugate product estimator derived in Eq (5.13) and reproduced below.

$$\hat{\theta}_\Delta = \angle \left\{ \sum_{m=0}^{N_p-1} a^*[m]z(mT_M) \right\}$$

Writing in *IQ* form,

$$\begin{aligned} I &\rightarrow \left\{ a^*[m]z(mT_M) \right\}_I = a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M) \\ Q &\uparrow \qquad \qquad \qquad \left\{ a^*[m]z(mT_M) \right\}_Q = a_I[m]z_Q(mT_M) - a_Q[m]z_I(mT_M) \end{aligned}$$

Compare the *Q* component of the above equation with cross product PED in Eq (5.32). They are very similar with the only difference being the use of de-rotated matched filter output $\{\hat{z}_I(mT_M), \hat{z}_Q(mT_M)\}$ instead of the uncompensated matched filter output $\{z_I(mT_M), z_Q(mT_M)\}$.

Now recall that when a complex number V is rotated towards the inphase axis, its quadrature component V_Q naturally moves towards zero. Consequently, most of the magnitude $\sqrt{V_I^2 + V_Q^2}$ appears on the inphase axis and contributes towards V_I .

Similarly in the above equation, as the *Q* term above approaches zero, the *I* term above converges towards its maximum. Considering that the de-rotated terms $\hat{z}_I(mT_M)$ and $\hat{z}_Q(mT_M)$ are being used instead of $z_I(mT_M)$ and $z_Q(mT_M)$, we can conclude that *the feedback structure or PLL is trying to drive the quadrature part of this conjugate product to zero*. No more phase offset induced rotation survives!

The name cross product PED originates from this cross-axes product $I \cdot Q - Q \cdot I$ being used as the expression of interest and driven to zero.

Mean Curve for Data-Aided PED

Before we discuss the mean curve for a cross product PED, recall that the Rx signal amplitude is another unknown parameter. This amplitude is a result of the gains and losses when the signal travels through the wireless channel and the Tx/Rx frontend components such as antennas, filters, amplifiers, mixers and so on. Previously, we omitted the Rx signal amplitude for simplicity by normalizing it to 1. We were justified

in doing so because the form of the phase difference PED involves a \angle operation hence rendering the amplitude of the Rx signal irrelevant. Nonetheless, in the current context, *the cross product PED directly involves signal values* and not their phases. Hence, it is imperative to include the Rx amplitude in our current calculations and denote it as γ .

The mean curve for a data-aided cross product PED is derived in Appendix 5.9 where the final expression is given by

$$\overline{e_D} = 2A^2\gamma \sin \theta_{\Delta:e} \quad (5.33)$$

This sinusoidal mean curve is drawn in Figure 5.26. Notice that it closely follows the linear relationship in the range $\{-\pi/4, +\pi/4\}$ before the curve flattens out around $\pm\pi/2$ and tends towards zero again till $\theta_{\Delta:e}$ reaches $\pm\pi$. However, the positive slope around zero only occurs at $\theta_{\Delta:e} = 0$, so there is no ambiguity in the final locked phase. Again, this stable locking point is indicated by arrows converging towards zero at the positive slope.

Recall from Chapter 4 that a negative slope does not generate a stable locking point. This is also indicated by arrows converging away from the zero at the negative slope. The dashed line – drawn for comparison – is the linear relation $\overline{e_D} = \theta_{\Delta:e}$ which naturally extends from $-\pi$ to $+\pi$ for $-\pi \leq \theta_{\Delta:e} \leq +\pi$ (it was the mean curve for phase difference PED).

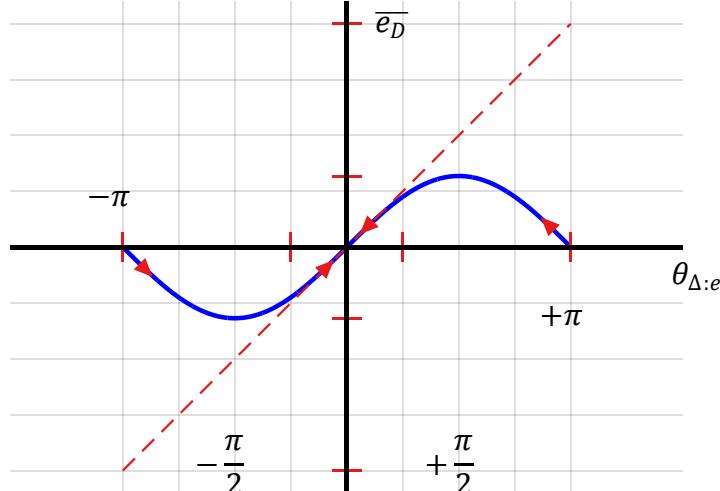


Figure 5.26: Mean curve for data-aided cross product PED. Observe the positive slope occurring only at $\theta_{\Delta:e} = 0$

Let us find out the reason behind this sinusoidal shape. First, consider the expression for the cross product PED.

$$e_D[m] = \gamma \left(a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M) \right)$$

Fixing $a_I[m]$ and $a_Q[m]$ to $\{+A, +A\}$ (a similar argument holds for any other pair as well with just a change of signs), the cross product phase error detector $e_D[m]$ unfolds as a difference between Q and I arms of the de-rotated matched filter output.

$$e_D[m] = \text{constant} \cdot \left(\hat{z}_Q(mT_M) - \hat{z}_I(mT_M) \right)$$

A de-rotated matched filter output leaves the remaining phase as $\theta_{\Delta:e}$. Starting from the top right reference symbol in Figure 5.27, an anticlockwise rotation reveals that for 0 to $\pi/4$, Q increases while I decreases, resulting in an overall increase in the difference $Q - I$. From $\pi/4$ to $\pi/2$, Q also decreases but I continues to decrease further at a rate faster than Q , thus resulting in an overall increase in $Q - I$. Following the same logic onwards along the red arrows, we can see that the mean curve for cross product PED follows a *sinusoidal shape*.

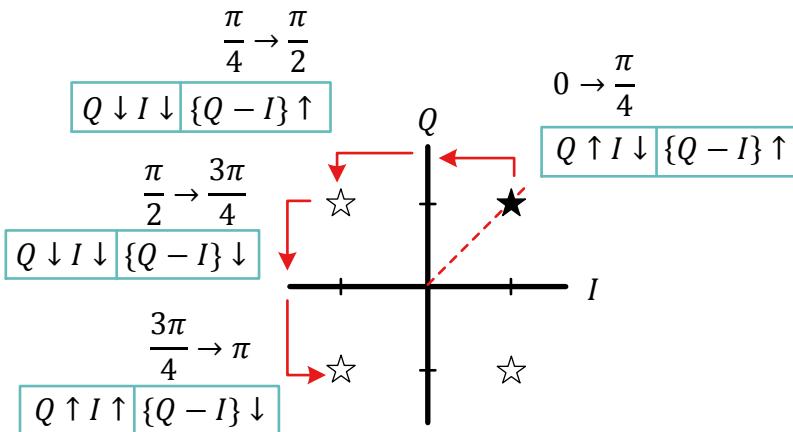


Figure 5.27: Why the mean curve for data-aided cross product phase error detector exhibits a sinusoidal shape

Note 5.8 A wrong path?

We are interested in the logic behind this sinusoidal shape because for large phase differences, the steering force generated by such an error signal becomes very small. It seems that the maximum correlation approach (or maximum likelihood) in this case has lead us to an inferior estimation procedure.

We saw the phase difference PED in Section 5.5.3 that is linear over the whole range $-\pi \leq \theta_{\Delta:e} + \pi$ and overcomes this limitation. Interestingly, we will find that the cross product PED is not that useless at all and why it is actually the most practical PED in implementing a phase synchronization system.

Remember that the inverse tangent version of phase difference PED in Eq (5.14) was derived through the same equation as

$$\hat{\theta}_{\Delta} = \tan^{-1} \left\{ \frac{\sum_{m=0}^{N_p-1} a_I[m]z_Q(mT_M) - a_Q[m]z_I(mT_M)}{\sum_{m=0}^{N_p-1} a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M)} \right\}$$

Here, the phase is explicitly taken out. However, when the cross product phase error detector takes the form

$$\sum_{m=0}^{N_p-1} a_I[m] \hat{z}_Q(mT_M) - a_Q[m] \hat{z}_I(mT_M) = 0,$$

which is quite similar to the numerator above, the phase appears implicitly within the de-rotated matched filter outputs $\hat{z}_I(mT_M)$ and $\hat{z}_Q(mT_M)$. Importantly, this expression assumes that

- the phase error is corrected once and for all,
- after de-rotation, all the energy appears in I arm of the conjugate product – the denominator, and
- the Q arm of the conjugate product does not survive anymore and it is legitimate to equate it to zero.

Since this is not what happens in the modified PLL here, we conclude that a feedback implementation that converges in phase steps was our own choice that created this sinusoidal relation.

The largest value in this mean curve is $2\gamma A^2$. Finally, the gain of cross product PED in data-aided mode can be seen from Eq (5.33) as

$$K_D = 2\gamma A^2 \quad (5.34)$$

because for $\theta_{\Delta:e} \approx 0$, $\sin \theta_{\Delta:e} \approx \theta_{\Delta:e}$ and hence $\bar{e}_D \approx 2\gamma A^2 \theta_{\Delta:e}$. In the factor $2A^2$, A is the symbol level while 2 appears due to the underlying modulation (QPSK in this case) and cancels out with the normalizing factor squared for that modulation ($1/\sqrt{2}$ here). This gain will be used in the design of a PLL employing a data-aided cross product PED.

Compare the mean curve of phase difference PED in Figure 5.18 with that of cross product PED in Figure 5.26. For $\gamma = 1$ and large phase deviations, the error signal generated by the phase difference PED (a linear shape) is much greater than the one generated by the cross product PED (a sinusoidal shape). Subsequently, a larger adjustment to the remaining phase helps in quicker acquisition.

Mean Curve for Decision-Directed PED

When the training symbols are not available, e.g., in a blind receiver or when the training sequence is finished and the tracking loops are enabled, the decisions from the Rx can be fed back to use as a replacement of known data. In a QPSK constellation, the decisions are expressed as

$$\begin{aligned} \hat{a}_I[m] &= A \times \text{sign} \left\{ \hat{z}_I(mT_M) \right\} \\ \hat{a}_Q[m] &= A \times \text{sign} \left\{ \hat{z}_Q(mT_M) \right\} \end{aligned}$$

Consequently, the training symbols $a[m]$ can be replaced with the corresponding decisions $\hat{a}[m]$ to produce a decision-directed cross product PED as

$$e_D[m] = \gamma \left(\hat{a}_I[m] \hat{z}_Q(mT_M) - \hat{a}_Q[m] \hat{z}_I(mT_M) \right) \quad (5.35)$$

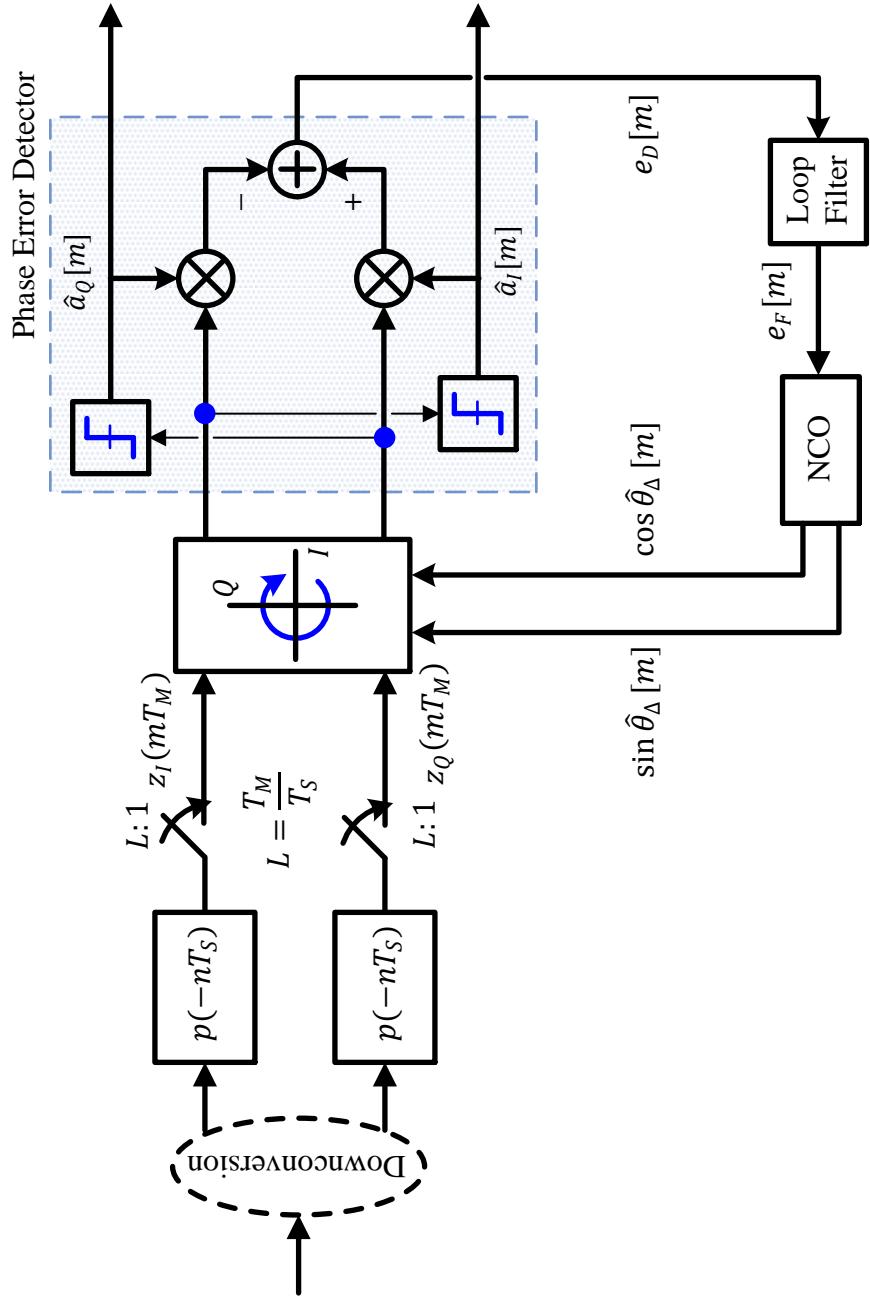


Figure 5.28: A block diagram for a QPSK carrier phase PLL employing the cross product PED in decision-directed mode

Summarizing the above findings, a block diagram of phase synchronization unit implementing a decision-directed cross product PED is illustrated in Figure 5.28.

The mean curve for this PED can be derived in a very similar manner as in the decision-directed case of phase difference PED in Section 5.5.3. Owing to the $\pi/2$ rotational symmetry of QPSK constellation, a phase ambiguity of $\pi/2$ shows up in the expression here as well.

The mean curve in the data-aided scenario in Eq (5.33) was

$$\bar{e_D} = 2\gamma A^2 \sin \theta_{\Delta:e}$$

In this decision-directed scenario, it should be the same as above except that there is an additional factor of a $\pi/2$ multiple as before. Therefore, comparing with Eq (5.27), the mean curve for a decision-directed cross product PED can be expressed as

$$\bar{e_D} = 2\gamma A^2 \begin{cases} \sin(\theta_{\Delta:e} + \pi) & -\pi \leq \theta_{\Delta:e} \leq -\frac{3\pi}{4} \\ \sin(\theta_{\Delta:e} + \frac{\pi}{2}) & -\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq -\frac{\pi}{4} \\ \sin(\theta_{\Delta:e}) & -\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{\pi}{4} \\ \sin(\theta_{\Delta:e} - \frac{\pi}{2}) & +\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{3\pi}{4} \\ \sin(\theta_{\Delta:e} - \pi) & +\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq +\pi \end{cases} \quad (5.36)$$

Trigonometric identities $\sin(A \pm \pi/2) = \pm \cos A$ and $\sin(A \pm \pi) = -\sin A$ can be used to simplify the above expression as

$$\bar{e_D} = 2\gamma A^2 \begin{cases} -\sin \theta_{\Delta:e} & -\pi \leq \theta_{\Delta:e} \leq -\frac{3\pi}{4} \\ +\cos \theta_{\Delta:e} & -\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq -\frac{\pi}{4} \\ +\sin \theta_{\Delta:e} & -\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{\pi}{4} \\ -\cos \theta_{\Delta:e} & +\frac{\pi}{4} \leq \theta_{\Delta:e} \leq +\frac{3\pi}{4} \\ -\sin \theta_{\Delta:e} & +\frac{3\pi}{4} \leq \theta_{\Delta:e} \leq +\pi \end{cases} \quad (5.37)$$

This mean curve is drawn in Figure 5.29. As opposed to the mean curve of data-aided cross product PED, it crosses zero with a positive slope (and hence stable lock points) at

$$\theta_{\Delta:e} = -\frac{\pi}{2}, 0, +\frac{\pi}{2}, \pi$$

which leads to a $\pi/2$ phase ambiguity. This is indicated by arrows converging towards zeros at positive slopes.

Consequently, the QPSK carrier phase PLL employing a decision-directed cross product PED can compensate for a carrier phase offset but it will be unknown which of the above four points it has locked to. Different techniques such as inserting a unique word or differential encoding/decoding are utilized for resolving this ambiguity.

Finally, compared with data-aided case where the largest value in the mean curve was $2\gamma A^2$, the mean curve here jumps at $\pi/4$ to yield the value at this discontinuity as

$$2\gamma A^2 \sin \frac{\pi}{4} = \sqrt{2}\gamma A^2$$

Similarly, the gain of cross product PED in decision-directed mode can be derived from Eq (5.37) as follows. For $\theta_{\Delta:e} \approx 0$, $\sin \theta_{\Delta:e} \approx \theta_{\Delta:e}$ and hence $\bar{e_D} \approx 2\gamma A^2 \theta_{\Delta:e}$. Thus,

$$K_D = 2\gamma A^2 \quad (5.38)$$

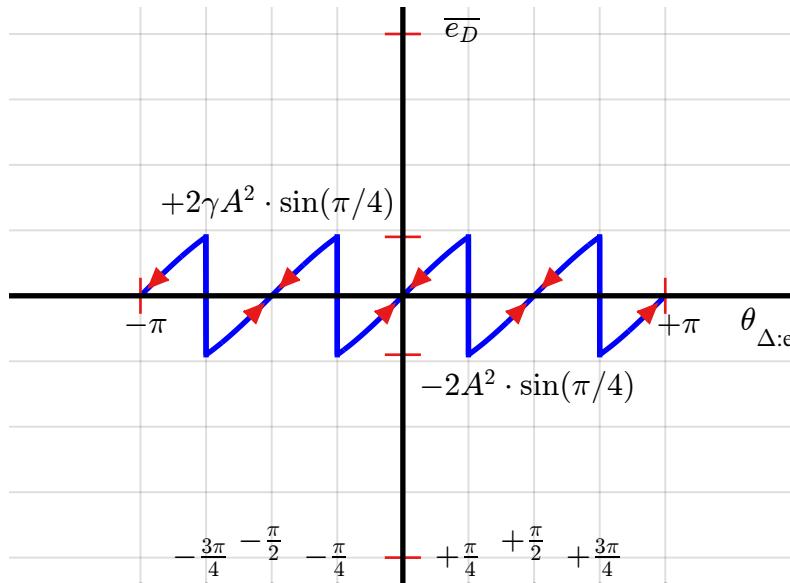


Figure 5.29: Mean curve for decision-directed cross product PED. Observe the $\frac{\pi}{2}$ phase ambiguity producing 4 stable lock points at $-\frac{\pi}{2}, 0, +\frac{\pi}{2}$ and π

This gain will be used in the design of a PLL employing a decision-directed cross product PED.

Example 5.2

Assume that our task is to recover a signal coming from a deep space mission with the following information.

- The modulation is unit energy QPSK, and
- There are 100 training symbols available at the start of the transmission.
- There is no impairment present in the link except a constant phase offset θ_Δ .
- To closely watch the phase recovery process, the SNR is high.

Let us choose a feedback implementation for this purpose, i.e., the Rx needs to be synchronized in phase with the Tx through a PLL. The actual loop noise bandwidth B_n is chosen based on various calculations that determine the acquisition time, noise performance, pull-in range, and so on. For this simple application at 1 sample/symbol, assume the following parameters.

$$B_n T_M = 0.05$$

$$\zeta = 0.8$$

Furthermore, a cross product phase error detector seems a good enough option. For a unit energy QPSK modulation,

$$A = \frac{1}{\sqrt{2}}$$

Finally, the Rx signal amplitude γ is assumed equal to 1 since an Automatic Gain Control adjusts it to a predetermined value. Consequently, from Eq (5.34), the phase detector

gain is unity as well.

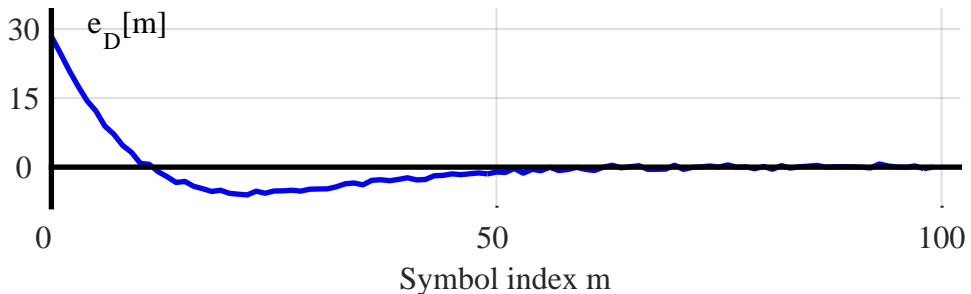
$$K_D = 2\gamma A^2 = 1$$

The NCO gain K_0 is already set to 1. Putting the above values of K_0 , K_D , ζ and B_n in Eq (4.7) yields the (approximate) PI loop filter coefficients below.

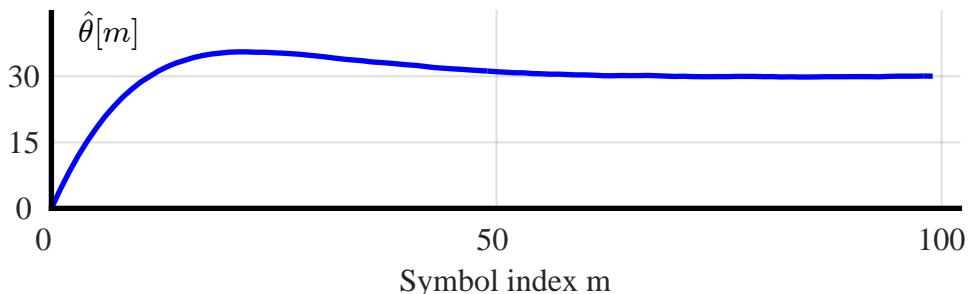
$$K_P = 0.1438$$

$$K_i = 0.0081$$

If we had chosen a smaller loop noise bandwidth B_n , the values of K_P and K_i would have been correspondingly smaller. These relatively large values affect a larger input e_F to the NCO and hence a larger correction. For a high SNR case, this implies a faster convergence to the equilibrium. These values are used to implement a PLL to recover the phase offset in the Tx signal.



(a) Phase error detector output $e_D[m]$ in degrees for the QPSK carrier phase PLL



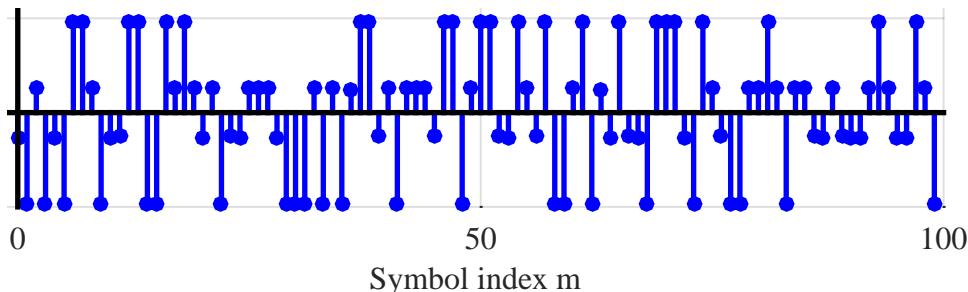
(b) Phase estimate $\hat{\theta}[m]$ in degrees at the output of the NCO. It can be seen to slowly converge to 30° , which is the phase offset of the incoming Tx signal in this example

Figure 5.30: Carrier phase PLL for QPSK modulation based on a cross product PED.
Notice the index m as the PLL is operating at 1 sample/symbol

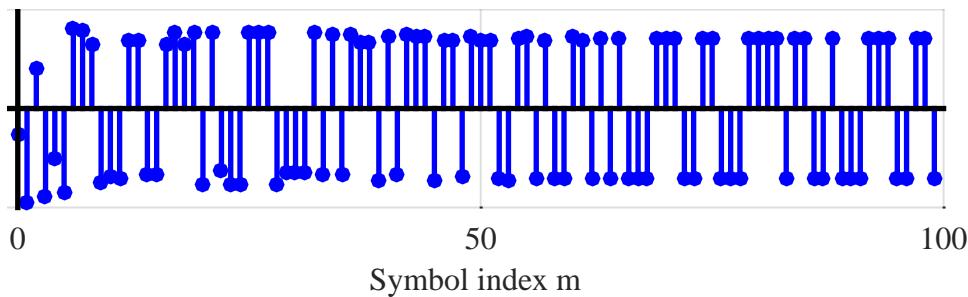
Phase Detector Output and Phase Estimate

The resulting signals $e_D[m]$ and $\hat{\theta}[m]$ are drawn in Figure 5.30a and Figure 5.30b, respectively. Notice that a training sequence with 100 symbols is of a reasonable length for the synchronization process. Both the phase error $e_D[m]$ and the phase estimate $\hat{\theta}[m]$ converge in around 100 symbols in a manner very similar to examples of a regular

PLL in Chapter 4. The overshoot in the response is due to the damping factor ζ being less than 1. For a wider B_n , the convergence would have been more rapid and probably a wise choice for that high an SNR.



(a) Effect of a constant phase offset on inphase part of matched filter output $z_I(mT_M)$



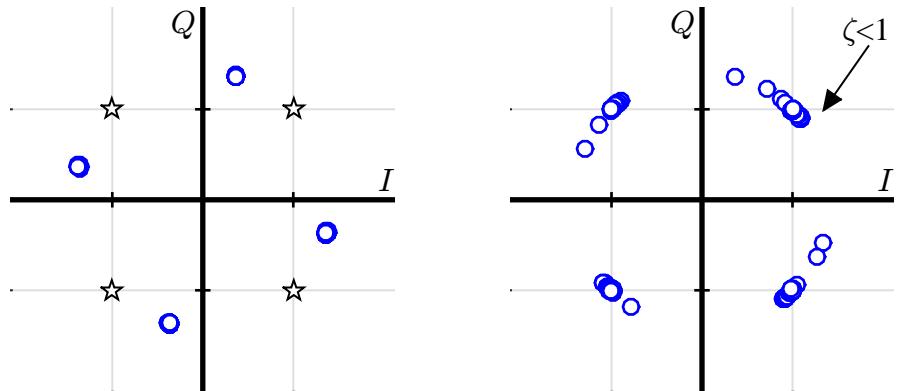
(b) Inphase part of de-rotated matched filter output $\hat{z}_I(mT_M)$ where the effect of phase convergence on symbol estimates can be seen by approaching $\pm 1/\sqrt{2}$ values

Figure 5.31: Phase recovery process

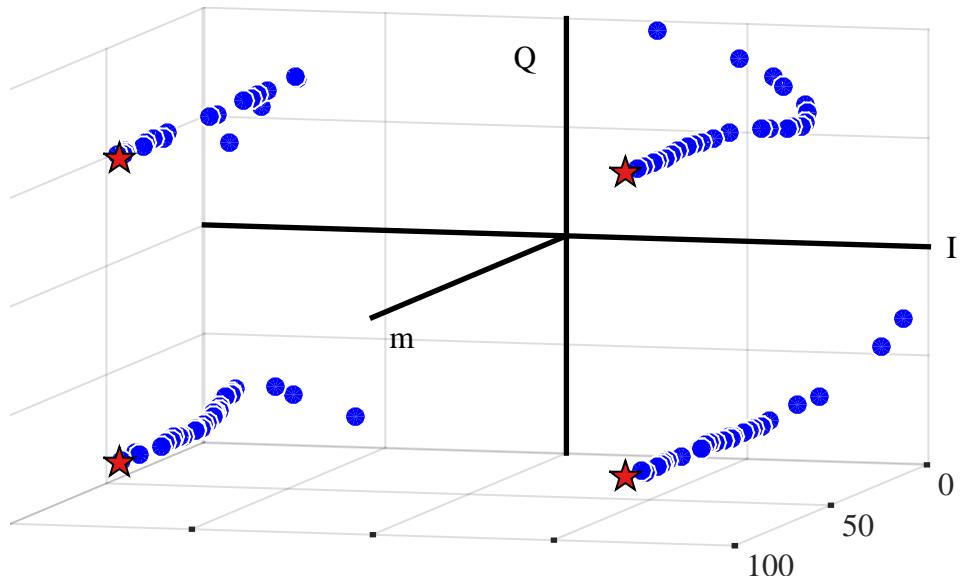
De-rotated Matched Filter Outputs

To observe the effect of phase rotation on matched filter outputs and that of the PLL in recovering the phase, we plot both the inphase part of the matched filtered signal $z_I(mT_M)$ and the inphase part of the de-rotated matched filter output $\hat{z}_I(mT_M)$ in Figure 5.31. Observe the effect of the phase rotation on downsampled symbol estimates as discussed in Section 5.1. Next, in the inphase part of de-rotated matched filter output $\hat{z}_I(mT_M)$, the effect of phase convergence on symbol estimates can be seen by approaching $\pm 1/\sqrt{2}$ values.

Finally, the scatter plot of $\hat{z}(mT_M)$ is illustrated in Figure 5.32b which displays the phase recovery process from a constellation point of view in 2D space. The samples $\hat{z}(mT_M)$ mapped on the original constellation diagram can be seen to converge towards the desired locations. Interestingly, the overshoot arising from choosing a damping factor $\zeta < 1$ displays itself in the process of the mapped symbol estimates passing through the constellation points and then returning backwards, just like the phase error detector output and phase estimate in Figure 5.30.



(a) Scatter plot of the original matched filter signal $z(mT_M)$ (b) Scatter plot of de-rotated matched filtered signal $\hat{z}(mT_M)$



(c) A dynamic scatter plot of $\hat{z}(mT_M)$ from Figure 5.32b. Observe how the convergence towards the actual constellation locations follows the same trajectory as $e_D[m]$ and $\hat{\theta}[m]$

Figure 5.32: The phase recovery process from a constellation point of view

Note 5.9 A dynamic scatter plot

This phase recovery process, its overshoot and final convergence are much more interesting to watch from a constellation point of view as it unfolds in a 3D space like a python, which I call a *dynamic scatter plot*. If this page was a 3D box and we could go towards the right side of the scatter plot, we would have gotten Figure 5.32c where the four constellation points are tracing an exact same trajectory as the phase error detector output $e_D[m]$ and the phase estimate $\hat{\theta}[m]$ in Figure 5.30. The varying amount of gap between some constellation points arises from the randomness of the training

sequence consisting of one of out four possible symbols during each T_M .

From this example, we also conclude that a PLL is able to acquire not only a constant phase offset but can also track a slowly varying phase, which by definition is a frequency offset. This is why we had an integrator in the PI loop filter of the PLL. The range of acquisition of this frequency offset is directly proportional to the loop noise bandwidth B_n . For larger frequency offsets, a separate frequency acquisition aid such as a frequency locked loop is required. We will discuss this in more detail in Chapter 6.

BPSK Modulation

Similar to the phase difference PED for BPSK modulation, a cross product PED for BPSK modulation can be constructed by equating $a_Q[m] = 0$ in QPSK case, see Eq (5.32). The expression for a data-aided PED is given by

$$e_D[m] = \gamma \cdot a_I[m] \hat{z}_Q(mT_M) \quad (5.39)$$

and that for a decision-directed PED by

$$e_D[m] = \gamma \cdot \hat{a}_I[m] \hat{z}_Q(mT_M) \quad (5.40)$$

A block diagram for implementing a decision-directed cross product PED is drawn in Figure 5.33. Clearly, as the quadrature component of $z(mT_M)$ goes to zero, so does the PED output $e_D[m]$ and the loop converges.

On the same note, the mean curves can also be derived as in QPSK. The mean curve for a data-aided cross product PED in BPSK case is written as

$$\bar{e}_D = \gamma A^2 \sin \theta_{\Delta:e}$$

where the factor 2 from QPSK expression is now absent because the normalizing factor in BPSK modulation is 1. Similarly, the mean curve for a decision-directed cross product PED is

$$\bar{e}_D = \gamma A^2 \sin \begin{cases} \theta_{\Delta:e} + \pi & -\pi \leq \theta_{\Delta:e} \leq -\frac{\pi}{2} \\ \theta_{\Delta:e} & -\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\frac{\pi}{2} \\ \theta_{\Delta:e} - \pi & +\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\pi \end{cases}$$

which can also be written as

$$\bar{e}_D = \gamma A^2 \begin{cases} -\sin(\theta_{\Delta:e}) & -\pi \leq \theta_{\Delta:e} \leq -\frac{\pi}{2} \\ \sin(\theta_{\Delta:e}) & -\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\frac{\pi}{2} \\ -\sin(\theta_{\Delta:e}) & +\frac{\pi}{2} \leq \theta_{\Delta:e} \leq +\pi \end{cases}$$

These curves are illustrated in Figure 5.34 which reveals the following observations. As opposed to QPSK modulation which has a $2\pi/4 = \pi/2$ phase ambiguity, a BPSK modulation has a $2\pi/2 = \pi$ phase ambiguity. Therefore, the mean curve of the decision-directed PED has two stable lock points, one at $\theta_{\Delta:e} = 0$ and the other at $\theta_{\Delta:e} = \pi$. Notice the directions of the arrows at $\theta_{\Delta:e} = \pm\pi$ (both of which are actually the same point). In data-aided case, the arrow manifests a negative slope, so the direction of movement of the PED output is away from the change in phase error $\theta_{\Delta:e}$. On the other hand, the decision-directed implementation produces a stable locking point at $\pm\pi$ as well. The phase ambiguity needs to be overcome through an external aid such as a unique word or differential encoding.

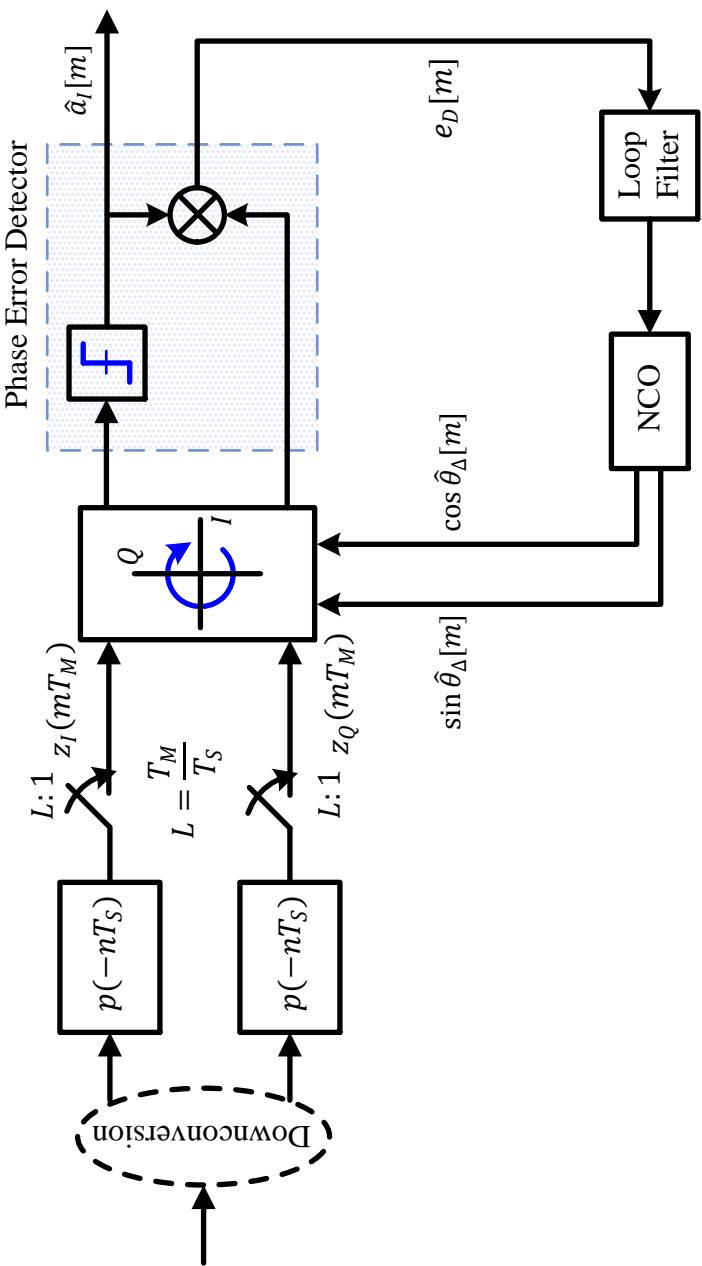


Figure 5.33: A block diagram for implementing the decision-directed cross product PED in a BPSK Rx

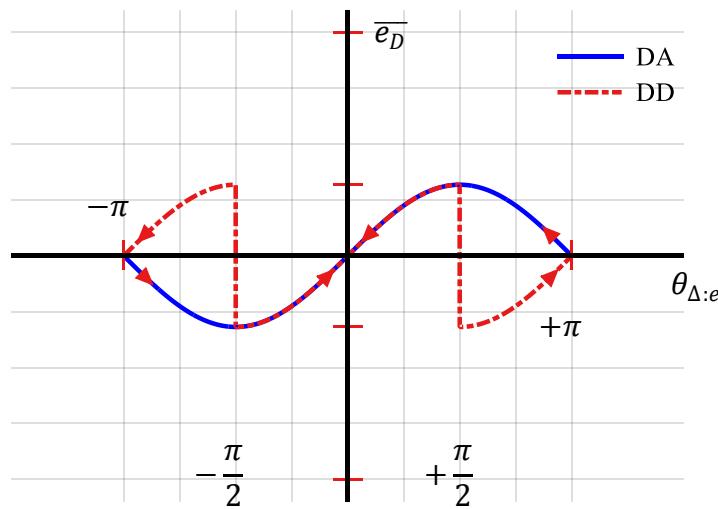


Figure 5.34: Mean curves for data-aided (DA) and decision-directed (DD) cross product phase error detectors for BPSK

5.5.6 A Comparison of Phase Error Detectors

In Section 5.5.5, we found that the sinusoidal shape of the mean curve for a cross product phase error detector arises due to incorporating only the quadrature part of the cross product between the data symbols and matched filter outputs. We saw in Figure 5.26 that the steering force generated by such an error signal with a sinusoidal mean curve becomes very small for large phase differences.

In addition, we established that the phase difference PED has that desirable linear mean curve as it utilizes all the information by using the inphase part of the cross product as well. However, computing its expression requires the following operations during each cycle (which is the same as the symbol time T_M in this discussion).

One Division operation A division operation is inherently slower than multiplication in a digital processor.

One four-quadrant inverse tangent operation There are three popular approaches to calculate a four-quadrant inverse tangent operation:

COordinate **R**otation **D**igital **C**omputer (**CORDIC**) A CORDIC is an extremely efficient algorithm to implement several functions in hardware because it only performs shift and add operations in an iterative manner, thus eliminating the need for explicit multipliers. However, running the algorithm for a specific number of iterations for each symbol duration necessitates a higher complexity budget.

Polynomial **A**pproximation The polynomial approximation theory solves for an order and a set of polynomial coefficients that closely approximate a given function. A high speed processor is required in the Rx in this case.

Look-Up Table A Look-Up Table of precomputed values can be stored in memory and accessed as needed during the transceiver operation. In this option, the penalty lies in the memory size of the processor. Two obvious design tradeoffs here are the size and accuracy of the Look-Up Table. In general, the larger the size, the more accurate the values are. A good solution is to store a small number of values and implement interpolation between the values to increase the accuracy.

Any of the above approaches entails a certain amount of complexity on the receiver's part.

In contrast, a cross product PED can be implemented with just two multiplications and a single addition (even the multiplications can be avoided by using the sign of $a_I[m]$ and $a_Q[m]$ while absorbing the scaling factor in PED gain). The main point is that the signal values are directly employed without any explicit phase computation.

More importantly, remember that in decision-directed mode of operation, there is a phase ambiguity of $\pi/2$ for a QPSK modulation producing four stable lock points at $-\pi/2$, 0 , $+\pi/2$ and π . For a small segment of phase error ranging from $-\pi/4$ to $+\pi/4$ (and similar repetitions thereof), the sinusoidal mean curve of a cross product PED is not far from linear owing to the small angle approximation $\sin A \approx A$ for small A .

To visually comprehend this point, the mean curves of the cross product and phase difference PEDs (for equal gains) are plotted in Figure 5.35 for a decision-directed operation. Given the fact that the loop mostly operates in a decision-directed fashion, it can be concluded that the cross product PED is not that useless at all and it is actually the most practical PED for implementing a phase synchronization system[†].

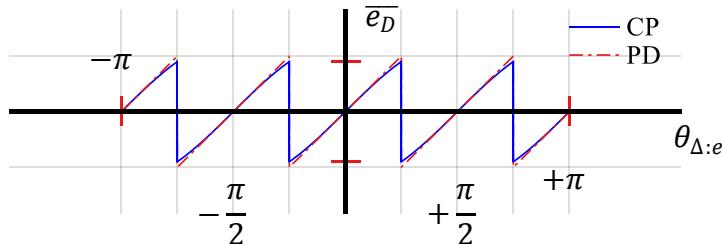


Figure 5.35: Mean curves for cross product (CP) and phase difference (PD) PED in decision-directed mode. Observe the linear overlap almost everywhere

5.6 Non-Data-Aided Techniques

As discussed above, the drawback of inserting a known sequence within the message is a decreased spectral efficiency of the system. To avoid this cost, a phase estimator (as well as estimators for other distortions) can be derived in a non-data-aided fashion. One option exercised above was a decision-directed strategy: to use the outputs of

[†]There is another difference in the gain between two PEDs in that the cross product PED depends on the Rx signal amplitude due to employing direct signal values instead of an explicit phase computation. However, the AGC in the Rx sets the amplitude at a reference value.

the decision device as a replacement of known data. However, this strategy cannot be utilized for initial phase acquisition at the start of the transmission.

Thus, the task at hand is to replace the conjugate product of the matched filter outputs $z(mT_M)$ and a known sequence $a^*[m]$ as in data-aided and decision-directed techniques with an operation that does not need the knowledge of $a[m]$. We start with an example of a QPSK constellation.

5.6.1 Feedforward: M-th Power Estimator

To understand the inner working of an M^{th} -power estimator, we start with a 4^{th} -power estimator for QPSK and then generalize it to MPSK constellations. Recall from the definition of complex numbers that in an IQ -plane, raising a complex number to a certain power *raises* the magnitude to that power while *multiples* the phase with that number. For a complex number V in polar form and its 4^{th} power,

$$\begin{aligned} |V^4| &= |V|^4 \\ \angle V^4 &= 4 \cdot \angle V \end{aligned} \tag{5.41}$$

The phases of the four points on a regular QPSK constellation are given by $\pm\pi/4$ and $\pm 3\pi/4$. Since

$$+\pi = -\pi = +3\pi = -3\pi,$$

multiplying these phases with 4 maps them all to the same location.

$$4 \cdot \left\{ \pm \frac{\pi}{4}, \pm \frac{3\pi}{4} \right\} = \pi$$

This phase multiplication, for QPSK symbols $a[m]$, can be written as

$$4 \cdot \angle a[m] = \pi$$

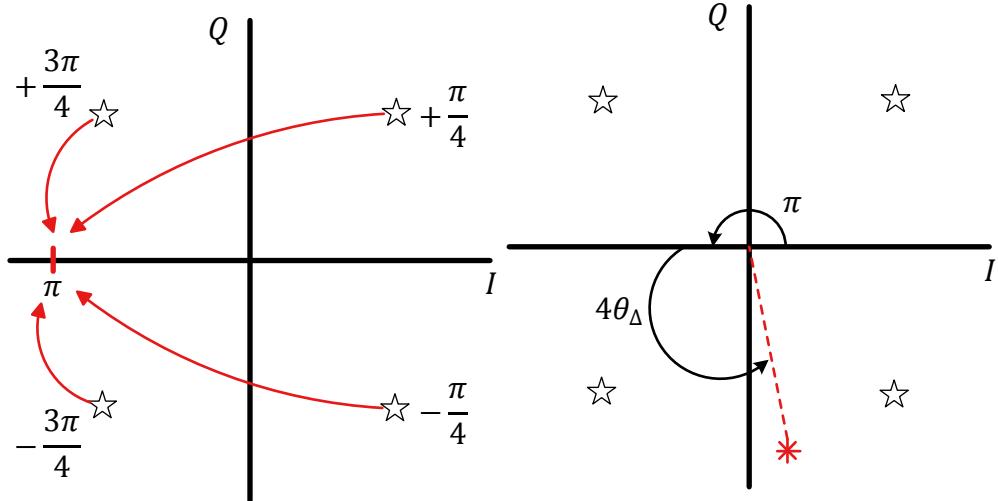
regardless of the value each $a[m]$ has. The underlying intuition behind this is the $2\pi/4$ *rotational symmetry* of QPSK constellation: rotating the constellation by an angle of $2\pi/4$ generates the same constellation. This can be implemented by raising $a[m]$ to the 4^{th} power and is drawn in Figure 5.36a. We can see that *the 4th power operation has removed the modulation from all data points* without any data knowledge at the Rx.

With this background, let us consider the case where the QPSK symbols at the Rx contain a phase offset θ_Δ as well. Recall from Eq (5.4) that the magnitude and phase of the matched filter output at symbol interval m can be written as

$$\begin{aligned} |z(mT_M)| &= \sqrt{a_I^2[m] + a_Q^2[m]} \\ \angle z(mT_M) &= \angle a[m] + \theta_\Delta \end{aligned}$$

where a mismatch of θ_Δ between incoming carrier and Rx oscillator is seen to leave the magnitude of the data symbols $a[m]$ unchanged and rotate their phases by an angle θ_Δ . Clearly, raising each output to the 4^{th} power yields

$$\begin{aligned} |z^4(mT_M)| &= (a_I^2[m] + a_Q^2[m])^2 \\ \angle z^4(mT_M) &= 4 \cdot \angle a[m] + 4\theta_\Delta = \pi + 4\theta_\Delta \end{aligned} \tag{5.42}$$



(a) Raising QPSK symbols to the 4th power maps all the phases to the same angle π

(b) Raising Rx QPSK symbols (contaminated with a phase offset θ_Δ) to the 4th power maps them all to $\pi + 4\theta_\Delta$

Figure 5.36: An illustration of wiping off the modulation process without any training information

This is illustrated in Figure 5.36b. From Eq (5.42), the summation of matched filtered outputs can be employed to average the noise and a non-data-aided phase estimator can be derived as

$$\hat{\theta}_\Delta = \frac{1}{4} \cdot \angle \left\{ \sum_{m=0}^{N_d-1} z^4(mT_M) \right\} - \frac{\pi}{4} \quad (5.43)$$

where N_d is the sequence length in symbols.

For another version of QPSK constellation, the symbols start from an angle 0 instead of $\pi/4$. Hence, its phase values are

$$0, +\frac{\pi}{2}, \pi, -\frac{\pi}{2}$$

which after multiplication by 4 all map to the an angle 0. In this case, the factor $\pi/4$ in Eq (5.43) is not required.

$$\hat{\theta}_\Delta = \frac{1}{4} \cdot \angle \left\{ \sum_{m=0}^{N_d-1} z^4(mT_M) \right\} \quad (5.44)$$

This is the most common form of 4th-power estimator encountered in the synchronization literature.

The above estimator works due to $2\pi/4$ rotational symmetry of the QPSK constellation. For a general M -PSK constellation with a rotational symmetry of $2\pi/M$ and

the starting phase of 0, the M^{th} -power phase estimator can be written as

$$\hat{\theta}_\Delta = \frac{1}{M} \cdot \angle \left\{ \sum_{m=0}^{N_d-1} z^M(mT_M) \right\} \quad (5.45)$$

A block diagram of this scheme is drawn in Figure 5.37. Notice the main differences from the conjugate product estimator: there is no training data, an M^{th} -power operation and a $1/M$ scaling factor.

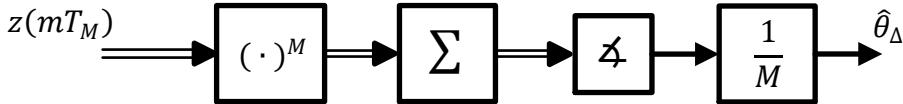


Figure 5.37: A block diagram of an M^{th} -power estimator. Notice the differences from the conjugate product estimator: no training data, the power operation and a $1/M$ scaling factor

Comments on Performance

Now let us turn our attention to what happens in the presence of noise. For the simple case of raising the matched filter output to the 4^{th} power and using the identity $(A + B)^4 = A^4 + 4A^3B + 6A^2B^2 + 4AB^3 + B^4$,

$$(signal + noise)^4 = signal^4 + 4 \times signal^3 \times noise + 6 \cdot signal^2 \times noise^2 + \\ 4 \cdot signal \times noise^3 + noise^4$$

Here, we have four terms that involve noise. Compare this with Eq (5.16) in the data-aided case where there was only one noise term. A consequence of the power operation is now clear: *significant noise enhancement in the estimation block*. In general, a larger M yields a larger phase jitter at the NCO output. Subsequently, the performance of M^{th} -power estimator is worse than the conjugate product estimator. This makes sense as well since the former is a non-data-aided scheme while the latter is data-aided. This loss in performance is the price paid for the absence of training sequence in the packet for increasing spectral efficiency.

5.6.2 Feedback: M-th Power Phase Error Detector

Again assume that a feedback (FB) system, like a Phase Locked Loop (PLL), is employed for synchronizing the Rx oscillator phase to the phase of the Tx signal. In this situation as well, the Rx PLL can be driven through the actual Tx signal itself and without any pilot tone if the phase shifts at symbol intervals arising from modulating data can be removed.

In the data-aided and decision-directed techniques discussed in Section 5.5, new phase error detectors were devised that removed the impact of modulation symbols

$\angle a[m]$ from $\angle z(mT_M)$ at each symbol interval, generating an output as a function of θ_Δ only. With this modification, the input to the loop filter became similar to a simple sinusoidal case and hence all the concepts about a PLL tracking a simple input sinusoid apply for the phase synchronization block as before.

However, those techniques exploited the knowledge of training data or decisions fed back from the detector for this purpose. When no such help is available, our initial idea to modify a simple PLL in non-data-aided mode is the following.

- At each symbol time mT_M , raise the matched filter output to M^{th} power and employ it as a phase error detector.
- At next symbol time $(m + 1)T_M$, de-rotate the matched filtered output through the NCO output.
- At this symbol time $(m + 1)T_M$, use this de-rotated matched filtered signal as the phase error detector input.

Subsequently transform the feedforward M^{th} -power estimator $\hat{\theta}_\Delta$ of Eq (5.45) into a feedback phase error detector output $e_D[m]$ by

- removing the summation, and
- replacing the matched filter output $z(mT_M)$ by de-rotated matched filter output $\hat{z}(mT_M)$

as

$$e_D[m] = \frac{1}{M} \cdot \angle \hat{z}^M(mT_M) \quad (5.46)$$

As opposed to wiping off the carrier phase offset once and for all as done in feedforward M^{th} -power estimator, this kind of implementation is essentially a strategy to converge towards the final solution in small incremental steps. The rest of the principles remain the same as in an ordinary PLL and data-aided/decision-directed approaches explained above. On the downside, the phase detection range of the M^{th} -power PED is reduced to $[-\pi/M, +\pi/M]$ due to the phase ambiguity. This is because raising a carrier to the M^{th} power generates a control signal that represents M times the actual phase difference.

This feedback strategy is described here because a well known phase synchronization technique for BPSK modulation, known as the *squaring synchronizer*, is a special case of M^{th} -power PED with $M = 2$.

$$e_D[m] = \frac{1}{2} \cdot \angle \hat{z}^2(mT_M)$$

The squaring operation removes the modulation and doubles the frequency which is locked through a standard PLL. The output of the PLL is divided by 2 to provide the carrier at the actual frequency which is then used to synchronously demodulate the Rx signal. A block diagram of a squaring synchronizer directly operating on the Rx BPSK signal is drawn in Figure 5.38. The squaring synchronizer has been extensively used for locking the carrier in conventional amplitude modulated systems.

Next, we discuss an alternative strategy known as the Costas loop.

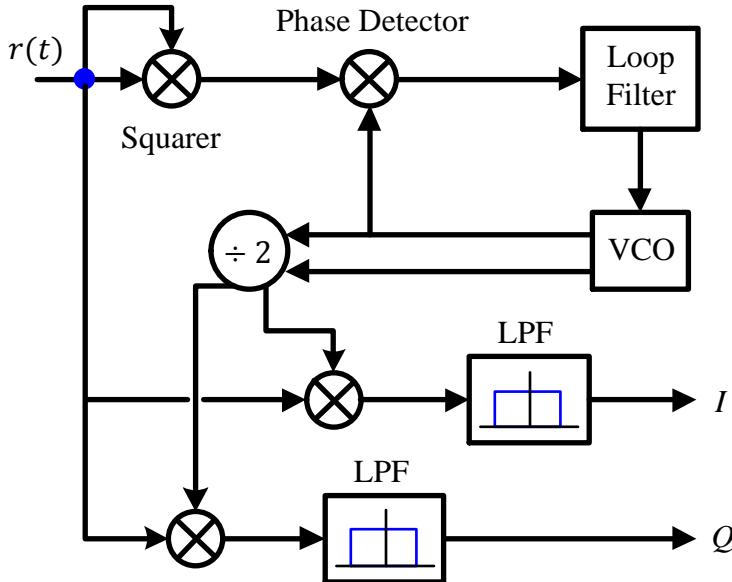


Figure 5.38: A block diagram for a squaring loop for BPSK demodulation

5.6.3 The Costas Loop: A Classic Solution

No discussion on synchronization can be complete without a description of the classic Costas loop – a carrier phase synchronization solution devised by John Costas at General Electric Company in 1956 [16] because of the impact it had on modem signal processing in general and carrier synchronization in particular. In words of Costas,

"It is unfortunate that many engineers tend to avoid phase locked systems. It is true that a certain amount of stability is a prerequisite but it has been determined by experiment that for this application that stability requirements of single-side band voice are more than adequate. Once a certain degree of stability is obtained, the step to phase lock is a simple one."

John Costas - Synchronous Communications [16]

Costas was one of the earliest scientists to demonstrate that the carrier phase could be reliably recovered from the Rx signal without the need of a pilot tone. Since the publication of his paper, numerous quadrature carrier synchronization loop structures have been developed but their overall structure can easily be traced back to the original work by Costas.

The main theme of this text is to focus on discrete-time techniques for synchronization. For this purpose, we will follow a continuous-time development due to two reasons.

- Although we deal with its digital implementations now, the Costas loop was originally a pure analog solution and an extensive literature based on continuous-time processing is available. With a similar treatment here, the reader can con-

nect the concepts with that original work.

- We have already analyzed a discrete-time implementation for phase synchronization for BPSK and QPSK modulations. Once we treat the Costas loop in continuous-time, we will connect it with that discrete-time counterpart discussed before.

BPSK Modulation

To understand the fundamental concept behind the Costas loop structure for BPSK modulation, we will follow a similar line previously discussed in discrete-time scenario. We start with the continuous-time version $v_I(t)$ of the sampled baseband signal $v_I(nT_S)$ given by

$$v_I(t) = \sum_i a_I[i] p(t - iT_M)$$

where $a_I[i]$ is the i^{th} symbol, $p(t)$ denotes a transmit pulse and T_M is the symbol time. The passband signal $r(t)$ is this baseband signal upconverted by a carrier wave with a phase offset of θ_Δ .

$$r(t) = v_I(t) \sqrt{2} \cos(2\pi F_C t + \theta_\Delta)$$

At the Rx, it is treated in I and Q arms by the sinusoids produced by a Voltage Controlled Oscillator (VCO) with $\hat{\theta}_\Delta$ as its phase reference. Referring to the terms in Figure 5.39,

$$\begin{aligned} I \rightarrow & \quad x_I(t) = r(t) \cdot \sqrt{2} \cos(2\pi F_C t + \hat{\theta}_\Delta) \\ & = v_I(t) \sqrt{2} \cos(2\pi F_C t + \theta_\Delta) \sqrt{2} \cos(2\pi F_C t + \hat{\theta}_\Delta) \end{aligned}$$

Using the identity $\cos A \cdot \cos B = 0.5 \{\cos(A - B) + \cos(A + B)\}$,

$$I \rightarrow \quad x_I(t) = v_I(t) \cos \theta_{\Delta:e} + \underbrace{\cos(2\pi 2F_C t + \theta_\Delta + \hat{\theta}_\Delta)}_{\text{Double frequency term}}$$

where $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$ is the carrier phase error. The double frequency term is removed by the lowpass filter in the I arm. We write its output as

$$I \rightarrow \quad z_I(t) = v_I(t) \cos \theta_{\Delta:e}$$

Notice that the quadrature carrier here has a positive sign in Figure 5.39 instead of the usual negative sign we encountered in discrete-time phase synchronization loops for complex signal processing. Following a similar line of reasoning and using $\cos A \cdot \sin B = 0.5 \{\sin(A + B) - \sin(A - B)\}$, the quadrature part of the downconverted and lowpass filtered signal is

$$Q \rightarrow \quad z_Q(t) = -v_I(t) \sin \theta_{\Delta:e}$$

The negative sign appears due to using a positive sign at the quadrature carrier. Ignoring the hard limiting operation in Figure 5.39 for now and multiplying the two expressions above generates the continuous-time error signal.

$$e_D(t) = -v_I^2(t) \cos \theta_{\Delta:e} \sin \theta_{\Delta:e} = -\frac{1}{2} v_I^2(t) \sin 2\theta_{\Delta:e} \quad (5.47)$$

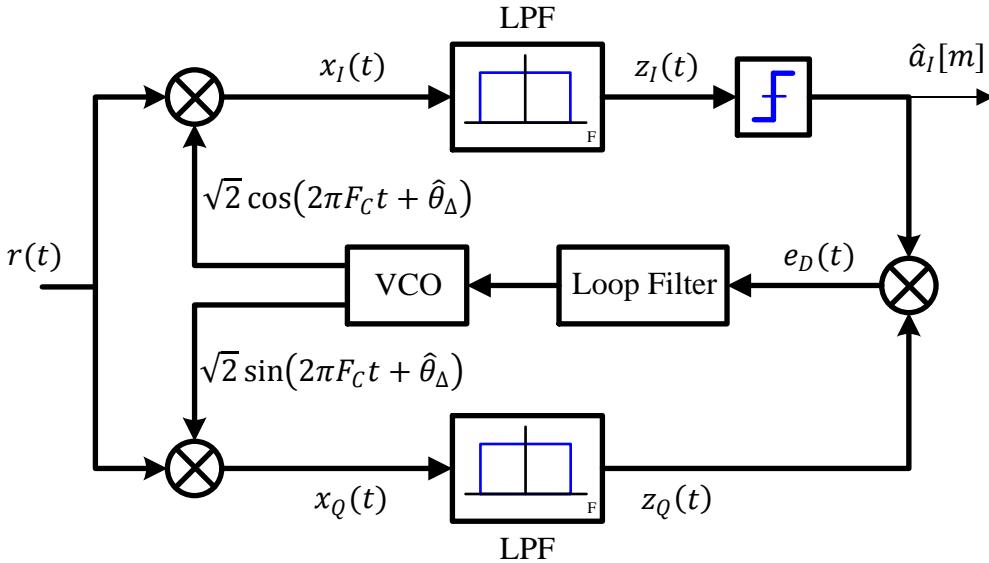


Figure 5.39: A Costas loop based phase synchronizer for BPSK modulation

where we have used the identity $\cos A \sin A = 0.5 \sin 2A$. Clearly, as long as $v_I^2(t)$ varies slowly, **the mean curve has a sinusoidal shape** which establishes a stable locking point around $\theta_{\Delta:e} = 0$.

For the purpose of understanding the operation of the Costas loop, assume for the moment the following.

- The Costas loop is reasonably close to the lock point, i.e., $\theta_{\Delta:e}$ is small. Then, the signal on the I rail after lowpass filtering is close to symbol value $a_I[m]$ at the right time due to $\cos \theta_{\Delta:e} \approx 1$ while that on the Q rail would be close to $\theta_{\Delta:e}$ due to $\sin \theta_{\Delta:e} \approx \theta_{\Delta:e}$.
- The pulse shape is a simple rectangle so that a symbol does not interfere with the neighbouring symbols, hence within a symbol duration

$$v_I^2(t) \approx a_I^2[m] = (\pm 1)^2 = 1,$$

Consequently, the phase error $\theta_{\Delta:e}$ in Eq (5.47) can be written as

$$e_D(t) \approx -\frac{1}{2}(2\theta_{\Delta:e}) = -\theta_{\Delta:e}$$

The negative sign in the error term above is catered for by either

- removing the negative sign before the VCO input, or
- interchanging the cosine and sine outputs of the VCO and treating the output of the sine as the I arm (this is the solution adopted in conventional Costas loop structures).

Finally, in another version of the Costas loop, the hard limiting operation in Figure 5.39 rejects the small variations usually induced by the signal in the opposite arm and the noise, without affecting a sinusoidal mean curve.

Having obtained this sinusoidal error term, the operation of a Costas loop pretty much mimics a standard PLL. Due to the $\sin(\cdot)$ term in the phase error detector, the Q rail output is of the same polarity as the I output for one direction of phase difference and opposite polarity for the other direction. Moreover, the sign of the I arm can be taken for making a symbol decision as well. Hence, this phase synchronization circuit provides data estimates as well which was a fundamental shift from the communication circuits of that time.

It has been shown that the performance of the Costas loop is similar to a squaring PLL described before. The advantage is that the signal processing operations performed through analog circuitry were required only at the carrier frequency, instead of twice the carrier frequency thus leading to a simpler circuit design.

QPSK Modulation

After understanding the operation of Costas loop for BPSK modulation, it is not difficult to extend its capability to a QPSK case. The Costas loop for QPSK phase synchronization is illustrated in Figure 5.40.

We start with the Rx signal $r(t)$ which includes the carrier wave with a phase offset of θ_Δ and a positive sign with the quadrature carrier.

$$r(t) = v_I(t)\sqrt{2} \cos(2\pi F_C t + \theta_\Delta) + v_Q(t)\sqrt{2} \sin(2\pi F_C t + \theta_\Delta)$$

At the Rx, it is treated in I and Q arms by the sinusoids produced by a Voltage Controlled Oscillator (VCO) with $\hat{\theta}_\Delta$ as its phase reference. Referring to the terms in Figure 5.40,

$$\begin{aligned} x_I(t) &= r(t) \cdot \sqrt{2} \cos(2\pi F_C t + \hat{\theta}_\Delta) \\ I \rightarrow &= \left\{ v_I(t)\sqrt{2} \cos(2\pi F_C t + \theta_\Delta) + \right. \\ &\quad \left. v_Q(t)\sqrt{2} \sin(2\pi F_C t + \theta_\Delta) \right\} \sqrt{2} \cos(2\pi F_C t + \hat{\theta}_\Delta) \end{aligned}$$

Using the identities $\cos A \cdot \cos B = 0.5 \{\cos(A - B) + \cos(A + B)\}$ and $\sin A \cdot \cos B = 0.5 \{\sin(A + B) + \sin(A - B)\}$,

$$I \rightarrow x_I(t) = v_I(t) \cos \theta_{\Delta:e} + v_Q(t) \sin \theta_{\Delta:e} + \text{Double frequency terms}$$

where $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$ is the carrier phase error. The double frequency terms are removed by the lowpass filter in the I arm. We write its output as

$$I \rightarrow z_I(t) = v_I(t) \cos \theta_{\Delta:e} + v_Q(t) \sin \theta_{\Delta:e}$$

Following a similar line of reasoning, the quadrature part of the downconverted signal is

$$\begin{aligned} x_Q(t) &= r(t) \cdot \sqrt{2} \sin(2\pi F_C t + \hat{\theta}_\Delta) \\ Q \rightarrow &= \left\{ v_I(t)\sqrt{2} \cos(2\pi F_C t + \theta_\Delta) + \right. \\ &\quad \left. v_Q(t)\sqrt{2} \sin(2\pi F_C t + \theta_\Delta) \right\} \sqrt{2} \sin(2\pi F_C t + \hat{\theta}_\Delta) \end{aligned}$$

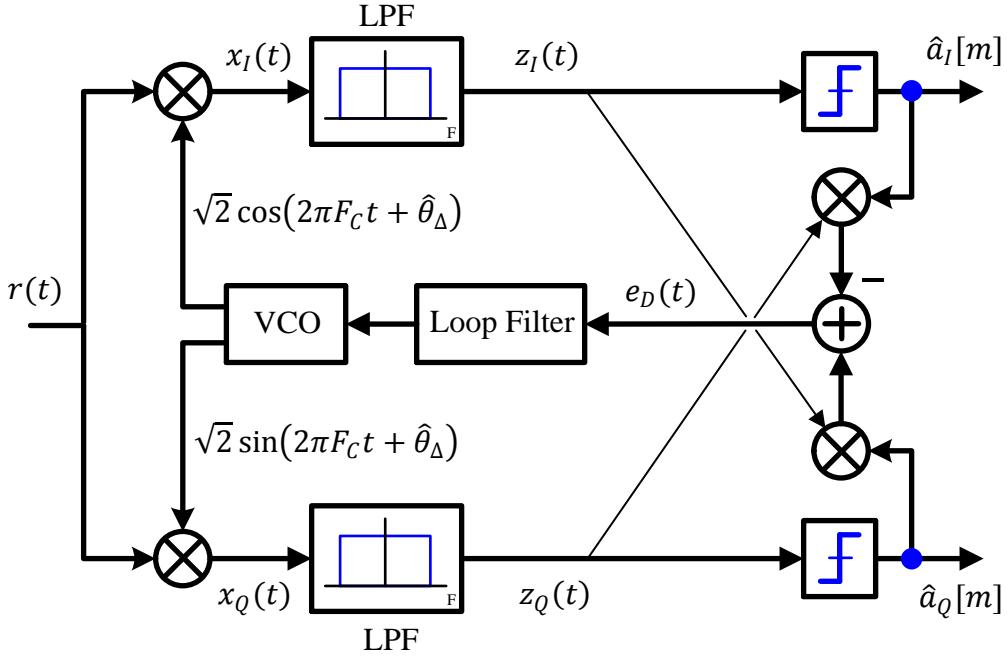


Figure 5.40: A Costas loop based phase synchronizer for QPSK modulation

Using the identities $\cos A \cdot \sin B = 0.5 \{\sin(A+B) - \sin(A-B)\}$ and $\sin A \cdot \sin B = 0.5 \{\cos(A-B) - \cos(A+B)\}$,

$$Q \rightarrow x_Q(t) = -v_I(t) \sin \theta_{\Delta:e} + v_Q(t) \cos \theta_{\Delta:e} + \text{Double frequency terms}$$

The double frequency terms are removed by the lowpass filter in the Q arm. We write its output as

$$Q \rightarrow z_Q(t) = -v_I(t) \sin \theta_{\Delta:e} + v_Q(t) \cos \theta_{\Delta:e}$$

Again assuming a rectangular pulse shape and a small phase difference (which implies $\cos \theta_{\Delta:e} \approx 1$ and $\sin \theta_{\Delta:e} \approx 0$), the I signal $v_I(t)$ is close to the symbol value $a_I[m]$ at the right time. Taking its sign is thus approximated as $\hat{a}_I[m]$. Similarly, the sign of the signal on the Q rail, $v_Q(t)$, is approximately $\hat{a}_Q[m]$.

$$\text{sign}\{z_I(t)\} \approx \hat{a}_I[m], \quad \text{sign}\{z_Q(t)\} \approx \hat{a}_Q[m]$$

The purpose of these sign operations is to reject the small variations usually induced by the signal in the opposite arm and the noise. Therefore, the error term from Figure 5.40 can be given as

$$\begin{aligned} e_D(t) &= \text{sign}\{z_Q(t)\}z_I(t) - \text{sign}\{z_I(t)\}z_Q(t) \\ &\approx \hat{a}_Q[m]z_I(t) - \hat{a}_I[m]z_Q(t) \end{aligned}$$

Compare the above expression with the decision-directed cross product phase error detector in Eq (5.35), which without normalization by γ is given by

$$e_D[m] = \hat{a}_I[m]\hat{z}_Q(mT_M) - \hat{a}_Q[m]\hat{z}_I(mT_M)$$

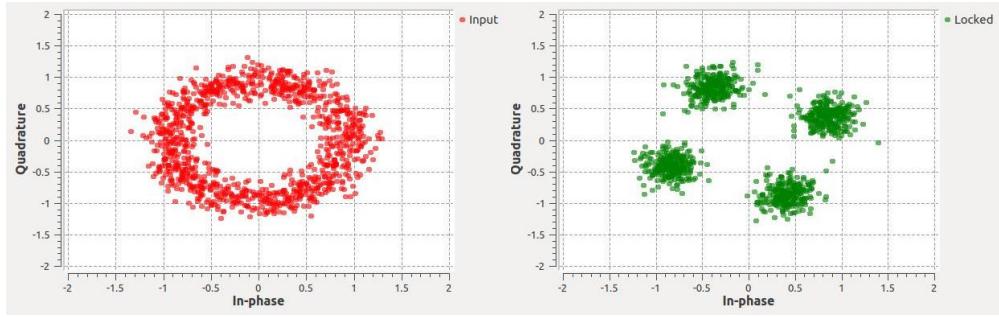


Figure 5.41: Output from the Costas Loop block after phase convergence

where the difference of a negative sign occurs due to the positive sign of the quadrature carrier in this case. This comparison shows that the Costas loop can be thought of an approximate solution to the maximum correlation (or maximum likelihood) technique. Due to this reason, the mean curve of the Costas loop is very similar to the mean curve of the cross product phase error detector in a decision-directed QPSK scenario drawn in Figure 5.29 with a $\pi/2$ phase ambiguity.

Exercise 5.1

In GNU Radio, a Costas loop block is available with the following parameters.

Loop Bandwidth: The loop bandwidth is the equivalent noise bandwidth of a PLL described in Section 4.6 and should be adjusted accordingly.

Order: The loop order depends on the modulation scheme: 2 for BPSK, 4 for QPSK and 8 for 8-PSK.

Use SNR: An estimate of SNR before forming the product helps in more accurate phase estimates when a $\tanh(\cdot)$ function is also employed instead of slicing the filter output (i.e., mapping it to the constellation point). This comes from maximum likelihood theory, see Ref. [2] for the role played by the SNR estimate and $\tanh(\cdot)$.

At the time of this writing, there is a slight misrepresentation of the Costas loop in the following statement of the GNU Radio documentation: “The Costas loop locks to the center frequency of a signal and downconverts it to baseband”. While this was true for an analog implementation of old days, the digital implementation works at baseband to cater for a phase offset and any residual (fine) frequency offsets, see Note 5.3 for an explanation.

For a loop bandwidth of $2\pi/100$, QPSK modulation and a normalized carrier frequency offset of -0.043 , Figure 5.41 shows the convergence of the loop. The output is rotated because the frequency offset is at the edge of its lock range.

5.7 Detection Aids

Next, we describe some miscellaneous topics on carrier phase synchronization that are helpful in understanding the operation of a carrier PLL in a real implementation.

5.7.1 Carrier Lock Detectors

A lock detector is a logic signal used in the Rx to indicate successful synchronization after which the demodulation process can be started. It also helps the Rx in keeping the track of a signal loss due to fading and re-acquisition when the signal returns. Furthermore, a PLL tends to use different parameters (or even different algorithms in a software defined radio) during acquisition as compared to tracking. Finally, a non-data-aided version of an algorithm might be employed during acquisition while a switch to decision-directed mode is made after the lock is achieved.

Therefore, a Rx needs a reliable indication of the lock state for carrier and timing recovery. This is usually implemented through comparing a lock metric with a threshold and forming a hypothesis based on their difference. The averaging over many symbols can be implemented in either a feedforward manner or a feedback fashion while the threshold λ is computed based on the minimum tolerable phase offset θ_Δ .

For a BPSK signal, we know that a phase rotation reduces the amplitude in the I arm while injecting this power into the Q arm. A perfectly locked PLL should have all the energy in its I arm, as shown in Figure 5.4. Utilizing this observation, we can construct a carrier lock detector from the matched filter outputs $z(mT_M)$ by taking the difference between $|z_I(mT_M)|$ and $|z_Q(mT_M)|$, averaging it over a long interval and then comparing it against a threshold λ .

$$LD(\theta_\Delta) = \sum_{m=0}^{N_d-1} \left\{ |z_I(mT_M)| - |z_Q(mT_M)| \right\} \begin{cases} \geq \lambda & \text{PLL in lock} \\ \text{otherwise} & \text{PLL unlocked} \end{cases}$$

This strategy will not work in the case of a QPSK modulation because both I and Q arms contain the signal energy from the modulated symbols. This energy should be equal in the case of a zero θ_Δ and hence a zero difference between the two can be taken as an indication of lock. For this purpose, a threshold λ close to zero can be chosen.

$$LD(\theta_\Delta) = \sum_{m=0}^{N_d-1} \left\{ |z_I(mT_M)| - |z_Q(mT_M)| \right\} \begin{cases} \leq \lambda & \text{PLL in lock} \\ \text{otherwise} & \text{PLL unlocked} \end{cases}$$

Instead of an absolute value, a magnitude squared operation can also be employed. Notice that the above lock detectors operate in a non-data-aided fashion. We can also design their decision-directed counterparts utilizing $\hat{a}[m]$.

From the above expressions, we realize that the lock detection indicator depends on the signal amplitude because it operates directly on the matched filter outputs. This can cause erroneous lock and unlock flags when the signal power varies. An alternative technique is to divide the lock expressions by the signal magnitude

$$\sqrt{z_I^2(mT_M) + z_Q^2(mT_M)}$$

that acts as a virtual AGC to normalize the magnitude and deliver an output free of dynamic range problems.

5.7.2 Resolving Phase Ambiguity

In the discussion on feedback phase error detectors, we saw in Eq (5.27) that the mean curve for a QPSK modulation crosses zero with a positive slope (and hence stable lock points) at

$$\theta_{\Delta:e} = -\frac{\pi}{2}, 0, +\frac{\pi}{2}, \pi$$

Therefore, the loop suffers from a $2\pi/4 = \pi/2$ phase ambiguity due to this rotational symmetry. Consequently, the QPSK carrier phase PLL employing a decision-directed PED can compensate for a carrier phase offset but it will be unknown which of the above four points it has locked to. Similarly, a BPSK modulation has a $2\pi/2 = \pi$ phase ambiguity. Therefore, the mean curve of the decision-directed PED has two stable lock points, one at $\theta_{\Delta:e} = 0$ and the other at $\theta_{\Delta:e} = \pi$.

Generalizing this observation, we conclude that a decision-directed PLL for an M -PSK modulation suffers from a phase ambiguity of $2\pi/M$ and offers M stable locking points. There are two commonly employed methods for resolving this ambiguity: *unique word* and *differential encoding/decoding*.

We start with the unique word as it is easier to understand. A unique word (UW) is just a sequence of known symbols in the data stream. After achieving phase lock, the Rx looks for this specific pattern with 0 as well as other possible phase offsets. The phase rotation with which this pattern is found is the phase offset that needs to be corrected in the data symbols as well.

Suppose that for a BPSK modulation scheme, the UW at the start of the transmission is given by

0 1 1 0 1

Since there are two stable lock points at $\theta_{\Delta:e} = 0$ and $\theta_{\Delta:e} = \pi$ (which just inverts the bit), the PLL at the Rx searches for two patterns: 0 1 1 0 1 and its rotated by π version 1 0 0 1 0. If the PLL finds the former, the lock has been acquired to the actual phase and nothing needs to be done. However, if the search output indicates the latter pattern, then all the detected data symbols need to be inverted for correct results.

To study differential encoding and decoding, recall that we discussed the polar representation of QAM in Eq (3.41) reproduced below.

$$\begin{aligned} s(t) &= \sum_m \sqrt{a_I^2[m] + a_Q^2[m]} \cdot p(t - mT_M) \cdot \sqrt{2} \cos \left(2\pi F_C t + \tan^{-1} \frac{a_Q[m]}{a_I[m]} \right) \\ &= \sum_m X[m] \cdot p(t - mT_M) \cdot \sqrt{2} \cos (2\pi F_C t + \phi[m]) \end{aligned}$$

where

$$\begin{aligned} X[m] &= \sqrt{a_I^2[m] + a_Q^2[m]} \\ \phi[m] &= \tan^{-1} \frac{a_Q[m]}{a_I[m]} \end{aligned}$$

In this polar form, we have a single sinusoid whose amplitude and phase are determined by some combination of symbols $a_I[m]$ and $a_Q[m]$ during every symbol time. For an M -PSK constellation, all the symbols lie on a circle and hence the amplitude $X[m]$ is constant for all symbols.

$$X[m] = \text{constant}$$

This makes sense as only the phase is changed for a PSK modulations scheme. It is easiest to understand differential encoding and decoding with the help of a BPSK modulation.

In a regular BPSK system, the phase for the polar form $\phi[m]$ is chosen according to the incoming bit. As an example,

$$\begin{array}{ll} 0 & \rightarrow 0 \\ 1 & \rightarrow \pi \end{array}$$

So in the following sequence of bits $b[m]$, the mapping is shown in Table 5.1.

Table 5.1: Bit to phase mapping in a regular BPSK

m	0	1	2	3	4	5	6	7	8
$b[m]$	0	1	1	0	1	1	1	0	1
$\phi[m]$	0	π	π	0	π	π	π	0	π

It is tempting to think from the name ‘differential encoding’ that the phase chosen for the polar form $\phi[m]$ depends on the change in bit sequence rather than the bit sequence itself. For instance, if the next bit is the same as the previous bit, the chosen phase is 0 while if the bit changes, the chosen phase is π . However, it is slightly more complicated in the sense that the change in bit sequence is with respect to the previously encoded bit, not the previous input bit. Let us see how.

Denote the encoded bit as $c[m]$ and let its starting value $c[-1] = 0$. Then, we can write

$$c[m] = b[m] + c[m - 1] \quad \text{mod } 2 \quad (5.48)$$

When we apply this relation to the input bit sequence $b[m]$ in Table 5.1, we get

$$\begin{aligned} c[0] &= b[0] + c[-1] = 0 \\ c[1] &= b[1] + c[0] = 1 \\ c[2] &= b[2] + c[1] = 0 \\ c[3] &= b[3] + c[2] = 0 \\ &\vdots \end{aligned}$$

The complete encoded sequence is illustrated in Figure 5.42 where the tilted arrows indicate the two terms to be summed according to the differential encoder in Eq (5.48) while the downwards arrows show the results of this addition.

Now we see how the decoding process works. From the differential encoding in Eq (5.48), we can write the differential decoding procedure as

$$\hat{b}[m] = \hat{c}[m] + \hat{c}[m - 1] \quad \text{mod } 2 \quad (5.49)$$

For the example sequence, assume that the PLL locks onto the incoming phase without any $\pi/2$ jump, i.e., the Rx sequence $\hat{c}[m]$ is the same as the Tx sequence $c[m]$.

m	0	1	2	3	4	5	6	7	8
$b[m]$	0	1	1	0	1	1	1	0	1
$c[-1] = 0$	0	1	0	0	1	0	1	1	0
$\phi[m]$	0	π	0	0	π	0	π	π	0

Figure 5.42: Bit to phase mapping in a differentially encoded BPSK

Then, applying the decoding relation in Eq (5.49) with an initial value of $\hat{c}[-1] = 0$,

$$\begin{aligned}\hat{b}[0] &= \hat{c}[0] + \hat{c}[-1] = 0 \\ \hat{b}[1] &= \hat{c}[1] + \hat{c}[0] = 1 \\ \hat{b}[2] &= \hat{c}[2] + \hat{c}[1] = 1 \\ \hat{b}[3] &= \hat{c}[3] + \hat{c}[2] = 0 \\ &\vdots\end{aligned}$$

The complete sequence $\hat{b}[m]$ is decoded in Figure 5.43. Notice that it is exactly the same as the original bit sequence $b[m]$, although only the first bit can be in error for a different $c[-1] = 1$. Now if the PLL locks onto the other lock point π as can happen in a BPSK system, the differential decoding output is still valid (except the first bit) as illustrated in Figure 5.43.

A block diagram for the implementation of this encoding and decoding process is drawn in Figure 5.44. One disadvantage of the differential encoding and decoding is that if one symbol is received in error and detected as $\hat{c}[m]$, then it will lead to two errors at the output of the final decoded sequence, as is evident from the relation in Eq (5.49), one for the current bit where it participates as $\hat{c}[m]$ and the other for the next bit where it participates as $\hat{c}[m - 1]$. At practical target BERs for a well designed system, the errors are very rare for the budgeted SNR, particularly more so in consecutive symbols and hence this approximately doubles the BER.

Exercise 5.2

GNU Radio uses generalized versions of these two blocks, ‘Differential Encoder’ and ‘Differential Decoder’. In the scenario described above, the modulus M is 2 for a binary scheme which can be extended for other values of M . Keep in mind that the addition at the decoder side is actually a subtraction since they are both the same for mod 2 operations.

PLL lock point = 0

m	0	1	2	3	4	5	6	7	8
$\hat{\phi}[m]$	0	π	0	0	π	0	π	π	0
$\hat{c}[-1] = 0$	0	1	0	0	1	0	1	1	0
$\hat{b}[m]$	0	1	1	0	1	1	1	0	1

PLL lock point = π

m	0	1	2	3	4	5	6	7	8
$\hat{\phi}[m]$	π	0	π	π	0	π	0	0	π
$\hat{c}[-1] = 0$	1	0	1	1	0	1	0	0	1
$\hat{b}[m]$	1	1	1	0	1	1	1	0	1

Figure 5.43: Phase to bit mapping in a differentially decoded BPSK. Notice that both stable lock points 0 and π for the PLL produce the same output $\hat{b}[m]$ (except the first bit)

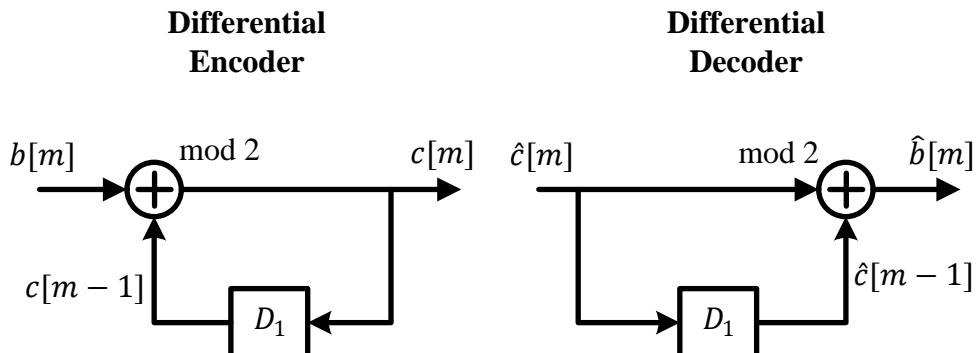


Figure 5.44: Block diagram for a differential encoder and decoder. The addition at the decoder side is actually a subtraction since they are both the same for mod 2 operations

5.7.3 Cycle Slips, Hangup and the Role of AGC

Regarding the discussion above, an account of some important phenomena is now in order.

Cycle slips A phenomenon that manifests itself in synchronization structures is *cycle slips*. Here, assume that the loop is in tracking mode, implying that $\hat{\theta}_\Delta[m]$ is fluctuating around the true offset θ_Δ . At some instance, a noise spike can create a sudden large phase deviation from this equilibrium point such that $\hat{\theta}_\Delta[m]$ gets attracted to another stable point (such as $-\pi/2$, $+\pi/2$ or π in a decision-directed scenario). The resultant phase correction in subsequent symbols will contain an additional multiple of $\pi/2$ thus ruining a series of symbol estimates in the future until a mechanism detects these errors, declares a failure and the loop reacquires the true phase.

Hangup A problem that frequently appears during phase acquisition in a feedback structure is a *hangup event*. Assume that the initial value of $\hat{\theta}_\Delta[m]$ is somewhere close to a negative slope zero, see Figure 5.26. Although the system does have a tendency to move to a stable point as indicated by the directional arrows close to $\pm\pi$, the error signal e_D around this location is small. This error signal is the main steering force behind acquisition but here it injects very marginal input into the subsequent blocks. As a consequence, $\hat{\theta}_\Delta[m]$ may dwell around this position for a long time, considerably increasing the acquisition time. A long acquisition time renders a communication failure in many scenarios and is highly undesirable.

Hangup is specifically related to a feedback synchronization system like a PLL. A feedforward procedure does not suffer from this problem by virtue of generating a single estimate in an open loop.

Role of AGC Recall that the gain of a phase difference PED is equal to unity because it employs the four-quadrant inverse tangent operations to compute the angle. Moreover, the gain K_D of both the data-aided and the decision-directed cross product PED is proportional to $2A^2$ where A is the symbol level and the factor 2 appears due to the underlying QPSK modulation and cancels out with the normalizing factor of that modulation (which is $1/\sqrt{2}$ in QPSK). The overall gain is unity again. However, the difference in this case is that the signal values are directly employed without any explicit phase computation. Consequently, the gain K_D in a cross product PED actually depends on the Rx signal amplitude γ as well, which is important in phase synchronization due to the following reason.

We discussed in the design of the phase locked loop in Chapter 4 that the phase error detector gain K_D is utilized in computing the loop filter coefficients (see Eq (4.7)) which in turn promise a certain target performance. Assume that the amplitude varies by a certain factor,

- the detector gain K_D increases,
- the slope of the S-curve increases by the same factor,
- the loop bandwidth increases,
- the noise entering the loop increases, and
- the PLL error variance increases by this factor.

Consequently, the PLL operating conditions only remain the same when the signal amplitude remains constant. For this purpose, there are two options available: estimate the amplitude γ , or fix it at a known value. If it is estimated, then it needs to be re-computed each time the Rx signal amplitude changes which happens very frequently in mobile channels. Then, this changed value is used to compute the updated gain K_D and subsequently the loop filter coefficients. To avoid this problem, a robust design includes an Automatic Gain Control (AGC) that adjusts the amplitude of the Rx signal to a constant value. This in turn enables fixed filter coefficients to achieve a desired PLL response.

5.8 The Small Picture

The crux of almost all the DSP techniques for phase synchronization discussed in this chapter is as follows. With the help of an unsynchronized local oscillator at the Rx, the incoming signal is downconverted to baseband with an arbitrary phase. After downconversion, the actual frequency of the ‘carrier’ is zero (or close to zero in case of a small frequency offset). Eventually, only $L = 1$ complex sample per symbol after matched filtering is required for symbol detection and the final matched filtered complex signal $z(t)$ is just a rotation of the *modulated* I and Q waveforms by that phase offset θ_Δ . After multiplying with the known symbol conjugate $a^*[m]$ or $\hat{a}^*[m]$ or raising to a power, the angle \angle out of the residual Q and I is the desired phase offset $\hat{\theta}_\Delta$. This process is shown in Figure 5.45. All the phase synchronization techniques described in this chapter either explicitly or implicitly accomplish this task.

5.9 Appendix

Here, we derive some relations utilized in the main text.

Effect of Carrier Phase Mismatch

In the absence of noise, the received signal for a passband waveform is the same as the transmitted signal except a carrier phase mismatch, i.e., *we ignore every other distortion in the Rx signal except the phase offset θ_Δ* .

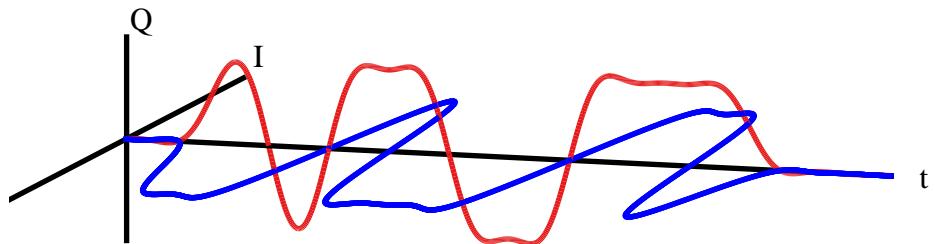
After bandlimiting the incoming signal through a bandpass filter, it is sampled by the ADC operating at $F_S = 1/T_S$ samples/second to produce

$$\begin{aligned} r(nT_S) &= v_I(nT_S)\sqrt{2}\cos(2\pi F_C nT_S + \theta_\Delta) - v_Q(nT_S)\sqrt{2}\sin(2\pi F_C nT_S + \theta_\Delta) \\ &= v_I(nT_S)\sqrt{2}\cos\left(2\pi \frac{k_C}{N}n + \theta_\Delta\right) - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left(2\pi \frac{k_C}{N}n + \theta_\Delta\right) \end{aligned} \quad (5.50)$$

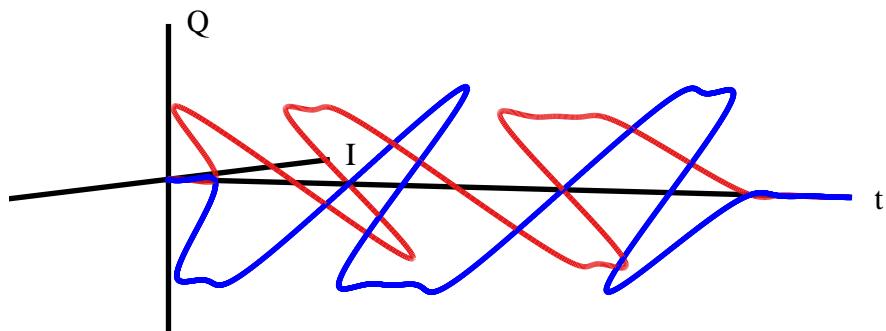
where the relation $F/F_S = k/N$ is used and k_C (in a DFT size of N) corresponds to carrier frequency F_C . Here, we assume a conceptual downconversion in discrete domain but in practice there are different techniques to accomplish this task that depend on the Rx architecture as explained in Chapter 10.

To produce a complex baseband signal from the real Rx signal $r(nT_S)$, the samples of this waveform are input to a mixer which multiplies them with discrete-time quadrature sinusoids $\sqrt{2}\cos 2\pi(k_C/N)n$ in the I arm and $-\sqrt{2}\cdot\sin 2\pi(k_C/N)n$ for Q arm. The reasons for including the factors $\sqrt{2}$ and a negative sign with $\sin(\cdot)$ were explained in Section 3.7.

$a_I[m]$ and $a_Q[m]$ after pulse shaping



$z[m]$: Matched filter output rotated by θ_Δ



$a^*[m]z[m]$

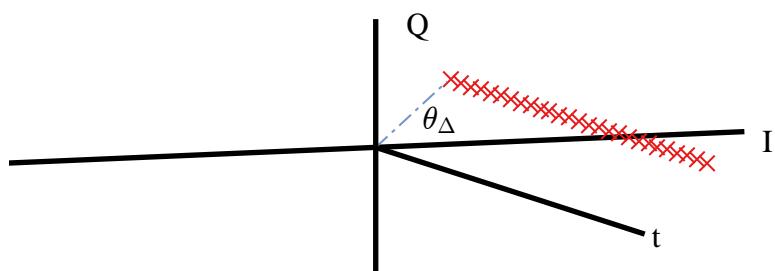


Figure 5.45: The small picture of phase synchronization

We continue the derivation for I part and encourage the reader to solve the Q part as an exercise due to their similarity.

$$\begin{aligned} x_I(nT_S) &= r(nT_S) \cdot \sqrt{2} \cos 2\pi \frac{k_C}{N} n \\ I \rightarrow &= \left\{ v_I(nT_S) \sqrt{2} \cos \left(2\pi \frac{k_C}{N} n + \theta_\Delta \right) - \right. \\ &\quad \left. v_Q(nT_S) \sqrt{2} \sin \left(2\pi \frac{k_C}{N} n + \theta_\Delta \right) \right\} \sqrt{2} \cos 2\pi \frac{k_C}{N} n \end{aligned}$$

Using the identities $\cos A \cdot \cos B = 0.5 \{\cos(A+B) + \cos(A-B)\}$ and $\sin A \cdot \cos B = 0.5 \{\sin(A+B) + \sin(A-B)\}$,

$$\begin{aligned} x_I(nT_S) &= v_I(nT_S) \left\{ \cos \theta_\Delta + \underbrace{\cos \left(2\pi \frac{2k_C}{N} n + \theta_\Delta \right)}_{\text{Double frequency term}} \right\} - \\ I \rightarrow &v_Q(nT_S) \left\{ \sin \theta_\Delta + \underbrace{\sin \left(2\pi \frac{2k_C}{N} n + \theta_\Delta \right)}_{\text{Double frequency term}} \right\} \end{aligned}$$

which can be expressed as

$$I \rightarrow x_I(nT_S) = v_I(nT_S) \cos \theta_\Delta - v_Q(nT_S) \sin \theta_\Delta + \underbrace{v_Q(nT_S) \sin \theta_\Delta + v_I(nT_S) \cos \theta_\Delta}_{\text{Double frequency terms}} \quad (5.51)$$

This is the downconverted inphase signal input to the matched filter. Following a similar line of reasoning, the quadrature part of the downconverted signal is

$$Q \rightarrow x_Q(nT_S) = v_Q(nT_S) \cos \theta_\Delta + v_I(nT_S) \sin \theta_\Delta + \underbrace{v_I(nT_S) \sin \theta_\Delta - v_Q(nT_S) \cos \theta_\Delta}_{\text{Double frequency terms}} \quad (5.52)$$

These two expressions will be utilized in the derivation of phase estimators later. The I matched filter output is written as

$$\begin{aligned} z_I(nT_S) &= x_I(nT_S) * p(-nT_S) \\ I \rightarrow &= \left\{ v_I(nT_S) \cos \theta_\Delta - v_Q(nT_S) \sin \theta_\Delta + \right. \\ &\quad \left. \text{Double frequency terms} \right\} * p(-nT_S) \end{aligned}$$

The double frequency terms in the above equation are filtered out by the matched filter $h(nT_S) = p(-nT_S)$, which also acts as a lowpass filter due to its spectrum limitation to $(1+\alpha)/2T_M$, where T_M is the symbol time. Writing the definitions of $v_I(nT_S)$ and $v_Q(nT_S)$ from Eq (5.2),

$$\begin{aligned} z_I(nT_S) &= \left\{ \sum_i a_I[i] p(nT_S - iT_M) \cos \theta_\Delta - \right. \\ I \rightarrow &\quad \left. \sum_i a_Q[i] p(nT_S - iT_M) \sin \theta_\Delta \right\} * p(-nT_S) \end{aligned}$$

Utilizing the definition of auto-correlation function $r_p(nT_S)$,

$$I \rightarrow z_I(nT_S) = \sum_i \left\{ a_I[i] \cos \theta_\Delta - a_Q[i] \sin \theta_\Delta \right\} r_p(nT_S - iT_M)$$

To generate symbol decisions, T_M -spaced samples of the matched filter output are required at $n = mL = mT_M/T_S$. Downsampling the matched filter output generates

$$\begin{aligned} I \rightarrow z_I(mT_M) &= z_I(nT_S) \Big|_{n=mL=mT_M/T_S} \\ &= \sum_i \left\{ a_I[i] \cos \theta_\Delta - a_Q[i] \sin \theta_\Delta \right\} r_p(mT_M - iT_M) \end{aligned}$$

For a square-root Nyquist pulse that satisfies no-ISI criterion of Eq (3.24), $r_p(mT_M - iT_M)$ is zero except for $i = m$ where its value is 1. Thus,

$$I \rightarrow z_I(mT_M) = a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta$$

A similar derivation for Q matched filter output yields the final expression for the symbol-spaced samples in the presence of phase offset θ_Δ .

$$\begin{aligned} I \rightarrow z_I(mT_M) &= a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \\ Q \uparrow z_Q(mT_M) &= a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \end{aligned}$$

Mean Curve for Phase Difference Data-Aided PED

To derive the mean curve, the first step is to consider how a phase offset θ_Δ rotates the matched filter output $z(mT_M)$ anticlockwise. Recall the phase rotation rule $I \cos \theta - Q \sin \theta$ and $Q \cos \theta + I \sin \theta$,

$$\begin{aligned} I \rightarrow z_I(mT_M) &= a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \\ Q \uparrow z_Q(mT_M) &= a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \end{aligned}$$

Next, recall how the phase rotation block in Figure 5.17 de-rotates the matched filter outputs by its updated estimate $\hat{\theta}_\Delta$ (see Eq (5.17), for example).

$$\begin{aligned} I \rightarrow \hat{z}_I(mT_M) &= z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \\ Q \uparrow \hat{z}_Q(mT_M) &= z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \end{aligned}$$

Let us plug the values of $z_I(mT_M)$ and $z_Q(mT_M)$ from the former equation into the latter. As a result, the de-rotated matched filter output can be represented as

$$\begin{aligned} I \rightarrow \hat{z}_I(mT_M) &= \left\{ a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \right\} \cos \hat{\theta}_\Delta + \\ &\quad \left\{ a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \right\} \sin \hat{\theta}_\Delta \end{aligned}$$

and

$$\begin{aligned} Q \rightarrow \hat{z}_Q(mT_M) &= \left\{ a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \right\} \cos \hat{\theta}_\Delta - \\ &\quad \left\{ a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \right\} \sin \hat{\theta}_\Delta \end{aligned}$$

Using the identities $\cos A \cdot \cos B \mp \sin A \cdot \sin B = \cos(A \pm B)$ and $\sin A \cos B \pm \cos A \sin B = \sin(A \pm B)$, we get in terms of $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$,

$$\begin{aligned} I \rightarrow \hat{z}_I(mT_M) &= a_I[m] \cos \theta_{\Delta:e} - a_Q[m] \sin \theta_{\Delta:e} \\ Q \uparrow \hat{z}_Q(mT_M) &= a_Q[m] \cos \theta_{\Delta:e} + a_I[m] \sin \theta_{\Delta:e} \end{aligned} \tag{5.53}$$

The above relation holds for any Phase Error Detector (PED) and we will use it to derive their mean curves. We start with the mean curve for a data-aided phase difference PED.

One version of the phase difference PED in Eq (5.20) was given as

$$e_D[m] = \tan^{-1} \frac{\left\{ a^*[m] \widehat{z}(mT_M) \right\}_Q}{\left\{ a^*[m] \widehat{z}(mT_M) \right\}_I} = \tan^{-1} \frac{a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M)}{a_I[m] \widehat{z}_I(mT_M) + a_Q[m] \widehat{z}_Q(mT_M)}$$

To find the expression for the mean curve, plug the expressions of $\widehat{z}_I(mT_M)$ and $\widehat{z}_Q(mT_M)$ from Eq (5.53) above in this equation. Thus, the denominator (I part) can be written as

$$\begin{aligned} & a_I[m] \widehat{z}_I(mT_M) + a_Q[m] \widehat{z}_Q(mT_M) \\ &= a_I[m] \left\{ a_I[m] \cos \theta_{\Delta:e} - a_Q[m] \sin \theta_{\Delta:e} \right\} + \\ & \quad a_Q[m] \left\{ a_Q[m] \cos \theta_{\Delta:e} + a_I[m] \sin \theta_{\Delta:e} \right\} \\ &= a_I^2[m] \cos \theta_{\Delta:e} - a_I[m] a_Q[m] \sin \theta_{\Delta:e} + a_Q^2[m] \cos \theta_{\Delta:e} + a_Q[m] a_I[m] \sin \theta_{\Delta:e} \\ &= (a_I^2[m] + a_Q^2[m]) \cos \theta_{\Delta:e} \end{aligned}$$

Similarly, the numerator (Q component) can be simplified as

$$a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M) = (a_I^2[m] + a_Q^2[m]) \sin \theta_{\Delta:e}$$

Plugging the numerator and denominator back in the PED expression yields

$$e_D[m] = \tan^{-1} \frac{\sin \theta_{\Delta:e}}{\cos \theta_{\Delta:e}} = \tan^{-1} \tan \theta_{\Delta:e} = \theta_{\Delta:e} \quad (5.54)$$

This expression is independent of the data symbols $a[m]$, therefore its average with respect to the data symbol remains the same. The final expression is thus given by

$$\overline{e_D} = \theta_{\Delta:e}$$

Mean Curve for Cross Product Data-Aided PED

In this case, we use Eq (5.53) that related the expressions of de-rotated matched filter outputs $\widehat{z}_I(mT_M)$ and $\widehat{z}_Q(mT_M)$ with phase error $\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$. Plugging in the expression for the cross product phase error detector in Eq (5.32) in the presence of an amplitude scaling factor γ as discussed above, we get

$$\begin{aligned} e_D[m] &= \gamma \left(a_I[m] \widehat{z}_Q(mT_M) - a_Q[m] \widehat{z}_I(mT_M) \right) \\ &= \gamma \left(a_I[m] \left\{ a_Q[m] \cos \theta_{\Delta:e} + a_I[m] \sin \theta_{\Delta:e} \right\} - \right. \\ & \quad \left. a_Q[m] \left\{ a_I[m] \cos \theta_{\Delta:e} - a_Q[m] \sin \theta_{\Delta:e} \right\} \right) \\ &= \left\{ a_I^2[m] + a_Q^2[m] \right\} \gamma \sin \theta_{\Delta:e} \end{aligned} \quad (5.55)$$

For a QPSK constellation, both the training symbols a_I and a_Q take values from a finite symbol set $\pm A$ yielding a total of four possible choices.

$$\{+A, +A\}, \quad \{-A, +A\}, \quad \{+A, -A\}, \quad \{-A, -A\}$$

Each such option has a 1/4 chance of occurring, hence averaging over all these values in relation to Eq (5.55) generates

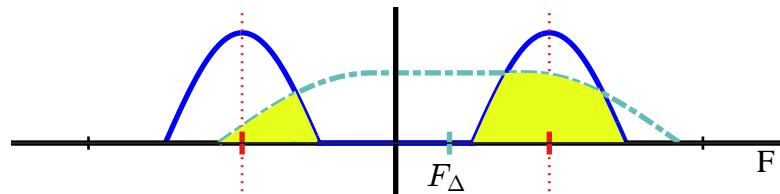
$$\overline{e_D} = \frac{1}{4} \left\{ (+A)^2 + (+A)^2 \right\} \gamma \sin \theta_{\Delta:e} + \frac{1}{4} \left\{ (-A)^2 + (+A)^2 \right\} \gamma \sin \theta_{\Delta:e} + \\ \frac{1}{4} \left\{ (+A)^2 + (-A)^2 \right\} \gamma \sin \theta_{\Delta:e} + \frac{1}{4} \left\{ (-A)^2 + (-A)^2 \right\} \gamma \sin \theta_{\Delta:e}$$

and the final expression is given by

$$\overline{e_D} = 2A^2 \gamma \sin \theta_{\Delta:e}$$

Chapter 6

Carrier Frequency Synchronization



“Excess bandwidth provides the energy required for timing and frequency synchronization.”

fred harris

Until Chapter 5, it was assumed that the Rx signal samples incur a phase offset that does not change significantly during the reception of the whole sequence. In reality, the Rx signal phase is varying with time. In general, the Rx signal contains both a phase offset that is the constant part of the signal phase and a frequency offset that is the part of the phase changing linearly with time. In this chapter, our target is to discuss the estimation and compensation of this Carrier Frequency Offset (CFO).

In physics, frequency is defined as the number of cycles per unit time and its units are cycles/second or Hz. Angular frequency is the rate of change of phase of a sinusoidal waveform and its units are radians/second that leads to the equation

$$2\pi f = \frac{\Delta\theta}{\Delta t}$$

where $\Delta\theta$ and Δt are the changes in angle and time, respectively. Due to this reason, a small *Carrier Frequency Offset (CFO)*, or F_Δ , in the received signal spins the constellation in a circle (or multiple circles for higher-order modulations). An example of such a spinning constellation will be shortly illustrated.

A Carrier Frequency Offset (CFO) usually arises due to two reasons:

Frequency mismatch between the Tx and Rx oscillators: In the discussion on pass-band modulation schemes like QAM and PSK so far, we assumed perfect down-conversion, i.e., the frequency of the local oscillator at the Rx is exactly the same as the carrier of the incoming signal. Obviously, this is not true in practice as no two devices are the same and there is always some difference between the manufacturer's nominal specification and the real frequency of that device. Moreover, this actual frequency keeps varying (slightly) with temperature, pressure, age and other factors.

Doppler effect: A moving Tx, moving Rx or any kind of movement around the channel creates a Doppler shift that creates a carrier frequency offset as well. The Doppler effect impacts the Rx signal in a different manner for a single path channel as compared to a multipath channel. We will learn more about it during the discussion of a wireless channel in Chapter 8.

Therefore, a specific subsystem needs to be designed at the Rx to acquire the exact frequency of the Tx signal. Next we examine the effect of a carrier frequency mismatch on the Rx signal sequence.

6.1 Effect of Carrier Frequency Mismatch

The accuracy of local oscillators in communication receivers is defined in terms of their *ppm (parts per million)* rating. The ppm rating at the oscillator crystal indicates how much its frequency may deviate from the nominal value. So 1 ppm is just what it says: 1 out of 10^6 parts, just like a percent is parts per hundred. To get a feel of how

big or small this number is, 10^6 seconds translate into

$$\frac{10^6 \text{ seconds}}{24 \text{ hours/day} \times 3600 \text{ seconds/hour}} \approx 11.5 \text{ days}$$

Assuming a realistic number for crystal inaccuracy as an example, a 20 ppm rating is equivalent to a deviation of 20 seconds every 11.5 days or $30 \times 20/11.5 \approx 52$ seconds each month. This might sound entirely harmless but for the purpose of typical wireless communication systems operating at several GHz of carrier frequency F_C , it is one of the major sources of signal distortion.

Consider an example of a wireless system operating at 2.4 GHz and with ± 20 ppm crystals, which is typical of many applications. The maximum deviation of the carrier frequency at the Tx or Rx can be

$$\pm \frac{20}{10^6} \times 2.4 \times 10^9 = \pm 48 \text{ kHz}$$

However, in the worst case scenario, the Tx can be 20 ppm above (or below) the nominal frequency, while the Rx can be 20 ppm below (or above) the nominal frequency, resulting in the overall difference of 40 ppm between the two. So the worst case CFO due to local oscillator mismatch in this example can be

$$2 \times 48 = 96 \text{ kHz}$$

Depending on the symbol rate $R_M = 1/T_M$, this induces a significant amount of distortion as we shortly see.

Keep in mind that this calculation is for basic precision and the actual frequency may vary depending on environmental factors, mainly the temperature and aging. Finally, a movement anywhere in the channel (whether by Tx, Rx or some other object within that environment) adds Doppler shift which can be up to several hundreds of Hz. Although this Doppler shift is much less than the oscillator generated mismatch, it severely distorts the Rx signal in an entirely different manner, as we will find out in Chapter 8.

Now we turn our attention towards Figure 6.1 to differentiate between three possibilities in which F_Δ can dictate the receiver design.

Case 1: Very Large CFO

This is the scenario encountered when CFO

$$F_\Delta > F_{\Delta,\max}$$

where $F_{\Delta,\max}$ is the maximum expected CFO between the Tx and Rx signal. It was illustrated in Figure 2.41 that an input signal at the Rx is filtered by an analog prefilter to remove the out-of-band noise as well as any interfering signal. Ideally, for a signal with bandwidth B , the frequency response of this prefilter $G(F)$ should be flat within the frequency range

$$|F| \leq B + F_{\Delta,\max}$$

so that the incoming signal can pass through in an undistorted manner[†]. The passband width of this prefilter is designed according to the maximum CFO $F_{\Delta,\max}$ expected at the Rx.

[†]We are assuming a flat wireless channel here as well. Actual wireless channels will be discussed later in Chapter 8.

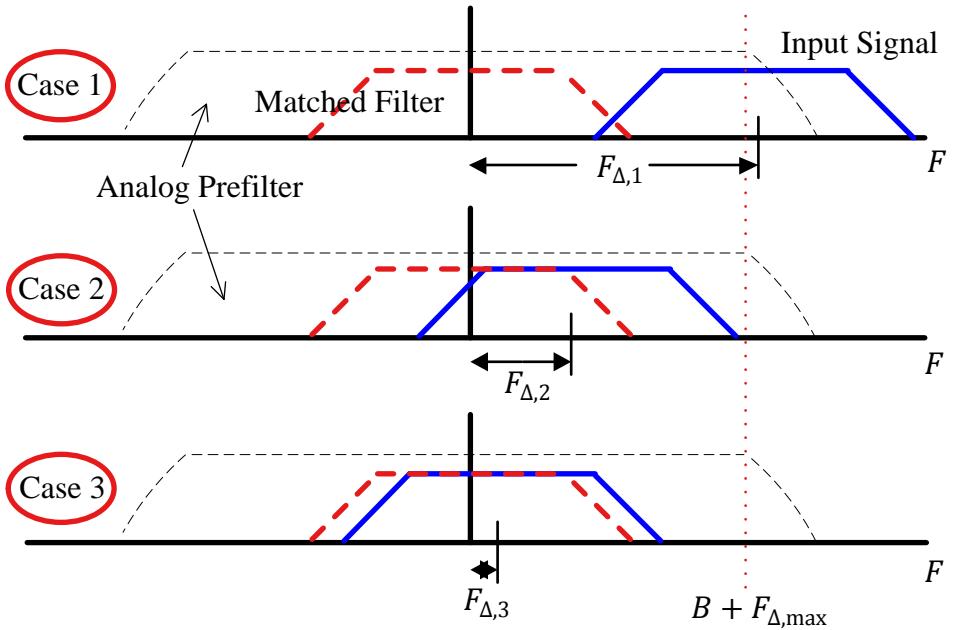


Figure 6.1: Three different cases for carrier frequency offset (1) $\text{CFO} > F_{\Delta,\max}$ (2) $0.15/T_M < \text{CFO} < F_{\Delta,\max}$ (3) $\text{CFO} < 0.15/T_M$

However, if the CFO is greater than $F_{\Delta,\max}$, then much of the actual intended signal will be filtered out by the analog prefilter and the Rx sampled signal will not even closely resemble the Tx signal as a linear function of Tx data. No amount of signal processing can then recover the signal. Since it is an outcome of poor system design, the only remedy is to redesign the system (particularly the analog frontend) with more accurate estimates and margins for CFO and other such random distortions.

Case 2: Large CFO

Here, the condition is given by

$$15\% \text{ of } \frac{1}{T_M} < F_{\Delta} < F_{\Delta,\max}$$

Since CFO F_{Δ} is less than $F_{\Delta,\max}$, the Rx signal is within the passband of the analog prefilter $G(F)$ and ideally suffers no distortion. However, remember from Section 3.4 that to maximize the SNR, the Rx signal must be passed through a matched filter. This is not possible in this case because much of the Rx signal bandwidth does not sufficiently overlap with the matched filter due to the CFO being greater than a certain percentage of the symbol rate $1/T_M$ which happens to lie around 15%. If applied, it would remove a significant portion of the incoming signal energy.

Since Rx signal cannot be matched filtered without distorting the signal, and the signal cannot be downsampled to 1 sample/symbol without matched filtering, it is easy to deduce that more than 1 sample/symbol (say, L) is required to trace the frequency offset.

Case 3: Small CFO

Finally, the CFO F_Δ here is

$$F_\Delta < 15\% \text{ of } \frac{1}{T_M}$$

When the signal is rotated by less than 15% of $1/T_M$, a CFO does not significantly distort the incoming signal on a symbol-by-symbol basis, and hence *the effect of this rotation on one symbol, although still significant, can now be tracked at symbol rate, or 1 sample/symbol*. In other words, the matched filtering keeps most of the signal intact and as a result, symbol boundaries can be marked first (a problem known as symbol timing synchronization which we discuss in Chapter 7) before carrier frequency is estimated at the symbol-spaced samples.

To see the effect of the carrier frequency offset F_Δ for this case, consider again a received passband signal consisting of two PAM waveforms in I and Q rails. We start with Figure 6.2 to clarify the signal notations used at various stages of the transceiver where $v_I(t)$ and $v_Q(t)$ are the continuous versions of sampled signals $v_I(nT_S)$ and $v_Q(nT_S)$ given by

$$\begin{aligned} I \rightarrow v_I(nT_S) &= \sum_i a_I[i] p(nT_S - iT_M) \\ Q \uparrow v_Q(nT_S) &= \sum_i a_Q[i] p(nT_S - iT_M) \end{aligned}$$

In the above equation, $a_I[i]$ and $a_Q[i]$ are the inphase and quadrature components of the i^{th} symbol, $p(nT_S)$ are the samples of a square-root Nyquist pulse while T_S and T_M are the sample time and symbol time, respectively.

The Rx signal can be written as

$$\begin{aligned} r(t) &= v_I(t)\sqrt{2} \cos [2\pi(F_C + F_\Delta)t + \theta_\Delta] - v_Q(t)\sqrt{2} \sin [2\pi(F_C + F_\Delta)t + \theta_\Delta] \\ &= v_I(t)\sqrt{2} \cos [2\pi F_C t + 2\pi F_\Delta t + \theta_\Delta] - \\ &\quad v_Q(t)\sqrt{2} \sin [2\pi F_C t + 2\pi F_\Delta t + \theta_\Delta] \end{aligned} \quad (6.1)$$

Here, the carrier offset can be seen as $2\pi F_\Delta t$ which is changing with time and there is a phase offset θ_Δ present. As derived in Appendix 6.6, the symbol-spaced matched filter output in the presence of a small frequency offset F_Δ is given as

$$\begin{aligned} I \rightarrow z_I(mT_M) &= a_I[m] \cos(2\pi F_0 m + \theta_\Delta) - a_Q[m] \sin(2\pi F_0 m + \theta_\Delta) \\ Q \uparrow z_Q(mT_M) &= a_Q[m] \cos(2\pi F_0 m + \theta_\Delta) + a_I[m] \sin(2\pi F_0 m + \theta_\Delta) \end{aligned} \quad (6.2)$$

where F_0 is the normalized Carrier Frequency Offset (nCFO): CFO normalized by the symbol rate $R_M = 1/T_M$.

$$F_0 = \frac{F_\Delta}{R_M} = F_\Delta T_M \quad (6.3)$$

From the phase rotation rule of complex numbers $I \cdot \cos \theta - Q \cdot \sin \theta$ and $Q \cdot \cos \theta + I \cdot \sin \theta$, we know that this expression is nothing but a continuous anticlockwise rotation

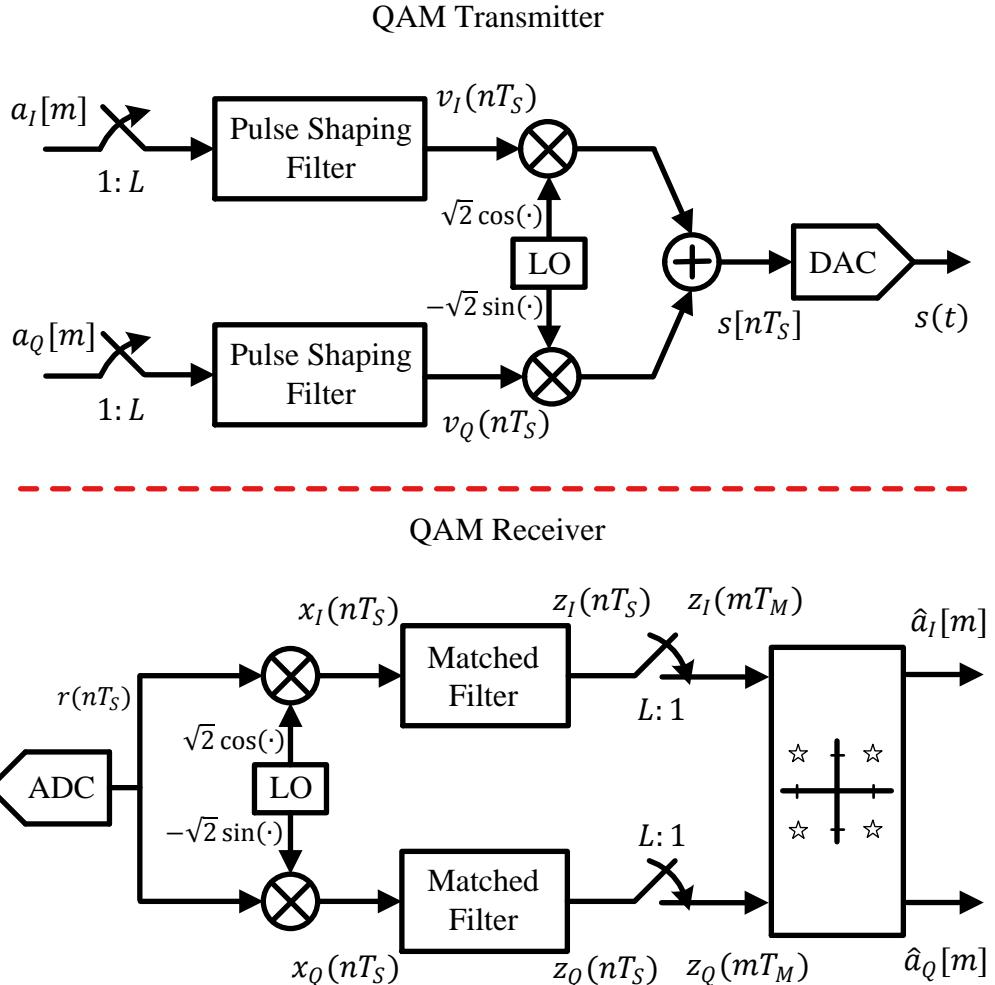


Figure 6.2: A block diagram of a QAM system to explain the signal notations at various blocks

by an angle $2\pi F_0 m + \theta_\Delta$. In polar form, this expression can be written as

$$\boxed{\begin{aligned}|z(mT_M)| &= \sqrt{a_I^2[m] + a_Q^2[m]} \\ \angle z(mT_M) &= \angle a[m] + 2\pi F_0 m + \theta_\Delta\end{aligned}} \quad (6.4)$$

The expression $2\pi F_0 m$ in the resultant phase being a function of symbol index m , we can conclude that a small frequency offset keeps the magnitude unchanged but continually rotates the desired output $a[m]$ on the constellation plane, *thus spinning the Rx constellation*. This is a natural outcome since the angular frequency is defined as the rate of change of phase. This is drawn for symbol-spaced samples in the scatter plot of Figure 6.3a for a 4-QAM constellation and Figure 6.3b for 16-QAM constellation.

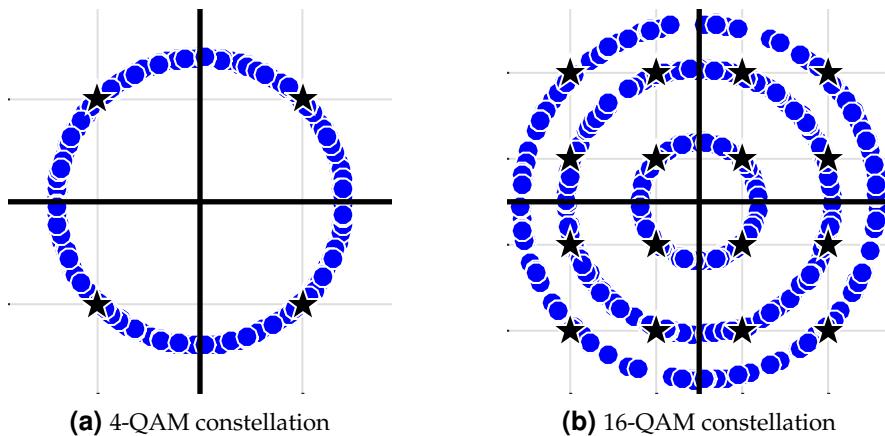


Figure 6.3: A small CFO on the matched filtered samples spins the constellation in circles

Note 6.1 The invisible dark matter

These rotating constellations remind me of an evidence presented by the astronomers about the existence of the dark matter [17]. Some galaxies are spinning so fast that their stars should fall off their outer edges, similar to Figure 6.3 which looks like many ping pong balls arranged in a merry-go-round. This is because the force of gravity from all the mass present in the galaxy is not enough to hold the stars together. Since this does not happen, it is believed that this additional mass comes from the invisible dark matter. Dark matter is considered to account for approximately 85% of the matter in the universe and it is indifferent to the entire electromagnetic spectrum, thus appearing invisible (or dark) to our measurement tools.

The eye diagram for a signal stream affected by a carrier frequency offset does not reveal much information since the continuously changing phase (overlapped many times for the purpose of eye diagram) appears to randomize the signal. For a single pulse shaped waveform, however, an overriding complex sinusoid can easily be identified. The above results are summarized in Table 6.1.

Keep in mind that we will use different notations for different representations of a

Table 6.1: Classification of CFO F_A

	Distortion	Matched Filter	Samples/Symbol
$F_\Delta > F_{\Delta,\max}$	Yes	No	None
$0.15/T_M < F_\Delta < F_{\Delta,\max}$	No	No	L
$F_\Delta < 0.15/T_M$	No	Yes	1

Carrier Frequency Offset (CFO) which will become clear from the context. For example,

$$\begin{aligned} \text{Actual CFO} &\rightarrow F_\Delta \\ \text{Discrete CFO (normalized)} F_\Delta T_M &\rightarrow F_0 \\ \text{Discrete frequency index for } F_0 \text{ in an } N\text{-point DFT} &\rightarrow k_0 \end{aligned}$$

6.2 The Big Picture

In Section 5.3, we said that carrier phase synchronization is done at the end of the Rx signal processing chain due to the very nature of the DSP implementation. And that almost all DSP based phase synchronization algorithms are timing-aided. Timing acquisition implies knowing the symbol boundaries in the Rx sampled waveform which is equivalent to identifying the optimal sampling instants where the eye opening is maximum and Inter-Symbol Interference (ISI) from the neighbouring symbols is zero, see Figure 5.9.

In the case of carrier frequency synchronization, this is not true. From Section 6.1, we know that a CFO can be large or small which leads to the following two synchronization strategies.

Non-timing-aided techniques For a large normalized CFO F_0 greater than 15% of the symbol rate $1/T_M$, digital processing of the incoming signal must be started without any knowledge of the symbol timing. Imagine in frequency domain, e.g., as in Figure 6.1, how a large CFO reduces the overlap between the incoming signal and the bandwidth processed by digital filters further down the road. At this stage, it is necessary to acquire the carrier frequency to a reasonably close value before any meaningful manipulations on the data can proceed including timing acquisition, necessarily leading to non-timing-aided algorithms.

A further consequence of the timing absence is that downsampling at symbol rate $1/T_M$ is not feasible since there is no zero-ISI sample yet located. Consequently, non-timing-aided frequency synchronization in general operates at a rate larger than $1/T_M$, say L samples/symbol.

Finally, no information about the data can be assumed even if there was a preamble or training sequence embedded in the Tx signal stream because the Rx cannot mark which samples of the Rx sequence correspond to those known symbols. Therefore, all non-timing-aided carrier frequency synchronization algorithms are non-data-aided as well.

Timing-aided techniques When the CFO is small enough (usually less than 15% of symbol rate $1/T_M$), timing can be acquired first from an oversampled and slowly rotating constellation. Timing-aided in turn implies that while the matched filtering occurs at L samples/symbol, any processing faster than at symbol rate $1/T_M$ to adjust the frequency is unnecessary. Therefore, the frequency synchronization block works at exactly the symbol rate $1/T_M$, or just 1 sample/symbol to minimize the computational complexity.

In this case, both data-aided and non-data-aided techniques are possible to implement depending on whether a training sequence is present or not. Having known the symbol boundaries, information about the training sequence can be

incorporated into designing a frequency synchronization algorithm. This scenario also holds during Rx operation in steady state (tracking mode) that implies small CFO values. An even smaller residual CFO can be tracked by a PLL if the loop filter includes an integrator component.

Due to this larger role of timing in the development of frequency synchronization algorithms, I classify the techniques here as timing-aided and non-timing-aided, as opposed to just data-aided/decision-directed and non-data-aided, which was the strategy adopted in carrier phase synchronization scenario, see Figure 5.10. The current classification is illustrated in Figure 6.4. Whether that technique requires information about data or not will be clear from the context.

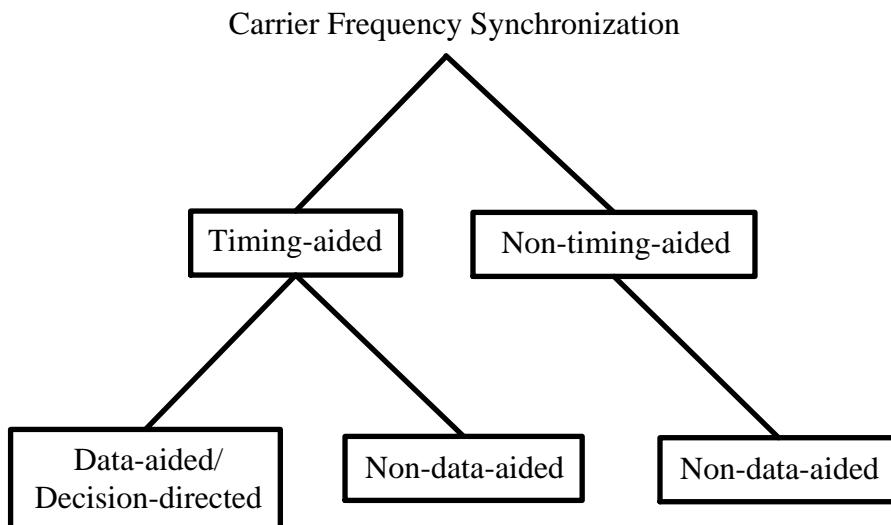


Figure 6.4: Classification of frequency synchronization algorithms. As opposed to the carrier phase, both timing-aided and non-timing-aided techniques can be utilized depending on the system requirements

Keep in mind that the noise power in non-timing-aided frequency estimation procedures is significantly higher than the timing-aided case. This is because the bandwidth of the analog prefilter shown in Figure 6.1 is considerably larger than the signal bandwidth (and the matched filter), thus allowing more noise affect the samples. Consequently, a very large averaging interval is usually required increasing the acquisition time. In many cases, a reasonable design is to adopt a two-stage strategy.

- In the first stage, a large CFO is significantly reduced by using a non-timing-aided technique. Although the CFO estimate thus generated is coarse, it is still good enough to allow matched filtering and timing acquisition from the discrete samples.
- In the second stage, a timing-aided scheme is applied to completely compensate for any residual (or fine) CFO left during the first stage.

Next, we move towards deriving techniques for estimating and compensating for this frequency offset.

6.3 Timing-Aided Techniques

As discussed above, a timing-aided scenario logically implies the following set of conditions.

- The CFO is small, i.e., $F_\Delta < 0.15/T_M$.
- The signal can be downsampled from L to 1 sample/symbol.
- The data symbols $a[m]$ are either known or a decision $\hat{a}[m]$ can be taken on them.

The estimators for large CFO running at L samples/symbol will be discussed later in the non-data-aided case. Continuing with correlation as the Master Algorithm for a digital communication system, we extend its applications from matched filtering and phase synchronization to frequency synchronization problem here. Most of the development in the timing-aided techniques here follows from the excellent Ref. [18].

6.3.1 Enter the Correlation

To focus on estimating the frequency offset, assume for now that

- $\theta_\Delta = 0$, as it does not affect the estimation of F_Δ that works on phase differences thus canceling θ_Δ out, and
- the sample rate F_S is L times the symbol rate $R_M = 1/T_M$, i.e., $F_S = L/T_M$.

The discussion from hereon follows similar to that in phase synchronization in Section 5.4.

In the presence of noise, the received and sampled signal is

$$r(nT_S) = s(nT_S) + \text{noise}$$

With a frequency offset F_Δ , the Tx signal $s(nT_S)$ is given by

$$\begin{aligned} s(nT_S) &= v_I(nT_S)\sqrt{2}\cos(2\pi(F_C + F_\Delta)nT_S) - \\ &\quad v_Q(nT_S)\sqrt{2}\sin(2\pi(F_C + F_\Delta)nT_S) \\ &= v_I(nT_S)\sqrt{2}\cos\left(2\pi\frac{F_C}{F_S}n + 2\pi\frac{F_\Delta}{F_S}n\right) - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left(2\pi\frac{F_C}{F_S}n + 2\pi\frac{F_\Delta}{F_S}n\right) \end{aligned}$$

where $v_I(nT_S)$ and $v_Q(nT_S)$ are the inphase and quadrature waveforms

$$\begin{aligned} I &\rightarrow & v_I(nT_S) &= \sum_m a_I[m]p(nT_S - mT_M) \\ Q &\uparrow & v_Q(nT_S) &= \sum_m a_Q[m]p(nT_S - mT_M) \end{aligned}$$

This relation can be modified as

$$\begin{aligned} s(nT_S) &= v_I(nT_S)\sqrt{2}\cos\left(2\pi\frac{k_C}{N}n + 2\pi\frac{F_0}{L}n\right) - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left(2\pi\frac{k_C}{N}n + 2\pi\frac{F_0}{L}n\right) \end{aligned} \tag{6.5}$$

where k_C corresponds to carrier frequency F_C and from the discussion regarding normalized CFO F_0 around Eq (6.3),

$$\frac{F_\Delta}{F_S} = \frac{F_\Delta}{L/T_M} = \frac{F_0}{L}$$

Next, we employ the same assumptions as in Section 5.4, namely a finite observation interval of N_0 symbols, perfect frame synchronization and no distortion at edges due to the extended pulse length. Then, the correlation between the Rx signal $r(nT_S)$ and our expected template signal $s(nT_S)$ from Eq (6.5) is

$$\begin{aligned}\text{corr}[0] &= \sum_{n=0}^{LN_0-1} r(nT_S)s(nT_S) \\ &= \sum_{n=0}^{LN_0-1} r(nT_S) \left\{ v_I(nT_S) \sqrt{2} \cos \left(2\pi \frac{k_C}{N} n + 2\pi \frac{F_0}{L} n \right) - \right. \\ &\quad \left. v_Q(nT_S) \sqrt{2} \sin \left(2\pi \frac{k_C}{N} n + 2\pi \frac{F_0}{L} n \right) \right\}\end{aligned}$$

Using the identities $\cos(A + B) = \cos A \cos B - \sin A \sin B$ and $\sin(A + B) = \sin A \cos B + \cos A \sin B$ and collecting common terms, we get

$$\begin{aligned}\text{corr}[0] &= \sum_{n=0}^{LN_0-1} r(nT_S) \sqrt{2} \cos 2\pi \frac{k_C}{N} n \cdot \\ &\quad \left\{ v_I(nT_S) \cos 2\pi \frac{F_0}{L} n - v_Q(nT_S) \sin 2\pi \frac{F_0}{L} n \right\} - \\ &\quad \sum_{n=0}^{LN_0-1} r(nT_S) \sqrt{2} \sin 2\pi \frac{k_C}{N} n \cdot \\ &\quad \left\{ v_Q(nT_S) \cos 2\pi \frac{F_0}{L} n + v_I(nT_S) \sin 2\pi \frac{F_0}{L} n \right\}\end{aligned}$$

From the Rx block diagram in Figure 6.2, a product of $r(nT_S)$ with the sinusoid $\sqrt{2} \cos 2\pi(k_C/N)n$ produces $x_I(nT_S)$ that is the I component in the top branch while the product of $r(nT_S)$ with $-\sqrt{2} \sin 2\pi(k_C/N)n$ generates $x_Q(nT_S)$ that is the Q part in the bottom branch. Substituting these terms along with the expressions for $v_I(nT_S)$ and $v_Q(nT_S)$ yields

$$\begin{aligned}\text{corr}[0] &= \sum_{n=0}^{LN_0-1} x_I(nT_S) \left(\sum_m a_I[m] p(nT_S - mT_M) \cos 2\pi \frac{F_0}{L} n - \right. \\ &\quad \left. \sum_m a_Q[m] p(nT_S - mT_M) \sin 2\pi \frac{F_0}{L} n \right) \\ &\quad + \sum_{n=0}^{LN_0-1} x_Q(nT_S) \left(\sum_m a_Q[m] p(nT_S - mT_M) \cos 2\pi \frac{F_0}{L} n + \right. \\ &\quad \left. \sum_m a_I[m] p(nT_S - mT_M) \sin 2\pi \frac{F_0}{L} n \right)\end{aligned}$$

Rearranging the above and recalling the fact that $p(nT_S)$ are the samples of a square-root

Nyquist pulse with support $-LG \leq n \leq LG$ samples,

$$\begin{aligned} \text{corr}[0] &= \sum_{m=0}^{N_0-1} a_I[m] \left\{ \underbrace{\sum_{n=(m-G)L}^{(m+G)L} \left(x_I(nT_S) \cos 2\pi \frac{F_0}{L} n + x_Q(nT_S) \sin 2\pi \frac{F_0}{L} n \right)}_{\tilde{x}_I(nT_S)} \cdot p(nT_S - mT_M) \right\} \\ &\quad + \sum_{m=0}^{N_0-1} a_Q[m] \left\{ \underbrace{\sum_{n=(m-G)L}^{(m+G)L} \left(x_Q(nT_S) \cos 2\pi \frac{F_0}{L} n - x_I(nT_S) \sin 2\pi \frac{F_0}{L} n \right)}_{\tilde{x}_Q(nT_S)} \cdot p(nT_S - mT_M) \right\} \end{aligned} \quad (6.6)$$

Observe the following regarding the above equation.

- From the conjugate multiplication rule $I \cdot I + Q \cdot Q$ and $Q \cdot I - I \cdot Q$, the signal $x(nT_S)$ is being de-rotated by a complex sinusoid of frequency F_0/L , the output of which is $\tilde{x}(nT_S)$.

$$\begin{aligned} I \rightarrow \quad \tilde{x}_I(nT_S) &= x_I(nT_S) \cos 2\pi \frac{F_0}{L} n + x_Q(nT_S) \sin 2\pi \frac{F_0}{L} n \\ Q \uparrow \quad \tilde{x}_Q(nT_S) &= x_Q(nT_S) \cos 2\pi \frac{F_0}{L} n - x_I(nT_S) \sin 2\pi \frac{F_0}{L} n \end{aligned} \quad (6.7)$$

- Next, this signal is correlated with the pulse shape $p(nT_S)$.
- Finally, after summing over the sample index n , we are left with a signal with **symbol rate spaced samples at times mT_M** , i.e., $L = 1$ sample/symbol now due to the timing-aided scenario.

As a consequence of above steps, the output of matched filter $z(mT_M)$ is defined as the correlation of $\tilde{x}(nT_S)$ with $p(nT_S)$, which is the same as convolution of $\tilde{x}(nT_S)$ with $p(-nT_S)$.

$$\begin{aligned} z_I(mT_M) &= \tilde{x}_I(nT_S) * p(-nT_S) \\ I \rightarrow \quad &= \sum_{n=(m-G)L}^{(m+G)L} \tilde{x}_I(nT_S) p(nT_S - mT_M) \end{aligned} \quad (6.8)$$

$$\begin{aligned} z_Q(mT_M) &= \tilde{x}_Q(nT_S) * p(-nT_S) \\ Q \rightarrow \quad &= \sum_{n=(m-G)L}^{(m+G)L} \tilde{x}_Q(nT_S) p(nT_S - mT_M) \end{aligned} \quad (6.9)$$

Then, the correlation result is given from Eq (6.6) by

$$\begin{aligned} \text{corr}[0] &= \sum_{m=0}^{N_0-1} \left\{ a_I[m] z_I(mT_M) + a_Q z_Q(mT_M) \right\} \\ &= \left\{ \sum_{m=0}^{N_0-1} a^*[m] z(mT_M) \right\}_I \end{aligned} \quad (6.10)$$

Recall from Section 3.7 that the downconverted signal $x(nT_S)$ is processed by the Rx through a matched filter $p(-nT_S)$ only. In light of the above discussion, the Rx processing now consists

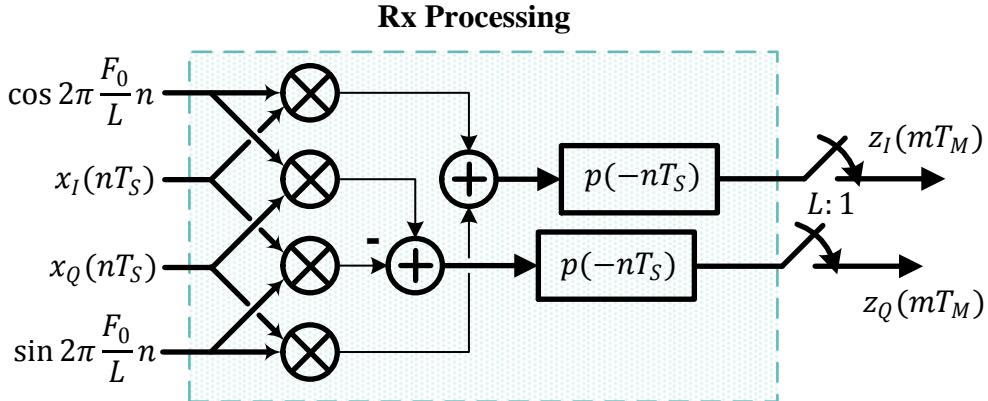


Figure 6.5: The Rx processing includes mixing $x(nT_S)$ with a complex sinusoid of frequency $-F_0/L$ before filtering with $p(-nT_S)$

of both mixing $x(nT_S)$ with a complex sinusoid of frequency $-F_0/L$, which becomes $-F_0$ after downsampling by L , as well as a matched filter $p(-nT_S)$ [†]. This is illustrated in Figure 6.5.

The only problem in implementing the above solution is that we do not know the value of F_0 which we solve in the subsequent sections. We will proceed with the above description to derive the frequency estimate \hat{F}_0 that maximizes the correlation. Recall that acquired timing in this section implies that if the Tx inserts known symbols within the message as a preamble or a training sequence, then the Rx can estimate the desired parameters through exploiting this ‘data’. This leads to a data-aided strategy of synchronization as below.

6.3.2 Feedforward: Brute Force Estimator

In burst mode wireless communications with short packets, the frequency offset F_Δ remains constant throughout the duration of the packet and a single shot estimator is enough for its compensation. Consequently, a feedforward (FF) strategy can be adopted through block processing of the whole symbol stream, where the primary task is to develop an expression for the CFO estimator \hat{F}_0 . Then, this estimate can be treated as the unknown original frequency offset with which the sequence can be de-rotated.

Notice that the normalized CFO F_0 is hidden in the samples $z(mT_M)$ and therefore there is no direct strategy that maximizes the correlation and yields a closed-form expression for the estimator. One viable method is to

- compensate $r(nT_S)$ with different trial values of \hat{F}_0 - Eq (6.7),
- form the matched filter output - Eq (6.8) and Eq (6.9),
- compute the correlation - Eq (6.10), and
- choose the value with the highest output.

[†]Interestingly, we could also combine this frequency rotation with the matched filter $p(-nT_S)$ to come up with a modified matched filter which would have been a true matched filter here.

When the length of the training sequence is N_p in this timing and data-aided mode, this correlation needs to be computed for N_p symbols. The frequency estimate can thus be written as

$$\begin{aligned}\hat{F}_0 &\Rightarrow \max \sum_{m=0}^{N_p-1} (a_I[m]z_I(mT_M) + a_Q[m]z_Q(mT_M)) \\ &\Rightarrow \max \left\{ \sum_{m=0}^{N_p-1} a^*[m]z(mT_M) \right\}_I\end{aligned}$$

where the effect of the accurate frequency correction appears in the matched filter output $z(mT_M)$. The effect of one candidate frequency shift and data conjugation on the incoming signal along with the matched filter is drawn in Figure 6.6.

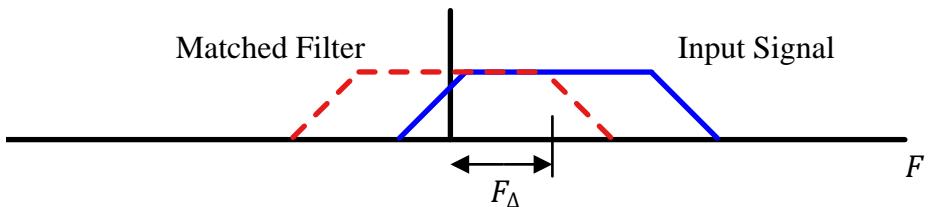


Figure 6.6: The frequency shifted and then data conjugated incoming signal along with the matched filter

In addition to the CFO F_Δ , if there is an additional phase rotation θ_Δ present in the Rx signal, then clearly from Figure 6.6, it is the magnitude of the correlation that maximizes the frequency estimate. The feedforward brute force estimator is subsequently given by

$$\hat{F}_0 \Rightarrow \max \left| \sum_{m=0}^{N_p-1} a^*[m]z(mT_M) \right| \quad (6.11)$$

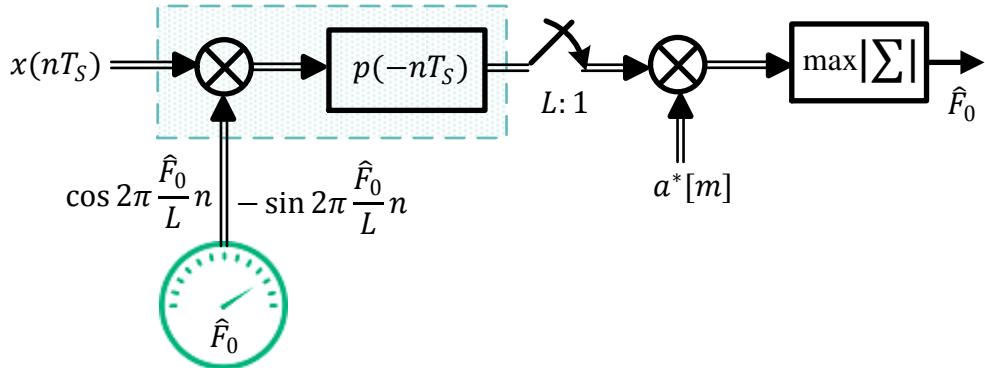
This also makes sense by looking at Figure 6.6: the larger the overlap, the lesser the CFO which ideally should be zero.

Due to the nature of the above expression, there is no closed-form expression of such an estimator. With fast digital signal processors available, the implementation strategy is to test a set of N test values,

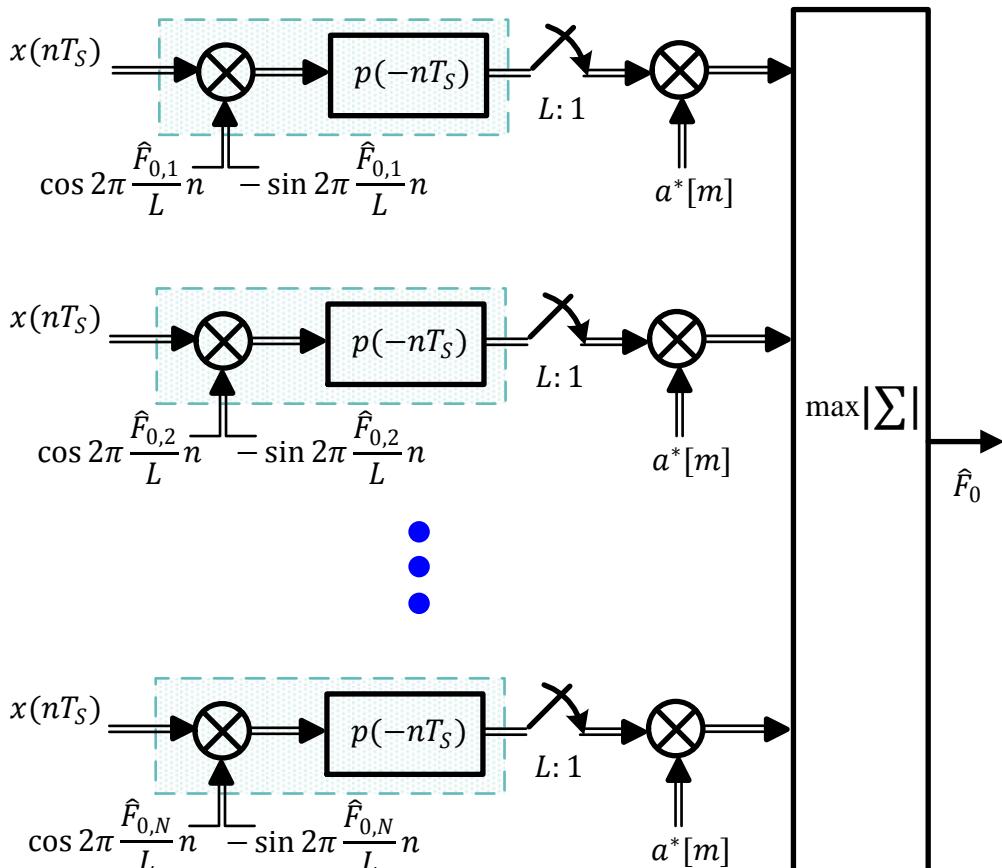
$$\hat{F}_{0,1}, \hat{F}_{0,2}, \dots, \hat{F}_{0,N},$$

evaluate Eq (6.11) for each of them and choose the one that maximizes the correlation sum. A block diagram of this brute force estimator is drawn in Figure 6.7 in the form of both a serial and a parallel search where the double arrows show complex operations.

Being a data-aided estimator maximizing the correlation, the feedforward brute force estimator exhibits a better performance than other estimators in a small CFO scenario. However, there are two issues that arise in its implementation.



(a) Serial (complex) implementation of CFO estimator



(b) Parallel (complex) implementation of CFO estimator

Figure 6.7: A serial and a parallel approach to feedforward brute force estimator

- From the set of N candidate parameters for \hat{F}_0 , consider from the definition that the CFO repaired and matched filter output $z(mT_M)$ needs to be computed for each candidate, i.e., $\hat{F}_{0,1}, \hat{F}_{0,2}, \dots$ after which the subsequent Rx operations are performed. This results in a huge computational complexity at the Rx that comes with a price, either in the form of
 - a high clock rate in a serial implementation shown in Figure 6.7a where more operations need to be performed in the same amount of time, or
 - more area in a parallel implementation shown in Figure 6.7b where multiple candidates are tested and the algorithm is executed for each of them in parallel.
- The relation in Eq (6.11) exhibits not a single global maximum, but many additional local maxima as well. Occasionally below a certain SNR, the correlation output $\text{corr}[0]$ gets so much distorted by noise that its maximum occurs significantly away from the actual CFO resulting in a large error. Such kinds of estimation errors naturally generate a burst of detection errors in the data stream.

This necessitates the need to look for more practical estimators. The difficulties encountered in implementing the brute force estimator can be solved for a small frequency offset ($F_0 \ll 1$) and data knowledge in the following manner.

6.3.3 Feedforward: The DFT Estimator

In Section 5.8, we said that correlation strips all the known parameters in its expression to reveal the only unknown parameter and isolate it. Hence, this process forms a relation of $r(nT_S)$ with the known parameters from which the unknown parameter can be estimated. Here again, the correlation result has indicated that $a^*[m]z(mT_M)$ is the route forward and we next see how this opens up an individual sinusoid with frequency F_Δ .

In the previous discussions, we have found that in addition to downconversion by the carrier frequency F_C , the input signal is rotated by F_0/L before entering as an input to the matched filter. The purpose here is to see the form of $z(nT_S)$ before applying the insight from correlation above. Using the rotation rule of complex numbers $I \cos \theta - Q \sin \theta$, the I matched filter output can be written as

$$I \rightarrow z_I(nT_S) = \left\{ \sum_i a_I[i] p(nT_S - iT_M) \cdot \cos 2\pi \frac{F_0}{L} n - \sum_i a_Q[i] p(nT_S - iT_M) \cdot \sin 2\pi \frac{F_0}{L} n \right\} * p(-nT_S)$$

Since the phase is not changing much during a symbol duration T_M , the complex sinusoid with frequency F_0/L can be approximated as a constant over the pulse shape $p(nT_S)$. Then, the sine and cosine can be taken out of the convolution expression and the I matched filter output can be written as

$$I \rightarrow z_I(nT_S) \approx \cos 2\pi \frac{F_0}{L} n \left\{ \sum_i a_I[i] p(nT_S - iT_M) * p(-nT_S) \right\} - \sin 2\pi \frac{F_0}{L} n \left\{ \sum_i a_Q[i] p(nT_S - iT_M) * p(-nT_S) \right\}$$

This is shown at the top of Figure 6.8 where the rotation by complex sinusoid F_0/L is now moved from the input of the matched filter $p(-nT_S)$ to its output. Applying the usual concept of pulse auto-correlation $r_p(nT_S)$,

$$I \rightarrow z_I(nT_S) \approx \cos\left(2\pi \frac{F_0}{L} n\right) \sum_i a_I[i] r_p(nT_S - iT_M) - \sin\left(2\pi \frac{F_0}{L} n\right) \sum_i a_Q[i] r_p(nT_S - iT_M)$$

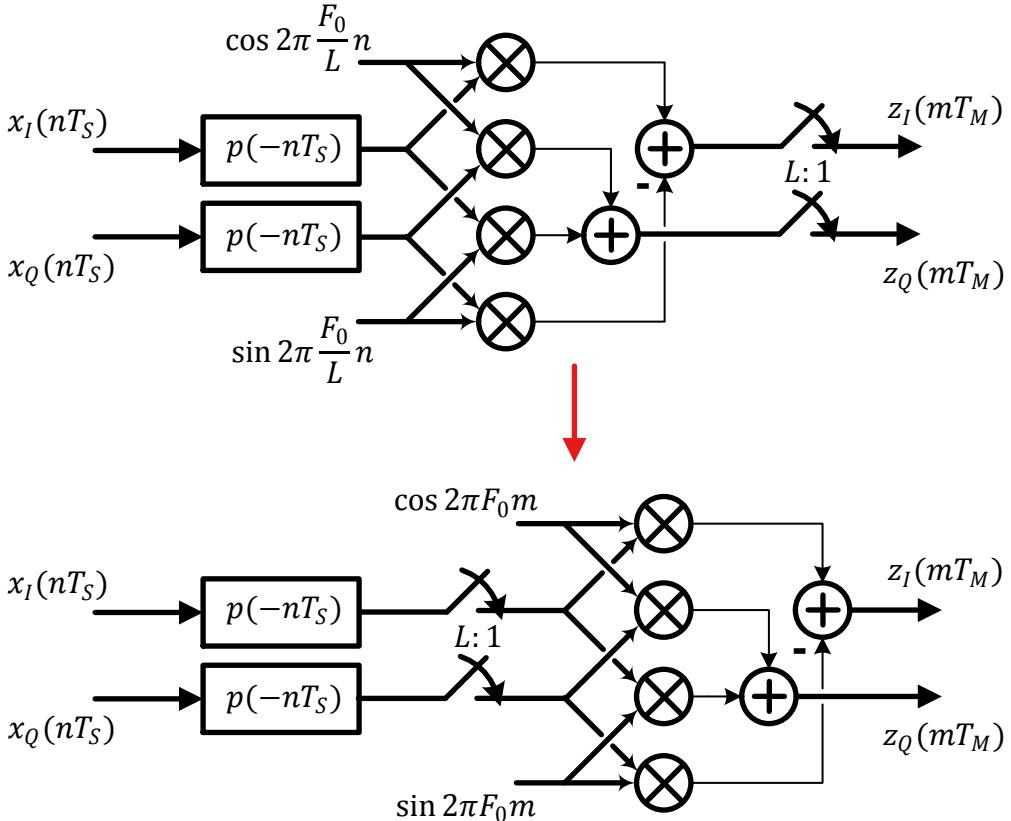


Figure 6.8: Small CFO implies that (top) it can be passed through the matched filter and (bottom) through the downampler

Now downampling by L at $n = mL$,

$$\left. \frac{F_0}{L} n \right|_{n=mL} = F_0 m$$

applying the Nyquist no-ISI criterion from Eq (6.50) and performing similar steps for the Q arm too, we have

$$I \rightarrow z_I(mT_M) = a_I[m] \cos 2\pi F_0 m - a_Q[m] \sin 2\pi F_0 m \\ Q \uparrow z_Q(mT_M) = a_Q[m] \cos 2\pi F_0 m + a_I[m] \sin 2\pi F_0 m \quad (6.12)$$

Here, $z(mT_M)$ is the matched filter output that explicitly brings the CFO rotation out instead of an implicit term within $z(mT_M)$ as was seen in Eq (6.11) and Figure 6.7a. This is drawn at the bottom of Figure 6.8 in two steps, first crossing the complex sinusoid past the matched filter and then through the downampler.

Since the data symbols $a[m]$ are known, their effect can be removed through conjugation. Furthermore, the training sequences in general come from a PSK constellation which implies that $a^*[m]a[m] = 1$. From Eq (6.12), if we form the signal

$$y(mT_M) = a^*[m]z(mT_M)$$

then applying the usual trigonometric identities, it gets stripped down to a bare sinusoid as

$$\begin{array}{ll} I \rightarrow & y_I[m] = y_I(mT_M) = \cos 2\pi F_0 m \\ Q \uparrow & y_Q[m] = y_Q(mT_M) = \sin 2\pi F_0 m \end{array} \quad (6.13)$$

What we have done so far is to *reduce the problem of finding the CFO F_0 in a modulated signal to estimating the frequency F_0 of a simple complex sinusoid embedded in noise*.

Digging Up the Complex Sinusoid

To visually comprehend the underlying signal level modifications, the Tx signal $s(t)$ in Figure 6.9 consists of fast variations due to the carrier and is downconverted at the Rx by a local oscillator having a frequency offset F_Δ with respect to the Tx. The downconverted, matched filtered and downsampled I and Q outputs $z_I(mT_M)$ and $z_Q(mT_M)$ are drawn next, where these are corrupted by additive noise at each stage after arriving at the Rx. Since the only distortion present is a frequency offset, removing the modulation reveals I and Q parts of a complex sinusoid whose frequency is the sought after parameter. This process of digging up the embedded sinusoid in the communications signal in Figure 6.9 is similar to that encountered in phase estimation process of Figure 5.14.

To further understand where this complex sinusoid came from, remember that a modulation removal process at symbol rate ideally results in all ones sequence, a DC signal in discrete time domain. The corresponding transform in frequency domain is an impulse at zero frequency. Therefore, after the mildly shifted incoming spectrum is presented the matched filter, the result is approximately an impulse at the location of CFO. This impulse in frequency domain corresponds to a complex sinusoid in time domain.

Locating the Complex Sinusoid in Noise

One of the most efficient methods to locate the above complex sinusoid in noise and consequently estimate its frequency is the Discrete Fourier Transform (DFT).

$$\begin{array}{ll} I \rightarrow & Y_I[k] = \sum_{n=0}^{N-1} \left[y_I[n] \cos 2\pi \frac{k}{N} n + y_Q[n] \sin 2\pi \frac{k}{N} n \right] \\ Q \uparrow & Y_Q[k] = \sum_{n=0}^{N-1} \left[y_Q[n] \cos 2\pi \frac{k}{N} n - y_I[n] \sin 2\pi \frac{k}{N} n \right] \end{array}$$

for each $k = -N/2, -N/2 + 1, \dots, -1, 0, 1, \dots, N/2 - 1$. Also recall from Section 1.10 that an N -point DFT of a complex sinusoid at each bin k is a sinc function

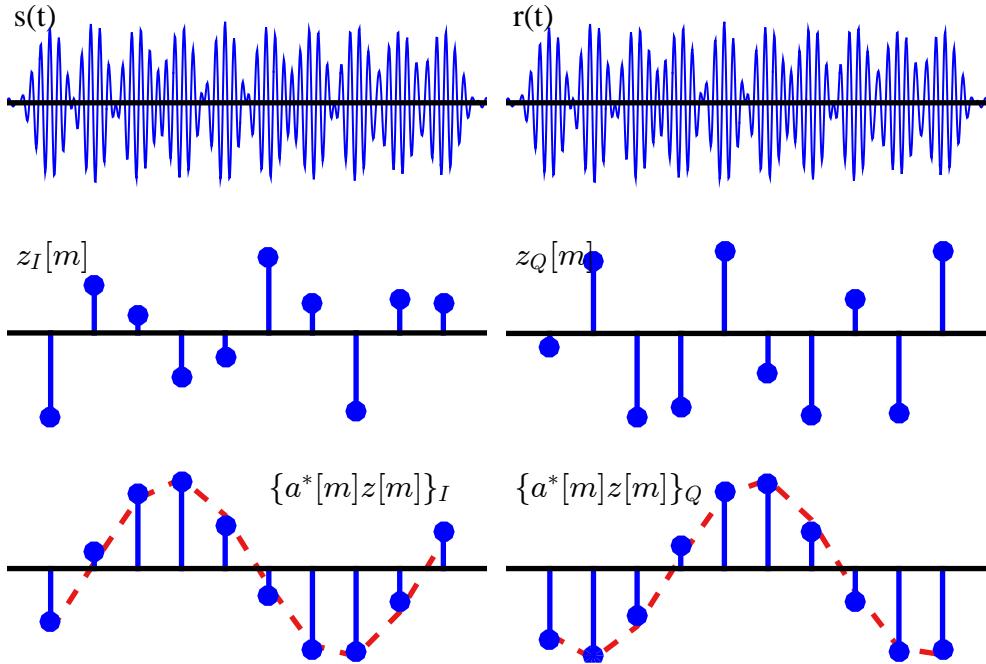


Figure 6.9: A signal level view of digging up the embedded sinusoid in communications signal through the correlation process

that has the maximum amplitude at k with zeros at all other bins $k' \neq k$. Therefore, the DFT acts as a bandpass filter at each frequency k and exhibits a peak corresponding to the frequency of the sinusoid embedded in noise. For k_0 corresponding to F_0 in a DFT size of N ,

$$\hat{k}_0 \Rightarrow \max |Y[k]| \quad (6.14)$$

For a small frequency offset, this relation greatly simplifies the CFO estimate. It can be implemented by taking a DFT of conjugated matched filter outputs $y(mT_M)$ and searching for the maximum peak. The frequency index k_0 corresponding to the maximum peak gives the desired estimate.

Notice that we have used the notation $y[n]$ instead of $y[m]$ in the DFT relation. This is because depending on the application, the DFT size N can or cannot be equal to the preamble length N_p . For a finer resolution in frequency domain, a sequence of zeros can be padded to $y[m]$ to form the signal $y[n]$ that is then taken into frequency domain. The advantage of a large N is a sharper peak, an example of which is drawn in Figure 6.10.

Note 6.2 Where did the simplification come from?

The difference between the brute force estimator of Eq (6.11) and the DFT estimator is that in the former, the matched filter output $z(mT_M)$ contains the CFO candidate \hat{F}_0 in an implicit manner. Therefore, $z(mT_M)$ needs to be computed for each CFO candidate

$$\hat{F}_{0,1}, \hat{F}_{0,2}, \dots, \hat{F}_{0,N},$$

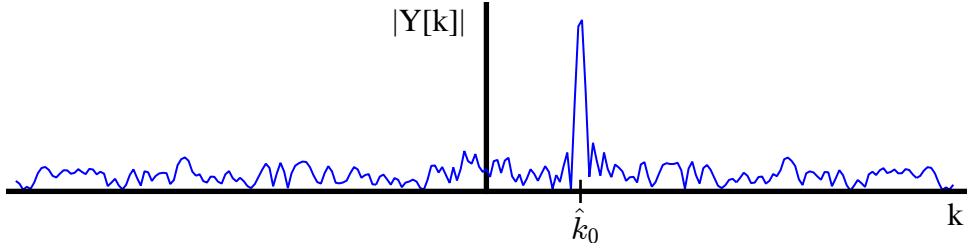


Figure 6.10: A plot of $|Y[k]|$ exhibits a maximum at the underlying CFO F_0 , represented by \hat{k}_0 in a DFT of size N

which is often prohibitively complex. In the latter, on the other hand, the CFO is small and it can be taken past the matched filter and the downampler. Hence, the matched filter output $z(mT_M)$ is just computed once and then its DFT can be taken to find the maximum. Moreover, this DFT can be implemented in the form of a Fast Fourier Transform (FFT) to reduce the complexity to a substantial extent.

6.3.4 Feedforward: Multiple Correlations Estimators

We have already discussed in Section 6.3.3 that the DFT is one of the most efficient methods to estimate the frequency of a complex sinusoid embedded in noise. Here, we explore other techniques based on auto-correlation of $y(mT_M) = a^*[m]z(mT_M)$ that provide a tradeoff in terms of accuracy and computational complexity as compared to a DFT estimator.

In Eq (6.13), it was seen that when the CFO F_Δ is small, the conjugate multiplication on matched filter outputs $y(mT_M) = a^*[m]z(mT_M)$ generates a complex sinusoid embedded in noise. Ignoring noise and any phase offset, we can write

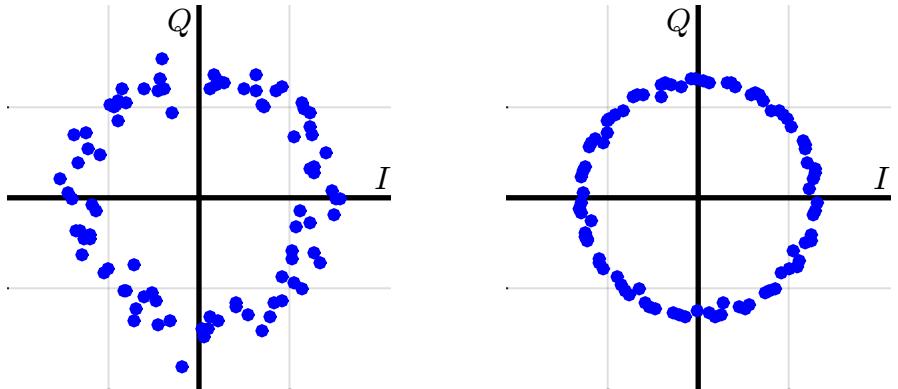
$$\begin{array}{ll} I \rightarrow & y_I(mT_M) = \cos 2\pi F_0 m \\ Q \uparrow & y_Q(mT_M) = \sin 2\pi F_0 m \end{array} \quad (6.15)$$

In time domain, this complex sinusoid was drawn in the last row of Figure 6.9. The problem with such a signal is that at low SNR, it is easily overwhelmed by noise. This can be readily observed through its scatter plot[†] shown in Figure 6.11a where the IQ samples are rotating in time domain in a circle with frequency F_0 .

We also saw earlier that the auto-correlation at lag 0, i.e., $\text{corr}[0]$, between the Rx signal $r(nT_S)$ and the expected signal template can be used to derive an optimal estimator. Here, we compute the auto-correlations of $y(mT_M)$ for all $j \neq 0$, i.e., $\text{corr}[j]$ for $1 \leq j \leq N_P - 1$ where N_P is the preamble length. First, notice that $y[(m-j)T_M]$ is given by

$$\begin{array}{ll} I \rightarrow & y_I[(m-j)T_M] = \cos 2\pi F_0(m-j) \\ Q \uparrow & y_Q[(m-j)T_M] = \sin 2\pi F_0(m-j) \end{array}$$

[†]Scatter plot of a complex signal is a plot of its Q samples drawn against its I samples.



(a) Scatter plot of $y(mT_M)$ – a complex sinusoid embedded in noise, see Eq (6.15)

(b) Scatter plot of auto-correlations of $y(mT_M)$, see Eq (6.17)

Figure 6.11: Auto-correlation of $y(mT_M)$ sufficiently suppress the noise to yield a better CFO estimator

Next, we employ the complex multiplication rule with a conjugate, $I \cdot I + Q \cdot Q$ and $Q \cdot I - I \cdot Q$ to find

$$\begin{aligned} I \rightarrow & \quad \{y(mT_M)y^*((m-j)T_M)\}_I = \cos 2\pi F_0 m \cdot \cos 2\pi F_0(m-j) + \\ & \quad \sin 2\pi F_0 m \cdot \sin 2\pi F_0(m-j) \\ Q \uparrow & \quad \{y(mT_M)y^*((m-j)T_M)\}_Q = \sin 2\pi F_0 m \cdot \cos 2\pi F_0(m-j) - \\ & \quad \cos 2\pi F_0 m \cdot \sin 2\pi F_0(m-j) \end{aligned}$$

which due to the identities $\cos A \cos B + \sin A \sin B = \cos(A - B)$ and $\sin A \cos B - \cos A \sin B = \sin(A - B)$ becomes

$$\begin{aligned} I \rightarrow & \quad \{y(mT_M)y^*((m-j)T_M)\}_I = \cos 2\pi F_0 j \\ Q \uparrow & \quad \{y(mT_M)y^*((m-j)T_M)\}_Q = \sin 2\pi F_0 j \end{aligned} \tag{6.16}$$

At first, it seems that there is no difference between Eq (6.15) and Eq (6.16), as the difference of indices m and j does not matter. However, this is not true.

Complex sinusoid with index m : The complex sinusoid with index m is one signal with N_P complex samples, each of which is embedded in noise. This was shown in Figure 6.11a.

Complex sinusoid with index j : On the other hand, the complex sinusoid with index j can be found through $N_P - j$ conjugate multiplication operations. For example, there are $N_P - 1$ complex samples at a lag of 1 by performing the operation $y(mT_M)y^*((m-1)T_M)$,

- first between $y^*(1T_M)$ and $y(0T_M)$,
- second between $y^*(2T_M)$ and $y(1T_M)$,
- ...
- $(N_P - 1)^{th}$ between $y^*[(N_P - 1)T_M]$ and $y[(N_P - 2)T_M]$.

Each of these contributes towards *forming 1 complex sample* of the index j sinusoid. In this example, the complex sample, $\cos 2\pi F_0 \cdot 1$ and $\sin 2\pi F_0 \cdot 1$ can then be obtained through averaging all the $N_p - 1$ samples above.

$$\text{corr}[1] = \frac{1}{N_p - 1} \sum_{m=1}^{N_p-1} y(mT_M)y^*((m-1)T_M)$$

Extending the same concept by taking the auto-correlation at lag j and averaging Eq (6.16) by the number of samples available,

$$\text{corr}[j] = \frac{1}{N_p - j} \sum_{m=j}^{N_p-1} y(mT_M)y^*((m-j)T_M), \quad 1 \leq j \leq N_p - 1 \quad (6.17)$$

Plugging Eq (6.16) above, we get

$I \rightarrow$	$\text{corr}_I[j] = \cos 2\pi F_0 j + \text{noise}, \quad 1 \leq j \leq N_p - 1$		(6.18)
$Q \uparrow$	$\text{corr}_Q[j] = \sin 2\pi F_0 j + \text{noise}, \quad 1 \leq j \leq N_p - 1$		

where the noise terms here is zero mean and has significantly less power due to the averaging operation. This complex sinusoid is drawn in Figure 6.11b. It can be observed that it is much smoother than the complex sinusoid in Figure 6.11a because the averaging operation as a filter reasonably suppressed the noise.

From here, there are two algorithms based on the above expression that are named behind their inventors, Fitz [19] and LR (Luise and Reggiannini) [20].

Fitz Method

Fitz started with the observation that the phase of this $\text{corr}[j]$ above should yield an estimate of the CFO.

$$\angle \text{corr}[j] = 2\pi F_0 j$$

The question here is the following: are we free to choose all values of j from 1 to $N_p - 1$ or is there an upper limit for the number of correlations we can employ? The answer implicitly comes from the sampling theorem as follows.

Denote the highest value of j that can be used by N_F where $N_F \leq N_p - 1$. Then, the largest phase of $\angle \text{corr}[j]$ is for $j = N_F$ and the maximum expected F_0 , say $F_{0,\max}$, and given by $2\pi|F_{0,\max}|N_F$. Clearly on an IQ-plane, this phase (anticlockwise direction) should be less than $2\pi/2 = \pi$ to avoid aliasing (just like $F_S/2$ can be the highest frequency according to the sampling theorem), or it will appear closer to 0 from the clockwise direction. In other words, the range of the resultant phase should be within $\{-\pi, +\pi\}$.

$$2\pi|F_{0,\max}|N_F < \pi, \quad N_F < \frac{1}{2|F_{0,\max}|}$$

Therefore, the largest expected CFO $F_{0,\max}$ determines the upper limit for the number of correlations employed. With this number N_F in hand, the estimation is straightfor-

ward. Averaging both sides of $\angle \text{corr}[j]$,

$$\begin{aligned}\frac{1}{N_F} \sum_{j=1}^{N_F} \angle \text{corr}[j] &= \frac{1}{N_F} \sum_{j=1}^{N_F} 2\pi F_0 j \\ &= \frac{1}{N_F} \cdot 2\pi F_0 \frac{N_F(N_F + 1)}{2} = \pi F_0(N_F + 1)\end{aligned}$$

The final CFO estimate can be written as

$$\hat{F}_0 = \frac{1}{\pi N_F(N_F + 1)} \sum_{j=1}^{N_F} \angle \text{corr}[j] \quad (6.19)$$

Notice the tradeoff involved above. A large N_F increases the estimation accuracy but reduces the estimation range $F_{0,\max}$.

LR Method

The LR method starts again with Eq (6.18) but instead of taking the angle first and average later, it takes the average first and angle later. For an $N_{LR} \neq N_F$ (as we find out soon),

$$\begin{array}{ll} I \rightarrow & \sum_{j=1}^{N_{LR}} \text{corr}_I[j] = \sum_{j=1}^{N_{LR}} \cos 2\pi F_0 j \\ Q \uparrow & \sum_{j=1}^{N_{LR}} \text{corr}_Q[j] = \sum_{j=1}^{N_{LR}} \sin 2\pi F_0 j \end{array}$$

where the term $1/N_{LR} \sum_{j=1}^{N_{LR}}$ noise is ignored.

The task now is to find the upper limit for N_{LR} . Compare the I and Q parts with Eq (1.63) and Eq (1.65) in Chapter 1. They are exactly the same, except a minus sign in the Q part with the following substitutions.

$$n_s \rightarrow 1, \quad L \rightarrow N_{LR}, \quad \frac{k}{N} \rightarrow F_0, \quad n \rightarrow j$$

Therefore, their sum is given by Eq (1.67) with no minus sign in the phase part.

$$\begin{aligned}\left| \sum_{j=1}^{N_{LR}} \text{corr}[j] \right| &= \frac{\sin \pi N_{LR} F_0}{\sin \pi F_0} \\ \angle \left\{ \sum_{j=1}^{N_{LR}} \text{corr}[j] \right\} &= \pi F_0 (N_{LR} + 1)\end{aligned}$$

From the phase above, the estimate is written as

$$\hat{F}_0 = \frac{1}{\pi(N_{LR} + 1)} \angle \left\{ \sum_{j=1}^{N_{LR}} \text{corr}[j] \right\} \quad (6.20)$$

Clearly, the ratio in the magnitude stays positive until the first zero crossing of the numerator that yields

$$\pi N_{LR} |F_{0,max}| < \pi, \quad N_{LR} < \frac{1}{|F_{0,max}|}$$

Notice from N_F of the Fitz method above, the LR method manifests twice the estimation range.

The above timing-aided schemes generate the most accurate CFO estimates compared to all other schemes due to the following factors.

- Availability of extra information in terms of data from the training sequence.
- Thinking in frequency domain, availability of timing also implies the downsampling of matched filter output at the rate of 1 sample/symbol, thus processing the Rx signal at the minimum possible bandwidth and consequently allowing the least amount of noise in the available samples.

Now we move towards feedback solutions for carrier frequency synchronization.

6.3.5 A Frequency Locked Loop (FLL)

In Section 4.7, we have already discussed that for a very small frequency offset, a Phase Locked Loop (PLL) is sufficient for driving the phase error between the Rx signal and the local oscillator output to zero. However, a PLL fails to operate for a large CFO and hence an acquisition aid must be invoked prior to handing the signal over to the PLL. Some of the common acquisition aids adopted for this purpose are the following.

Frequency sweeping: In this technique, the frequency of the local oscillator is swept over an uncertainty interval during acquisition until a phase lock is achieved. Then, the frequency sweeping is stopped to avoid an unnecessary increase in the phase variance.

Multi-stage acquisition: It consists of using a large filter bandwidth during the acquisition process and then switching to a narrow loop filter after the lock. Or a large CFO is significantly reduced by using a non-timing-aided technique in the feedforward mode. A timing-aided scheme is applied later to compensate for any residual CFO.

Frequency Locked Loop (FLL): In an FLL, a Frequency Error Detector (FED) generates an error signal corresponding to the difference in frequency between a reference and an output waveform to drive the loop similar to a PLL. An FLL can be used in parallel with a PLL such that the former operates during acquisition and loss of lock while the latter during steady state tracking.

A block diagram of an FLL is drawn in Figure 6.12. Just like a PLL, an FLL consists of the following components.

Frequency Error Detector (FED) As mentioned above, a frequency error detector determines the frequency difference between a reference input waveform and a locally generated waveform. According to the amount of that difference, it generates an error signal denoted as $e_D[n]$.

Loop Filter A loop filter sets the dynamic performance limits of an FLL as well. Moreover, it helps filter out noise and irrelevant frequency components generated in the frequency error detector. Its output signal is denoted as $e_F[n]$.

Numerically Controlled Oscillator (NCO) As discussed earlier in Section 4.5, an NCO generates a local discrete-time and discrete-valued waveform with a phase as close to the phase of the reference signal as possible. For a continuously changing phase, i.e., a frequency difference, an NCO consistently changes its phase, thus tracking the frequency of the reference.

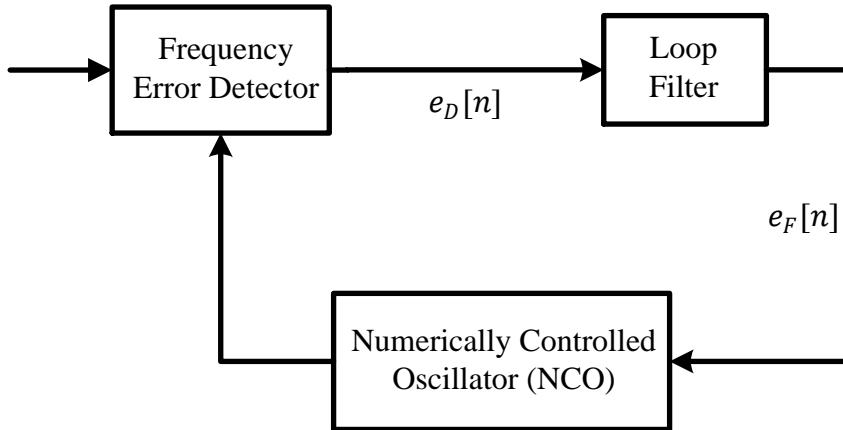


Figure 6.12: Basic structure of a Frequency Locked Loop (FLL)

In such a feedback setting with established components of an FLL in place (loop filter and NCO), our focus is now on devising a suitable frequency error detector that can generate an error signal $e_D[n]$ proportional to the frequency difference to enable the tracking of the reference frequency. During the discussion of timing-aided feedback systems here, the frequency error detector operates at the symbol rate, so the error signal $e_D[m]$ it generates is in terms of symbol time m .

6.3.6 Feedback: Phase Based Frequency Error Detector

In this section, we discuss a frequency locked loop with the following set of conditions.

- The CFO is small enough that timing can be acquired before its compensation. However, this CFO is still large enough that a PLL alone cannot do the job.
- With correct timing, Nyquist no-ISI condition is also satisfied and each symbol spaced sample is (almost) independent from the contribution of the neighbouring symbols.
- No training sequence is available and the Rx makes decisions on the optimal symbol spaced samples. Given the role of CFO, these decisions need not be correct.

Consider the block diagram in Figure 6.13 that describes the implementation of the overall FLL in terms of complex signals. Later, the same FLL will be drawn for real signals in Figure 6.17. The matched filter output is downsampled by L to produce a signal at the symbol rate. During each symbol time m , this signal is multiplied with the complex conjugate decisions $\hat{a}^*[m]$ from the symbol detector and fed to the Frequency Error Detector (FED). The FED produces an error signal $e_D[m]$ that is upsampled by L to match the sample rate of the loop filter and the NCO. Subsequently, it is smoothed out by the loop filter and drives an NCO that accordingly adjusts the phase of its output waveform at the sample rate T_S .

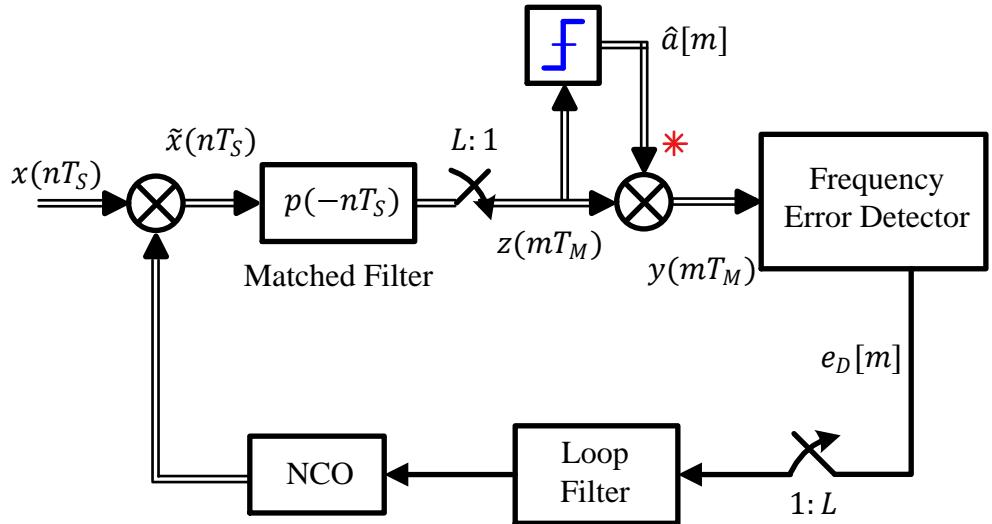


Figure 6.13: A block diagram for the implementation of a phase based FED in terms of complex signals

Now we develop an understanding of how this structure eliminates the CFO. Ignoring noise and referring back to Section 6.1, the Rx signal in the presence of a Carrier Frequency Offset (CFO) $F_\Delta/F_S = F_0/L$ is given by

$$r(nT_S) = v_I(nT_S)\sqrt{2} \cos\left(2\pi \frac{k_C}{N} n + 2\pi \frac{F_0}{L} n\right) - \\ v_Q(nT_S)\sqrt{2} \sin\left(2\pi \frac{k_C}{N} n + 2\pi \frac{F_0}{L} n\right)$$

where k_C corresponds to carrier frequency F_C and $v_I(nT_S)$ and $v_Q(nT_S)$ are the inphase and quadrature waveforms

$$\begin{aligned} I &\rightarrow & v_I(nT_S) &= \sum_m a_I[m] p(nT_S - mT_M) \\ Q &\uparrow & v_Q(nT_S) &= \sum_m a_Q[m] p(nT_S - mT_M) \end{aligned}$$

We also saw in Section 6.1 that after downconversion with the frequency F_C , the output signal $x(nT_S)$ in Eq (6.47) and Eq (6.48) is nothing but the product of shaped symbol stream $v(nT_S)$ with a complex sinusoid of frequency F_0/L (using the multiplication rule of complex numbers

$I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$.

$$\begin{aligned} I &\rightarrow x_I(nT_S) = v_I(nT_S) \cos 2\pi \frac{F_0}{L} n - v_Q(nT_S) \sin 2\pi \frac{F_0}{L} n \\ Q &\uparrow x_Q(nT_S) = v_Q(nT_S) \cos 2\pi \frac{F_0}{L} n + v_I(nT_S) \sin 2\pi \frac{F_0}{L} n \end{aligned} \quad (6.21)$$

Coming to the FLL here, on multiplication by another complex sinusoid of frequency $-\hat{F}_0/L$ generated by the NCO in Figure 6.13, the matched filter input is

$$\begin{aligned} I &\rightarrow \tilde{x}_I(nT_S) = x_I(nT_S) \cos 2\pi \frac{\hat{F}_0}{L} n + x_Q(nT_S) \sin 2\pi \frac{\hat{F}_0}{L} n \\ Q &\uparrow \tilde{x}_Q(nT_S) = x_Q(nT_S) \cos 2\pi \frac{\hat{F}_0}{L} n - x_I(nT_S) \sin 2\pi \frac{\hat{F}_0}{L} n \end{aligned}$$

Now we plug the expression for $x(nT_S)$ from Eq (6.21) and use the identities $\cos A \cdot \cos B \mp \sin A \cdot \sin B = \cos(A \pm B)$ and $\sin A \cos B \pm \cos A \sin B = \sin(A \pm B)$. Then, the input signal to the matched filter is given in terms of $F_{0:e} = F_0 - \hat{F}_0$ by

$$\begin{aligned} I &\rightarrow \tilde{x}_I(nT_S) = v_I(nT_S) \cos 2\pi \frac{F_{0:e}}{L} n - v_Q(nT_S) \sin 2\pi \frac{F_{0:e}}{L} n \\ Q &\uparrow \tilde{x}_Q(nT_S) = v_Q(nT_S) \cos 2\pi \frac{F_{0:e}}{L} n + v_I(nT_S) \sin 2\pi \frac{F_{0:e}}{L} n \end{aligned}$$

which after opening up $v(nT_S)$ becomes

$$\begin{aligned} I &\rightarrow \tilde{x}_I(nT_S) = \sum_m a_I[m] p(nT_S - mT_M) \cos 2\pi \frac{F_{0:e}}{L} n - \\ &\quad \sum_m a_Q[m] p(nT_S - mT_M) \sin 2\pi \frac{F_{0:e}}{L} n \\ Q &\uparrow \tilde{x}_Q(nT_S) = \sum_m a_Q[m] p(nT_S - mT_M) \cos 2\pi \frac{F_{0:e}}{L} n + \\ &\quad \sum_m a_I[m] p(nT_S - mT_M) \sin 2\pi \frac{F_{0:e}}{L} n \end{aligned}$$

After matched filtering, T_M -spaced samples of the matched filter output are required at $n = mL = mT_M/T_S$ that is performed by $L : 1$ downampler in Figure 6.13. The discrete frequency after downsampling appears to be $F_{0:e}$ now since $(F_{0:e}/L)n|_{n=mL} = F_{0:e}m$. By virtue of the available timing, Nyquist no-ISI criterion is also applicable.

$$\begin{aligned} I &\rightarrow z_I(mT_M) = a_I[m] \cos 2\pi F_{0:e} m - a_Q[m] \sin 2\pi F_{0:e} m \\ Q &\uparrow z_Q(mT_M) = a_Q[m] \cos 2\pi F_{0:e} m + a_I[m] \sin 2\pi F_{0:e} m \end{aligned}$$

Afterwards, these T_M -spaced samples are input to the symbol detector that generates symbol decisions $\hat{a}[m]$. Assuming a QPSK constellation with a unit amplitude, the data symbols $a[m]$ are just phase shifts of $\pm\pi/2$ and $\pm3\pi/2$ which are rotating anticlockwise with a frequency $F_{0:e}$. Thus, we can write

$$\angle z(mT_M) = \angle a[m] + 2\pi F_{0:e} m$$

The detector makes the decisions based on the minimum distance rule. At time m , it selects the constellation symbol $\hat{a}[m]$ closest to the matched filter output $z(mT_M)$. This is where the symbol decision $\hat{a}[m]$ is drawn for $z(mT_M)$ where $\angle z(mT_M) - \angle \hat{a}[m]$

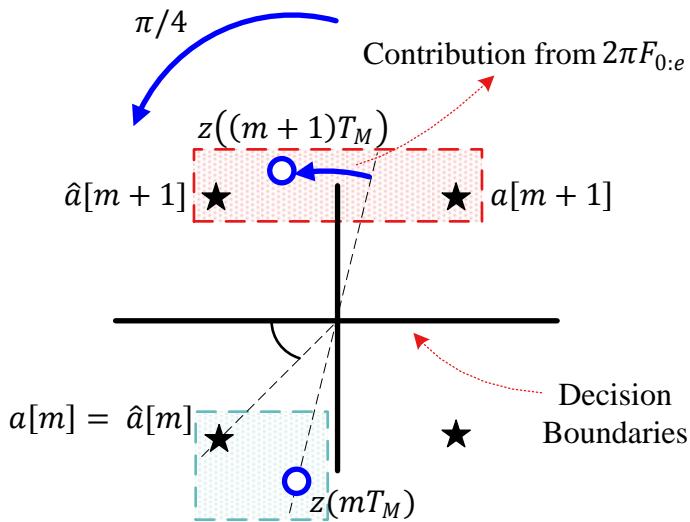


Figure 6.14: Minimum distance rule based detector decisions in a QPSK constellation. The role of $2\pi F_{0:e}$ in rolling over the matched filter output across the symbol boundary is also shown

is a little less than $+\pi/4$, as illustrated by the shaded green box at the bottom of Figure 6.14.

$$\angle z(mT_M) = \angle a[m] + 2\pi F_{0:e}m, \quad \text{and} \quad \hat{a}[m] = a[m]$$

Now at time $m+1$,

$$\begin{aligned} \angle z((m+1)T_M) &= \angle a[m+1] + 2\pi F_{0:e}(m+1), \\ &= \angle a[m+1] + 2\pi F_{0:e}m + 2\pi F_{0:e}, \quad \text{and} \quad \hat{a}[m+1] \neq a[m+1] \end{aligned}$$

It is evident that the factor $2\pi F_{0:e}$ has rolled over the matched filter output $z((m+1)T_M)$ across the symbol boundary, as illustrated by the shaded red box at the top of Figure 6.14. However,

$$|\angle z((m+1)T_M) - \angle \hat{a}[m+1]| < \pi/4$$

This characteristic can be utilized to devise a frequency error detector as follows.

- Guided by the correlation principle, multiply the matched filter output $z(mT_M)$ with conjugate of the detector decision $\hat{a}^*[m]$ to cancel the phase contribution of the modulation.

$$y(mT_M) = \hat{a}^*[m] \cdot z(mT_M)$$

which implies

$$\angle y(mT_M) = -\angle \hat{a}[m] + \angle z(mT_M)$$

Since the detector makes decisions based on the nearest constellation point, *the above term always lies within the range $(-\pi/4, +\pi/4)$* . To prove this point,

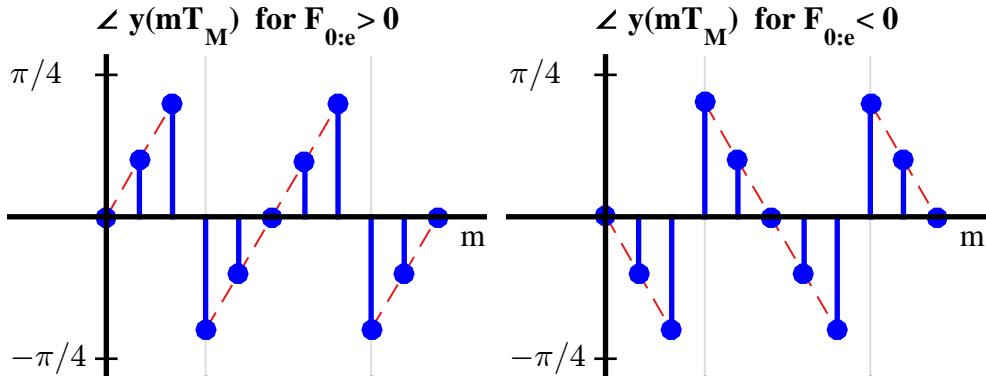


Figure 6.15: $\angle y(mT_M)$ as a function of symbol time m for $F_{0:e} = \pm 0.05$

imagine $\angle y(mT_M)$ being greater than $+\pi/4$. This implies that the difference between $\angle z(mT_M)$ and $\angle \hat{a}[m]$ is greater than $\pi/4$ which in a QPSK constellation of Figure 6.14 means that the detector has not selected the nearest constellation point, which is not true.

- Next, we draw $\angle y(mT_M)$ in Figure 6.15 for $F_{0:e} = +0.05$ on the left and for $F_{0:e} = -0.05$ on the right. As expected, $\angle y(mT_M)$ is an increasing ramp for $F_{0:e} > 0$ and a decreasing ramp for $F_{0:e} < 0$. We need a frequency error detector that at least leads the NCO to the right direction of frequency error movement.

Here, the interesting point to note is that the form of the frequency error detector is still not finalized, since an average of the above mentioned signal will tend towards zero for both a positive and a negative $F_{0:e}$! A little amount of processing is required to configure a proper FED such that on average, its output has the same sign as $F_{0:e}$.

- To produce an error signal with the same sign as $F_{0:e}$, all we need to do is to ensure that in $\angle y(mT_M)$ plots of Figure 6.15, there should be more positive samples than the negative samples for a positive $F_{0:e}$ and vice versa. For this purpose, a phase based FED can be devised as

$$e_D[m] = \begin{cases} \angle y(mT_M) & -\lambda < \angle y(mT_M) < \lambda \\ e_D[m-1] & \text{otherwise} \end{cases} \quad (6.22)$$

where λ is a threshold value, the absolute value of which is less than $\pi/4$.

To understand the operation of phase based frequency error detector, consider Figure 6.16 where $e_D[m]$ is plotted along with $\angle y(mT_M)$ (in dashed green line) for $\lambda = \pi/6$. Observe that the average of $e_D[m]$ is positive for a positive $F_{0:e}$ and negative for a negative $F_{0:e}$. The reason can be grasped by looking at $e_D[m]$ for a positive $F_{0:e}$. While $e_D[m] < \angle y(mT_M)$ when $\angle y(mT_M)$ goes past the threshold λ , the loss incurred is much smaller as compared to the gain when $e_D[m] > 0$ at the next symbol where $\angle y(mT_M)$ rolls over towards $-\pi/4$. The cumulative gain is hence positive. A corresponding argument holds for a negative $F_{0:e}$.

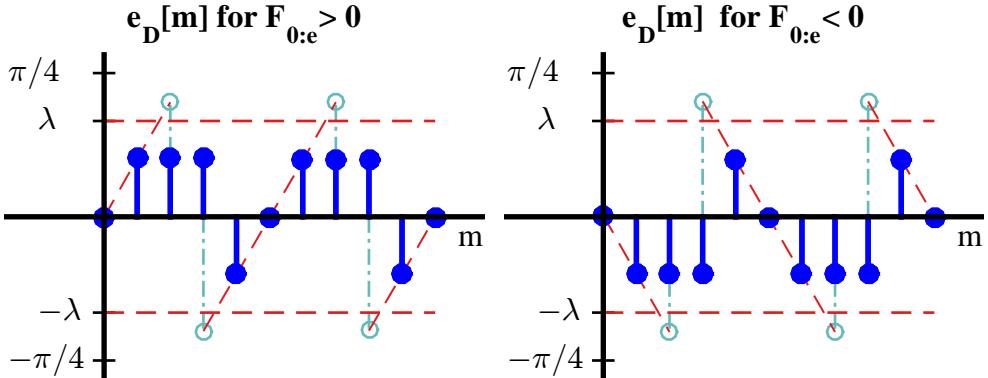


Figure 6.16: Operation of the phase based frequency error detector for $\lambda = \pi/6$. Clearly, the average of $e_D[m]$ is positive for a positive $F_{0:e}$ and negative for a negative $F_{0:e}$. Original phase $\angle y(mT_M)$ is also shown in dashed green line

With this understanding in place, an FLL employing the phase based FED is illustrated in Figure 6.17 in terms of real signals which is the same FLL drawn in Figure 6.13 for complex signals. As far as the block diagram is concerned, complex signals are easier to understand. However, any practical implementation employs operations involving real signals. Here, the notation T_M as before represents a delay of one sample. Since the FED operates at symbol rate $1/T_M$, it implies a delay of one symbol.

Finally, from the error signal in Eq (6.22),

$$-\lambda < \angle y(mT_M) < \lambda \quad \rightarrow \quad |\angle y(mT_M)| < \lambda$$

Now if $|\angle y(mT_M)|$ is lesser than λ , it is selected as the current error sample, otherwise the previous error sample $e_D[m - 1]$ is selected. So for this purpose, a mux is employed which is a shorthand for a multiplexer commonly used in electronic circuits. Here, a 2×1 mux has two input lines at its top, one select line at the left and one output line at the bottom. As a hardware counterpart of an if-else loop in software, the mux switches one of the two input lines through to the common output line by the application of a control signal.

Above, the operation of the phase based FED is explained for a QPSK modulation. The same idea can easily be extended to an M -PSK constellation by observing that $\angle(mT_M)$ lies between $-\pi/M$ and $+\pi/M$. The error signal $e_D[m]$ thus can be generated by choosing a threshold λ less than π/M .

Phase Based FED for QAM Modulation

For PSK constellations, the decision regions are very simple which is not the case for QAM constellations. Therefore, a little ingenuity is required to extend the idea of phase based FED to QAM modulation schemes. For a 16-QAM constellation, for example, one possible solution is as follows.

Observe from Figure 6.18 that a 16-QAM constellation consists of 2 QPSK constellations, one in the interior and one at the exterior, plus an 8-PSK constellation in the center. The strategy chosen is as follows.

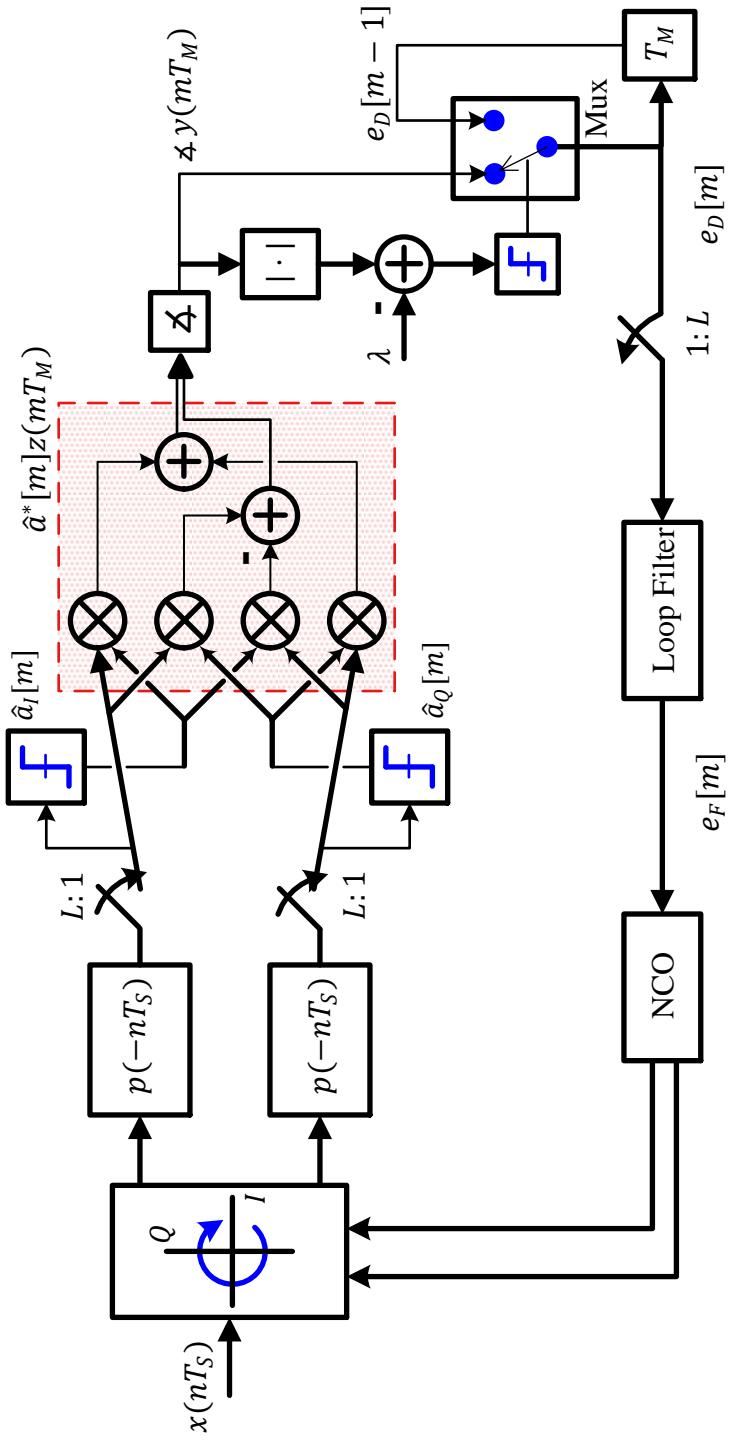


Figure 6.17: A block diagram for the implementation of a phase based FED in terms of real signals

When the detector makes a decision in favour of any one of the two QPSK constellations, the FED output $e_D[m]$ is computed as before in Eq (6.22). However, when the decision falls in the region defined by 8-PSK constellation, it can be left unchanged. Thus, the expression for a phase based FED of a 16-QAM is written as

$$e_D[m] = \begin{cases} \angle y(mT_M) & -\lambda < \angle y(mT_M) < \lambda \text{ and} \\ & y(mT_M) \text{ decision is in QPSK-1/QPSK-2} \\ e_D[m-1] & \text{otherwise} \end{cases}$$

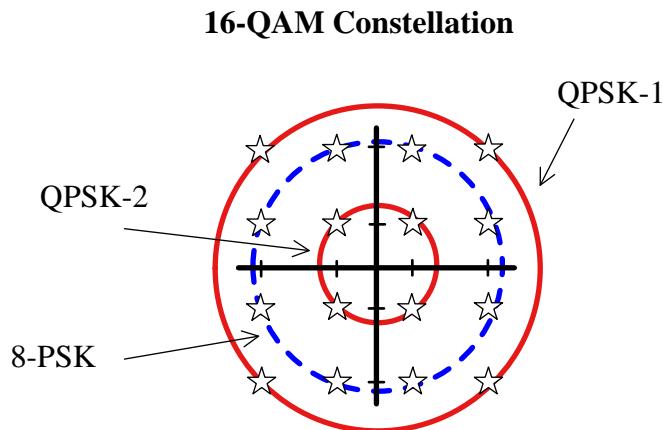


Figure 6.18: A 16-QAM constellation can be considered as 2 QPSK constellations, one in the interior and one at the exterior, plus an 8-PSK constellation in the center

To see the effect of such a FED on the carrier frequency synchronization for a 16-QAM modulation, the matched filter output $z(mT_M)$ and the FLL output are drawn in Figure 6.19 for a Square-Root Raised Cosine pulse with excess bandwidth $\alpha = 0.5$. The parameter λ is chosen to be $\pi/6$ while a damping factor $\zeta = 1/\sqrt{2}$ and a loop noise bandwidth of $B_n T_M = 3\%$ drive the FLL response at an $E_b/N_0 = 15$ dB. Notice the three PSK constellations that form the 16-QAM are clearly visible which perhaps led towards the discovery of this carrier recovery scheme. Finally, after the convergence in Figure 6.19b, the constellation is still affected by a small phase offset and a PLL is needed to drive this error towards zero.

A similar strategy with appropriate modification can be adopted for other higher-order QAM modulation schemes.

6.4 Non-Timing-Aided Techniques

Now we lift the small CFO assumption and allow the CFO to vary in an even wider range. As discussed earlier, a natural consequence of this setup is that no timing and hence no data can be acquired due to unknown symbol boundaries.

no timing \Rightarrow no data

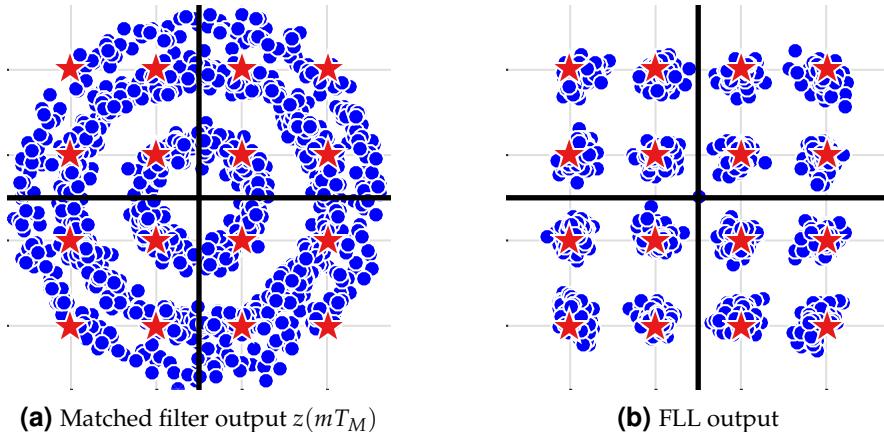


Figure 6.19: Results for a phase based FED applied to a 16-QAM constellation for carrier frequency recovery at $E_b/N_0 = 15$ dB. The threshold λ is chosen to be $\pi/6$ while the damping factor is $\zeta = 1/\sqrt{2}$ with a loop noise bandwidth of $B_n T_M = 3\%$

We discuss both feedforward and feedback techniques that bring the CFO down to an extent that timing can be established. This is usually the first stage of frequency acquisition. Subsequently, any of the timing-aided CFO correction methods discussed earlier can be applied in a feedforward or feedback manner, or a PLL is sufficient to compensate for the remaining CFO if it was reduced within a small range. In both options described above, a CFO correction is essentially a two-step process:

1. a non-timing-aided coarse CFO correction, and
2. a timing-aided fine CFO compensation which have been described before.

We start with a very simple feedforward non-timing-aided method known as delay and multiply technique.

6.4.1 Feedforward: Delay and Multiply Technique

We discussed earlier that a shorter acquisition time is required in burst mode communications. This is even more critical in the case of CFO acquisition because a modulated communications signal essentially means nothing before a coarse CFO correction. It is desirable then to apply a feedforward technique with a short acquisition time and a wide acquisition range. Delay and multiply is one such technique that has proved very effective in fast frequency estimation.

The main idea is that the frequency of any complex sinusoid can be estimated by taking a phase difference between any of its two samples at known time instants, say T_2 and T_1 . This is because by definition, the phase difference is given by

$$\Delta\phi = 2\pi F(T_2 - T_1)$$

Since T_2 and T_1 are known and $\Delta\phi$ is estimated through some method, the underlying frequency is

$$F = \frac{\Delta\phi}{2\pi(T_2 - T_1)}$$

Our task now is to somehow convert the modulated Rx signal into a signal resembling a complex sinusoid (embedded in noise) so that the phase $\Delta\phi$ can be estimated.

Consider a Rx signal downconverted by F_C that in accordance with the notation adopted in this text is $x(nT_S)$ (see Figure 6.2) and contains a carrier frequency offset of F_Δ which after sampling at a rate $F_S = 1/T_S$ gets scaled as

$$2\pi F_\Delta t \Big|_{t=nT_S} \rightarrow 2\pi F_\Delta nT_S$$

The discussion here is in terms of the actual CFO F_Δ and not normalized CFO F_0 due to the absence of timing. As discussed before, $x(nT_S)$ is a product of shaped symbol stream $v(nT_S)$ with a complex sinusoid of frequency F_Δ , see Eq (6.47) and Eq (6.48). Using the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$,

$$\begin{aligned} I &\rightarrow x_I(nT_S) = v_I(nT_S) \cos 2\pi F_\Delta nT_S - v_Q(nT_S) \sin 2\pi F_\Delta nT_S \\ Q &\uparrow \quad x_Q(nT_S) = v_Q(nT_S) \cos 2\pi F_\Delta nT_S + v_I(nT_S) \sin 2\pi F_\Delta nT_S \end{aligned} \quad (6.23)$$

Assume that it is passed through a simple lowpass filter with sufficiently wide bandwidth to accommodate a worst case CFO $F_{\Delta,\max}$ so that it suppresses the noise and interference but lets the signal contents pass through undistorted.

Next, we form a conjugate difference between the two samples of $x(nT_S)$ spaced a specific time interval apart, say iT_S where i is an integer.

$$y(nT_S) = x(nT_S)x^*(nT_S - iT_S)$$

To find the relationship of $y(nT_S)$ with F_Δ , a block diagram for the implementation of delay and multiply technique is drawn in Figure 6.20.

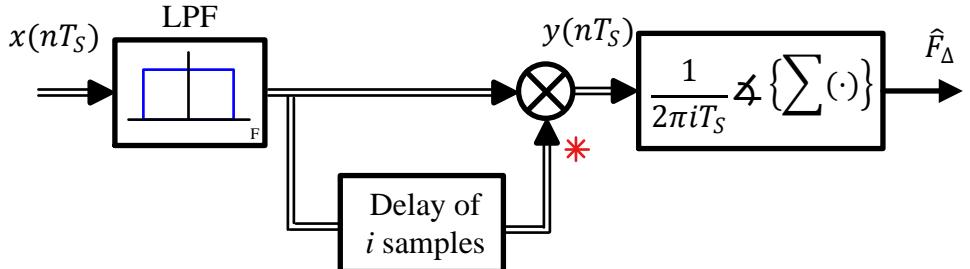


Figure 6.20: A block diagram for the implementation of delay and multiply method operating at a rate F_S

Breaking $y(nT_S)$ into inphase and quadrature components through the conjugate multiplication rule $I \cdot I + Q \cdot Q$ and $Q \cdot I - I \cdot Q$ yields

$$\begin{aligned} I &\rightarrow y_I(nT_S) = x_I(nT_S)x_I(nT_S - iT_S) + x_Q(nT_S)x_Q(nT_S - iT_S) \\ Q &\uparrow \quad y_Q(nT_S) = x_Q(nT_S)x_I(nT_S - iT_S) - x_I(nT_S)x_Q(nT_S - iT_S) \end{aligned}$$

Using the two identities $\cos A \cdot \cos B \pm \sin A \cdot \sin B = \cos(A \mp B)$ and $\sin A \cos B \pm \cos A \sin B = \sin(A \pm B)$ and plugging the value of $x(nT_S)$ from Eq (6.23), $y(nT_S)$ can

be written as[†]

$$\begin{array}{ll} I \rightarrow & y_I(nT_S) = W_I(nT_S) \cos 2\pi F_\Delta iT_S - W_Q(nT_S) \sin 2\pi F_\Delta iT_S \\ Q \uparrow & y_Q(nT_S) = W_Q(nT_S) \cos 2\pi F_\Delta iT_S + W_I(nT_S) \sin 2\pi F_\Delta iT_S \end{array} \quad (6.24)$$

It is obvious from rotation rule of complex numbers that $y(nT_S)$ is just an anticlockwise rotation of the complex signal $W(nT_S)$ defined as

$$W(nT_S) = v(nT_S)v^*(nT_S - iT_S)$$

Therefore, its I and Q components $W_I(nT_S)$ and $W_Q(nT_S)$ are given as

$$\begin{array}{ll} I \rightarrow & W_I(nT_S) = v_I(nT_S)v_I(nT_S - iT_S) + v_Q(nT_S)v_Q(nT_S - iT_S) \\ Q \uparrow & W_Q(nT_S) = v_Q(nT_S)v_I(nT_S - iT_S) - v_I(nT_S)v_Q(nT_S - iT_S) \end{array}$$

Comparing $x(nT_S)$ from Eq (6.23) with $y(nT_S)$ in Eq (6.24), it seems that both are quite similar. So why did we take the conjugate product? This is because $v(nT_S)$ in the expression for $x(nT_S)$ contains the phase shifts arising from the data as well. On the other hand, a short delay of i samples in a reasonably oversampled signal implies that the modulation is wiped out in $y(nT_S)$. For a unit energy signal,

$$\begin{array}{ll} I \rightarrow & W_I(nT_S) \approx v_I^2(nT_S) + v_Q^2(nT_S) = 1 \\ Q \uparrow & W_Q(nT_S) \approx 0 \end{array}$$

Plugging them back in Eq (6.24) produces the following expression.

$$\begin{array}{ll} I \rightarrow & y_I(nT_S) = \cos 2\pi F_\Delta iT_S \\ Q \uparrow & y_Q(nT_S) = \sin 2\pi F_\Delta iT_S \end{array}$$

which is (as in the timing-aided cases) a complex sinusoid of frequency F_Δ . Thus, the CFO information can be extracted from the phase of $y(nT_S)$.

For a sampled sequence of length N_d , we need to sum these samples to average out the noise. Therefore,

$$\begin{aligned} 2\pi \hat{F}_\Delta iT_S &= \angle \left\{ \sum_{n=0}^{N_d-1} y(nT_S) \right\} \\ &= \angle \left\{ \sum_{n=0}^{N_d-1} x(nT_S)x^*(nT_S - iT_S) \right\} \end{aligned}$$

From here, the CFO estimate \hat{F}_Δ is given by

$$\hat{F}_\Delta = \frac{1}{2\pi iT_S} \angle \left\{ \sum_{n=0}^{N_d-1} x(nT_S)x^*(nT_S - iT_S) \right\} \quad (6.25)$$

[†]This can be verified as follows. Since Eq (6.23) is $x(nT_S) = v(nT_S) \exp(j2\pi F_\Delta nT_S)$,

$$\begin{aligned} y(nT_S) &= x(nT_S)x^*(nT_S - iT_S) = v(nT_S) \exp(j2\pi F_\Delta nT_S) v^*(nT_S - iT_S) \exp(j2\pi F_\Delta (-nT_S + iT_S)) \\ &= v(nT_S)v^*(nT_S - iT_S) \exp(j2\pi F_\Delta iT_S) = W(nT_S) \exp(j2\pi F_\Delta iT_S) \end{aligned}$$

The most commonly used implementation of delay and multiply method is to choose $i = 1$ which modifies the above estimator as

$$\hat{F}_\Delta = \frac{1}{2\pi T_S} \angle \left\{ \sum_{n=0}^{N_d-1} x(nT_S) x^* [(n-1)T_S] \right\}$$

The signal bandwidth after downconversion is equal to $(1 + \alpha)/2T_M$ where α is the excess bandwidth or roll-off factor in a Nyquist pulse. For an example of $F_{\Delta,\max}$ equal to $1/T_M$ and $\alpha = 0.5$, the total flat response required for the lowpass filter is

$$\begin{aligned} B_{\text{Lowpass}} &= F_{\Delta,\max} + \frac{1 + \alpha}{2T_M} = \frac{1}{T_M} + \frac{1 + 0.5}{2T_M} \\ &= \frac{1.75}{T_M} \approx \frac{2}{T_M} \end{aligned}$$

The sample rate thus chosen is $F_S = 4/T_M$ here. This yields another form of the estimator as a function of T_M .

$$\hat{F}_\Delta = \frac{4}{2\pi T_M} \angle \left\{ \sum_{n=0}^{N_d-1} x \left(n \frac{T_M}{4} \right) x^* \left[(n-1) \frac{T_M}{4} \right] \right\}$$

An inherent limitation of delay and multiply technique as well as all non-data-aided CFO estimators is the performance degradation due to the wide bandwidth of the Rx filter. This is because in the absence of a reasonably accurate CFO compensation, the Rx bandwidth needs to be kept as large as allowed by $F_{\Delta,\max}$ to accommodate the Rx signal that can be anywhere within that range. Subsequently, the signal suffers not only from a large amount of noise but also from adjacent channel interference. In addition, matched filtering is also not always possible to enhance the Signal-to-Noise Ratio (SNR) except in some cases.

Nevertheless, the performance of the delay and multiply method is very close to the non-data-aided estimate derived through the maximum correlation principle such as the one discussed in the next section. Therefore, delay and multiply method with one sample difference finds its applications in burst mode receivers that need a coarse CFO estimate first up in the shortest amount of time.

6.4.2 Feedback: Derivative Frequency Error Detector

In the absence of timing and data, once a valid signal starts arriving, the timing offset can lie anywhere in the range $(0, T_M)$. The effect of an incorrect timing offset will be explained in detail in Chapter 7. Suffice it to say that the matched filter output is usually nowhere close to the actual symbol value, as opposed to how we defined it in a timing-aided scenario. Therefore, downsampling the CFO repaired and matched filter output at symbol rate $1/T_M$ is not feasible and we concentrate on this signal sampled at a rate $F_S = 1/T_S$. This sample rate is related to the symbol rate as $F_S = L/T_M$ where L is the number of samples/symbol.

I	\rightarrow	Inphase matched filter output	\rightarrow	$z_I(nT_S)$
Q	\uparrow	Quadrature matched filter output	\rightarrow	$z_Q(nT_S)$

The CFO F_Δ is related to normalized CFO F_0 targeted in timing-aided scenario as

$$\frac{F_0}{L} = \frac{F_\Delta T_M}{L} = \frac{F_\Delta}{F_S}$$

The objective here is to determine a CFO estimate \hat{F}_Δ from the samples $x(nT_S)$ and $z(nT_S)$, i.e., the matched filter input and output, respectively. This feedback process is also known as *Automatic Frequency Control (AFC)*.

Subsequently, we discuss a derivative frequency error detector generating an error signal for tracking the CFO within a Frequency Locked Loop (FLL). For a true understanding of this topic, I highly recommend reading Section 3.6 on a pulse shaping filter first.

Effect of CFO on Signal Energy

We remember from Eq (6.10) that the correlation output is given by

$$\text{corr}[0] = \left\{ \sum_{m=0}^{N_0-1} a^*[m] z(mT_M) \right\}_I$$

To find the best CFO estimate in a non-data-aided setting when both the data and timing are unknown, one clue from the above equation is to remove the modulation by multiplication with a value similar in sign. So we replace $a[m]$ with the matched filter output $z(nT_S)$, thus leading to the correlation output as $\sum_n |z(nT_S)|^2$ which is the input signal energy.

To see why it works, the starting point is the Parseval's relation introduced in Eq (2.27) that relates the energy of a signal $s[n]$ in time domain with that of its DFT $S[k]$.

$$E_s = \sum_{n=0}^{N-1} |s[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |S[k]|^2$$

Recall that the energy in CFO repaired and matched filtered output can be found in either time or frequency domain using the relation above.

$$E_z = \sum_{n=0}^{N_d-1} |z(nT_S)|^2 = \frac{1}{N_d} \sum_{k=0}^{N_d-1} |Z[k]|^2$$

where N_d is the total number of symbols. It is important to differentiate N_d in a non-timing-aided and non-data-aided scenario here from N_P which is the training length in a timing-aided case. Also, a summation in discrete-time samples is the same as area under the curve in a continuous-time signal. This fact is exploited next for computing the energy E_z in frequency domain.

Assuming that the channel has a flat frequency response (we learn about wireless channels in Chapter 8), we claim that this energy is maximum for a CFO estimate $\hat{F}_\Delta = F_\Delta$. Although this can be proved mathematically[†], we make an intuitive argument.

The Tx spectrum is shaped by a square-root Nyquist pulse $P[k]$ in frequency domain, a conjugated version of which is used at the Rx side for matched filtering. However, the Rx signal is additionally shifted by an unknown CFO F_Δ by nature. This is

[†]Beginning with Taylor series expansion of the log likelihood function and later applying the Schwartz inequality.

depicted without any modulation through a blue solid line in the left column of Figure 6.21.

Next, it is presented to another square-root Nyquist filter centred at frequency 0, which has a frequency response $P^*[k]$ shown as red dashed line in Figure 6.21. The pulse shape is real and even, thus $P[k]$ as well as $P^*[k]$ are also real and even. In frequency domain, the output of a filter is its frequency response multiplied with that of the incoming signal. Therefore, only the overlapped region survives the filtering operation.

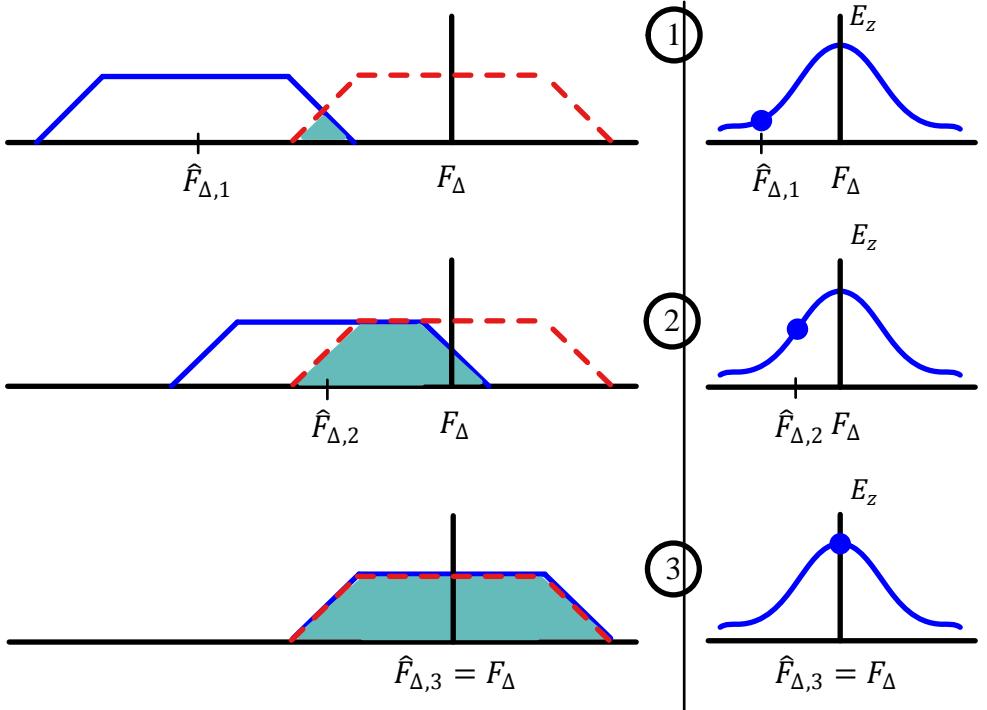


Figure 6.21: As the CFO estimate \hat{F}_Δ converges towards the actual CFO F_Δ , the area under the overlapped region increases and so does the energy in the CFO repaired and matched filtered signal E_z , reaching its maximum at $\hat{F}_\Delta = F_\Delta$

Now in the right column, Figure 6.21 shows the area under the magnitude squared of overlapped region which corresponds to the energy of the signal $z(nT_S)$ at the matched filter output. With this understanding, we take into account the following three cases.

1. For a large difference $\hat{F}_\Delta - F_\Delta$ as in the first case, the area under the overlapped region of the shaped signal and the matched filter is very small and consequently the energy E_z of this output signal is low.

$$P^*[k] \cdot P[k - k_0] < |P[k]|^2$$

Clearly, not compensating for the CFO generates an output signal with less than optimal energy.

2. As the CFO estimate \hat{F}_Δ slides towards the second case of a reduced CFO difference, the overlapped area increases and so does the energy in $z(nT_S)$.
3. When \hat{F}_Δ approaches F_Δ in the third case, both the shaping filter and the matched filter completely overlap, the result is a perfect Nyquist spectrum and the signal energy attains its maximum value $P[k]^2$.

The argument here suffices for a general understanding of the nature of the problem. In summary, energy in the CFO repaired and matched filtered output $\sum_n |z(nT_S)|^2$ starts from zero for a very large difference between CFO and its estimate, increases towards a maximum value at $\hat{F}_\Delta = F_\Delta$ and then rolls down again in a symmetric fashion. Similarly, for the general case of a sequence of pulse shaped data symbols, we need to square the Rx signal to wipe out the modulation and compute the signal energy. Squaring is a very interesting operation that exploits the underlying structure in the signal for synchronization purpose. However, I think that a more appropriate place to discuss the effect of squaring is during timing synchronization and hence it is detailed in Chapter 7.

In light of the above discussion, *the target of a non-timing-aided CFO compensation technique is to maximize this energy*. As hinted earlier, it turns out that a non-data-aided estimator that maximizes this energy is the same as the one derived through the maximum correlation principle.

Maximizing the Signal Energy

We have learned so far that both feedforward and feedback techniques can be employed to compensate for the CFO. As will become clear with the nature of the problem, a feedback strategy is suited to carrier acquisition in this setup which implies that instead of finding a closed-form expression for \hat{F}_Δ , we form a frequency error detector proportional to F_Δ that drives an FLL maximizing $|z(nT_S)|^2$ in a recursive manner. Note the absence of summation here. It would have been there for a feedforward scheme but the strategy here is to compute an error signal for each feedback iteration of an FLL, just like we did before for a PLL.

After removing the summation, our goal is to maximize the energy

$$|z(nT_S)|^2 = \{z(nT_S) \cdot z^*(nT_S)\} \quad (6.26)$$

with respect to the CFO. The derivation is simple both in terms of mathematics and intuition. Refer to the definition of a derivative introduced in Section 2.6 for further discussion. Using the notation \cdot for a derivative from here onwards, we can proceed from Eq (6.26) as follows.

$$\begin{aligned} \frac{d}{d\hat{F}_\Delta} |z(nT_S)|^2 &= \frac{d}{d\hat{F}_\Delta} \{z(nT_S) \cdot z^*(nT_S)\} \\ &= z(nT_S) \left\{ \dot{z}(nT_S) \right\}^* + \dot{z}(nT_S) z^*(nT_S) \\ &= z(nT_S) \left\{ \dot{z}(nT_S) \right\}^* + \left(z(nT_S) \left\{ \dot{z}(nT_S) \right\}^* \right)^* \\ &= 2 \left\{ z(nT_S) \cdot \left\{ \dot{z}(nT_S) \right\}^* \right\}_I \end{aligned} \quad (6.27)$$

where the last equation follows from the fact that a complex signal added with its conjugate produces twice its inphase component, see Eq (1.23).

To find the maximum of the curve, we equate the derivative to zero. On the other hand, the purpose of an FLL is to drive an error signal to zero as well. Combining these two facts, we can utilize this derivative as an error signal (neglecting an irrelevant factor of 2).

$$e_D[n] = \left\{ z(nT_S) \cdot \left\{ \dot{z}(nT_S) \right\}^* \right\}_I \quad (6.28)$$

Such an error signal is known as a *Derivative Frequency Error Detector (FED)*, commonly known as a maximum likelihood FED. The name derivative FED arises due to utilizing the derivative of the matched filter output $\dot{z}(nT_S)$.

Whilst the error signal is obtained, finding the derivative of a time-varying Rx signal after CFO repaired matched filtering in real-time is not a desirable signal processing operation. We want to convert the error signal, which is the output of a Frequency Error Detector (FED), into a more practical formation.

The Two Filters

Going towards a simplified error term, we use a rather simple argument. For a complete derivation, refer to the text by Meyr et al in Ref. [21].

First, notice the set of operations in Figure 6.22 in which only the blocks in QAM Tx and Rx related to CFO and its compensation are drawn while assuming negligible impact from the frontend. Observe the opposite directions in which F_Δ and \hat{F}_Δ are rotating their respective input signals. Now there can be two ways to proceed from here.

- As proposed by Eq (6.28), pass the CFO compensated and matched filtered output $z(nT_S)$ through another filter that computes the derivative of its input in frequency domain. One option described in Section 2.6 is the differentiator

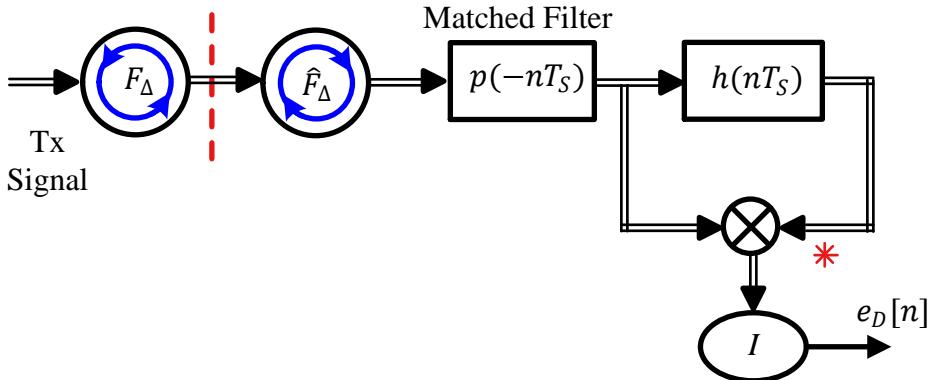
$$h[n] = \frac{1}{2} \{ +1, 0, -1 \}$$

Here, the matched filter and the frequency domain derivative filter appear in series as shown in Figure 6.22a.

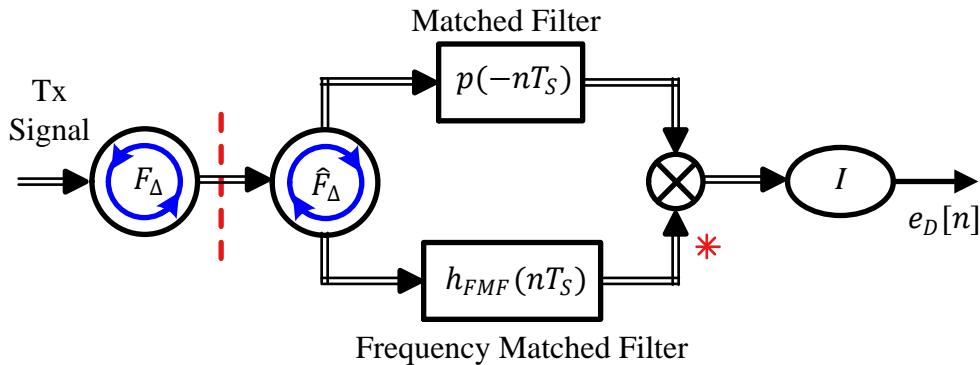
- As far as the inphase component of $z(nT_S) \cdot (\dot{z}(nT_S))^*$ is concerned, it makes sense to create a separate filter based on the pre-computed derivative of the matched filter with respect to frequency. We call it a *Frequency Matched Filter* $H_{FMF}(F)$ and denote its impulse response by $h_{FMF}(nT_S)$.

$$H_{FMF}(F) = \frac{d}{dF} H_{MF}(F)$$

$$h_{FMF}(nT_S) = \text{iDFT} \left\{ H_{FMF}(F) \right\}$$



(a) A serial approach: match filter the signal and compute the derivative



(b) A parallel approach: matched filter and frequency matched filter

Figure 6.22: A simplified FED produces the same inphase component as the former in Eq (6.28)

To see why this approach works, consider the matched filter output through the differentiator.

$$\begin{aligned}
 \dot{z}(nT_S) &= z(nT_S) * h(nT_S) \\
 &= \left\{ \tilde{x}(nT_S) * p(-nT_S) \right\} * h(nT_S) \\
 &= \tilde{x}(nT_S) * \underbrace{\left\{ p(-nT_S) * h(nT_S) \right\}}_{h_{FMF}(nT_S)}
 \end{aligned}$$

where $\tilde{x}(nT_S)$ is the frequency compensated matched filter input. Here, the two filters, namely the matched filter and the frequency matched filter, appear in parallel as shown in Figure 6.22b. The expression ‘The Two Filters’ is obviously taken from ‘The Two Towers’ in The Lord of the Rings by J. R. R. Tolkien.

It must be remembered that the two approaches in Figure 6.22a and Figure 6.22b are equal only with respect to inphase component of their product error signal that suffices for our purpose of extracting an FED. For a hint of

a quadrature component generated in the former approach, see the first term of Eq (6.51) in the Appendix on Meyr's derivation in Ref. [21]. While we have avoided rigorous derivations in this text, the purpose of this appendix is to show the process behind the simple expressions and that the quadrature component is different in the two approaches.

Let us denote the output of the frequency matched filter as $z_{FMF}(nT_S)$. In light of the above discussion, the matched filter output $z(nT_S)$ in Eq (6.28) can instead be multiplied with the output of the frequency matched filter $z_{FMF}(nT_S)$ and employed as an error signal $e_D[n]$ in the frequency locked loop.

$$e_D[n] = \left\{ z(nT_S) \cdot z_{FMF}^*(nT_S) \right\}_I \quad (6.29)$$

When this error signal is made a part of an FLL, it is commonly known as a special case of a *Quadrincorrelator Rx* which eventually drives the CFO estimate towards the correct solution. While the two filters employed here are the matched filter and the frequency matched filter, generalizing this concept implies that a quadrincorrelator can involve a product of any two filters that satisfy some specific design criteria.

Note 6.3 Serial vs parallel approach

Although both approaches above require two filters, the difference is that the serial approach incurs an additional delay because the input for computing the derivative is not available until the matched filter generates its output. On the other hand, the parallel approach uses two filters operating on the same input samples at the same time. An FLL or a PLL is an iterative solution in which any additional delay within the loop impacts its performance and hence the parallel approach proves more useful.

As an example, let us construct a continuous-time frequency matched filter for a square-root Nyquist pulse, say a Square-Root Raised Cosine (SRRC).

A Frequency Matched Filter

In Eq (3.32) of Section 3.6, we derived the frequency domain formula for the SRRC pulse (here acting as a matched filter) as

$$H_{MF}(F) = \begin{cases} \sqrt{T_M} & 0 \leq |F| \leq \frac{1-\alpha}{2T_M} \\ \sqrt{T_M} \cos \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(|F| - \frac{1-\alpha}{2T_M} \right) \right\} & \frac{1-\alpha}{2T_M} \leq |F| \leq \frac{1+\alpha}{2T_M} \\ 0 & |F| \geq \frac{1+\alpha}{2T_M} \end{cases} \quad (6.30)$$

The most important point to notice in the above equation is that the transition band of a Square-Root Raised Cosine pulse is a quarter cycle of a cosine with width α/T_M (this transition band is a half-cosine in a Raised Cosine filter). This can be seen as follows. From the equation, the inverse period of the cosine is $T_M/4\alpha$ and hence the period in frequency domain is $4\alpha/T_M$. However, it exists in the region

$$\frac{1+\alpha}{2T_M} - \frac{1-\alpha}{2T_M} = \frac{\alpha}{T_M} \quad (6.31)$$

which shows that the transition band is a quarter cycle of a cosine.

To derive our frequency matched filter, we take the derivative of Eq (6.30) for three distinct regions of the spectrum, remembering that the cosine below is in frequency domain, not time domain and the derivative is with respect to frequency in a frequency matched filter. Thus, the derivative of an SRRC pulse in frequency domain, i.e., a frequency matched filter $H_{FMF}(F)$, is

$$H_{FMF}(F) = \begin{cases} 0 & 0 \leq |F| \leq \frac{1-\alpha}{2T_M} \\ \sqrt{T_M} \left(-2\pi \cdot \frac{T_M}{4\alpha} \right) \sin \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(+F - \frac{1-\alpha}{2T_M} \right) \right\} \\ \quad + \frac{1-\alpha}{2T_M} \leq F \leq +\frac{1+\alpha}{2T_M} \\ \sqrt{T_M} \left(+2\pi \cdot \frac{T_M}{4\alpha} \right) \sin \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(-F - \frac{1-\alpha}{2T_M} \right) \right\} \\ \quad - \frac{1+\alpha}{2T_M} \leq F \leq -\frac{1-\alpha}{2T_M} \\ 0 & |F| \geq \frac{1+\alpha}{2T_M} \end{cases} \quad (6.32)$$

This frequency matched filter is drawn in Figure 6.23b for $\alpha = 0.25$ corresponding to an SRRC pulse in Figure 6.23a. Looking at the frequency response, it is easy to understand why the frequency matched filter is often called a band edge filter[†]. From the figure, it is also clear that *only the transition bandwidth contributes towards forming the frequency matched filter*. For excess bandwidth $\alpha = 0$, there would have been no frequency matched filter. In other words,

“The excess bandwidth not only controls the spectrum of the Tx signal but also provides the energy for synchronization purpose.”

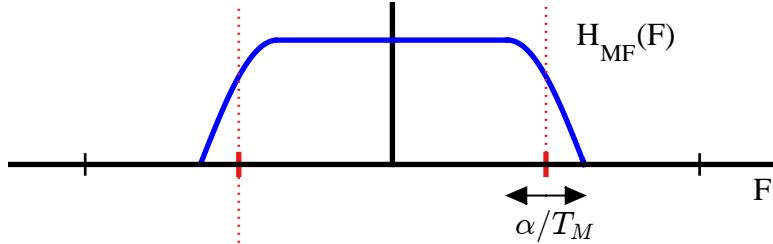
Observe that the frequency matched filter is indeed a quarter cycle of a negative sine wave in frequency domain. This can be more clearly visualized if you imagine α increasing towards 1 in Figure 6.23b that will join the two band edges at the origin. This negative sine arises due to the transition band of the original matched filter being a quarter cycle of a cosine with width α/T_M .

The Rx Structure

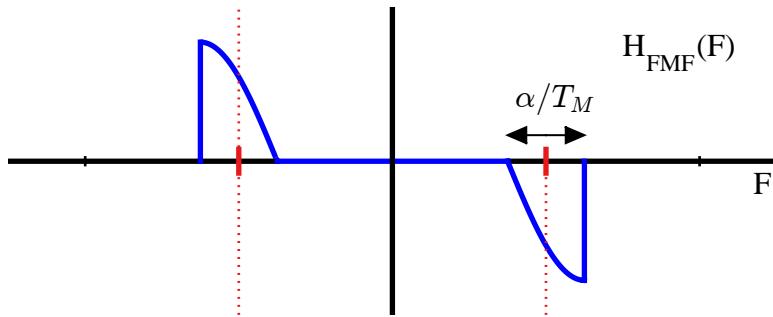
Now we look into the Rx structure that incorporates such a frequency error detector. For this purpose, we take a quick recap of the notations used. Recall from the Rx block diagram in Figure 6.2 that $x(nT_S)$ in I and Q branches is given from the Rx signal $r(nT_S)$ by

$$\begin{aligned} I &\rightarrow x_I(nT_S) = r(nT_S) \cdot \sqrt{2} \cos 2\pi F_C n T_S \\ Q &\uparrow x_Q(nT_S) = -r(nT_S) \cdot \sqrt{2} \sin 2\pi F_C n T_S \end{aligned}$$

[†]We will shortly see that other band edge filters are possible and in fact more effective, depending on the nature of the application at hand.



(a) A Square-Root Raised Cosine (SRRC) pulse as a matched filter. The transition band is important here



(b) Continuous spectrum of the frequency matched filter that consists of two band edges (the band edges look like one shark going to the right side and the other going to the left)

Figure 6.23: A matched filter in continuous frequency domain along with the corresponding frequency matched filter for excess bandwidth $\alpha = 0.25$

Ignoring noise and referring back to Section 6.1, this downconverted signal $x(nT_S)$ in the presence of a Carrier Frequency Offset (CFO) $F_\Delta/F_S = F_0/L$ is given by the product of shaped symbol stream $v(nT_S)$ with a complex sinusoid of frequency F_0/L (using the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$) as we saw in Eq (6.47) and Eq (6.48).

$$\begin{aligned} I &\rightarrow & x_I(nT_S) &= v_I(nT_S) \cos 2\pi \frac{F_0}{L} n - v_Q(nT_S) \sin 2\pi \frac{F_0}{L} n \\ Q &\uparrow & x_Q(nT_S) &= v_Q(nT_S) \cos 2\pi \frac{F_0}{L} n + v_I(nT_S) \sin 2\pi \frac{F_0}{L} n \end{aligned}$$

This is the signal input to the FLL here in Figure 6.24. On multiplication by another complex sinusoid of frequency $-\hat{F}_0/L$ generated by the NCO, the matched filter input is

$$\begin{aligned} I &\rightarrow & \tilde{x}_I(nT_S) &= x_I(nT_S) \cos 2\pi \frac{\hat{F}_0}{L} n + x_Q(nT_S) \sin 2\pi \frac{\hat{F}_0}{L} n \\ Q &\uparrow & \tilde{x}_Q(nT_S) &= x_Q(nT_S) \cos 2\pi \frac{\hat{F}_0}{L} n - x_I(nT_S) \sin 2\pi \frac{\hat{F}_0}{L} n \end{aligned}$$

Now we plug the expression for $x(nT_S)$ and use the identities $\cos A \cdot \cos B \mp \sin A \cdot$

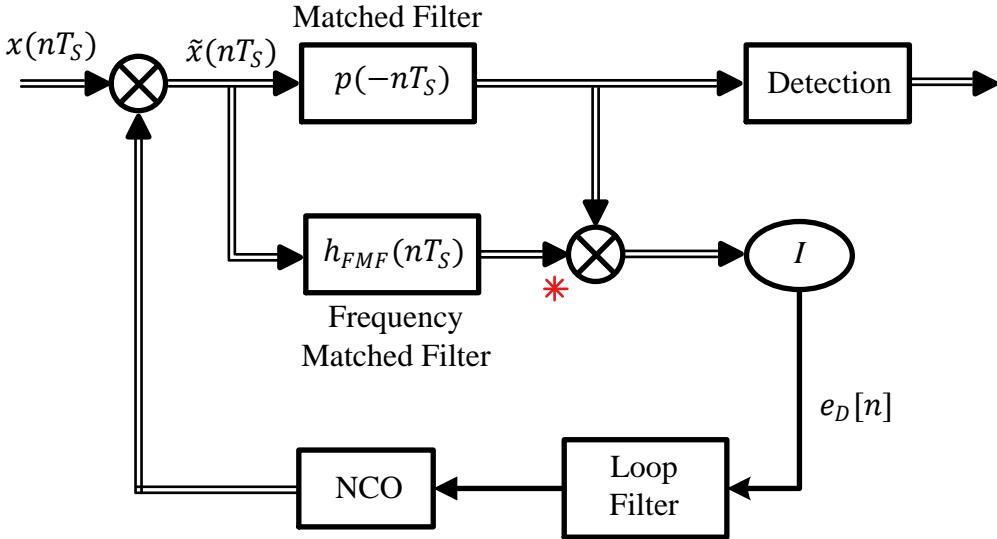


Figure 6.24: A frequency locked loop with a frequency matched filter. Conjugate product of its output with that of the matched filter output generates the error signal

$\sin B = \cos(A \pm B)$ and $\sin A \cos B \pm \cos A \sin B = \sin(A \pm B)$.

$$\begin{aligned} I &\rightarrow \tilde{x}_I(nT_S) = v_I(nT_S) \cos 2\pi \frac{F_{0:e}}{L} n - v_Q(nT_S) \sin 2\pi \frac{F_{0:e}}{L} n \\ Q &\uparrow \quad \tilde{x}_Q(nT_S) = v_Q(nT_S) \cos 2\pi \frac{F_{0:e}}{L} n + v_I(nT_S) \sin 2\pi \frac{F_{0:e}}{L} n \end{aligned}$$

Here, the input signal to the matched filter is given in terms of $F_{0:e} = F_0 - \hat{F}_0$ which is the error signal tracked by the FLL in Figure 6.24. In other words, the signal $\tilde{x}(nT_S)$ is mixed with $F_{0:e}$ in this setup.

Just like the impulse response and frequency response of a matched filter are

$$\begin{aligned} \text{impulse response : } & h_{MF}(nT_S) \\ \text{frequency response : } & H_{MF}(F) \\ \text{output } z(nT_S) = & \tilde{x}(nT_S) * h_{MF}(nT_S), \end{aligned} \tag{6.33}$$

the impulse response and frequency response of the frequency matched filter are

$$\begin{aligned} \text{impulse response : } & h_{FMF}(nT_S) \\ \text{frequency response : } & H_{FMF}(F) \\ \text{output } z_{FMF}(nT_S) = & \tilde{x}(nT_S) * h_{FMF}(nT_S) \end{aligned} \tag{6.34}$$

where $h_{FMF}(nT_S)$ could (ideally) be generated through taking an inverse Fourier Transform of the frequency matched filter already computed in frequency domain, which we will discuss soon.

The dynamics and behaviour of the FLL thus formulated are not much different than those of a PLL. Needless to say, its non-timing-aided and non-data-aided nature

warrants long acquisition and settling times as compared to aided frequency synchronization procedures. Do remember on the other hand that catching up with a frequency offset is much quicker for an FLL than with a phase offset for a PLL.

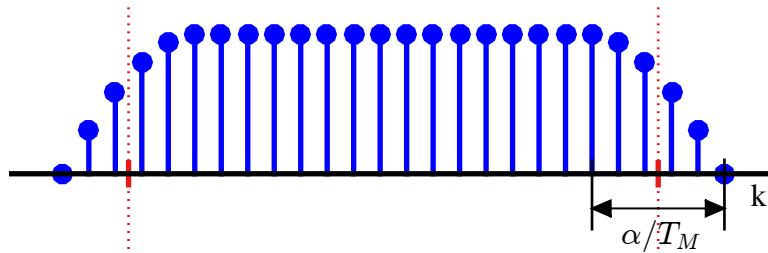
Drawback of a Frequency Matched Filter

As far as implementation is concerned, the actual coefficients of the discrete-time frequency matched filter $h_{FMF}(nT_S)$ are required. For this purpose, we first generate the discrete spectrum and then convert it into time domain coefficients through an inverse Discrete Fourier Transform (iDFT).

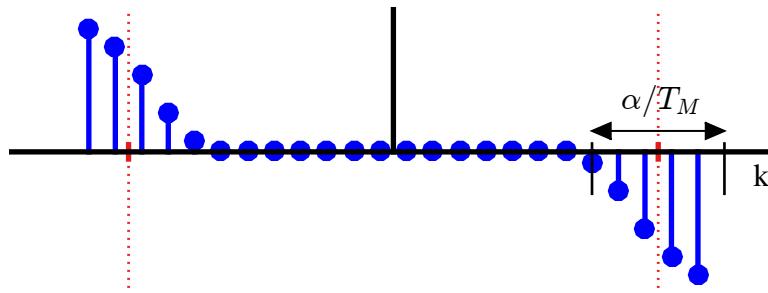
In Section 2.6, we found the coefficients of a filter that can be employed to output the derivative of a signal in discrete-time. Our problem is in frequency domain though and as a result, we have

$$H[k] = \frac{1}{2} \{+1, 0, -1\}$$

When our matched filter – a square-root Nyquist pulse shown in Figure 6.25a – is convolved with such a filter in frequency domain, the coefficients of a discrete frequency matched filter are obtained which are plotted in Figure 6.25b.



(a) A Square-Root Raised Cosine (SRRC) pulse as a matched filter $H_{MF}[k]$. The transition band is important here



(b) Spectrum of the frequency matched filter $H_{FMF}[k]$ that consists of two band edges

Figure 6.25: A matched filter in discrete frequency domain along with the corresponding frequency matched filter for excess bandwidth $\alpha = 0.25$

There are a few observations to make in this Figure.

A good approximation? We noted in Section 2.6 that using only two samples of the input signal to compute the derivative and ignoring all the remaining samples serves the purpose well if the input signal is at least five times oversampled as compared to the minimum limit set by the Nyquist theorem.

Fall at edges Looking at the matched filter itself, the transition band dictates that the frequency matched filter suddenly falls to zero at the band edges, as is visible in Figure 6.25b. While convolving with any FIR filter, the group delay causes the output signal to be delayed by a number of samples given by (assuming an odd filter length)

$$D = \frac{\text{Filter length} - 1}{2}$$

As a result, D samples of the output at the start – when the filter response is moving into the input sequence – are discarded. Similarly, the last D samples – when the filter is moving out of the input sequence – are discarded as well. In this case of a derivative filter $0.5\{+1, 0, -1\}$ above,

$$D = \frac{3 - 1}{2} = 1$$

sample at both edges has been discarded.

Going into time domain The frequency matched filter has two parts: one at the edge of the positive band and the other at the edge of the negative band, and hence it is a special case of a band edge filter. Since the transition band of our matched filter contains a quarter cycle of a cosine, the frequency matched filter consists of a quarter cycle of a sine wave in both the positive and negative bands which was seen during the discussion in continuous domain.

This quarter cycle of the sine ends abruptly at the edges of the transition band and hence cannot be implemented. If we just want to see how its impulse response looks like, observe that this discontinuity at the edges is equivalent to multiplication of a sinusoid with a rectangular window of width α/T_M in frequency domain. Consequently, it triggers convolution in time domain between two impulses arising due to the sinusoid and two sinc signals arising due to the rectangular windows. We do not discuss the details of how to conceptually produce the time domain signal from impulses and sinc signals, as they are similar to those we discuss in the context of band edge filters later.

The I and Q components of this impulse response $h_{FMF}(nT_S)$ are plotted in Figure 6.26 which was obtained by taking the iDFT of the frequency matched filter of Figure 6.25b. Such a signal does not decay fast enough and gives rise to a never ending impulse response that can be observed through the slowly decaying long tails.

We conclude that a better alternative is needed for a realizable filter. Next, we see that the frequency matched filter can be modified into a more sensible form.

Construction of Modified Frequency Matched Filter

As explained before, between $\pm(1 - \alpha)/T_M$ and $\pm(1 + \alpha)/T_M$, the spectral shape of a matched filter $H_{MF}(F)$ is a quarter cycle of a cosine, see the shaded region in Figure 6.27a of width α/T_M . Consequently, the frequency matched filter, being the derivative of a cosine, is also a quarter cycle of a sine within that region. Recall that the origin of the difficulty to realize the impulse response for a frequency matched filter lies in the sudden discontinuity at frequencies $\pm(1 + \alpha)/T_M$ on the edges of the spectrum.

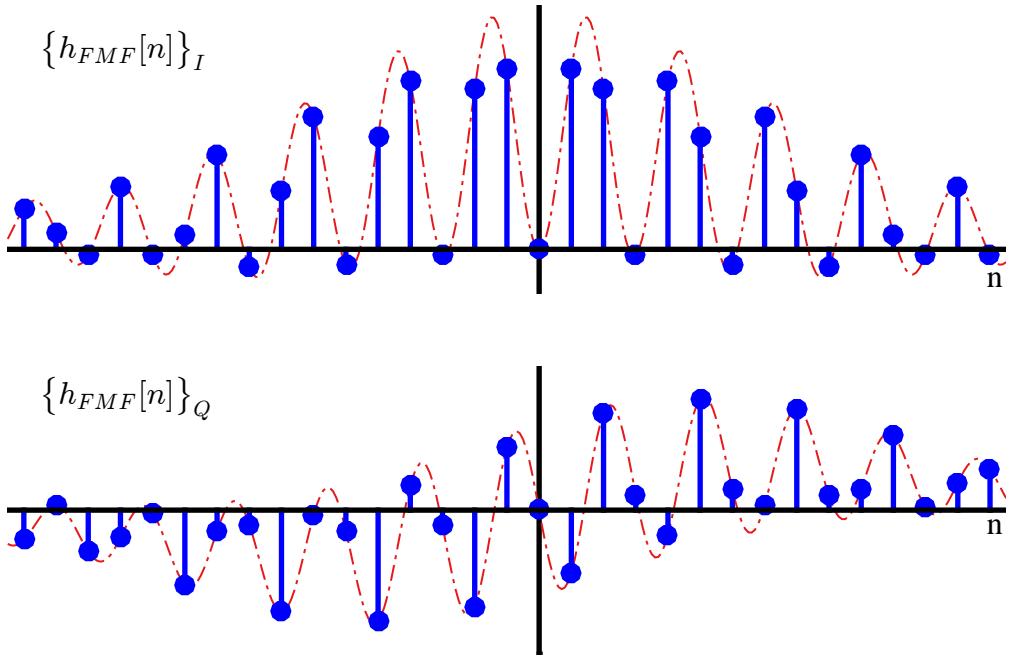


Figure 6.26: I and Q components of the impulse response $h_{FMF}(nT_S)$ computed by taking iDFT of the frequency matched filter with $\alpha = 0.25$ of Figure 6.25b

One approach to solve this problem is to continue the spectral shape past the discontinuity in the same trajectory such that the final spectral shape denoted by $\tilde{H}_{FMF}(F)$ forms a half cycle of a sine wave because the derivative of the quarter cycle of a cosine in the square-root Nyquist spectrum is a sine. This is drawn in Figure 6.27b for $\alpha = 0.25$, where the shaded region is the original frequency matched filter $H_{FMF}(F)$. This increases the total frequency span of one side of the filter to $2\alpha/T_M$. A wider bandwidth and a relatively smooth transition towards zero imply a reduced time span and hence we will soon see that the time domain response of such a filter falls off quite rapidly as compared to the original frequency matched filter. The edges of the modified frequency matched filter now end at

$$\frac{1+\alpha}{2T_M} + \frac{\alpha}{T_M} = \frac{1+3\alpha}{2T_M}$$

with a total range from $\pm(1-\alpha)/2T_M$ to $\pm(1+3\alpha)/2T_M$.

If we ignore the scaling factors for a moment (which can always be normalized later), we can write the frequency response of the modified frequency matched filter

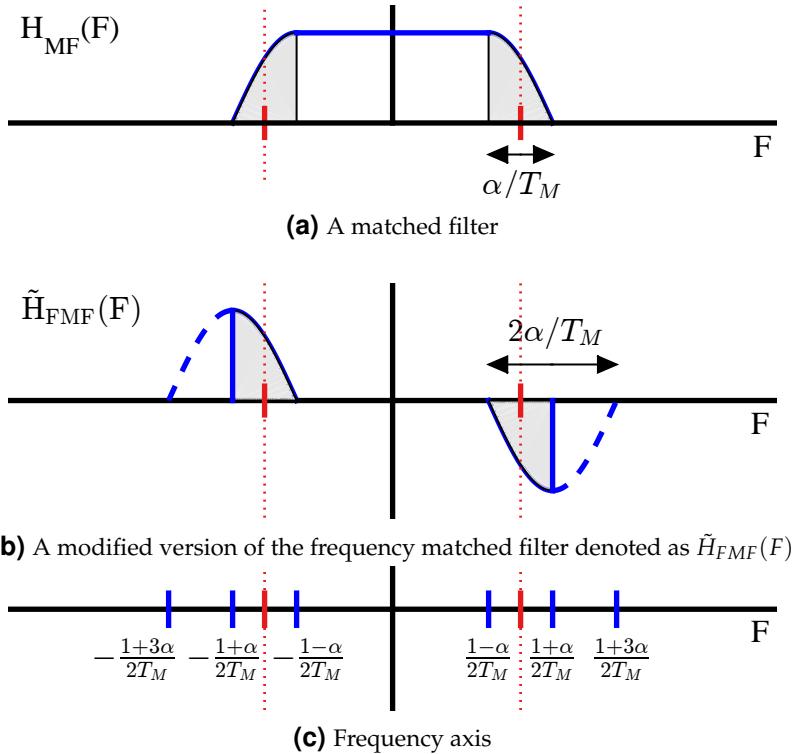


Figure 6.27: A frequency matched filter H_{MF} is extended into a modified frequency matched filter $\tilde{H}_{FMF}(F)$ through continuing the sine wave of the transition band for $\alpha = 0.25$

$\tilde{H}_{FMF}(F)$ from Eq (6.32) as

$$\tilde{H}_{FMF}(F) = \begin{cases} 0 & 0 \leq |F| \leq \frac{1-\alpha}{2T_M} \\ -\sin \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(+F - \frac{1-\alpha}{2T_M} \right) \right\} & +\frac{1-\alpha}{2T_M} \leq F \leq +\frac{1+3\alpha}{2T_M} \\ +\sin \left\{ 2\pi \cdot \frac{T_M}{4\alpha} \left(-F - \frac{1-\alpha}{2T_M} \right) \right\} & -\frac{1+3\alpha}{2T_M} \leq F \leq -\frac{1-\alpha}{2T_M} \\ 0 & |F| \geq \frac{1+3\alpha}{2T_M} \end{cases} \quad (6.35)$$

The impulse response and the output of $\tilde{H}_{FMF}(F)$ are given by $\tilde{h}_{FMF}(nT_S)$ and $\tilde{z}_{FMF}(nT_S)$, respectively. Next, we explain the problem with the modified frequency matched filter.

Problem with the Modified Frequency Matched Filter

Having the modified frequency matched filter at hand, now we can construct an error signal as a product of the matched filter output and the modified frequency matched filter output, as was done before in Eq (6.29).

$$e_D[n] = \left\{ z(nT_S) \cdot \tilde{z}_{FMF}^*(nT_S) \right\}_I \quad (6.36)$$

where the term $\tilde{z}_{FMF}(nT_S)$ represents the output of the modified frequency matched filter $\tilde{h}_{FMF}(nT_S)$. The corresponding Rx structure is very similar to what we encountered in Figure 6.24, with the only difference being the replacement of frequency matched filter $h_{FMF}(nT_S)$ with a modified frequency matched filter $\tilde{h}_{FMF}(nT_S)$.

The error signal above computes the inphase component of the product $z(nT_S) \cdot \tilde{z}_{FMF}^*(nT_S)$. Let us find out the spectrum of such a signal which is drawn in Figure 6.28a for a zero CFO and in Figure 6.28b for a non-zero CFO. Excess bandwidth α in both cases is equal to 0.25 and $L = 4$ samples/symbol.

There is no spectral line at $F = 0$ in Figure 6.28a since there is no CFO present. Ideally, the error signal spectrum should contain a single line at DC proportional to the CFO. Now while the spectral line at DC corresponding to the CFO in Figure 6.28b is small, the spectrum around it seems to possess a substantial amount of noise, as far as the output of an FED is concerned. Since the signal at these frequencies arises due to interactions among data symbols themselves, it is often termed as *self noise*, also known as *modulation noise* or *pattern noise*.

Self noise is a result of the interactions of the signal itself as explained now. The error signal in time domain is formed by the product of the matched and modified frequency matched filter outputs. This product in time domain is a convolution in frequency domain between the spectrum of the matched filter output (which spans the full signal bandwidth) and that of the modified frequency matched filter output (which is limited in bandwidth). This convolution implies sliding one over the other and hence the resultant signal possesses the maximum amount of undesirable spectral components, as evident from Figure 6.28b. The main point is that the problem here is not with the modified frequency matched filter but with the matched filter that corrupts the resultant error signal.

Note 6.4 How good is maximum correlation?

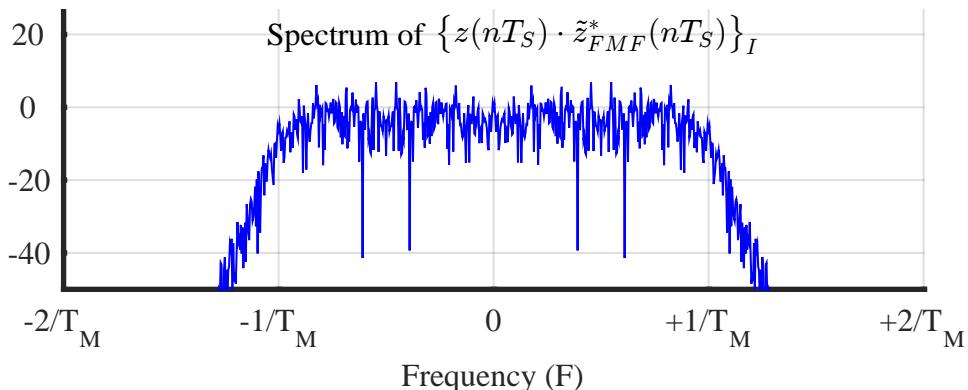
The derivative FED, on which the above modifications are based, is founded on the maximum correlation (which comes from maximum likelihood) theory. Since the maximum likelihood technique is founded on the AWGN characteristic of the noise entering the Rx system, it outputs an algorithm that deals with this noise satisfying a criterion for optimality. For the random data sequence, *it does not care about the self noise component*. So it is a *common mistake* to assume that the maximum likelihood or maximum correlation technique is a black box which outputs the best possible algorithm for any specific scenario. The reality is that it is only as good as the model over which it is constructed, just like a computer even the best of which is only as good as the code written for execution.

We conclude that a derivative FED, even the one implementing a modified frequency matched filter, is not very suitable for a practical implementation. However, it

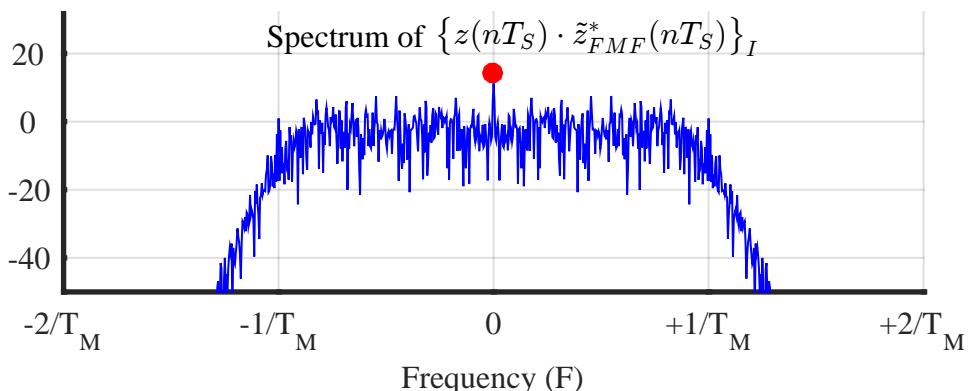
does lead us to some better FEDs with efficient synthesis of a CFO synchronization block as explained next.

6.4.3 Feedback: Band Edge FLL

In this section and onwards, we discuss band edge filters that help acquiring the frequency and timing offsets from the Rx waveform. Truly, this is one of the most delightful topics for a person seeking to understand the field of signal synchronization. Unfortunately, in the digital and wireless communications literature, there is very little information available on this topic and my target is to explain this concept from the fundamentals.



(a) In the presence of a zero CFO (Carrier Frequency Offset)



(b) In the presence of a non-zero CFO

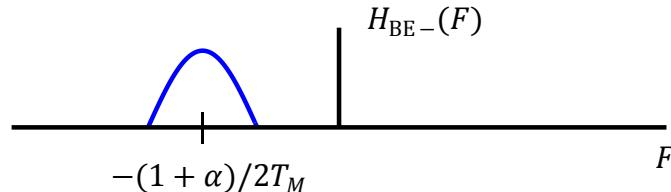
Figure 6.28: Spectrum of the error signal as the inphase component of the product between the matched filter output $z(nT_S)$ and conjugate of modified frequency matched filter output $\tilde{z}_{FMF}^*(nT_S)$, generated for $L = 4$ samples/symbol and excess bandwidth $\alpha = 0.25$

Construction of Band Edge Filters

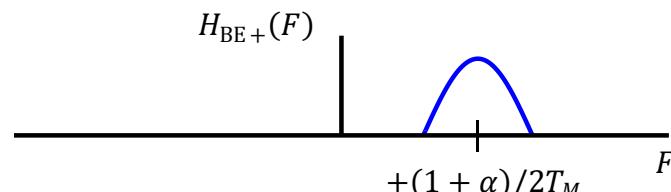
There are two routes to take from here. We can either start with the intuitive process behind the generation of the Carrier Frequency Offset (CFO) spectral line and subsequently construct a more efficient Frequency Error Detector (FED). Or we can construct the FED first and later explore the brilliant intuition behind it. I take the later route due to the convenience of having some results in hand for the intuitive explanation part.

The generation of self noise was attributed to the fact that $e_D[n] = \{z(nT_S) \cdot \tilde{z}_{FMF}^*(nT_S)\}_I$ in Eq (6.36) is formed by the product of the matched filter and modified frequency matched filter outputs in time domain. Equivalently, the convolution in frequency domain includes the contribution from the wide bandwidth of $h_{MF}(nT_S)$ sliding over the limited bandwidth of $\tilde{h}_{FMF}(nT_S)$. The solution is to find a way to eliminate the matched filter entirely while forming the FED output. For this purpose, one hint we have is that the matched filter region outside the transition band adds nothing useful but self noise and hence *the matched filter component, out of the two that form the product in the error signal, should be modified as well*. Proceeding in this manner, certainly we are looking for another filter with a restricted band! When we find such a filter, we can form the FED signal by the product of its output with that of the modified frequency matched filter, as we discover now.

To remove the transition bands, let us first break down our modified frequency matched filter $\tilde{H}_{FMF}(F)$ developed in Figure 6.27b and Eq (6.35) into two parts, one an upper band edge filter $H_{BE+}(F)$ and the other a lower band edge filter $H_{BE-}(F)$. The lower band edge filter is exactly the same as the lower band of $\tilde{H}_{FMF}(F)$ in Figure 6.27b, while the upper band edge filter is its replica on the positive side of the spectrum. These two filters are shown in Figure 6.29.

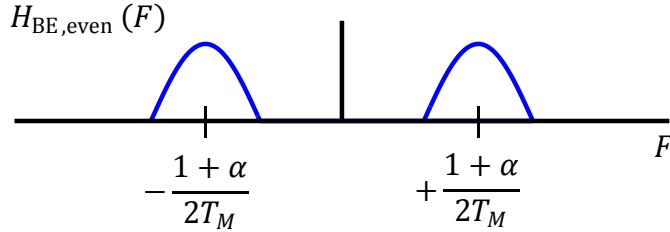


(a) Spectrum of the lower band edge filter

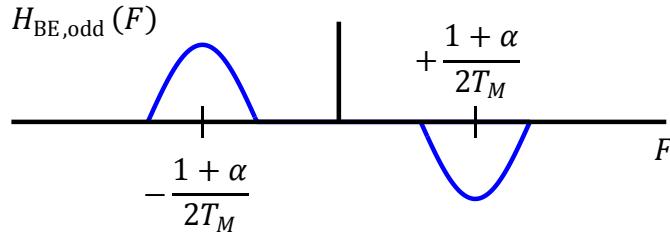


(b) Spectrum of the upper band edge filter

Figure 6.29: Breaking the modified frequency matched filter into two parts for $\alpha = 0.25$



(a) Spectrum of the even band edge filter



(b) Spectrum of the odd band edge filter

Figure 6.30: Forming even and odd band edge filters for $\alpha = 0.25$. Odd band edge filter is actually the modified frequency matched filter, while even band edge filter is later used in the error signal in place of the matched filter

Clearly, our modified frequency matched filter $\tilde{H}_{FMF}(F)$ can be constructed by just subtracting $H_{BE+}(F)$ from $H_{BE-}(F)$. Using this hint, we can also form another filter through summing these two basic filters. In this process, we obtain two entirely new filters:

- a sum band edge filter that has an even symmetric frequency response, and
- a difference band edge filter that has an odd symmetric frequency response.

As a consequence, they are known as *even band edge filter* and *odd band edge filter*. They can be denoted by $H_{BE,even}(F)$ and $H_{BE,odd}(F)$, respectively, and are illustrated in Figure 6.30.

$$H_{BE,even}(F) = H_{BE-}(F) + H_{BE+}(F)$$

$$H_{BE,odd}(F) = H_{BE-}(F) - H_{BE+}(F)$$

(6.37)

where

$$H_{BE,odd}(F) = \tilde{H}_{FMF}(F)$$

i.e., the odd band edge filter $H_{BE,odd}(F)$ of Figure 6.30b is – not surprisingly – our modified frequency matched filter $\tilde{H}_{FMF}(F)$ discussed earlier.

In light of the above expressions, we can introduce the outputs of even and odd

band edge filters as

$$\begin{aligned} z_{\text{BE,even}}(nT_S) &= z_{\text{BE-}}(nT_S) + z_{\text{BE+}}(nT_S) \\ z_{\text{BE,odd}}(nT_S) &= z_{\text{BE-}}(nT_S) - z_{\text{BE+}}(nT_S) \end{aligned} \quad (6.38)$$

where $z_{\text{BE-}}(nT_S)$ and $z_{\text{BE+}}(nT_S)$ are the outputs of the lower and upper band edge filters, respectively. Next, we move towards forming an error signal with the available signals.

Forming the error signal

Recall that the original Derivative FED was formed through taking the derivative of $|z(nT_S)|^2$ in Eq (6.28) with respect to the CFO estimate, which was subsequently modified through replacing $\dot{z}(nT_S)$ with $z_{FMF}(nT_S)$ in Eq (6.29) and then with $\tilde{z}_{FMF}(nT_S)$ in Eq (6.36). To summarize,

$$e_D[n] \rightarrow \left\{ z(nT_S) \cdot \left\{ \dot{z}(nT_S) \right\}^* \right\}_I \quad (6.39)$$

$$\rightarrow \left\{ z(nT_S) \cdot z_{FMF}^*(nT_S) \right\}_I \quad (6.40)$$

$$\rightarrow \left\{ z(nT_S) \cdot \tilde{z}_{FMF}^*(nT_S) \right\}_I \quad (6.41)$$

Denoting the outputs of the odd and even band edge filters by $z_{\text{BE,odd}}(nT_S)$ and $z_{\text{BE,even}}(nT_S)$, respectively, we employ the following two facts.

- The odd band edge filter $H_{\text{BE,odd}}(F)$ is exactly the same as the modified frequency matched filter $\tilde{H}_{FMF}(F)$. Therefore, its output $\tilde{z}_{FMF}^*(nT_S)$ can be replaced with $z_{\text{BE,odd}}(nT_S)$ in Eq (6.41).
- Recalling the symmetry in the spectrum of the matched filter, we can note that our even band edge filter $H_{\text{BE,even}}(F)$ is also even symmetric. The role of the matched filter in forming the error signal is to remove the effect of modulation. For this purpose, utilizing the whole bandwidth is not necessary and only the transition band suffices here as far as the CFO estimate \hat{F}_Δ is concerned. Therefore, the matched filter output $z(nT_S)$ can be replaced with $z_{\text{BE,even}}(nT_S)$ in Eq (6.41).

After performing the above two substitutions, Eq (6.41) can be modified to yield our *Band Edge Frequency Error Detector (FED)* as

$$e_D[n] = \left\{ z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S) \right\}_I \quad (6.42)$$

A block diagram for the implementation of a band edge FED is drawn in Figure 6.31 where even and odd band edge filter outputs are constructed through addition and subtraction of the outputs from the upper and lower band edge filters, respectively. The inphase part of the error signal then drives the loop.

After reaching the final expression of our band edge FED, one wonders why three initial FEDs were modified in the first place. I summarize below our findings in this regard.

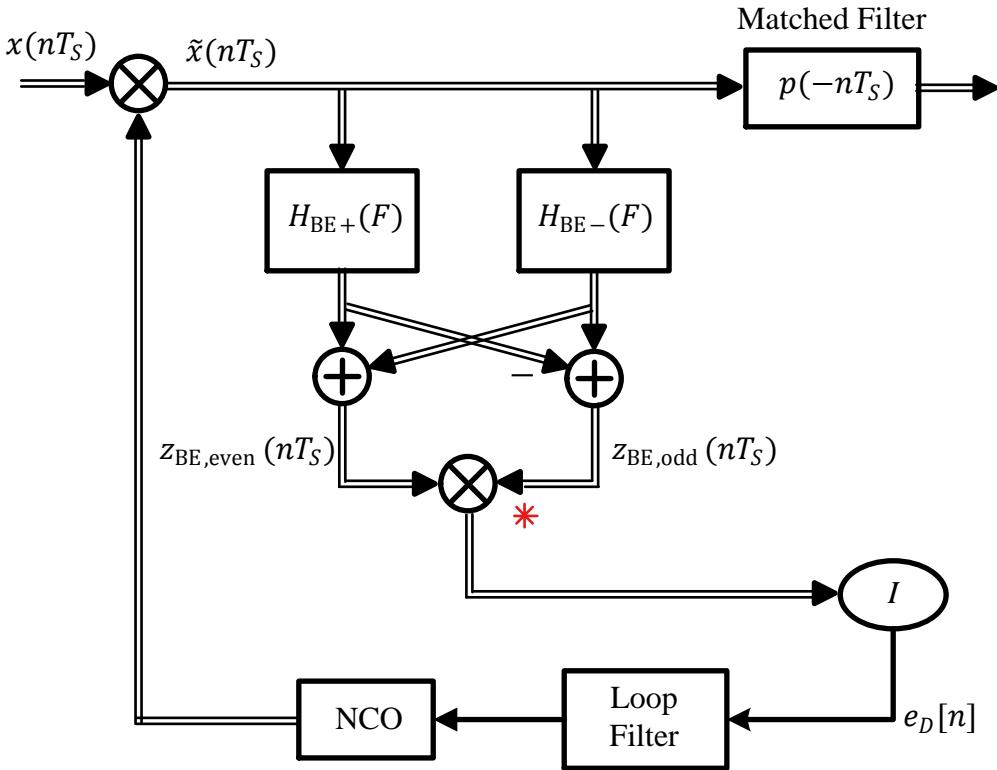


Figure 6.31: A frequency locked loop with band edge filters. Even and odd band edge filter outputs are constructed through addition and subtraction of the outputs from the upper and lower band edge filters, respectively

Note 6.5 Why many FEDs?

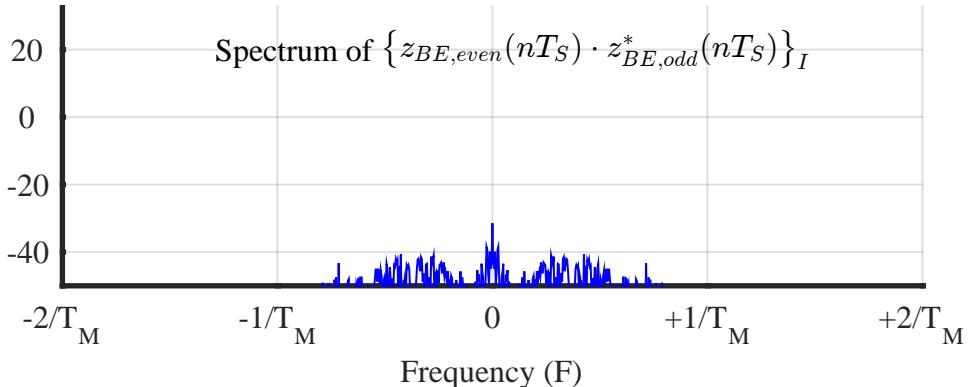
Starting from the derivative FED in Eq (6.39), we proceed as follows.

Derivative FED – Eq (6.39) In addition to the injection of self noise, the simplest reason to avoid a derivative FED is that it computes the derivative of a time-varying Rx signal in real-time, which includes Gaussian noise as well. For a sufficiently oversampled signal and utilizing the first central difference method, the derivative is computed by taking the difference which directly leads to doubling of the noise variance and hence noise power^a. It is better to differentiate a fixed known signal and utilize it for this purpose, as was done with frequency matched filter.

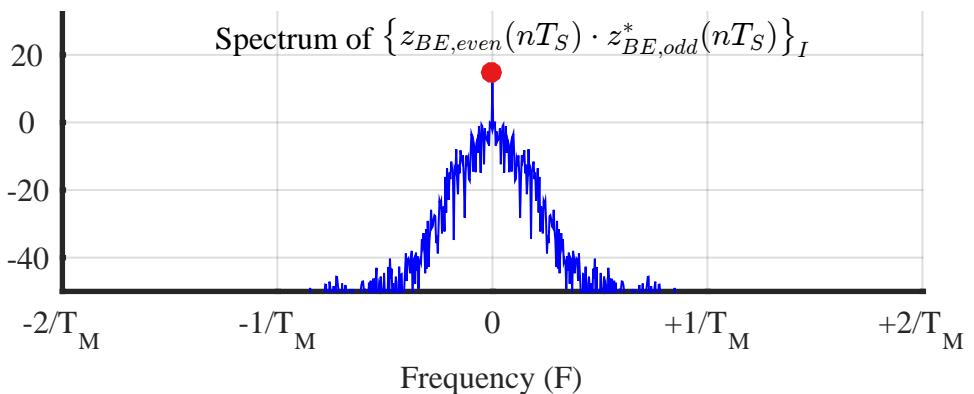
Frequency matched filter – Eq (6.40) A frequency matched filter consists of a quarter cycle of a sine that ends abruptly at the edges of the transition band and hence cannot be implemented.

Modified frequency matched filter – Eq (6.41) While a modified frequency matched filter avoids the discontinuity, the error signal contains a significant level of self noise generated by convolution of its frequency domain output with that of the matched filter spectrum.

Band edge filters – Eq (6.42) Here, the convolution in frequency domain between the



(a) In the presence of a zero CFO (Carrier Frequency Offset)



(b) In the presence of a non-zero CFO

Figure 6.32: Spectrum of $e_D[n]$ as the *inphase component* of the product between the even band edge filter output $z_{BE,even}(nT_S)$ and conjugate of odd band edge filter output $z_{BE,odd}^*(nT_S)$, generated for $L = 4$ samples/symbol and excess bandwidth $\alpha = 0.25$. Compare with Figure 6.28. Later, Figure 7.67 draws its quadrature component that aids in timing acquisition

two filter outputs only contains a minimum level of self noise due to the absence of the full bandwidth matched filter. This is shown in Figure 6.32a for a zero CFO and Figure 6.32b for a non-zero CFO with an excess bandwidth $\alpha = 0.25$. Compare this with the spectrum of the error signal formed by matched and modified frequency matched filters in Figure 6.28 and observe how self noise has been reduced to a significantly small level by eliminating the matched filter from the FLL.

A fifth FED Soon we will encounter a fifth form of FED that is of interest for CFO estimation and known as a differential power measurement FED. It will nicely help in completing the picture of above frequency error detectors.

^aThe variance of a sum or difference of two independent Gaussian random variables is twice the original.

In Chapter 7, we will have another look into the origin of this spectral line at DC corresponding to the CFO when we discuss how the multiplication of two terms that form the band edge error signal leads to a convolution in frequency domain.

Impulse Response of Band Edge Filters

The discussion until now was in frequency domain. The question now is to find the impulse response of the band edge filters. We will focus on an Finite Impulse Response (FIR) filter implementation due to their linear phase property over the entire bandwidth. For this purpose, we first draw a baseband version of the spectral lobe we encountered in the band edge filters and shift this spectrum to the desired location later. This is done in Figure 6.33.

Clearly, the filter translated to baseband is nothing but a simple half-cosine. To find its support, we recall from Figure 6.27c that the width of this spectral lobe is $2\alpha/T_M$. Here, the complete period of the half-cosine, i.e., $4\alpha/T_M$, is also shown. Its mathematical expression is given by

$$H_{BE,\text{baseband}}(F) = \begin{cases} \cos\left(2\pi \cdot \frac{T_M}{4\alpha} \cdot F\right) & \text{for } -\frac{\alpha}{T_M} \leq F \leq +\frac{\alpha}{T_M} \\ 0 & \text{everywhere else} \end{cases} \quad (6.43)$$

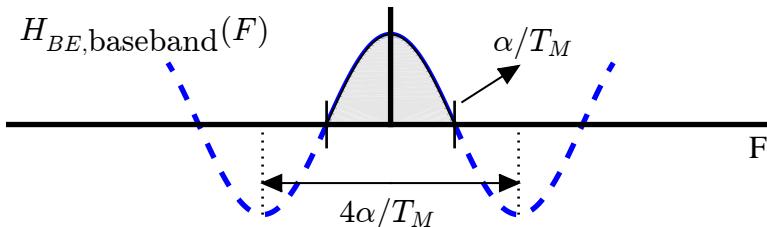


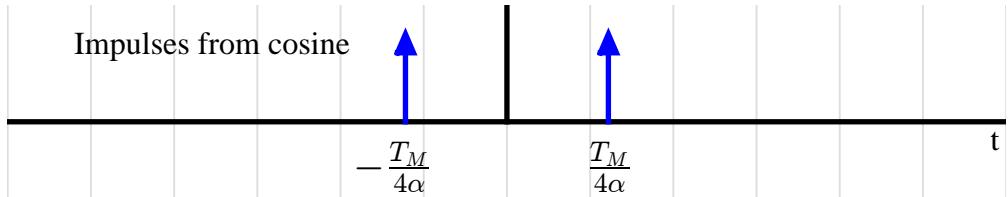
Figure 6.33: Baseband version of the band edge filter, which is nothing but a simple half-cosine of period $4\alpha/T_M$ and width $2\alpha/T_M$

To construct the impulse response corresponding to such a frequency response, we trace the following sequence of steps.

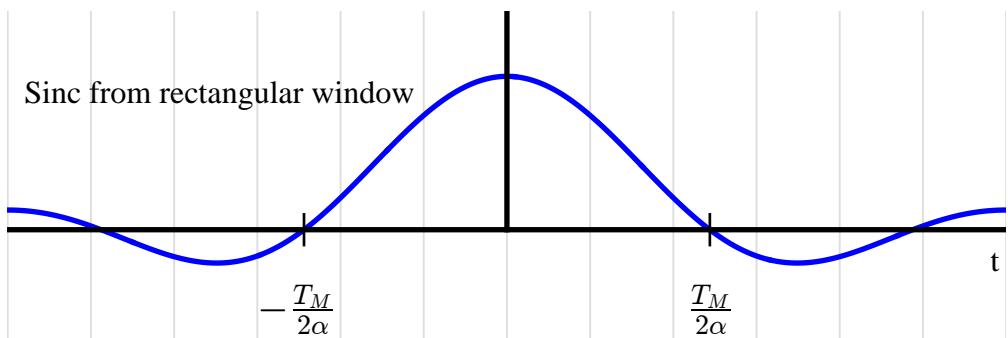
- In frequency domain, the cosine in the above expression has a frequency support only from $-\alpha/T_M$ to $+\alpha/T_M$ as shown in Figure 6.33, not from $-\infty$ to ∞ .
- In frequency domain, this is equivalent to multiplication with a rectangular window of width $2\alpha/T_M$.
- Multiplication in frequency domain is convolution in time domain of the following two signals:

Two impulses: In time domain, $\cos(2\pi \cdot T_M/(4\alpha) \cdot F)$ results in two impulses at time locations $\pm T_M/4\alpha$. This is drawn in Figure 6.34a.

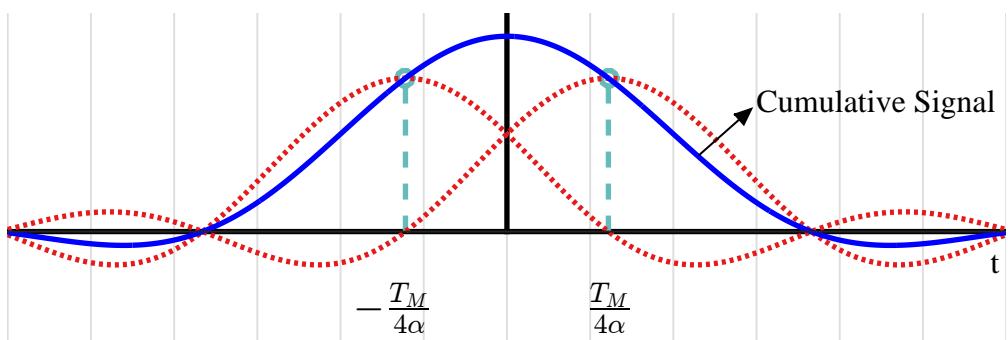
One Sinc: The rectangular window of width $2\alpha/T_M$ is a sinc signal in time domain with zero crossings at the reciprocal width, i.e., at integer multiples of $\pm T_M/2\alpha$, which is plotted in Figure 6.34b.



(a) Two impulses in time domain come from cosine in frequency domain with period $4\alpha/T_M$



(b) Sinc in time domain is a result of multiplying the cosine in frequency domain with a rectangular window of width $2\alpha/T_M$



(c) Convolution of sinc with two impulses produces two sincs centered at locations of those impulses

Figure 6.34: Construction of the impulse response of the band edge filter (baseband version)

- Convolution of a signal with an impulse is the signal itself.
- Such a convolution generates two sinc signals: one at time $+T_M/4\alpha$ while the other at time $-T_M/4\alpha$. When these two sinc signals are added together, we get the impulse response of a baseband band edge filter. This is illustrated in Figure 6.34c. Notice that this is very similar to what we found during the formation of a Raised Cosine pulse shaping filter in Section 3.6.
- Finally, the actual impulse responses of the following band edge filters are required: lower band edge filter $h_{BE-}(nT_S)$, upper band edge filter $h_{BE+}(nT_S)$, even

band edge filter $h_{BE,even}(nT_S)$ and odd band edge filter $h_{BE,odd}(nT_S)$. Starting from the baseband impulse response in Figure 6.34c, we proceed as follows.

- ◊ Build the impulse response $h_{BE-}(nT_S)$ of the lower band edge filter $H_{BE-}(F)$ by shifting the baseband impulse response to band edge frequency $-0.5(1 + \alpha)/T_M$ through multiplication with a complex sinusoid of frequency $-0.5(1 + \alpha)/T_M$. I and Q components of $h_{BE-}(nT_S)$ are drawn in Figure 6.35a.
- ◊ Form the impulse response $h_{BE+}(nT_S)$ of upper band edge filter $H_{BE+}(F)$ by shifting the baseband impulse response to band edge frequency $+0.5(1 + \alpha)/T_M$ through multiplication with a complex sinusoid of frequency $+0.5(1 + \alpha)/T_M$. I and Q components of $h_{BE+}(nT_S)$ are drawn in Figure 6.35b.

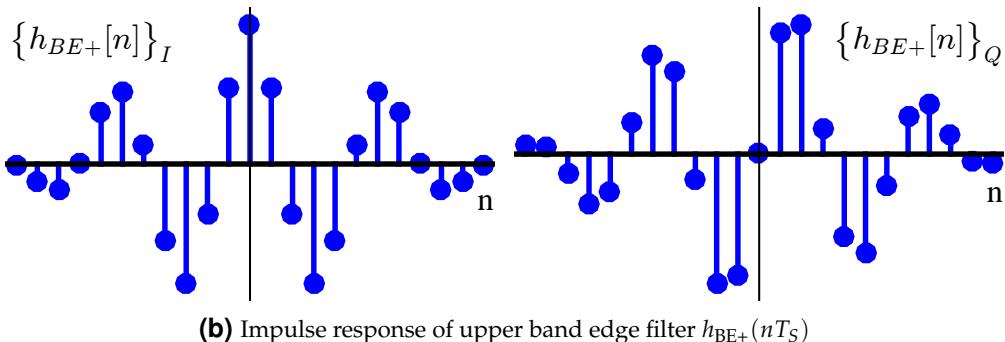
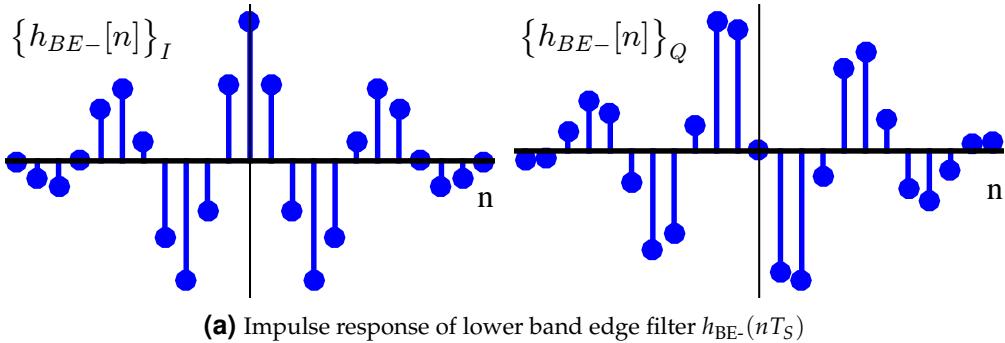


Figure 6.35: I and Q components of impulse responses of lower band edge filter $h_{BE-}(nT_S)$ and upper band edge filter $h_{BE+}(nT_S)$, respectively

A discussion on these impulse responses is now in order.

- Notice that the I components of the lower and upper band edge filters are the same, while the Q component of one is a negative version of the other. This makes sense because both impulse responses are generated from a basic time series and the difference is the angle of rotation of the complex sinusoid $\pm 0.5(1 + \alpha)/T_M$. I part of such a sinusoid is a cosine which is an even signal $\cos(\pm A) = \cos A$ that subsequently generates exactly the same I components. On the other hand, Q part of such a sinusoid is a sine which is an odd signal $\sin(\pm A) = \pm \sin A$ that subsequently generates the Q part of one as a negative of the other.

■ Referring to how even and odd band edge filters are created from lower and upper band edge filters as in Eq (6.37),

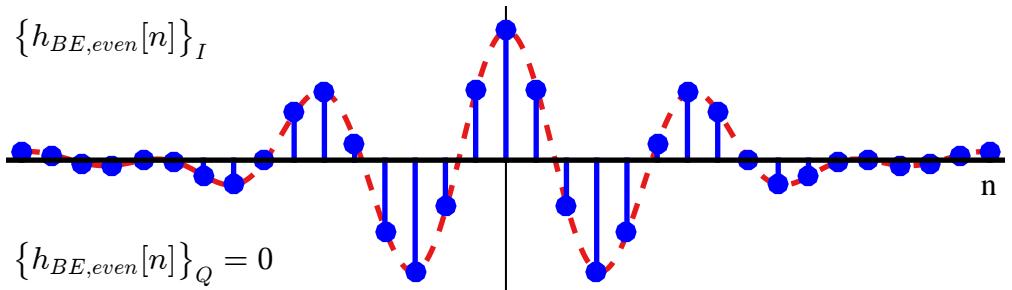
- ◊ add the above two impulse responses together to create even band edge filter $h_{BE,even}(nT_S)$, and
- ◊ subtract the upper from the lower to create odd band edge filter $h_{BE,odd}(nT_S)$.

The outcomes of the above procedure for excess bandwidth $\alpha = 0.25$ are illustrated in Figure 6.36.

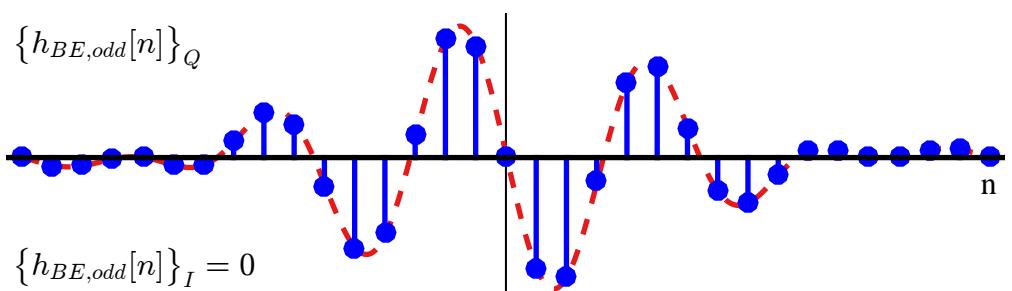
- Observe that since $h_{BE,even}(nT_S)$ is a sum of $h_{BE-}(nT_S)$ and $h_{BE+}(nT_S)$, the Q components in Figure 6.35 cancel out, while the opposite is true for $h_{BE,odd}(nT_S)$. As a verification, remember from Section 2.9 that a real and even signal has a real and even inverse transform that is true for $h_{BE,even}(nT_S)$. A similar argument for a real and odd signal holds true for $h_{BE,odd}(nT_S)$ in regards to Q component.
- Compare the above responses with that of the frequency matched filter in Figure 6.26 and see how the transients are significantly smoothed out. Hence, it is a realizable set of filters.

Intuition Behind Working of the Band Edge Filters

Until now, we traced the mathematical steps and their easy explanations. However, we could have arrived at the same set of filters by utilizing the intuition behind how band edge filters work. We describe the intuition by following our previous work and



(a) Impulse response of the even band edge filter $h_{BE,even}(nT_S)$. Notice that the Q part is zero



(b) Impulse response of the odd band edge filter $h_{BE,odd}(nT_S)$. Notice that the I part is zero

Figure 6.36: Impulse responses of even and odd band edge filters for $\alpha = 0.25$

then explain why the mathematical formulation was not that bad and had to offer its own advantages.

Let us start with the band edge FED in Eq (6.42) reproduced below.

$$e_D[n] = \left\{ z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S) \right\}_I$$

and plug in the definitions of even and odd band edge filters from Eq (6.38) reproduced below as well.

$$\begin{aligned} z_{\text{BE,even}}(nT_S) &= z_{\text{BE-}}(nT_S) + z_{\text{BE+}}(nT_S) \\ z_{\text{BE,odd}}(nT_S) &= z_{\text{BE-}}(nT_S) - z_{\text{BE+}}(nT_S) \end{aligned}$$

The error signal thus becomes

$$\begin{aligned} e_D[n] &= \left\{ \left(z_{\text{BE-}}(nT_S) + z_{\text{BE+}}(nT_S) \right) \left(z_{\text{BE-}}(nT_S) - z_{\text{BE+}}(nT_S) \right)^* \right\}_I \\ &= \left\{ z_{\text{BE-}}(nT_S) z_{\text{BE-}}^*(nT_S) - z_{\text{BE+}}(nT_S) z_{\text{BE+}}^*(nT_S) + \right. \\ &\quad \left. z_{\text{BE+}}(nT_S) z_{\text{BE-}}^*(nT_S) - z_{\text{BE-}}(nT_S) z_{\text{BE+}}^*(nT_S) \right\}_I \\ &= \left\{ |z_{\text{BE-}}(nT_S)|^2 - |z_{\text{BE+}}(nT_S)|^2 + \right. \\ &\quad \left. \left(z_{\text{BE+}}(nT_S) z_{\text{BE-}}^*(nT_S) - \left(z_{\text{BE+}}(nT_S) z_{\text{BE-}}^*(nT_S) \right)^* \right) \right\}_I \quad (6.44) \end{aligned}$$

The first two terms are the magnitude squared values and hence have no Q component. The last term in the brackets is a difference of two complex signals, one of which is a conjugate of the other. Recalling the definition of a conjugate,

$$\begin{array}{ll} I & \rightarrow & \{V^*\}_I = V_I \\ Q & \uparrow & \{V^*\}_Q = -V_Q \end{array}$$

we can see that the I component of both a complex signal and its conjugate are the same. Therefore, this I part cancels out in the last term of Eq (6.44) and we are left with the error signal $e_D[n]$ as

$$e_D[n] = |z_{\text{BE-}}(nT_S)|^2 - |z_{\text{BE+}}(nT_S)|^2 \quad (6.45)$$

In words, this simple equation tells us that as far as CFO estimation is concerned, a *band edge FED is just an energy difference between the outputs of lower and upper band edge filters*. Therefore, it is also known as a *Differential Power Measurement (DPM) FED* and can be alternatively obtained through computing the energy difference between the two filter outputs.

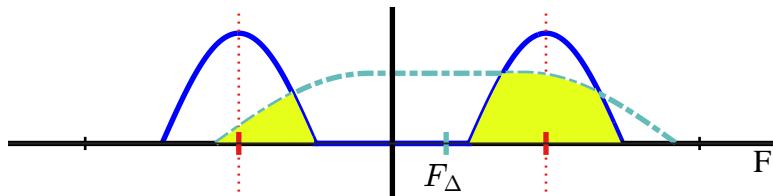
To understand its principle of operation, consider Figure 6.37a.

- When the Tx signal is shifted in frequency by a positive F_Δ , the incoming spectrum is plotted as a green dashed line in Figure 6.37a. It is clear from the shaded

regions that the output energy from the upper band edge filter is larger than that from the lower band edge filter. Consequently, the difference in their powers instructs the FLL to further downconvert the incoming signal, thus increasing its CFO estimate \hat{F}_Δ . A reverse argument holds for a negative F_Δ .

- On the other hand, a zero carrier frequency offset $F_\Delta = 0$ implies that the input spectrum arrives right at the middle of the our band edge filters. The energies in both filter outputs are equal due to their symmetry and their difference renders the error signal zero.

The energy difference perspective reminds of me of the street hawkers during my childhood days. They used to roam the streets selling random stuff and carried a weighing scale – very similar to the one shown in Figure 6.37b. The weights were placed on one side of the balance and the objects on the other. Naturally, the heavier of the two pulled the balance on its side, exactly like a CFO affected communication signal entering into a band edge FLL.



(a) A CFO generates more energy in one band edge filter



(b) A DPM-FED works like a weighing balance of old days

Figure 6.37: Logic behind a Differential Power Measurement (DPM) FED generating an error signal for CFO estimation

Note 6.6 Why the complicated route?

A question at this stage is that if a CFO could be detected by a simple pair of upper and lower band edge filters $H_{BE+}(F)$ and $H_{BE-}(F)$, why did we take the complicated route of constructing even and odd band edge filters, namely $H_{BE,even}(F)$ and $H_{BE,odd}(F)$?

The answer is that such a detector suffices when only the CFO estimation is of interest. The I part of the band edge error signal, a product of even and odd band edge filter outputs in Eq (6.42), lead us to the current simpler expression. Had we investigated the Q component, we would have found a very useful timing error detector that beautifully complements the frequency error detector generated by I part. That we

will do in Chapter 7.

Rx Structure

Based on the above description, a block diagram for the implementation of a band edge FLL is illustrated in Figure 6.38. It consists of two parallel branches, one with an upper band edge filter and the other with a lower band edge filter. The error signal is generated through the difference in the output energy of the two filters that drives an FLL towards gaining frequency lock with the incoming reference. This Rx structure is also known as a *dual filter detector*.

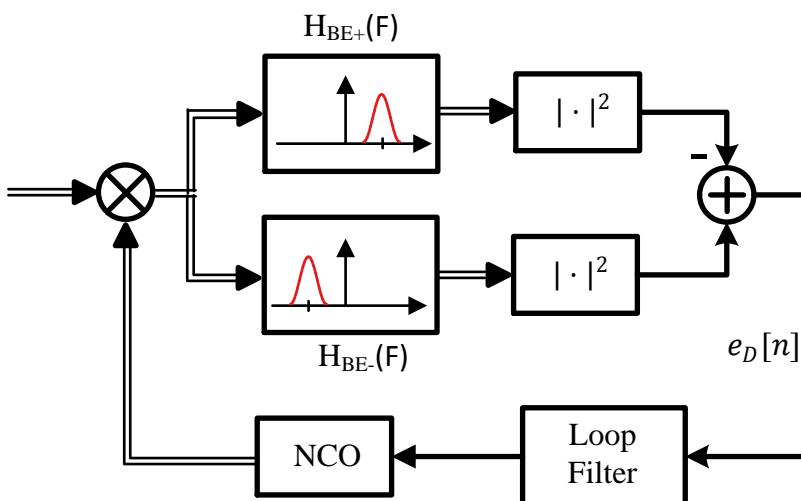


Figure 6.38: A frequency locked loop with a band edge FED: a dual filter detector

One advantage a band edge FLL has over other schemes discussed before is that it is better suited to frequency compensation in a wireless channel. We will discuss wireless channels in detail in Chapter 8. As a final remark, there is a little chance of confusion between this dual filter detector and an even band edge filter. As depicted in Figure 6.38, a dual filter detector computes the difference of two energies in its respective arms whereas the output of an even band edge filter arises from a simple convolution with a single filter.

Exercise 6.1

In GNU Radio, the block ‘FLL Band-Edge’ implements the above concepts within a frequency locked loop with the following parameters.

Samples Per Symbol: Recall that with large frequency offsets, matched filtering and downsampling to 1 sample/symbol is not an option. Therefore, the input to this block is L samples/symbol.

Filter Rolloff Factor: This is the excess bandwidth α of the pulse shaping filter and determines the bandwidth occupied by the band edge filters, see Figure 6.33.

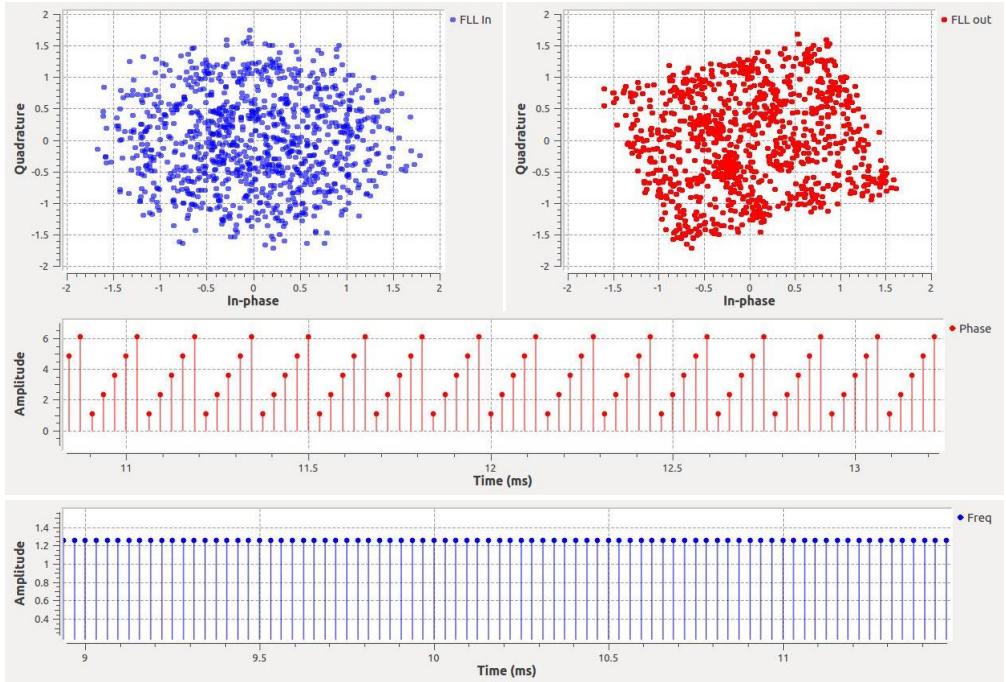


Figure 6.39: Output from the FLL Band-Edge block after frequency convergence

Prototype Filter Size: As evident from Figure 6.36, a large filter size better approximates the practical band edge spectrum with lower sidelobes.

Loop Bandwidth: See Section 4.6 for details. Keep in mind that the loop bandwidth determines the pull-in range, acquisition time and steady state performance of the loop and frequency locked loops converge much faster than the phase locked loops.

Next, we implement an FLL in GNU Radio for a 16-QAM modulation, a large normalized CFO of -0.2 , a loop bandwidth of $2\pi/250$, excess bandwidth $\alpha = 0.5$ and samples/symbol $L = 4$ for which the results are shown in Figure 6.39. The output constellation is not downsampled to $L = 1$ since the timing has not been recovered yet. Notice that the *compensated frequency offset* is very close to $2\pi(+0.2) = 1.25$ and hence the frequency is seen to converge at the right value. Furthermore, we have a gradually rising phase offset which jumps quickly and rolls over between 0 and 2π for large frequency offsets.

6.5 The Small Picture

Here, we explain a summary of the frequency synchronization in wireless communication systems described in this chapter. With the help of an unsynchronized local oscillator at the Rx, the incoming signal is downconverted to baseband that now contains an arbitrary Carrier Frequency Offset (CFO) F_Δ .

Small CFO: In this case, the Rx signal can be matched filtered and downsampled to $L = 1$ sample/symbol for further processing. Here, the matched filtered complex signal $z(t)$ is the modulated I and Q waveforms rotated by a complex sinusoid of frequency F_Δ . After multiplying with the known symbol conjugate $a^*[m]$ or $\hat{a}^*[m]$ or raising to a power, the residual I and Q consist of that complex sinusoid only. Then, any conventional technique to estimate the frequency of a sinusoid embedded in noise can be applied. This is true not only for the conjugate product estimator and the multiple correlations estimator, even a phase based FED follows the gradually increasing or decreasing angle of this complex sinusoid.

Large CFO: This invites the energy maximization perspective of the matched filtered signal $z(nT_S)$ for which we process the signal before matched filtering in a prescribed manner.

$$\sum_n |z(nT_S)|^2$$

Since energy is just a sum of magnitude squared values of a signal, we will learn in Chapter 7 that squaring a signal in time domain gives rise to convolution in frequency domain which exhibits its peak at the point of maximum overlap, i.e., at the frequency F_Δ . The peak of a squared signal can be found through the product between the signal and its derivative. For example, a derivative frequency error detector, a frequency matched filter and band edge filters generate an error signal based on this principle which guides the frequency locked loop. On the other hand, phase based frequency estimation schemes (such as delay and multiply) implicitly implement this squaring operation to produce a feedforward estimate.

6.6 Appendix

Here, we derive some relations utilized in the main text.

Effect of Carrier Frequency Mismatch for a Small CFO

In the absence of noise, the sampled version of Eq (6.1) becomes

$$\begin{aligned} r(nT_S) &= v_I(nT_S)\sqrt{2}\cos\left[2\pi F_C nT_S + 2\pi F_\Delta nT_S + \theta_\Delta\right] - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left[2\pi F_C nT_S + 2\pi F_\Delta nT_S + \theta_\Delta\right] \\ &= v_I(nT_S)\sqrt{2}\cos\left[2\pi \frac{k_C}{N}n + 2\pi \frac{F_\Delta}{F_S}n + \theta_\Delta\right] - \\ &\quad v_Q(nT_S)\sqrt{2}\sin\left[2\pi \frac{k_C}{N}n + 2\pi \frac{F_\Delta}{F_S}n + \theta_\Delta\right] \end{aligned} \quad (6.46)$$

where the relation $F/F_S = k/N$ is used and k_C corresponds to carrier frequency F_C . To produce a complex baseband signal from the Rx signal, the samples of this waveform are input to a mixer which multiplies them with discrete-time quadrature sinusoids $\sqrt{2}\cos 2\pi(k_C/N)n$ in the I arm and $-\sqrt{2}\cdot \sin 2\pi(k_C/N)n$ for Q arm.

The normalized CFO (nCFO) was defined in Eq (6.3) as

$$F_0 = \frac{F_\Delta}{R_M} = F_\Delta T_M$$

For an Rx operating at L samples per symbol, $F_S = L/T_M$, so

$$\frac{F_\Delta}{F_S} = \frac{F_\Delta}{L/T_M} = \frac{F_0}{L}$$

Next, using the identities $\cos A \cdot \cos B = 0.5 \{\cos(A+B) + \cos(A-B)\}$ and $\sin A \cdot \cos B = 0.5 \{\sin(A+B) + \sin(A-B)\}$ and ignoring the double frequency terms as they are filtered out, the signal in the I rail after downconversion can be written as

$$\begin{aligned} I \rightarrow x_I(nT_S) &= r(nT_S) \cdot \sqrt{2} \cos 2\pi \frac{k_C}{N} n \\ &= v_I(nT_S) \cos \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) - v_Q(nT_S) \sin \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \end{aligned} \quad (6.47)$$

This is the downconverted inphase signal input to the matched filter[†]. The matched filter output is written as

$$\begin{aligned} I \rightarrow z_I(nT_S) &= x_I(nT_S) * p(-nT_S) \\ &= \left\{ v_I(nT_S) \cos \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) - \right. \\ &\quad \left. v_Q(nT_S) \sin \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \right\} * p(-nT_S) \end{aligned}$$

Plugging the expressions for $v_I(nT_S)$ and $v_Q(nT_S)$,

$$\begin{aligned} I \rightarrow z_I(nT_S) &= \sum_i \left\{ a_I[i] p(nT_S - iT_M) \cos \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) - \right. \\ &\quad \left. a_Q[i] p(nT_S - iT_M) \sin \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \right\} * p(-nT_S) \end{aligned}$$

Observe that the terms $\cos 2\pi(F_0/L)n + \theta_\Delta$ and $\sin 2\pi(F_0/L)n + \theta_\Delta$ are a function of n and hence cannot be taken out of the convolution sum between the input and the matched filter response. *The simple pulse based matched filter is not matched anymore!* Only for a very small frequency offset (which is the case we are addressing here)

$$F_\Delta \ll \frac{1}{T_M}, \quad \text{or} \quad F_0 \ll 1,$$

the above equation can be approximated as

$$\begin{aligned} I \rightarrow z_I(nT_S) &\approx \cos \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \sum_i a_I[i] p(nT_S - iT_M) * p(-nT_S) - \\ &\quad \sin \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \sum_i a_Q[i] p(nT_S - iT_M) * p(-nT_S) \end{aligned} \quad (6.49)$$

This is because then cosine and sine do not change much during a symbol interval for small CFO and can be taken out of the convolution expression. Applying the usual concept of pulse

[†]Following a similar line of reasoning, the quadrature part of the downconverted signal is

$$Q \rightarrow x_Q(nT_S) = v_Q(nT_S) \cos \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) + v_I(nT_S) \sin \left(2\pi \frac{F_0}{L} n + \theta_\Delta \right) \quad (6.48)$$

where the double frequency terms are again ignored.

auto-correlation $r_p(nT_S)$,

$$\begin{aligned} I \rightarrow z_I(nT_S) &\approx \cos\left(2\pi\frac{F_0}{L}n + \theta_\Delta\right) \sum_i a_I[i] r_p(nT_S - iT_M) - \\ &\quad \sin\left(2\pi\frac{F_0}{L}n + \theta_\Delta\right) \sum_i a_Q[i] r_p(nT_S - iT_M) \end{aligned}$$

Recall that a small CFO implies that the phase varies little for a symbol duration and hence Nyquist no-ISI criterion is approximately valid, i.e., the auto-correlation of the pulse shape down-sampled at symbol rate $r_p(mT_M)$ is given by

$$r_p(mT_M) \approx \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases} \quad (6.50)$$

Downsampling by L implies $n = mL = mT_M/T_S$,

$$\left. \frac{F_0}{L}n \right|_{n=mL} = F_0m$$

After performing similar steps for the Q arm too, we get

$$\begin{aligned} I \rightarrow z_I(mT_M) &= a_I[m] \cos(2\pi F_0 m + \theta_\Delta) - a_Q[m] \sin(2\pi F_0 m + \theta_\Delta) \\ Q \uparrow z_Q(mT_M) &= a_Q[m] \cos(2\pi F_0 m + \theta_\Delta) + a_I[m] \sin(2\pi F_0 m + \theta_\Delta) \end{aligned}$$

Derivative Frequency Error Detector

Throughout this text, we have avoided rigorous derivations but the purpose of this appendix is to show the process behind the simple expression of the derivative frequency error detector and that the quadrature component is different in the two approaches of Figure 6.22.

Denoting the downconverted signal by $x(nT_S)$, the matched filter by $h_{MF}(nT_S) = p(-nT_S)$ and its output by $z(nT_S)$, we can write

$$z(nT_S) = \sum_l x(lT_S) e^{-j2\pi\hat{F}_\Delta lT_S} h_{MF}(nT_S - lT_S)$$

Differentiating the above expression yields

$$\begin{aligned} \dot{z}(nT_S) &= \frac{d}{d\hat{F}_\Delta} z(nT_S) = \sum_l x(lT_S) (-j2\pi lT_S) e^{-j2\pi\hat{F}_\Delta lT_S} h_{MF}(nT_S - lT_S) \\ &= j2\pi \sum_l x(lT_S) \{(-n + n - l)T_S\} e^{-j2\pi\hat{F}_\Delta lT_S} h_{MF}(nT_S - lT_S) \\ &= -j2\pi nT_S \sum_l x(lT_S) e^{-j2\pi\hat{F}_\Delta lT_S} h_{MF}(nT_S - lT_S) - \\ &\quad \sum_l x(lT_S) e^{-j2\pi\hat{F}_\Delta lT_S} \{ -j2\pi(nT_S - lT_S) h_{MF}(nT_S - lT_S) \} \end{aligned}$$

The above equation can be written as

$$\begin{aligned} \dot{z}(nT_S) &= -j2\pi nT_S \left\{ x(nT_S) e^{-j2\pi\hat{F}_\Delta nT_S} * h_{MF}(nT_S) \right\} - \\ &\quad \left\{ x(nT_S) e^{-j2\pi\hat{F}_\Delta nT_S} * \underbrace{(-j2\pi nT_S \cdot h_{MF}(nT_S))}_{h_{FMF}(nT_S)} \right\} \\ &= -j2\pi nT_S \cdot z(nT_S) - z_{FMF}(nT_S) \end{aligned} \quad (6.51)$$

where $z_{FMF}(nT_S)$ is the output of a *Frequency Matched Filter* defined as

$$h_{FMF}(nT_S) = -j2\pi nT_S \cdot h_{MF}(nT_S) \quad (6.52)$$

While in itself Eq (6.51) does not tell much, we invoke a property of Fourier Transform.

$$-j2\pi t x(t) \xrightarrow{\mathcal{F}} \frac{d}{dF} X(F)$$

where F is the continuous frequency variable and $X(F)$ is the Fourier Transform of $x(t)$. Assume for a moment that the quantities in Eq (6.51) are continuous-time and apply the above property to view what happens in frequency domain.

$$\mathcal{F}\left\{\dot{z}(t)\right\} = \frac{d}{dF} Z(F) - Z_{FMF}(F)$$

Here, $Z(F)$ and $Z_{FMF}(F)$ are the Fourier Transforms of $z(t)$ and $z_{FMF}(t)$, respectively. Also, applying the same property to Eq (6.52), the filter frequency response is

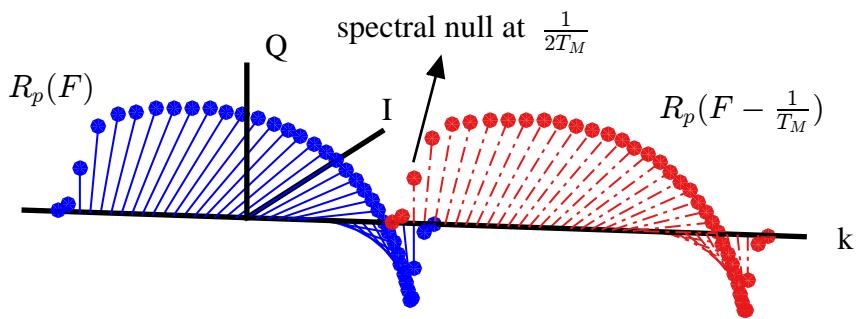
$$H_{FMF}(F) = \frac{d}{dF} H_{MF}(F)$$

which is the frequency domain derivative of our matched filter. What this all means is that in frequency domain, the derivative of the matched filter output with respect to the CFO estimate \hat{F}_Δ is the difference between the derivative of the matched filter output with respect to frequency F and output of a filter whose frequency response is the derivative of the matched filter itself. For a CFO estimate $\hat{F}_\Delta = F_\Delta$, this difference is zero.

Finally, plugging the expression in Eq (6.51) back in the error signal $e_D[n]$ of Eq (6.28), the quadrature component gets eliminated.

Chapter 7

Timing or Clock Synchronization



“Sometimes you are in sync with the times, sometimes you are in advance, sometimes you are late.”

Bernardo Bertolucci

Timing synchronization is one of the most fascinating topics in the field of digital communications. On the bright side, numerous scientists have contributed towards its body of knowledge due to its crucial role in the successful implementation of communication and storage systems. On the not-so-bright side, this knowledge has grown to an extent that it has also become the least understood and puzzling topic in the grand scheme of things. My objective in this chapter is to simplify the problem as well as its solutions in a clear and intelligible manner.

Note 7.1 Heart of a digital Rx

In a digital communications Rx, the timing synchronization plays a similar role as that of a heart in a human body by providing periodic ticks that keep the system running in a coherent manner.

Symbol timing synchronization is also known as symbol *timing recovery*, *clock synchronization* and *clock recovery*. The word *symbol* is very important here because time synchronization is a broader term that can refer to other kinds of synchronization as well such as network synchronization among computer or microcontroller clocks, packet synchronization in telecommunication systems, and so on.

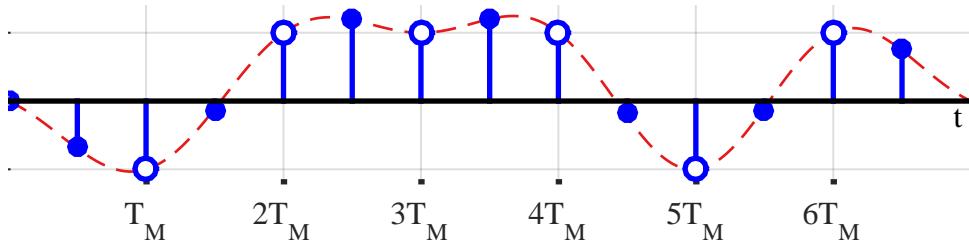
The problems of carrier phase and frequency synchronization are relatively easier to understand. After all, a carrier phase offset is just a one-time rotation – while a carrier frequency offset implies a continuous rotation – of the Rx complex samples. Timing misalignment is a little different and can be explained as follows.

One of the foremost tasks of a digital Rx is to determine the starting point of the useful component in a sampled Rx signal, as many of its initial samples are noise only. It is the job of a frame synchronization unit to determine symbol number 1 in the Rx sequence.

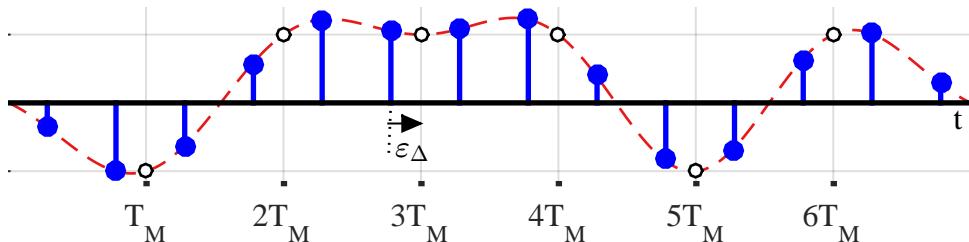
Once the frame synchronization is achieved, the job of the symbol timing synchronization block is to establish the optimal sampling instant within a symbol. The Rx needs to sample the received sequence every T_M seconds at exactly these instants when the eye opening is maximum (points of no ISI from the neighbouring symbols). However, the peak sample at the end of the matched filter output for each symbol duration is *not known in advance* at the Rx. This is illustrated in Figure 7.1 for a signal sampled at 2 samples/symbol.

Figure 7.1a is a simple case of exactly 2 samples/symbol, one of which was at the maximum eye opening. In reality, this peak sample does not coincide at all with a generated sample at the Rx. This is because the ADC just samples the incoming continuous waveform without any information about the symbol boundaries. This is illustrated in Figure 7.1b where ε_Δ is the timing phase offset by which the Rx clock is misaligned in reference to the Tx clock. An efficient mechanism needs to be invoked to construct the *missing samples* at the exact symbol boundaries, shown as black and white dots in Figure 7.1b.

As a first solution, it is possible to transmit both the data and the clock in a Tx waveform (for example, the data at the I channel and the clock signal at the Q channel in a QAM waveform) so that the Rx can produce the samples that are exactly aligned



(a) A Rx sampled the signal at exactly 2 samples/symbol, one of which coincides (fortunately) with the optimal sampling instants (integer multiples of T_M) shown as white circles



(b) Same as above, but with a fractional delay from the optimal instants. The fractional sampling instant ε_Δ within a symbol is known as symbol timing phase offset

Figure 7.1: The meaning of timing synchronization in a sampled sequence

with the symbol boundaries according to the Tx. However, it requires sacrificing the bandwidth and power because the resources dedicated to send the clock signal could be used to send more data as well. To avoid wasting the precious resources in this context, techniques to extract the Tx clock from the noisy received data waveform itself need to be devised.

In the days of analog signal processing and the early years of digital processing, symbol timing synchronization was accomplished by generating a clock signal that is aligned (in both phase and frequency) with the clock used to generate the data at the Tx. This was a Voltage Controlled Clock (VCC) driven by a filtered timing error detector output, just like a Voltage Controlled Oscillator (VCO) driven by a filtered phase error detector output in a PLL. After the proliferation of digital signal processing, this definition is not that helpful due to the fixed rate sampling and interpolation which we will explain in detail in this chapter.

In light of the above discussion, the goal of digital timing synchronization is to produce L samples per symbol at the matched filter output such that one of those samples is aligned with the maximum eye opening. We begin with the effect of symbol timing mismatch between the Rx signal and the local sampling clock on the detection mechanism.

7.1 Effect of Symbol Timing Mismatch

As usual, the effect of a symbol timing offset ε_Δ can be visualized in both time and frequency domains.

Time Domain View

To understand the role of a symbol timing mismatch in time domain, we begin with an M -PAM signal as

$$s(t) = \sum_i a[i] p(t - iT_M)$$

where $a[i]$ is the i^{th} data symbol, $p(t)$ is a square-root Nyquist pulse with support $-LG \leq t \leq LG$ samples, L is the number of samples/symbol, G is the group delay or one-sided pulse length in symbols and T_M is the symbol time. Ignoring every other distortion at the Rx except a *Symbol Timing Offset (STO)* ε_Δ , the received signal $r(t)$ is given as

$$r(t) = \sum_i a[i] p(t - iT_M - \varepsilon_\Delta) \quad (7.1)$$

In the above equation, the range of ε_Δ is defined as a fractional value within a symbol interval T_M .

$$-0.5T_M \leq \varepsilon_\Delta \leq +0.5T_M$$

which is why it is known as a *timing phase*. Note that we are defining the symbol timing offset ε_Δ in units of the symbol time T_M . When there is a need of a *normalized Symbol Timing Offset (nSTO)* in an expression, we will use the notation ε_0 which implicitly carries a factor of T_M within itself in terms of a percentage.

$$-0.5 \leq \varepsilon_0 = \frac{\varepsilon_\Delta}{T_M} \leq +0.5$$

Note 7.2 Frame synchronization

A natural question at this stage can be the following. The form of the expression $p(t - mT_M - \varepsilon_\Delta)$ clearly signifies ε_Δ as the time delay from the moment the Tx constructs the modulated signal to the moment the Rx delivers its sampled version to the baseband unit. The delays in the Tx frontend, the wireless channel and the Rx frontend all can easily account for many symbol intervals, so why is ε_Δ defined as a fraction of T_M ? The answer is that *locating the correct symbol to an integer level* (basically symbol number 1 in a Rx sequence) is the job of the frame synchronization block.

Integer multiples of T_M	\rightarrow	Frame synchronization
Fractional T_M	\rightarrow	Symbol timing synchronization

A frame synchronization unit handles this task through correlation, either of a stored preamble with an incoming preamble or two training sequences within the Rx signal itself.

Now, we examine what happens when the Rx samples this signal in Eq (7.1) through an ADC operating at a rate of $F_S = 1/T_S$ samples/second, the output of which is

$$r(nT_S) = \sum_i a[i] p(nT_S - iT_M - \varepsilon_\Delta)$$

This signal is input to a matched filter $h(nT_S) = p(-nT_S)$.

$$\begin{aligned} z(nT_S) &= \left\{ \sum_i a[i] p(nT_S - iT_M - \varepsilon_\Delta) \right\} * p(-nT_S) \\ &= \sum_i a[i] r_p(nT_S - iT_M - \varepsilon_\Delta) \end{aligned} \quad (7.2)$$

where $r_p(nT_S)$ comes into play from the definition of the pulse auto-correlation, see Eq (3.6). This is plotted in Figure 7.2 for a 25% timing offset where the sampling is not being performed at optimal instants. We will next zoom into this figure for a detailed discussion.

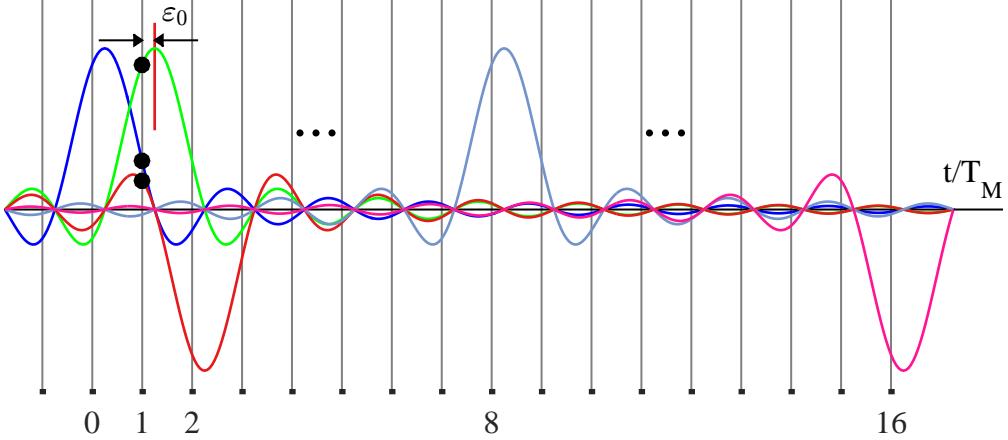


Figure 7.2: Symbol Timing Offset (STO), shown here as 25% of a symbol time T_M , causing ISI in the neighbouring symbols

To generate symbol decisions, T_M -spaced samples of the matched filter output are required at $n = mL = mT_M/T_S$. Downsampling the matched filter output generates

$$\begin{aligned}
 z(mT_M) &= z(nT_S) \Big|_{n=mL=mT_M/T_S} \\
 &= \sum_i a[i] r_p(mT_M - iT_M - \varepsilon_\Delta) = \sum_i a[i] r_p[(m-i)T_M - \varepsilon_\Delta] \\
 &= \underbrace{a[m] r_p(-\varepsilon_\Delta)}_{\text{Term 1}} + \underbrace{\sum_{i \neq m} a[i] r_p[(m-i)T_M - \varepsilon_\Delta]}_{\text{Term 2}}
 \end{aligned} \tag{7.3}$$

The above equation reveals that the optimal samples at the matched filter output can be broken into the following two terms:

Term 1: Desired Signal is given by $a[m]r_p(-\varepsilon_\Delta)$. From Section 3.6, we know that the maximum output of a Nyquist pulse auto-correlation is 1 and it is delivered at instant 0. Therefore, the effect of $r_p(-\varepsilon_\Delta)$ is to reduce the desired symbol estimate $a[m]$ by an amount proportional to the Symbol Timing Offset (STO) ε_Δ .

Term 2: Inter-Symbol Interference (ISI) is the accumulation of contributions from all other symbols in the neighborhood of the current symbol, up to the pulse group delay. This contribution, again, depends on the timing offset ε_Δ .

By zooming in on the terms $i = m - 1$, $i = m$ and $i = m + 1$, we can also write Eq (7.3) as

$$\begin{aligned}
 z(mT_M) &= a[m]r_p(-\varepsilon_\Delta) + \\
 &\quad \cdots + a[m-1]r_p(T_M - \varepsilon_\Delta) + a[m+1]r_p(-T_M - \varepsilon_\Delta) + \cdots
 \end{aligned}$$

The consequences of the above equation can be observed by the shaded region in Figure 7.3[†] which is a zoomed in version of Figure 7.2. Here,

- the current symbol has reduced in amplitude, and
- adjacent two ISI terms can be seen as interfering with the current symbol estimate.

The STO here is $0.25T_M$ which for a sample rate of $4/T_M$ means an offset of 1 sample. I recommend you to carefully look at this figure to understand the interference from the other symbols for an incorrect sampling instant.

The vertical dashed lines illustrate the actual symbol boundaries which is where the samples should be taken. However, due to the timing offset, the contributions from $a[m - 1]$ and $a[m + 1]$ are not zero. The amount of their interference is dictated by this offset while the sign is governed by the modulating data values.

Eye Diagram and Scatter Plot with Timing Offset

When many such scaled pulse shapes are added together to form a Pulse Amplitude Modulation (PAM) waveform, we can see the behavior more clearly through sampling an eye diagram for different timing offsets. Figure 7.4 illustrates an example for three different ε_Δ :

$$\varepsilon_\Delta = 0, 0.25T_M \text{ and } -0.5T_M$$

which we discuss below. At a rate of L samples/symbol, these values of ε_Δ correspond to 0, $L/4$ and $-L/2$ samples, respectively. Obviously, a symbol timing offset can fall anywhere between the samples as well. In this figure, pay close attention to the solid and dotted arrows as they reveal different characteristics for the symbol timing.

$\varepsilon_\Delta = 0$ corresponds to the perfect symbol estimates at the center of the eye diagram, which is at the end of the symbol interval T_M in the actual matched filter output (recall from Section 3.8 that an eye diagram is drawn within an interval $-0.5T_M \leq t \leq +0.5T_M$). This is where the eye opening on average is at its maximum and no ISI from the neighboring symbols distorts this optimal sample. A timing error samples the eye diagram at some other instant.

$\varepsilon_\Delta = -0.5T_M$ corresponds to the maximum symbol timing offset a digital system can have. In the eye diagram, it is the left edge of the interval, as drawn in Figure 7.4, or the middle of the symbol interval T_M in an actual symbol stream. Any ε_Δ lesser than that will come around from the right side of the eye diagram (and the integer 1 will be deleted from the integer output of the frame synchronization block). For example, an offset of $\varepsilon_\Delta = -0.6T_M$ will appear as $+0.4T_M$. Without any aid from the frame synchronization block, we would have encountered a cycle slip.

In Section 3.8, it was shown that tracing a single transition in an eye diagram gives information about 3 symbols: Past, NOW and Future. For this reason, a different sequence of +1s and -1s carves a different path on the eye diagram. This is the reason an eye diagram has different traces coming to the same +1 value: coming from top indicates the previous symbol being +1 and middle for previous symbol being -1. The traces leaving a +1 value follow a similar logic:

[†]If you feel a bit confused about $r_p(\pm T_M - \varepsilon_\Delta)$, refer to Section 1.2.

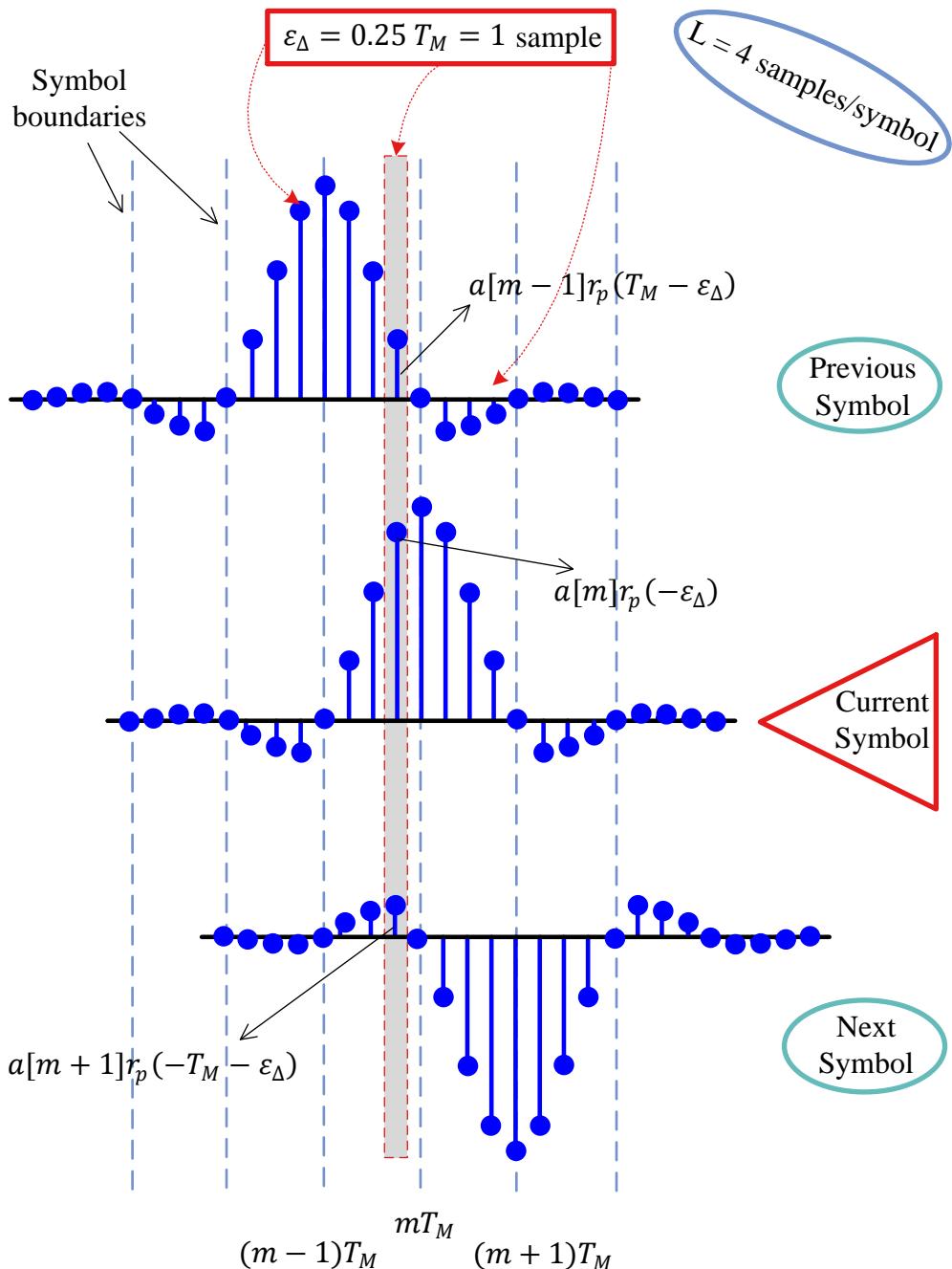


Figure 7.3: Symbol Timing Offset (STO), shown here as 25% of a symbol time T_M or 1 sample for $L = 4$ samples/symbol, manifests itself as (1) a reduction in desired symbol amplitude, and (2) ISI from neighboring symbols

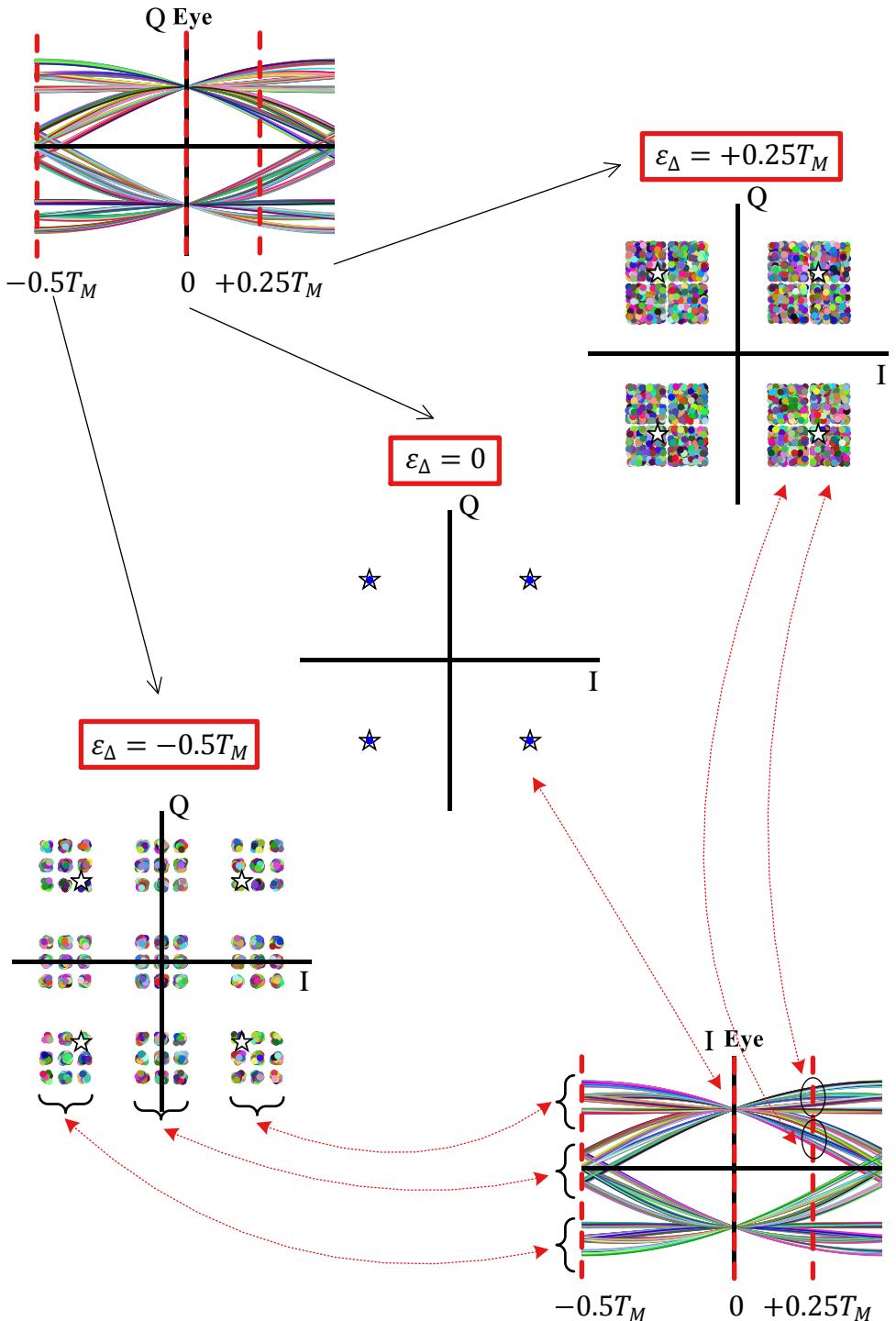


Figure 7.4: Effect of Symbol Timing Offset (STO) on symbol estimates for $\varepsilon_\Delta = 0$, $+0.25T_M$ and $-0.5T_M$. At a rate of L samples/symbol, these ε_Δ correspond to 0 , $L/4$ and $-L/2$ samples, respectively

Table 7.1: Analogy between carrier phase offset θ_Δ and symbol timing offset ε_Δ

	Carrier Phase Offset θ_Δ	Symbol Timing Offset ε_Δ
Minimum value	$-\pi$	$-0.5T_M$
Maximum value	$+\pi$	$+0.5T_M$
Unique interval	2π	T_M

leaving towards the top indicates the next symbol being +1 and middle for next symbol being -1.

But in Figure 7.4, another three distinct routes can be identified at the bottom of Q Eye from a -1 to a -1 transition. This is due to the sequence values coming before the previous symbol. So a +1, -1, -1 will follow a slightly different path as compared to a -1, -1, -1, and so on. Considering a similar contribution arriving from the I branch at the top, middle and bottom, a total of $3 \times 3 = 9$ modulation points for each constellation point arise in the scatter plot.

Finally, notice that the magnitude of these points in the scatter plot tends to be greater than the ideal constellation point denoted by a star. This can be recognized by looking at the eye diagram at $-0.5T_M$ where two out of three paths have a higher magnitude than the ideal point at the sampling instant.

$\varepsilon_\Delta = +0.25T_M$ corresponds to sampling at the quarter symbol interval on the right half of the eye diagram. Two distinct patterns can be distinguished in the scatter plot, which correspond to the eye diagram, say for example, out of a +1 value, those heading towards the top and those heading towards the middle. A total of $4 = 2 \times 2$ distinct regions for each ideal constellation point in the scatter plot arise due to a similar contribution from I arm (indicated by the ellipses and the arrows).

For this region, scatter plot shows lesser magnitude on average than the ideal constellation point. This is again evident from the corresponding eye diagram.

The above study tells us an interesting analogy between the carrier phase offset and symbol timing offset. Just like the phase is defined between $-\pi$ to $+\pi$ and everything else is wrapped around, a symbol timing offset is defined between $-0.5T_M$ and $+0.5T_M$ and everything else wraps around. On that account, T_M is to symbol timing what 2π is to carrier phase. This analogy is documented in Table 7.1.

Note 7.3 The middle sample

In Figure 7.4, it is evident that while most symbol timing offsets produce a random spread around the constellation point, the sample in the middle of the symbol interval T_M corresponding to $\varepsilon_\Delta = \pm 0.5T_M$ has some structure in it. *This middle sample is of paramount significance* when we shortly discuss symbol timing synchronization procedures operating at $L = 2$ samples/symbol.

Frequency Domain View

Until now, the explanation rotated around the time domain. To see what effect it has in the frequency domain, first consider a square-root Nyquist pulse shaping filter alone instead of a shaped data sequence. Since this pulse shape is a real and even signal in time domain, it is also a real and even signal in frequency domain. Recall from Section 1.8 that a right shift of n_0 samples in a time series changes the phase of its DFT by $-2\pi(k/N)n_0$ for each k .

$$\begin{aligned} \text{Time shift } s[(n - n_0) \bmod N] &\longrightarrow -2\pi \frac{k}{N} n_0 \quad \text{Phase shift} \\ &\text{for each } k = -N/2, \dots, -1, 0, 1, \dots, N/2 - 1 \end{aligned}$$

For a continuous-time case, this is given by

$$\text{Time shift } s(t - \varepsilon_\Delta) \longrightarrow -2\pi F\varepsilon_\Delta \quad \text{Phase shift} \quad (7.4)$$

Based on this concept, this was shown in Eq (1.61) as

$$\begin{aligned} \text{Shift in one domain} &\longrightarrow \text{Multiplication by a complex sinusoid} \\ &\text{in the other domain} \end{aligned} \quad (7.5)$$

i.e., the DFT is multiplied with a frequency domain complex sinusoid as ε_Δ is fixed and the frequency varies. We conclude that a phase shift of $-2\pi F\varepsilon_\Delta$ represents multiplication in frequency domain of the signal with a complex sinusoid of inverse period ε_Δ (inverse period by definition is frequency; however, we are already in frequency domain and hence this expression).

We know that in time domain, a signal mixed (multiplied) with a complex sinusoid of a certain frequency causes a spectral shift in frequency domain (this is how spectrum is shared worldwide, e.g., a garage door opener works at a frequency of around 400 MHz and our WiFi modem at 2.4 GHz). *Owing to time frequency duality, a shift in time domain implies multiplication with a complex sinusoid in frequency domain.*

Armed with this knowledge, it is easy to see that a pulse shaping filter gets rotated by $-2\pi F\varepsilon_\Delta$ in frequency domain and the resampling must compensate for this rotation by adjusting its phases in the opposite direction. This is plotted in Figure 7.5. The overall result of the frequency response product of this and the phase adjusted matched filter is that their phases get added and hence canceled out, leaving everything to appear on I -axis (which is the core concept of correlation as we saw in Chapter 1).

The figure for a whole data sequence would be too complicated to draw. However, you can imagine the view with the help of Figure 7.5 and linearity of the amplitude scaling by the modulator.

Next, we extend this concept in relation to the sample rate F_S . This is because the final decision device (threshold detector) operates at a symbol rate $R_M = 1/T_M$, i.e., T_M -spaced samples of the matched filter output are required to generate symbol decisions and our final target is to investigate what happens when we directly sample the signal at $F_S = 1/T_M$. As before, the pulse shape suffices to study this effect and a whole modulated sequence is not needed.

$L = 2$ samples/symbol: For this purpose, we start with the sample rate F_S equal to L samples/symbol where $L \geq 2$. Figure 7.6a draws the spectrum of a

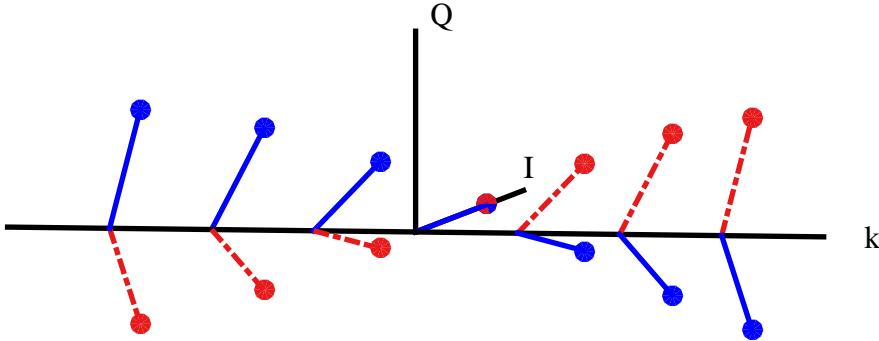


Figure 7.5: In frequency domain, the square-root Nyquist pulse gets phase shifted by $-2\pi(k/N)\varepsilon_\Delta$. To compensate, a matched filter should exhibit an opposite phase

Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ which is sampled at $F_S = 2/T_M$, or 2 samples/symbol. The replicas arising as a result of sampling are naturally $2/T_M$ apart and hence produce spectral “holes” here that cannot be filled regardless of the spectral shape. Recall from Nyquist no-ISI criterion in frequency domain here that the cumulative spectrum as in Eq (3.25) must be a constant function of frequency from $-\infty$ to $+\infty$.

$$\sum_{i=-\infty}^{\infty} R_p(F + \frac{i}{T_M}) = \dots + R_p(F + \frac{1}{T_M}) + R_p(F) + R_p(F - \frac{1}{T_M}) + \dots = T_M \quad (7.6)$$

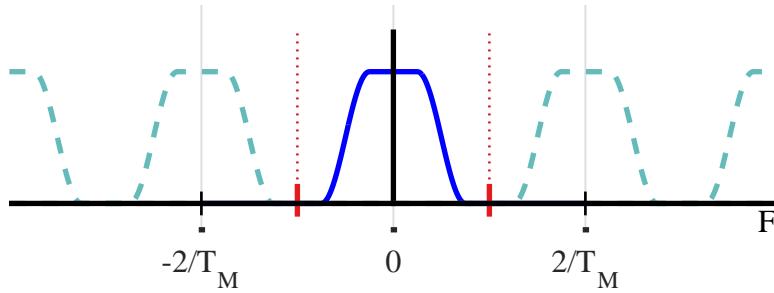
Thus, ISI can only be avoided at $L = 1$ sample/symbol which we explore next.

$L = 1$ sample/symbol, $\varepsilon_\Delta = 0$: Now we draw the same spectrum but for a sample rate of $F_S = 1/T_M$, or 1 sample/symbol but no timing offset in Figure 7.6b. The cumulative spectrum is flat because there is no rotation in frequency domain arising from ε_Δ . Thus, Nyquist no-ISI criterion is satisfied and the resulting symbol-spaced samples in time domain will exhibit no ISI.

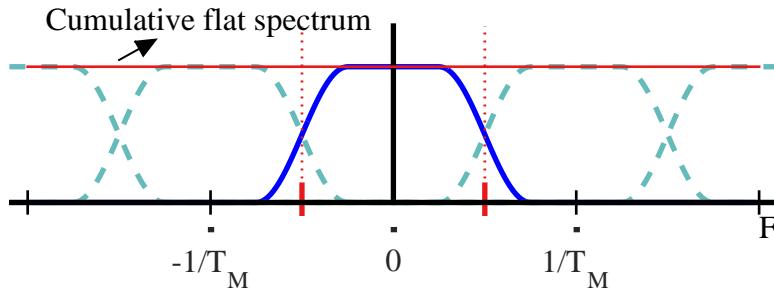
$L = 1$ sample/symbol, $\varepsilon_\Delta \neq 0$: Finally, for $F_S = 1/T_M$, or $L = 1$ sample/symbol, but with a timing offset of $\varepsilon_\Delta = 0.2T_M$, Figure 7.6c illustrates a slight dip in the cumulative spectrum. Nyquist no-ISI criterion is violated as a result. The descent gets deeper with increasing ε_Δ until it reaches zero at $\varepsilon_\Delta = 0.5T_M$. We conclude that a timing shift of ε_Δ puts the spectrum of the pulse auto-correlation $R_p(F)$ and its $1/T_M$ shifted version $R_p(F \pm 1/T_M)$ out of phase. As a consequence, the spectrum reduces in amplitude over the portion of frequency overlap, developing into a notch as ε_Δ approaches $0.5T_M$.

Mathematically, we can see why there is a reduction in spectral amplitude around $1/2T_M$. From Eq (7.6), the spectrum between 0 and $1/T_M$ after sampling is given by

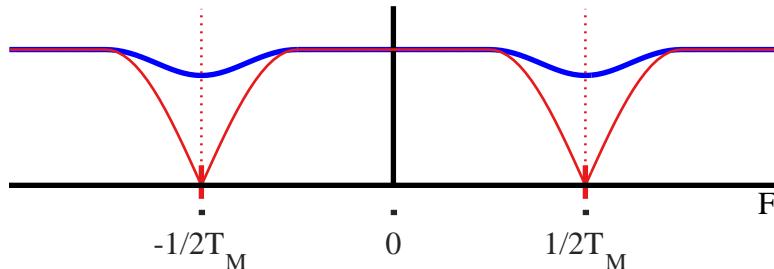
$$R_p(F) + R_p\left(F - \frac{1}{T_M}\right) \quad 0 \leq F \leq \frac{1}{T_M}$$



(a) Spectrum of a Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ sampled at $F_S = 2/T_M$, or 2 samples/symbol



(b) Spectrum of a Raised Cosine pulse and its $1/T_M$ -shifted versions with excess bandwidth $\alpha = 0.5$ sampled at $F_S = 1/T_M$, or 1 sample/symbol where $\varepsilon_\Delta = 0$



(c) Cumulative spectrum of a Raised Cosine pulse and its $1/T_M$ -shifted versions with excess bandwidth $\alpha = 0.5$ sampled at $F_S = 1/T_M$, or 1 sample/symbol where $\varepsilon_\Delta = 0.2T_M$ (thick line) and $\varepsilon_\Delta = 0.5T_M$ (thin line), the latter of which creates a null at $\pm 1/2T_M$

Figure 7.6: Effect of ε_Δ on the cumulative spectrum for $F_S = 1/T_M$, or $L = 1$. Clearly, Nyquist no-ISI criterion is violated unless $\varepsilon_\Delta = 0$

Since the pulse is real and even, the pulse auto-correlation $r_p(nT_S)$ has a real and even Fourier Transform $R_p(F)$ with zero phase, see Section 2.9. Then, a delay of ε_Δ injects the following phase shift according to Eq (7.4).

$$\text{Phase shift in } R_p(F) = -2\pi F \varepsilon_\Delta \quad (7.7)$$

However, the phase shift in its replica at $1/T_M$ is

$$\begin{aligned}\text{Phase shift in } R_p \left(F - \frac{1}{T_M} \right) &= -2\pi \left(F - \frac{1}{T_M} \right) \varepsilon_\Delta \\ &= -2\pi F \varepsilon_\Delta + 2\pi \frac{1}{T_M} \varepsilon_\Delta\end{aligned}\quad (7.8)$$

The term $2\pi(1/T_M)\varepsilon_\Delta$ for $\varepsilon_\Delta = 0.5T_M$ becomes

$$2\pi \frac{1}{T_M} \cdot 0.5T_M = \pi$$

Hence, Eq (7.8) yields

$$\text{Phase shift in } R_p \left(F - \frac{1}{T_M} \right) = -2\pi F \varepsilon_\Delta + \pi$$

which is exactly opposite to that of the original spectrum $R_p(F)$, see Eq (7.7). A phase shift of π is the same as multiplication by -1 . Hence, the cumulative spectrum *for* $\varepsilon_\Delta = 0.5T_M$ is given by

$$R_p(F) - R_p \left(F - \frac{1}{T_M} \right) \quad 0 \leq F \leq \frac{1}{T_M}$$

This subtraction of spectral components reduces their amplitudes. Particularly, at $F = 1/2T_M$, the phase shift of $R_p(F)$ is

$$-2\pi F \varepsilon_\Delta = -2\pi \frac{1}{2T_M} \cdot 0.5T_M = -\frac{\pi}{2}$$

while the phase shift of $R_p(F - 1/T_M)$ is

$$-2\pi F \varepsilon_\Delta + \pi = -2\pi \frac{1}{2T_M} \cdot 0.5T_M + \pi = -\frac{\pi}{2} + \pi = +\frac{\pi}{2}$$

This $\mp\pi/2$ phase relationship is more clearly drawn in Figure 7.7 for a small excess bandwidth $\alpha = 0.1$ for clarity. Their addition generates the spectral null you saw in Figure 7.6c. Not drawn in this figure is the alias at the negative side for clear illustration. Notice here that more spectral cancellation happens for large excess bandwidths and less cancellation for small excess bandwidths. Due to this reason, symbol timing strategies operating at $L = 1$ sample/symbol usually exhibit better performance for low excess bandwidths.

7.2 The Big Picture

A few critical details regarding the timing synchronization are in order which provide a wider picture of the timing synchronization problem in a digital communication system.

Timing compensation A major difference between carrier and timing offsets is that compensating for a carrier offset is very simple. One just needs to de-rotate

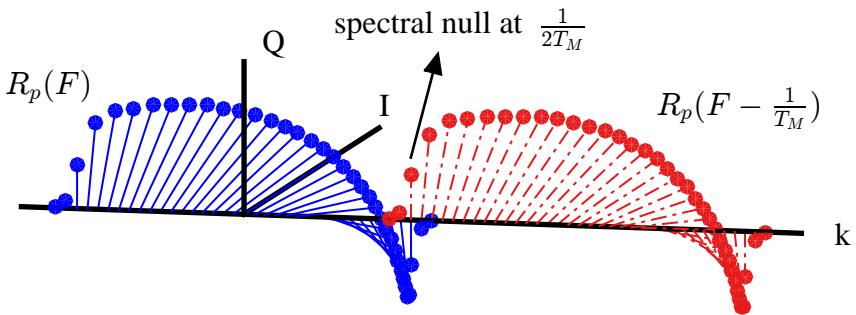


Figure 7.7: Spectrum of the Nyquist pulse $R_p(F)$ and its $1/T_M$ shifted version $R_p(F - 1/T_M)$ exhibit a spectral null at $F = 1/2T_M$ for $\epsilon_\Delta = 0.5T_M$. Excess bandwidth $\alpha = 0.1$ is chosen for clarity

(clockwise rotation) the complex samples by a constant angle to compensate for the carrier phase offset while this de-rotation needs to be continuous for adjusting a carrier frequency offset. This can essentially be viewed as a complex multiplication.

On the other hand, after a fixed rate sampling process, a Symbol Timing Offset (STO) needs re-computation of the signal values at optimal instants (maximum eye opening) as shown in Figure 7.8. This is a task handled by an interpolation block that is discussed in Section 7.8.

Timing phase and frequency offsets Just like a carrier phase and frequency offset, the clock used to sample the incoming continuous-time signal at a rate F_S contains a phase and frequency offset as compared to the Tx clock as well. However, there is a difference between the treatment of timing phase and frequency offsets as compared to the carrier phase and frequency offsets.

As discussed in Section 6.1, the accuracy of the local oscillators in communication receivers is defined in terms of ppm (parts per million) such as 1 ppm is 1 out of 10^6 parts. For example, with $F_C = 1.9$ GHz and ± 20 ppm crystals,

$$\pm \frac{20}{10^6} \times 1.9 \times 10^9 = \pm 38 \text{ kHz}$$

For the purpose of typical wireless communication systems operating at several hundreds of MHz or GHz of carrier frequency F_C , it is one of the major sources of signal distortion.

On the other hand, typical symbol rates in a wireless system are on the order of several MHz (some wireless systems are breaking into GHz symbol rates now). Consider an example of a wireless system operating at a symbol rate of 5 MHz and with ± 20 ppm crystals. The maximum deviation of the timing frequency at the Tx or Rx can be

$$\pm \frac{20}{10^6} \times 5 \times 10^6 = \pm 100 \text{ Hz}$$

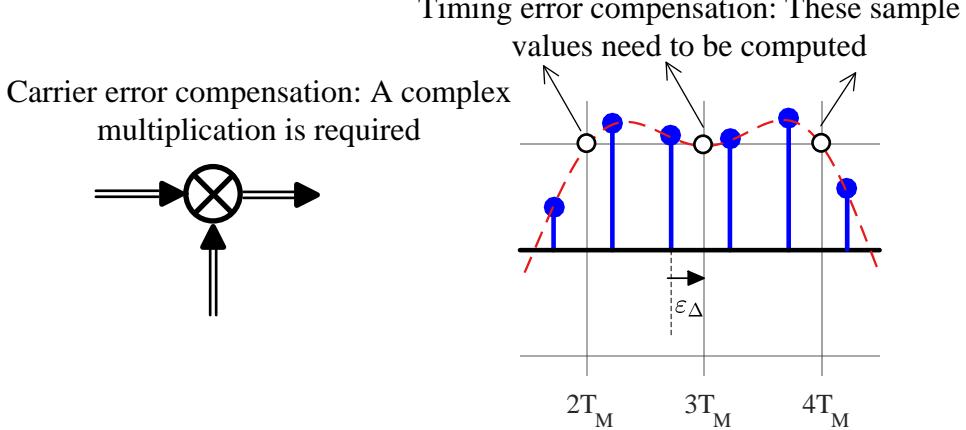


Figure 7.8: For carrier compensation, a complex rotation or multiplication is required while for timing error compensation, new sample values at optimal instants need to be computed by using the neighbouring samples

Consequently, a symbol timing frequency offset poses much less of a distortion as compared to a carrier frequency offset. Nonetheless, it is necessary to compensate in most wireless systems to match the Tx symbol rate due to extra samples injected into or deleted from the symbol stream.

Mode of operation In the phase synchronization problem we categorized the techniques as data-aided and non-data-aided, while the discussion in carrier frequency synchronization revolved around timing-aided and non-timing-aided scenarios. Here, both non-data-aided and data-aided (or decision-directed) techniques are essential where the former can be used for the acquisition stage while the latter during the tracking.

Moreover, both feedforward and feedback techniques are possible in this context although feedback timing recovery schemes are far more popular due to the underlying random variation in the clock and a need for regular computations of the optimal values.

Modulation type During the discussion on carrier phase and frequency synchronization, the focus was on passband linear modulation schemes such as Phase Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM). Here, we will concentrate on baseband schemes such as Pulse Amplitude Modulation (PAM) because it is easier to understand symbol timing synchronization in that context. Since symbol timing errors affect both I and Q arms in a similar manner, it is straightforward to extend the concepts learned for PAM to PSK and QAM. Nevertheless, some implications in passband systems need to be understood when symbol timing and carrier phase need to be jointly estimated.

7.3 Enter the Correlation

So far, we have applied *the Master Algorithm*, namely the correlation, to derive the matched filter for optimal detection and estimators for carrier phase synchronization as well as carrier frequency synchronization. Here, we apply the principle of maximum correlation for solving the timing synchronization problem.

Ignoring all other distortions, the Rx signal $r(t)$ is contaminated by a timing offset ε_Δ only. This signal is sampled at the Rx at a rate of $1/T_S$ samples each second to yield the discrete-time signal $r(nT_S)$. In a feedback system like a Timing Locked Loop (TLL), we are looking for a Timing Error Detector (TED) that maximizes the correlation of the Rx signal $r(nT_S)$ with an expected signal template. In the current scenario of a timing shift, this expected signal template $s(nT_S)$ is simply given as

$$s(nT_S) = \sum_{m=0}^{N_0-1} a[m] p(nT_S - mT_M - \varepsilon_\Delta)$$

where N_0 is the number of available symbols. The Rx signal is

$$r(nT_S) = s(nT_S) + \text{noise}$$

Here, $a[m]$ is the m^{th} PAM symbol, $p(nT_S)$ are the samples of a square-root Nyquist pulse with support $-LG \leq n \leq LG$ samples, L is the number of samples/symbol, G is the group delay or one-sided pulse length in symbols and T_S and T_M are the sample time and symbol time, respectively, that bear the relation $L = T_M/T_S$ samples/symbol.

The correlation between $r(nT_S)$ and its expected template $s(nT_S)$ is defined as

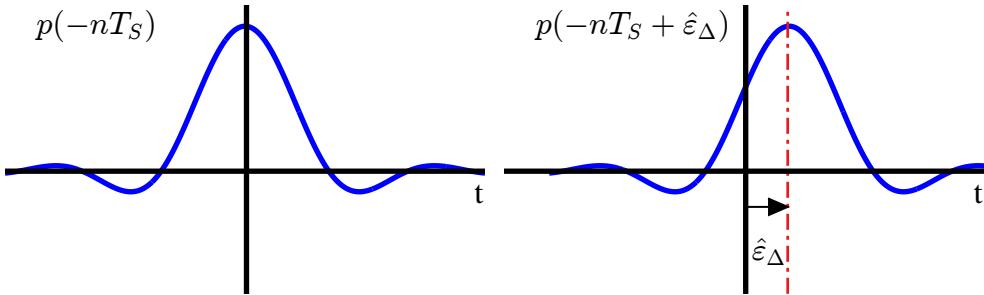
$$\begin{aligned} \text{corr}[j] &= r(nT_S) \diamond s(nT_S) \\ &= \sum_{n=-\infty}^{\infty} r(nT_S) s^*(nT_S - j) \end{aligned}$$

Simplifications come in the above equation by utilizing the facts that $j = 0$ is determined through the frame synchronization block and both the PAM modulated data symbols $a[m]$ and the pulse shape $p(nT_S)$ are real with zero quadrature parts. Replacing ε_Δ with its estimate $\hat{\varepsilon}_\Delta$ in our signal template $s(nT_S)$, we can write

$$\begin{aligned} \text{corr}[0] &= \sum_n r(nT_S) s^*(nT_S) \\ &= \sum_n r(nT_S) \cdot \sum_{m=0}^{N_0-1} a[m] p(nT_S - mT_M - \hat{\varepsilon}_\Delta) \end{aligned}$$

Now the above equation can be modified as

$$\begin{aligned} \text{corr}[0] &= \sum_{m=0}^{N_0-1} a[m] \sum_n r(nT_S) p\{-(mT_M + \hat{\varepsilon}_\Delta - nT_S)\} \\ &= \sum_{m=0}^{N_0-1} a[m] \underbrace{\sum_n r(nT_S) h_{MF}(mT_M + \hat{\varepsilon}_\Delta - nT_S)}_{\text{Matched filter output sampled at } mT_M + \hat{\varepsilon}_\Delta} \end{aligned} \quad (7.9)$$



(a) A square-root Nyquist pulse $p(-nT_S)$ as a matched filter (b) Matched filter with a delay of $\hat{\epsilon}_\Delta$ to match the timing offset ϵ_Δ

Figure 7.9: Resampling the matched filter output at $mT_M + \hat{\epsilon}_\Delta$ is the same as filtering with $p(-nT_S + \hat{\epsilon}_\Delta)$

Here, the matched filter output $z(mT_M + \hat{\epsilon}_\Delta)$ is the convolution of the Rx signal $r(nT_S)$ with the matched filter $h_{MF}(nT_S) = p(-nT_S)$ shifted in time by $\hat{\epsilon}_\Delta$ and is defined as[†]

$$z(mT_M + \hat{\epsilon}_\Delta) = \sum_n r(nT_S) p\left\{ - (mT_M + \hat{\epsilon}_\Delta - nT_S) \right\} \quad (7.10)$$

Recall that carrier frequency synchronization in Section 6.3.1 lead us to compensate for the CFO first and matched filter second. On a similar note, here we could define a filter that compensates for timing first and then match filter the Rx signal. Combined in one filter, this modified matched filter could be a pulse shaping filter shifted from the center by an offset $\hat{\epsilon}_\Delta$, as illustrated in Figure 7.9b. However, when we have the option of resampling the regular matched filter output $z(nT_S)$ at time instant $mT_M + \hat{\epsilon}_\Delta$, we take this more straightforward route and discuss this resampling strategy in detail in Section 7.8 on interpolation.

In light of the above discussion, the correlation result is given from Eq (7.9) by

$$\text{corr}[0] = \sum_{m=0}^{N_0-1} a[m] \cdot z(mT_M + \hat{\epsilon}_\Delta) \quad (7.11)$$

where the output of the matched filter needs to be sampled at $mT_M + \hat{\epsilon}_\Delta$ instead of mT_M . *This is the fundamental equation for timing synchronization problem*, the significance of which cannot be overestimated. Most timing strategies start from exploiting this equation or its variant into a workable digital circuit.

The main problem now is that we do not know the value of $\hat{\epsilon}_\Delta$ which we solve in subsequent sections. An interesting differentiating feature here is as follows.

Carrier phase synchronization In the case of carrier phase synchronization in Chapter 5, we mainly categorized the techniques in data-aided and non-data-aided

[†]In a realistic scenario of joint timing and phase synchronization, the absolute value of this expression is computed that leads to a carrier phase independent timing recovery. After recovering the timing, the carrier phase can be recovered as discussed before. This reduces this 2-dimensional search to a 1-dimensional case.

kinds. These are the two logical options because in digital modems, carrier phase is usually synchronized at the end of signal processing chain at 1 sample/symbol when the timing has been recovered.

Carrier frequency synchronization For carrier frequency synchronization in Chapter 6, a CFO can be large or small which lead to the main categorization as being non-timing-aided and timing-aided. For a large CFO, digital processing of the incoming signal must be started without any knowledge of the symbol timing. On the other hand, timing can be acquired first from an oversampled and slowly rotating constellation when the CFO is small.

Symbol timing synchronization In symbol timing synchronization, there is little need for such kinds of classifications. In general, the nature of the timing problem fits well in a feedback setup and decision-directed techniques are of significance due to the continual need to follow the slowly varying Tx symbol clock.

Having said that, for a short packet in burst mode, the timing offset ε_Δ is almost constant throughout the duration of the packet. Here, a feedforward strategy can be adopted through block processing of the whole symbol stream, where the primary task is to develop an expression for the timing estimator $\hat{\varepsilon}_\Delta$. Then, this estimate can be treated as the unknown original timing offset to be compensated through interpolating the samples.

7.3.1 Feedforward: Brute Force Estimator

Notice that the timing offset ε_Δ is hidden in the samples $z(mT_M + \hat{\varepsilon}_\Delta)$ and therefore there is no direct strategy that maximizes the correlation in Eq (7.11) to yield a closed-form expression for the estimator. Despite this fact, this expression derived through the correlation principle will guide us to many practical timing estimators, which we will explore later.

One viable method is to test a set of N parameters

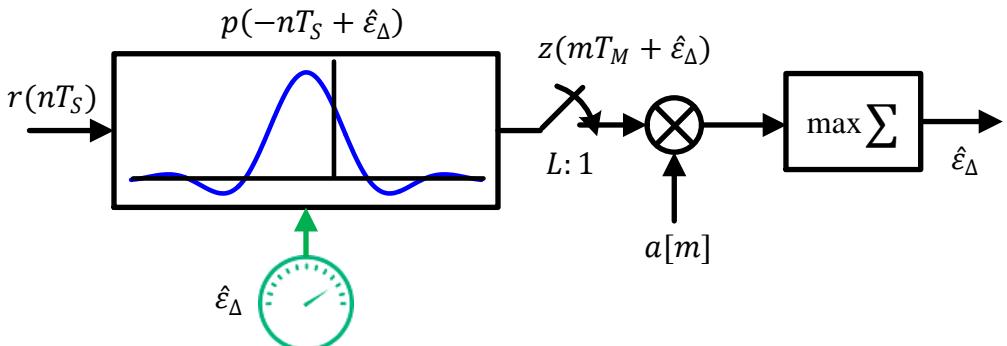
$$\hat{\varepsilon}_{\Delta,1}, \hat{\varepsilon}_{\Delta,2}, \dots, \hat{\varepsilon}_{\Delta,N},$$

by forming a set of N matched filters, computing the correlation Eq (7.11) for each of them and choosing the one that maximizes the correlation sum. Referring to Figure 7.10, the two possible approaches are

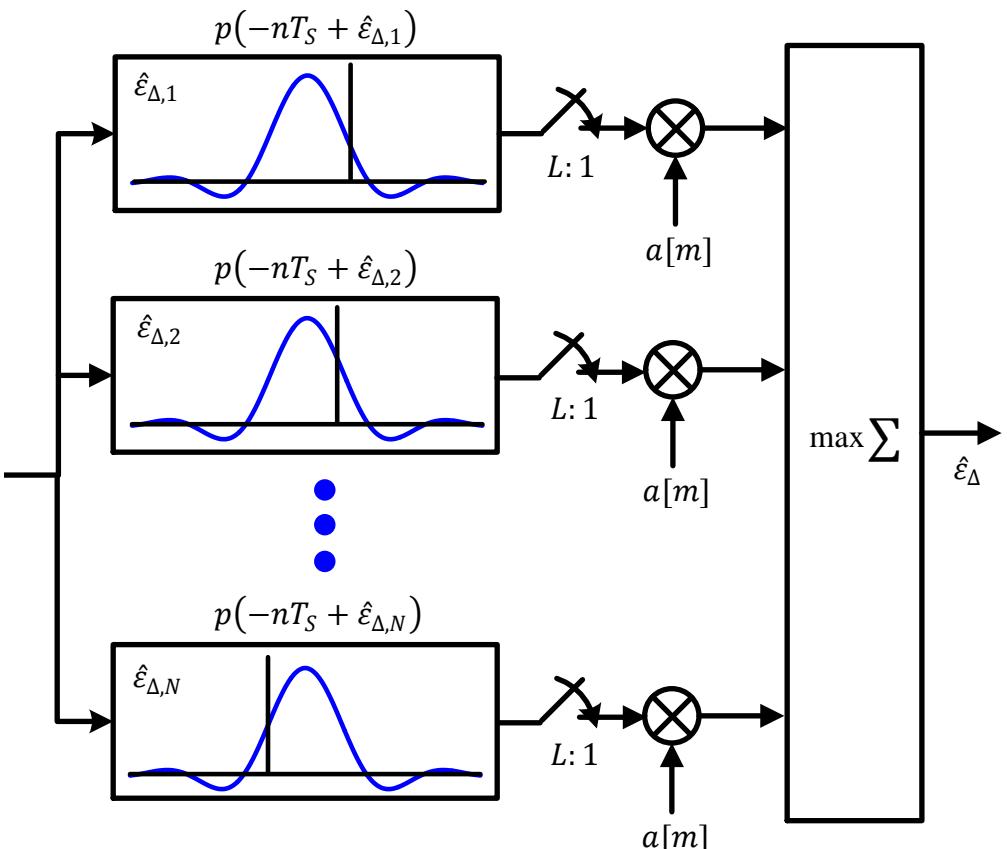
- a serial implementation shown in Figure 7.10a where more operations need to be performed in the same amount of time, or
- a parallel implementation shown in Figure 7.10b where multiple candidates are tested by executing the algorithm in parallel for each of them.

Thanks to the Moore's law, the brute force timing error estimator can be employed in modern high speed digital modems. Nevertheless, its high complexity still necessitates the need to look for more practical estimators. Interestingly, polyphase clock synchronization studied in Section 7.13 works on a similar principle of testing a parallel set of candidate timing offsets but elegantly bypasses its computational complexity problem.

We discuss the prevalent methods for this purpose next.



(a) Serial implementation of the timing estimator



(b) Parallel implementation of the timing estimator

Figure 7.10: A serial and a parallel approach to the feedforward brute force estimator

7.4 Why Squaring is Fundamental to Timing Synchronization

The *fundamental problem of synchronization* – first encountered in Section 5.2 – is revisited here in a different context.

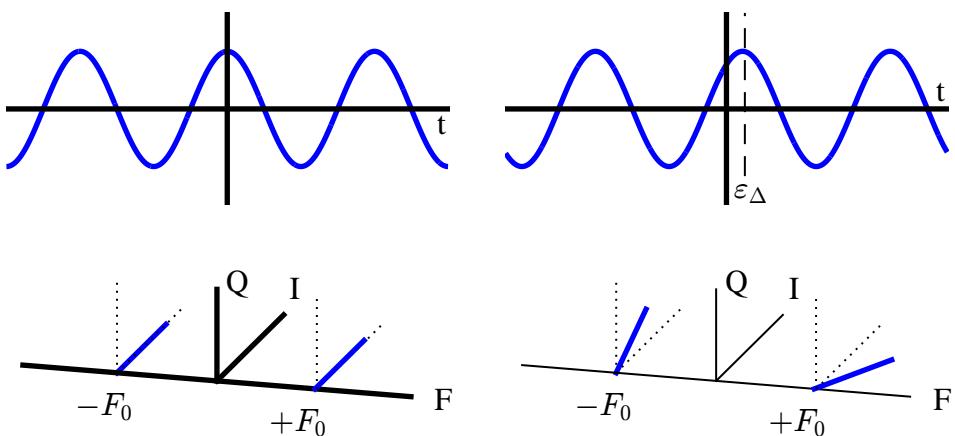
We start with the observation that a sequence of +1s and -1s makes a perfect clock alternating between a high and a low value. Imagine a 2-PAM signal consisting of a random binary sequence. It is input directly into a Timing Locked Loop (TLL) designed to estimate and compensate for the phase and frequency of the timing clock. The TLL will try to lock onto the timing phase within a specific duration. However, *this ‘clock’ does not alternate at every symbol boundary due to the modulating data* (when there is no data transition due to consecutive +1s or consecutive -1s), never allowing our mechanism to converge. This is the fundamental problem a timing synchronization system needs to address.

7.4.1 The Sinusoid Perspective

This is a very interesting story that begins with a simple sinusoid.

A Simple Sinusoid

To understand the operation of timing recovery, consider the most basic of signals: a sinusoid. Recall from Section 1.4 that a cosine in time domain is represented by two impulses in frequency domain as drawn in Figure 7.11a. Here, a zero phase at both impulses needs our attention. Now what happens when this sinusoid is shifted in time by ε_Δ ? According to Figure 7.11b, a specific phase shift appears in one such impulse while the same phase shift but with an opposite sign occurs in the other impulse. Let us find out why.



(a) A cosine wave with zero delay in time

(b) A cosine wave with delay ε_Δ

Figure 7.11: Effect of a timing shift ε_Δ in frequency domain

Owing to the relation between time shift and phase shifts encountered in Eq (7.4), a time shift ε_Δ in this cosine manifests itself as a rotation of $2\pi(k/N)\varepsilon_\Delta$ in those impulses. For a continuous-time case, this rotation becomes $2\pi F\varepsilon_\Delta$. Thus, the phases of the frequency domain impulses depend on the value of the time shift. To see this, Figure 7.11b was drawn with the following parameters.

$$F_0 = 100, \quad F_S = 1000, \quad \varepsilon_\Delta = -1$$

The value of $\varepsilon_\Delta = -1$ here signifies a delay or a right time shift. Therefore, the phase shifts for the impulses at $\pm F_0$ in this figure are given by

$$\pm 2\pi \frac{F_0}{F_S}(-1) \cdot \frac{180^\circ}{\pi} = \mp 36^\circ$$

where we have used the relation $k/N = F/F_S$. As a sanity check, for a delay of a quarter period, i.e., $\varepsilon_\Delta = T_0/4 = 1/4F_0$, the phase shift becomes

$$-2\pi F_0 \varepsilon_\Delta = -2\pi F_0 \frac{1}{4F_0} = -\frac{\pi}{2}$$

and the cosine wave becomes a sine wave.

To appreciate this point, recall from Eq (7.5) that a shift in time domain manifests itself as a multiplication with a complex sinusoid in frequency domain. The spectrum of a time shifted sinusoid in Figure 7.11b is redrawn in Figure 7.12 where the multiplication of the two spectral impulses with a complex sinusoid *in frequency domain* is clearly shown. A sinusoid in time lives at spectral impulses $\pm F_0$ and hence this spectral product just ‘samples’ this rotating complex sinusoid at those two frequencies only.

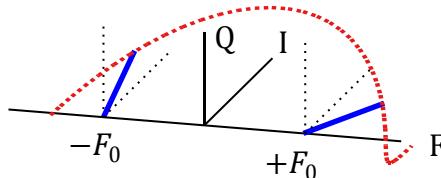


Figure 7.12: Two spectral impulses ‘sample’ the frequency domain complex sinusoid and get shifted by opposite phases

Clearly, if such a sinusoid is an input to our timing recovery device, *the phase of the sinusoid yields the desired timing offset $\hat{\varepsilon}_\Delta$* . This is what some communication Rx used to get when a clock signal was transmitted along with the modulated signal in a separate band. However, it costs bandwidth and power resulting in an inefficient utilization of resources. Then, the engineers figured out better methods for clock synchronization that extract the Tx clock from the Rx signal itself.

At this stage, a reader might think that a time domain description of the delay or advance such as at the top of Figure 7.11 is enough for the purpose of understanding and a frequency domain view was not necessarily required. Nevertheless, as

As we advance through this chapter, we will see the role of spectral impulses for timing synchronization from both a historical and a present-day perspective. You will find answers to interesting questions such as

- how can a ‘phase’ locked loop acquire the timing,
- why a bandpass filter has been used for timing recovery in the past, and
- how band edge filters acquire timing jointly with the carrier frequency offset.

Now we turn our attention towards a periodic input signal.

An Alternating Input Sequence

Consider an alternating (and hence periodic) data sequence with the following bit-symbol mapping.

$$10101010 \dots \rightarrow +1, -1, +1, -1, +1, -1, +1, -1, \dots$$

In modem jargon, this is known as a *dotting sequence*. In many communication systems, the training sequences often consist of 0xA (1010) and 0x5 (0101). In any case, the concepts hold true for any periodic input data sequence.

After pulse shaping at the Tx and matched filtering at the Rx, the signal can be written as

$$z(nT_S) = \sum_m a[m] r_p(nT_S - mT_M - \varepsilon_\Delta)$$

where $z(nT_S)$ is the matched filter output[†], $a[m]$ is the m^{th} data symbol (alternating between $+1$ and -1 in this case), $r_p(t)$ is a Raised Cosine pulse which is the auto-correlation of a square-root Nyquist pulse with support $-LG \leq t \leq LG$ samples and T_M is the symbol time. This is plotted for excess bandwidth $\alpha = 1$ in Figure 7.13a where the $+1, -1, +1, \dots$ data sequence is clearly seen.

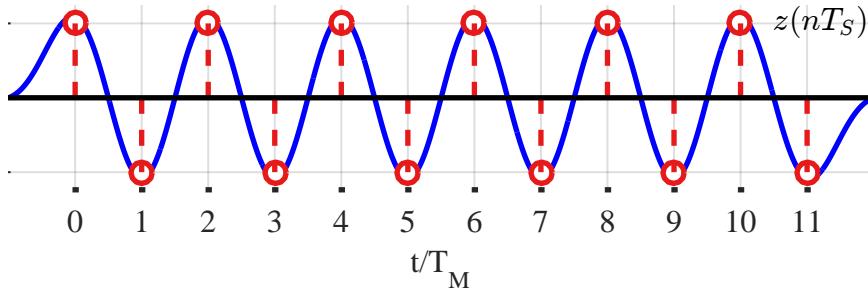
Due to the spectral smoothing of the alternating $+1$ s and -1 s sequence through the pulse shape, *the resulting signal essentially becomes like a sinusoid* with period $2T_M$ and hence frequency $1/2T_M$. This is the reason why the frequency $1/2T_M$ plays such a central role in the spectrum of all these single-carrier systems. There is some asymmetry in the first and the last symbol that becomes negligible with increasing sequence length. The role of excess bandwidth or roll-off factor α in timing synchronization will be discussed later and we continue with $\alpha = 1$ for now.

Remember from Figure 7.11 that the spectrum of a sinusoid with frequency F consists of spectral lines at $\pm F$. When the pulse shaped alternating sequence is taken into frequency domain in Figure 7.13b, we can see that two spectral lines are generated at $+1/(2T_M)$ and $-1/(2T_M)$, respectively. When there is a time shift ε_Δ in the Rx signal, the spectral lines exhibit a phase change of

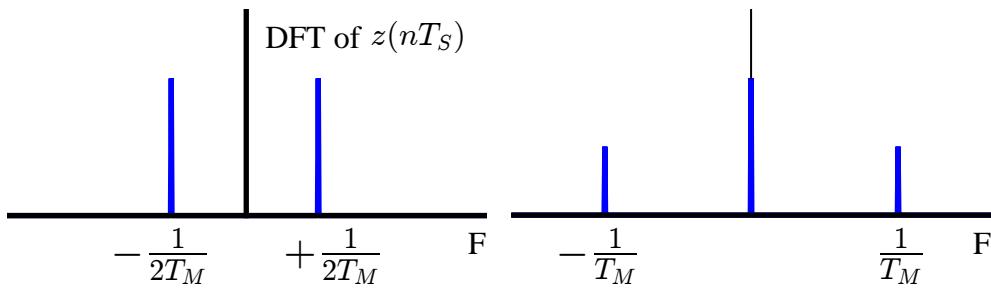
$$\mp 2\pi \cdot \frac{1}{2T_M} \cdot \varepsilon_\Delta$$

Consequently, an alternating sequence can provide a symbol timing estimate in the same manner as a simple sinusoid explained earlier: the time shift ε_Δ can be estimated from the phase of the spectral line.

[†]We will not explicitly bring timing offset ε_Δ into this discussion right now to keep our focus on generating timing lines. Obviously, the timing offset ε_Δ is then given by the phase of these spectral impulses and will appear soon in our explanation.



(a) The matched filtered alternating sequence essentially becomes a sinusoid with period $2T_M$



(b) Spectral lines at frequencies $\pm 1/(2T_M)$ due to data periodicity

(c) DFT of $|z(nT_S)|^2$: squaring generates spectral lines at frequencies $\pm 1/T_M$

Figure 7.13: An alternating data sequence both pulse shaped and matched filtered by a Square-Root Raised Cosine with excess bandwidth $\alpha = 1$ in time and frequency domains

What happens if we *square this signal*? Using the identity,

$$\cos^2 \theta = \frac{1}{2} (1 + \cos 2\theta)$$

our timing lines appear at $2(\pm 1/(2T_M)) = \pm 1/T_M$ along with a DC term. This is drawn in Figure 7.13c. From the above equation, the magnitude of the DC term is twice that of the sinusoid.

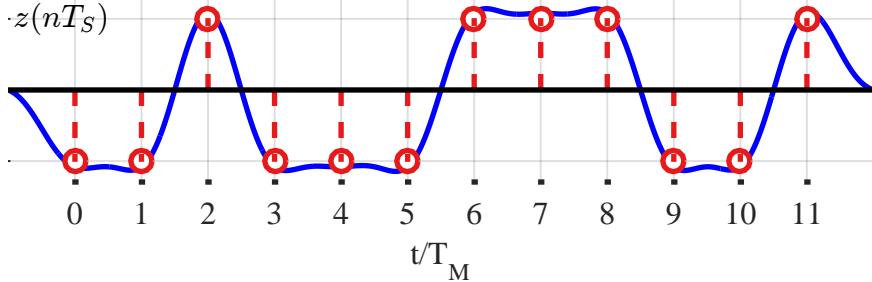
Keep in mind that until now, we have just discovered the presence of this timing line, we do not know yet how to use it for timing recovery.

Random Data

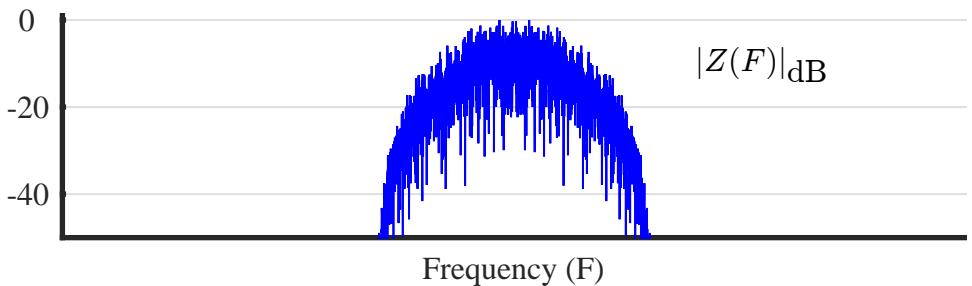
A Raised Cosine pulse shaped binary PAM random data sequence with excess bandwidth $\alpha = 1$ is illustrated in Figure 7.14a. The question is how to extract the spectral line without any alternating +1 and -1? Remember that if we take the DFT of a random signal, it will be different each time according to the exact realization of the sequence, so the spectrum of a random signal is the Fourier Transform of its average auto-correlation function instead of the signal itself.

Just temporarily, we take the DFT of this signal to observe how it looks like in frequency domain. A short sequence like in Figure 7.14a has some structure in it. So the DFT of a long sequence for true randomness is shown in Figure 7.14b on a dB

scale. Note that if I draw this figure again, I would have gotten a different spectrum according to the random data generated in my next run. The purpose here is not only to show the randomness in this spectrum but it will also help us later during our discussion on how squaring in time affects the frequency domain.



(a) Matched filtered random binary PAM sequence



(b) Spectrum of a long random sequence. No spectral lines are generated

Figure 7.14: A random data sequence both pulse shaped and matched filtered by a Square-Root Raised Cosine with excess bandwidth $\alpha = 1$

The main point to take is that no spectral lines are generated in such a case and consequently *no timing information is provided by the raw sequence itself*. Next, we discuss the squaring operation in time and frequency domains without heavy mathematics.

Squaring is Multiplication in Time Domain

With random binary PAM data arriving at the Rx, the outcome of the correlation discussion – see Eq (7.11) in Section 7.3 – says that the following expression needs to be maximized.

$$\sum_{m=0}^{N_0-1} a[m] \cdot z(mT_M + \hat{\epsilon}_\Delta)$$

Since the binary data $a[m]$ is unknown, a logical option is to use the matched filter output $z(mT_M + \hat{\epsilon}_\Delta)$ in place of $a[m]$, giving rise to

$$\sum_{m=0}^{N_0-1} |z(mT_M + \hat{\epsilon}_\Delta)|^2$$

(7.12)

The above expression is the squared matched filter output which is downsampled at symbol rate. Before downsampling at symbol rate, temporarily we want to examine the underlying high rate signal $z(nT_S)$ as

$$\sum_n |z(nT_S)|^2$$

Two comments are in order here.

- The actual derivation of Eq (7.12) involves log likelihood functions, logarithm of $\cosh(\cdot)$ and its approximations (interestingly, $|z(nT_S)|^2$ appears in one such approximation). But we steer clear of the intense mathematics and cover the underlying concepts through simple logic. Interested readers can refer to the excellent Ref. [21] for a detailed derivation.
- While squaring the signal value and its absolute value are the same for the PAM sequence assumed here, the absolute value squared is a more general operation that comes into play in passband modulations such as PSK and QAM to remove the effect of any phase rotations (remember, timing is recovered before the phase in digital modems). Obviously in such complex signals, $z(mT_M + \hat{\epsilon}_\Delta)$ in Eq (7.11) comes with a conjugate sign that leads to the absolute value squared.

Now let us square the matched filter output for a random binary PAM modulation to produce $|z(nT_S)|^2$ as illustrated in Figure 7.15 where it is evident that an excess bandwidth of $\alpha = 1$ was used to highlight the symbol rate sinusoid that appears during *data transitions*. Intuitively, during these data transitions, the signal behaves like a sinusoid very similar to the case of an alternating sequence. After squaring, the binary PAM values $+1$ and -1 map to the same value, leaving the squared sinusoid in between. A squared sinusoid is also a sinusoid at twice the rate along with a DC term because

$$\cos^2 \theta = \frac{1}{2} (1 + \cos 2\theta) \quad (7.13)$$

and $2(\pm 1/(2T_M)) = \pm 1/T_M$. On the other hand, when there is no transition, the squared signal is almost flat (for large excess bandwidth) and does not exhibit any sinusoidal behaviour. All this is evident from the horizontal grid line at $+1$ in Figure 7.15.

In summary, squaring produces in time domain a sinusoid at symbol rate $1/T_M$ (that originates during data transitions) as well as a DC term. This DC term can be traced back to

- the DC offset of $1/2$ from Eq (7.13) due to the negative values becoming positive as well, and
- the (almost) constant terms during no data transitions. While this is only true for $\alpha = 1$ (since an overshoot appears for smaller excess bandwidths), the concept still remains true.

Keep in mind that squaring is not the only non-linearity that generates a spectral line at symbol rate. Other options such as absolute value, fourth power or logarithmic non-linearities can also be used.

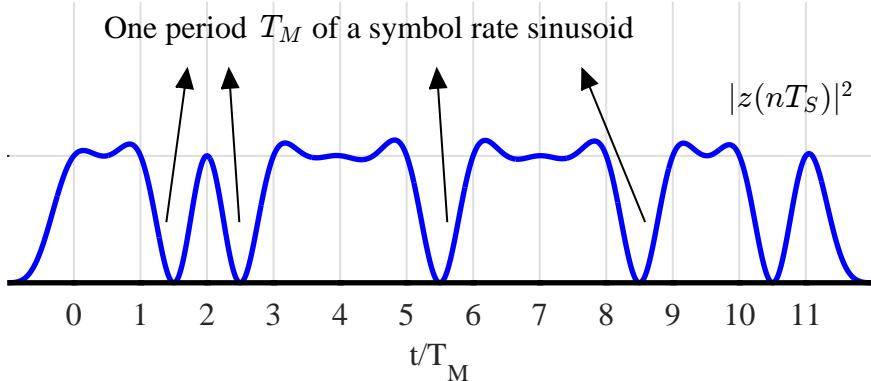


Figure 7.15: Squaring a matched filtered random binary PAM sequence. An excess bandwidth of $\alpha = 1$ is used to highlight the symbol rate sinusoid

Eye Diagram Perspective

First, recall that for a binary PAM sequence $a[m]$, the probability of both $+1$ and -1 is the same and equal to $1/2$. As a result, the average value is zero.

$$\text{Mean}\{a[m]\} = \frac{1}{2}(+1) + \frac{1}{2}(-1) = 0 \quad (7.14)$$

while $a^2[m] = +1$. Now we take the average value of the squared Rx signal and separate the two cases: one of square terms and the other of cross terms as happens in a squaring operation such as $(A + B)^2 = A^2 + B^2 + 2AB$.

$$\begin{aligned} \text{Mean } |z(nT_S)|^2 &= \text{Mean} \left\{ \sum_m a[m] r_p(nT_S - mT_M - \varepsilon_\Delta) \right\}^2 \\ &= \text{Mean} \left\{ \sum_m a^2[m] |r_p(nT_S - mT_M - \varepsilon_\Delta)|^2 \right\} + \\ &\quad 2 \sum_m \sum_{i \neq m} \text{Mean}(a[m]) \text{Mean}(a[i]) r_p(nT_S - mT_M - \varepsilon_\Delta) r_p(nT_S - iT_M - \varepsilon_\Delta) \\ &= \sum_m |r_p(nT_S - mT_M - \varepsilon_\Delta)|^2 \end{aligned} \quad (7.15)$$

where the second term goes to zero due to mean of the symbol sequence being zero as in Eq (7.14).

If our target was to find the peak of a single pulse, it is a straightforward task due to its symmetry. However, a whole data sequence shaped and matched filtered through a pulse is not symmetric. What happens if we take some help from the eye diagram perspective?

Figure 7.16 displays the squared eye diagram for a binary PAM sequence filtered with a Raised Cosine filter (hence including both the pulse shaping and matched filter) of excess bandwidth $\alpha = 0, 0.5$ and 1 , while Figure 7.16d draws their average values along the vertical axis according to Eq (7.15). Notice that the average value for $\alpha = 0$ is flat with no dependence on ε_Δ . Therefore, it provides no relevant information for timing synchronization. The flat response can be traced back to the fact that a sinc signal is

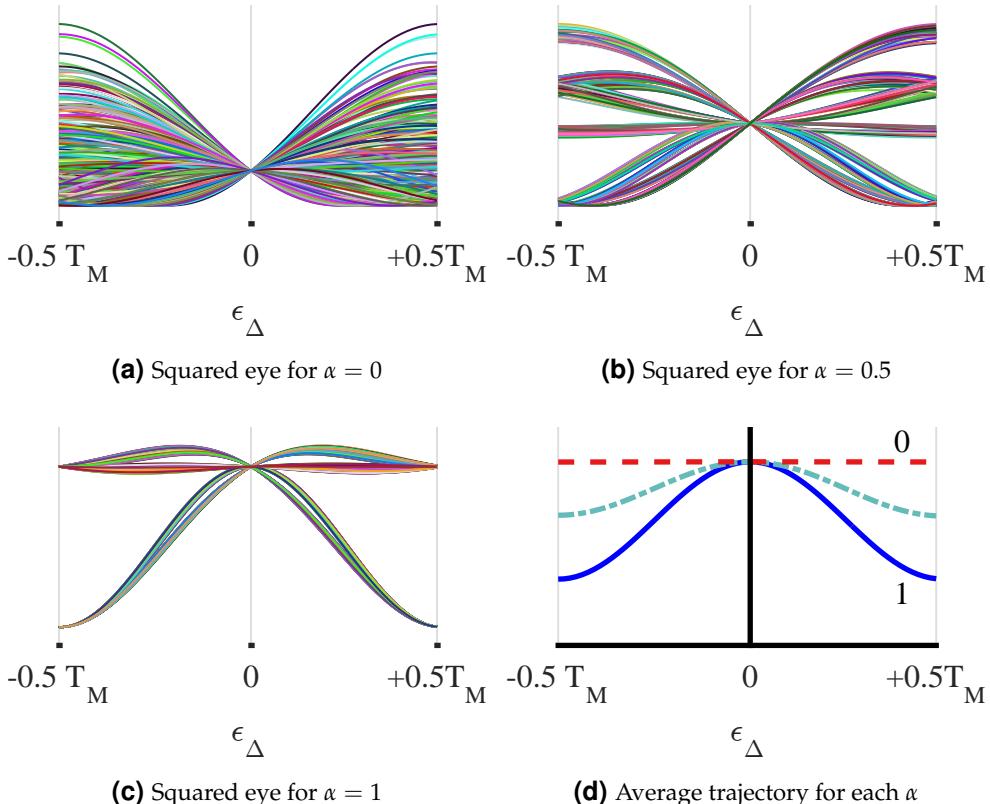


Figure 7.16: Square eye diagrams for a binary PAM sequence of 400 symbols shaped with Raised Cosine with excess bandwidths $\alpha = 0, 0.5$ and 1

the perfect interpolator, as it is the inverse DFT of a rectangular spectrum used to filter out the spectral replicas after interpolation, see Section 2.7.2. After squaring, both $+1$ and -1 map to the same value and hence the sinc signal, on average, perfectly interpolates between them maintaining the same value throughout the curve.

On the other hand, $\alpha = 0.5$ and $\alpha = 1$ provide suitable timing information due to the dependence of their average curves on ϵ_Δ . This gives us a clue that perhaps *generation of a timing line is not necessary for finding the optimal sampling instant*. Instead, standard mathematical methods (such as a derivative) for $|z(nT_S)|^2$ can be used to approach its maximum. In fact, several timing estimation techniques exploit this curve for synchronization purpose, as we will find out later.

Squaring is Convolution in Frequency Domain

Let us observe how this squared matched filter output $|z(nT_S)|^2$ looks in frequency domain. The curve is plotted in Figure 7.17a. If we consider the randomness in the Fourier transform of the simple matched filter output in Figure 7.14b, the result here seems remarkable: out of the random spectrum, three spectral lines have arisen as a result of squaring the signal.

There is a proper mathematical method to see their existence that is based on

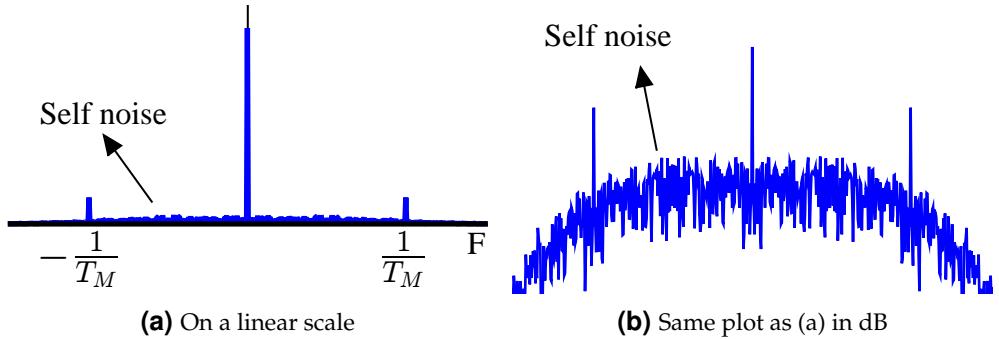


Figure 7.17: Spectrum of a squared matched filtered random binary PAM sequence $|z(nT_S)|^2$. An excess bandwidth of $\alpha = 1$ is used to highlight the symbol rate sinusoid

Fourier series but we describe the intuitive route in detail.

- First, see the squared signal in Figure 7.15. As described before, it is dominated by a DC term and a sinusoidal term for a sufficiently long sequence. In individual capacity, both the constant and the sinusoid are periodic signals and hence the composite signal contains an underlying periodicity.
 - We know that sampling in time domain at integer multiples of T_S creates *periodic* spectral replicas in frequency domain at integer multiples of the sample rate F_S , i.e., sampling in time domain generates periodicity in frequency domain. The reverse is also true: *periodicity in time domain produces samples in frequency domain*.
 - Bringing this analogy here, the periodicity in time domain by the symbol time T_M thus creates ‘samples’ in frequency domain at gaps of symbol rate, i.e., integer multiples of $\pm 1/T_M$ as seen in Figure 7.17a (the actual mathematical argument exploits the cyclostationarity in the underlying random process and hence its Fourier series representation).

However, the combination of the pulse shape and matched filter is bandlimited to

$$B_{z(nT_S)} = \frac{1+\alpha}{2T_M}$$

Now squaring is multiplication in time domain and hence convolution in frequency domain.

$$|z(nT_S)|^2 = z(nT_S) \cdot z(nT_S) \xrightarrow{\mathsf{F}} Z(F) * Z(F)$$

which doubles the bandwidth to

$$B_{|z(nT_S)|^2} = \frac{1+\alpha}{T_M} \leq \frac{2}{T_M}$$

As a consequence, any extra spectral ‘samples’ (arising from the discontinuities) at gaps of inverse symbol rate die down after the first such component. Only the DC term and the $\pm 1/T_M$ ‘samples’ survive and those at $\pm 2/T_M$ onwards are filtered out.

To understand it in an intuitive manner, start from the fact that squaring in time is convolution in frequency domain. So the spectrum can be divided into three components, as it is either the noise part, the DC term or the impulses representing the symbol rate sinusoid, as illustrated in Figure 7.17a.

Self noise: The matched filter output $z(nT_S)$ comes from a shaped long binary random sequence. From the definition of DFT, its spectrum, i.e., $|Z(F)|$, therefore contains a summation of complex sinusoids with different frequencies and amplitudes scaled by the random sequence. These sinusoids thus add constructively at some points and destructively at others. As a consequence, it contains several short peaks and troughs in a random manner.

During convolution, one spectrum is fixed at zero while the other slides from $-\infty$ to $+\infty$. Each convolution sample comes from the summation of products where the spectral regions overlap. Such spectral regions, when not completely aligned, produce a summation of several random samples and thus add up to a low value. Since the signal at these frequencies arises due to interactions among data symbols themselves, it is often termed as *self noise*, also known as *modulation noise* or *pattern noise*.

This does not seem much in an ideal Figure 7.17a but it is redrawn on a dB scale in Figure 7.17b for clarity. Self noise is a significant problem, particularly for short term variations, low excess bandwidths and higher-order QAM transmissions. However, several techniques have been devised to prevent it from distorting the time domain symbol rate sinusoid.

DC term Convolution result at time zero is produced when the sliding spectra completely overlap each other. A complete overlap of such random peaks and troughs gives rise to a large DC term where everything adds up in phase.

Impulses $\pm 1/T_M$ At the Tx, the PAM data sequence is generated at a rate of 1 sample/symbol. Consequently, upsampling by L creates L spectral replicas within the region $\pm F_S/2 = \pm L/2T_M$. An example for a random binary PAM sequence and $L = 4$ is illustrated in Figure 7.18 where the spectrum within the circle can be seen to repeat at $\pm 1/T_M, \pm 2/T_M, \dots$. It now possesses a symmetry around $\pm 1/2T_M$ at the edges of this circle.

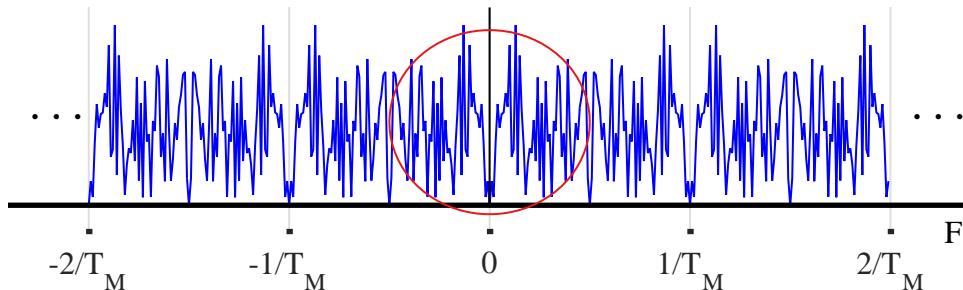
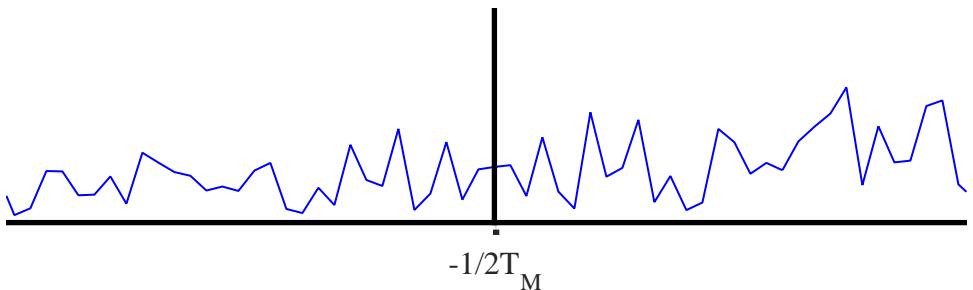


Figure 7.18: Spectrum of an $L = 4$ upsampled binary PAM data sequence. Notice the periodic repetitions of the spectrum within the circle

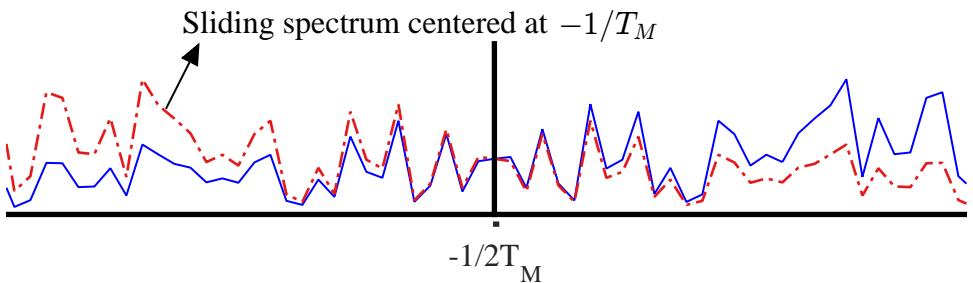
Also recall from Section 3.6 on pulse shaping filter that according to the Nyquist no-ISI criterion in frequency domain, a pulse shape with an odd sym-

metry around the point $\pm 1/2T_M$ is required. The spectrum $|Z(F)|$ is created through multiplication of such a pulse shape with the spectrum of an upsampled random binary PAM sequence.

As a result, $|Z(F)|$ is odd symmetric around $\pm 1/(2T_M)$, i.e., the values are the same but with magnitude going from low to high at $-1/2T_M$ according to the pulse shape. To show this segment, a zoomed in version of $|Z(F)|$ around $-1/(2T_M)$ is drawn in Figure 7.19a. During convolution that arises from squaring the signal in time domain, when the sliding spectrum reaches $-1/T_M$, it overlaps with a similar curve but with lower values which is drawn in Figure 7.19b. Their addition after point by point product generates a structure within all this randomness that is the spectral line at $-1/T_M$.



(a) Zoomed in version of the spectrum $|Z(F)|$ of the matched filtered random binary PAM sequence. Observe the symmetry around $-1/(2T_M)$



(b) Overlap during convolution. One spectrum is centered at zero while the other at $-1/T_M$

Figure 7.19: Sliding of spectra $|Z(F)|$ during convolution that arises from squaring the signal

Observe that the height of impulses in Figure 7.17a is lesser than that of an alternating sequence in Figure 7.13c. The DC term here comes from both the squaring operation $\cos^2 \theta = 0.5(1 + \cos 2\theta)$ and symbols where no data transitions occur. Also, the number of data transitions for a very long sequence is approximately half of the sequence length.

As a final remark, we said in Section 6.4.2 that a squaring operation exploits the underlying structure in the signal for synchronization purpose. Now we can easily see from the above description that the original spectrum in the presence of a CFO is centered at F_Δ . During the squaring operation (after repairing the CFO and matched filtering), a spectral convolution occurs which reaches its maximum at a complete

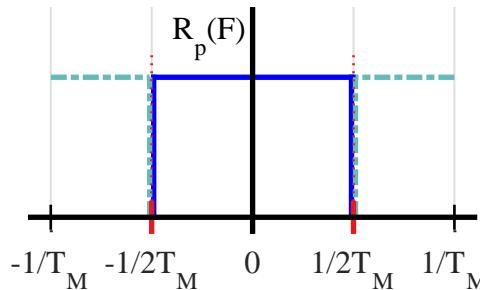
overlap, thus generating a maximum for the case where repairing was done by F_Δ . Consequently, a squaring operation leads to practical CFO estimators as we saw in Chapter 6 in the form of a derivative frequency error detector and band edge frequency locked loop. The role of band edge filters will continue in this chapter in Section 7.12.

7.4.2 Role of Excess Bandwidth

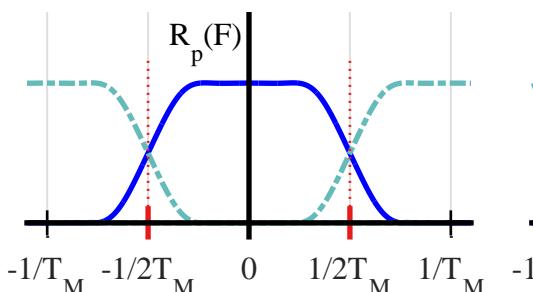
At this time, we discuss the role of excess bandwidth or roll-off factor in the generation of spectral timing lines. We begin with the frequency domain perspective.

Frequency Domain View

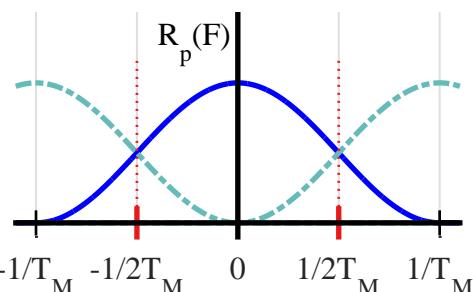
From the last section, we know that a squaring operation in time domain actuates a convolution in frequency domain of the signal spectrum with its replica. The resultant spectrum at $F = \pm 1/T_M$ is determined by the overlap of the two spectra at the instant when the sliding spectrum reaches $\pm 1/T_M$. The image at that moment is captured in Figure 7.20 for three excess bandwidths, $\alpha = 0, 0.5$ and 1 .



(a) $\alpha = 0$ implies no overlap and hence no timing information



(b) The overlapped region for $\alpha = 0.5$ is relatively small



(c) $\alpha = 1$ generates the largest possible overlap between the spectra

Figure 7.20: The role of excess bandwidth towards generating the spectral line for three different values. A larger α produces a larger overlapped region, thus resulting in a taller impulse

It is clear that the larger the excess bandwidth, the larger the overlapped region between the two spectra and hence taller the resultant timing line. Recalling the band

edge filters for frequency synchronization in Section 6.4.3 that exploit the excess bandwidth as well, this is the reason why you will often hear the following saying from fred harris.

“Excess bandwidth provides the energy required for timing and frequency synchronization.”

fred harris

It should be kept in mind that this is true for timing schemes derived from the above principles. Indeed, there are indeed timing error detectors that benefit from a deficiency of excess bandwidth and their performance deteriorates for higher values of α . Mueller and Muller (M&M) timing error detector discussed in Section 7.11 is one such example.

Time Domain View

Coming back to the time domain, we explain the case for $\alpha = 1$ and it is straightforward to generalize the concept for other values of α . Recall from the discussion on pulse shaping filters in Figure 3.25 of Section 3.6 that the expression for a Raised Cosine filter in frequency domain for $\alpha = 1$ is given by

$$R_p(F) = \frac{T_M}{2} \left[1 + \cos \left(2\pi \cdot \frac{T_M}{2} \cdot |F| \right) \right] \quad -\frac{1}{T_M} \leq F \leq +\frac{1}{T_M} \quad (7.16)$$

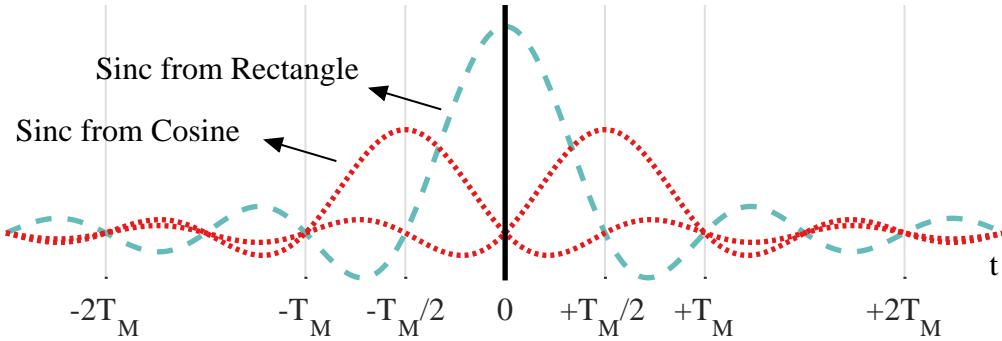
We also covered the following steps:

- In time domain, the constant term translates to an impulse at time location 0, and $\cos(\cdot)$ results in two impulses with half that amplitude at time locations (inverse period) $\pm T_M/2$.
- The rectangular window of width $2/T_M$ that defines its support in frequency is a sinc signal in time domain with zero crossings at integer multiples of $\pm T_M/2$.
- Convolution of a signal with an impulse is the signal itself. When convolved with sinc arising from the window, this generates three sinc signals: one at time 0 (convolution of the above sinc with impulse from constant term) while two at time $\pm T_M/2$ (convolution of the above sinc with impulses from the cosine).

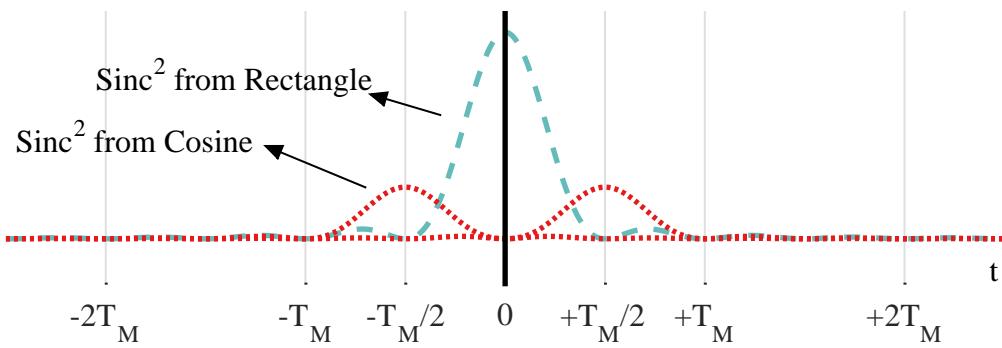
The resulting time domain signal was plotted in Figure 3.27a, which is now redrawn in Figure 7.21a for convenience without the cumulative signal (that is irrelevant to our discussion here). When its square is drawn in Figure 7.21b (cross terms not shown), it clear from their period that the extra sinc signals – arising from the spectral cosine responsible for excess bandwidth – provide the energy to $1/T_M$ sinusoidal signal. These additional sincs are absent for $\alpha = 0$ and very small for low values of α .

Another interesting explanation on the effect of excess bandwidth α in time domain is seen from Figure 7.16d. It is evident that a large excess bandwidth α provides a larger slope in a squared and averaged eye and hence more timing information.

Now we move on to the topic of recovering the symbol timing from the spectral lines generated above.



(a) Raised Cosine pulse formed by sum of inverse transform of a windowed constant and windowed cosine



(b) Square of the above signals without cross terms. Notice the $1/T_M$ sinusoidal component

Figure 7.21: Time domain formation of a Raised Cosine pulse for $\alpha = 1$

7.4.3 Clock Recovery in Analog Modems

Having recovered the spectral line at symbol rate $1/T_M$, the question is how to utilize this signal for successful symbol timing extraction. In the pre-DSP era, most of the signal processing required for the Rx operation was done in analog stages. This kind of architecture is shown in Figure 7.22. Analog modules are used to accomplish the mathematical operations for communication purposes such as summation, multiplication and filtering in general. The Rx signal is only digitized at the very end of signal processing chain, i.e., just before the symbol decision at 1 sample/symbol. Of course, the rise of DSP for accomplishing the same tasks kept pushing the ADC closer and closer to the antenna.

For the purpose of timing recovery, two tasks need to be accomplished to get a pair of clean impulses at $\pm 1/T_M$: extracting the symbol rate clock from the squared signal as well as removing the undesired components such as a DC term and the self noise. Both of these tasks can be accomplished by adopting either of the following strategies.

Bandpass filter The simplest solution is to employ a narrow bandpass filter around the timing lines $\pm 1/T_M$ that filters out most of the undesired spectral components. This is shown in Figure 7.23a for excess bandwidth $\alpha = 1$. Being a

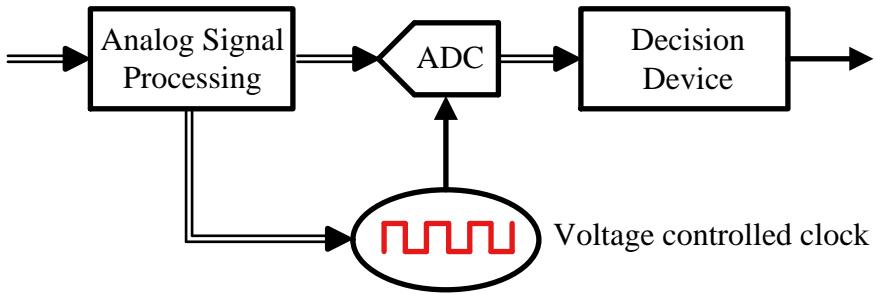


Figure 7.22: Analog approach adjusts the sampling clock using continuous-time signal processing

sinusoid, its phase offset from time zero is the same as the timing offset ε_Δ . The output signal then drives a clock generation circuit based on the zero crossings of the sinusoid that prompts the ADC to sample the matched filtered signal at optimal instants. As the timing offset ε_Δ slowly varies, the phase of the filter output sinusoid also varies accordingly.

Phase Locked Loop (PLL) The PLL solution to timing recovery problem is to vary the sampling instant of a voltage controlled clock through analog signal processing. The input to this PLL is the squared matched filter output that contains $\pm 1/T_M$ spectral lines. Again, the phase offset of the generated sinusoid is the target timing offset ε_Δ and a simple PLL like the one in Chapter 4 is enough for timing synchronization.

At the heart of this PLL sits a timing error detector that produces a control signal to adjust the phase of the voltage controlled clock. Assume that the timing error detector multiplies the PLL input with the clock output which possesses a spectrum as two impulses at $\pm 1/T_M$ but no DC term. Referring to Figure 7.23b, it is straightforward to deduce that when the two signals – one from Figure 7.23a and the other the same without the DC term – are close to each other, convolution of these impulses in frequency domain moves the spectral contents in a

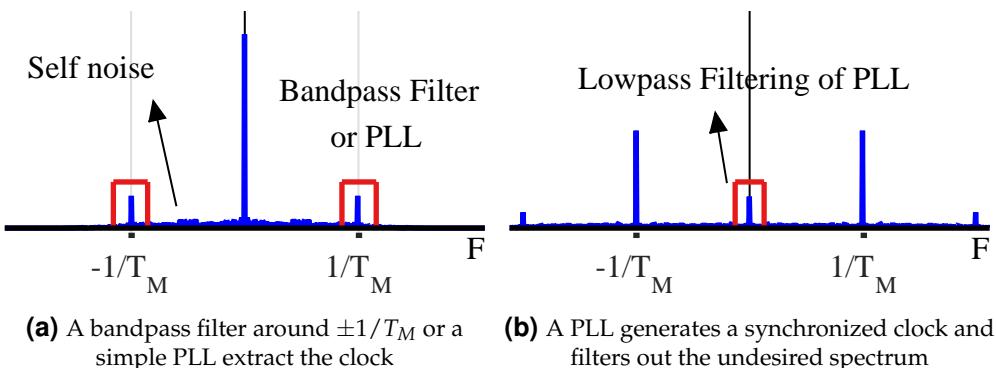


Figure 7.23: Methods to extract the synchronized clock from the spectral line

manner such that only the contribution from the timing lines appears at DC while the input DC term gets shifted to $\pm 1/T_M$ (this spectral convolution arises from the multiplication in the error detector and is in addition to the convolution from squaring). The short spectral lines at $\pm 2/T_M$ are also due to this convolution. The PLL then essentially acts as a lowpass filter around it. In a comparison, a PLL offers some advantages over a bandpass filter for timing synchronization.

In light of the discussion above, probably the oldest method to estimate the symbol timing is squaring the baseband signal. As we find next, the basic principle is to remove the modulation and generate a spectral line at the symbol rate which can then be used to track the Tx clock. The main advantages were as follows.

- At a time when baseband processing was in the realm of analog circuits, squaring the waveform could be implemented just through a multiplier.
- Furthermore, a squaring operation gave rise to mathematically tractable results, something communication theorists always desire. There are several reasons of this tendency. Having equations in hand gives the designer insights into the fundamental tradeoffs behind the system. Deriving performance benchmarks through those closed-form expressions also provides a tool to compare different methods for the same task. Obviously, closed-form expressions are not the ultimate goals in themselves and a balance needs to be found out.

This kind of procedure perfectly matched with an analog clock recovery setup shown in Figure 7.22 that provides the instant to sample the Rx waveform at 1 sample/symbol at the maximum eye opening.

Discrete-time processing opened the doors for better timing recovery schemes as we discover throughout this chapter. As an ever increasing number of transistors within the same area consistently keeps bringing the digital processing cost down, the use of analog circuits to control the timing is not economically feasible anymore and methods from digital processing have gradually become the norm. Everything that was implemented by ‘real’ circuit elements got replaced with the lines of code in a microprocessor that perform the same operations right out of the pure mathematical equations.

Before moving on to the actual timing locked loops, we explain a feedforward estimator that employs the squaring approach to extract the timing delay estimate for a short packet.

7.4.4 Feedforward: Digital Filter and Square Estimator

As described before, the analog modems utilize the squared signal to generate a clock in synchronization with the Tx symbol rate. After all, they had to sample only once: at the maximum eye opening. For a digital modem where digital signal processing techniques are applied to an already sampled signal, a direct estimate of the symbol timing offset $\hat{\varepsilon}_\Delta$, proposed in Ref. [22], can be obtained as follows.

Assume that the Rx signal is sampled at a rate of $F_S = 1/T_S = L/T_M$ where T_M is the symbol time and L is the oversampling factor, or samples/symbol.

$$r(nT_S) = \sum_m a[m] p(nT_S - mT_M - \varepsilon_\Delta)$$

Here, $p(nT_S)$ is a square-root Nyquist pulse such as a Square-Root Raised Cosine. Since the spectral line after squaring appears at $\pm 1/T_M$, the parameter L should be chosen according to the sampling theorem, i.e., more than twice the highest frequency component of the input signal, i.e.,

$$F_S = \frac{L}{T_M} > 2 \cdot \frac{1}{T_M} \quad \Rightarrow \quad L > 2$$

For this reason, *L must be an integer greater than 2*. It is a common practice to choose $L = 4$ for this purpose due to a simple estimator architecture, as we shortly see.

After converting the signal into digital domain, it is matched filtered with a similar but flipped pulse shape $p(-nT_S)$. That accounts for the ‘digital filter’ part in the name of this scheme.

$$\begin{aligned} z(nT_S) &= r(nT_S) * p(-nT_S) \\ &= \left\{ \sum_m a[m] p(nT_S - mT_M - \varepsilon_\Delta) \right\} * p(-nT_S) \\ &= \sum_m a[m] r_p(nT_S - mT_M - \varepsilon_\Delta) \end{aligned}$$

where $r_p(nT_S)$ is the Nyquist pulse such as a Raised Cosine. Since the sample rate is chosen large enough, no information is lost in digital transformation and squaring the digital signal produces the same timing lines as before. And hence the name *digital filter and square timing recovery*.

To locate this spectral line, we take the Discrete Fourier Transform (DFT) of the squared signal.

$$Y[k] = \text{DFT } |z(nT_S)|^2$$

Exploiting the fact that $|z(nT_S)|^2$ is a real signal with zero Q component, the DFT definition yields

$$\begin{aligned} I \rightarrow Y_I[k] &= \sum_{n=0}^{N-1} |z(nT_S)|^2 \cos 2\pi \frac{k}{N} n \\ Q \uparrow Y_Q[k] &= - \sum_{n=0}^{N-1} |z(nT_S)|^2 \sin 2\pi \frac{k}{N} n \end{aligned} \tag{7.17}$$

However, all DFT components $k = -N/2, \dots, -1, 0, 1, \dots, N/2 - 1$ of the sequence are not required because the matched filter output is bandlimited to $(1 + \alpha)/2T_M$ while the squared signal due to convolution in frequency domain is bandlimited to

$$B_{|z(nT_S)|^2} = \frac{1 + \alpha}{T_M} \leq \frac{2}{T_M}$$

Consequently, only the DFT component corresponding to $F = \pm 1/T_M$ should be enough for timing purpose. So instead of computing the full DFT, only a single such component can be calculated if its position k is known. Now we exploit one of the most useful relations from Eq (1.51), Chapter 1 to travel between discrete and continuous frequencies $k/N = F/F_S$. This gives the timing lines at the following discrete frequencies.

$$\pm \frac{F}{F_S} = \pm \frac{1/T_M}{1/T_S} = \pm \frac{1/(LT_S)}{T_S} = \pm \frac{1}{L}$$

Assume that a data packet contains a total of N_d symbols. Ignoring the group delay G on either side of the matched filtered and squared signal, the length of the sequence is $N = L \cdot N_d$. Since the above expression F/F_S is equal to k/N , we get

$$\frac{k}{N} = \pm \frac{1}{L} \quad \rightarrow \quad k = \pm N_d$$

This is the DFT component corresponding to our spectral line. Consequently, we plug it into DFT Eq (7.17) above.

$$\begin{aligned} I &\rightarrow Y_I[N_d] = \sum_{n=0}^{N-1} |z(nT_S)|^2 \cos 2\pi \frac{N_d}{N} n \\ Q &\uparrow Y_Q[N_d] = - \sum_{n=0}^{N-1} |z(nT_S)|^2 \sin 2\pi \frac{N_d}{N} n \end{aligned}$$

By using $N = L \cdot N_d$,

$$\begin{aligned} I &\rightarrow Y_I[N_d] = \sum_{n=0}^{N-1} |z(nT_S)|^2 \cos 2\pi \frac{1}{L} n \\ Q &\uparrow Y_Q[N_d] = - \sum_{n=0}^{N-1} |z(nT_S)|^2 \sin 2\pi \frac{1}{L} n \end{aligned} \tag{7.18}$$

Recall that the phase of this spectral component produces our estimate of timing delay, see Figure 7.11b and a delay in time manifests itself as sinusoidal phase, i.e., a rotation of spectral impulses.

$$\hat{\varepsilon}_\Delta = -\frac{1}{2\pi} \angle Y[N_d]$$

To avoid an extra phase term from the asymmetrical summation from $n = 0$ to $N - 1$, a symmetrical sum from $n = -N/2$ to $N/2 - 1$ can also be taken. The above expression seems to be a remarkable result since *it generates a timing estimate from a phase estimate!* Looking back at Figure 7.11, this is not a surprise.

As you might have noticed, we have skipped some mathematical details to rigorously derive the above estimator. The above estimator requires at least $L = 4$ samples/symbol and feedforward solutions have been proposed to reduce it to $L = 2$ samples/symbol at the cost of extra complexity. Observe from Eq (7.18) that for this particular value of L equal to 4, the cosine and sine decompose into following simple relations.

$$\begin{aligned} \cos \frac{2\pi}{4} n &= \cos \frac{\pi}{2} n = 1, 0, -1, 0, \dots \\ \sin \frac{2\pi}{4} n &= \sin \frac{\pi}{2} n = 0, 1, 0, -1, \dots \end{aligned}$$

The main attraction of this choice is that a multiplication-free realization of the timing estimator is obtained, a block diagram of which is drawn in Figure 7.24 with T_S representing one sample delay. Match the negative signs of the cosine and sine above with

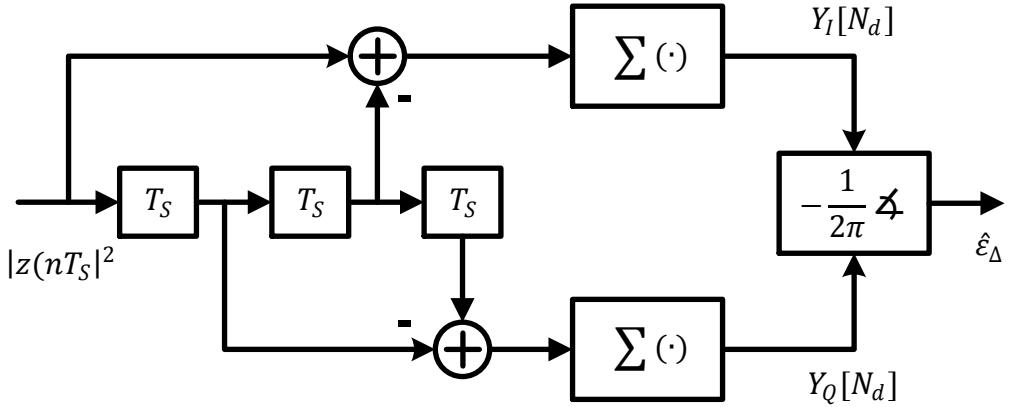


Figure 7.24: A block diagram for the implementation of a digital filter and square timing recovery for $L = 4$ samples/symbol

those in the block diagram. The negative signs in the Q arm are inverted due to the negative sign in Eq (7.18).

As far as the discussion above is concerned, this estimate can be utilized for timing recovery in a short packet. For other scenarios, emulating an analog approach of reconstructing a digital symbol rate clock is not our purpose and instead we want to know where the optimal sampling instant corresponding to the maximum eye opening during each symbol is located. This can be done in a very simple manner with just $L = 2$ or even $L = 1$ sample/symbol, for which we design a feedback solution by utilizing the insights developed through the squaring approach.

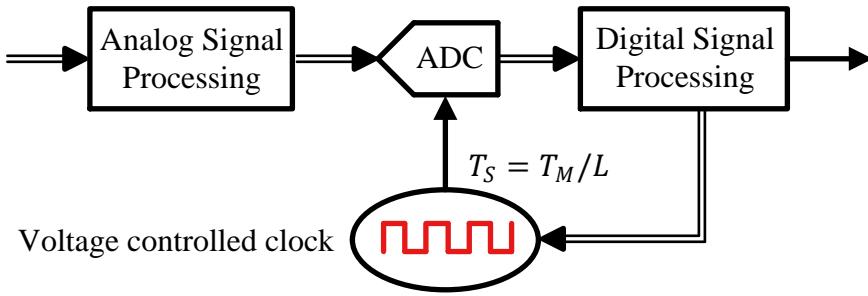
7.5 A Timing Locked Loop (TLL)

Digital communication systems by necessity need to sample the Rx waveform to produce a digital output at symbol rate $1/T_M$. There are three fundamental approaches to solve this problem.

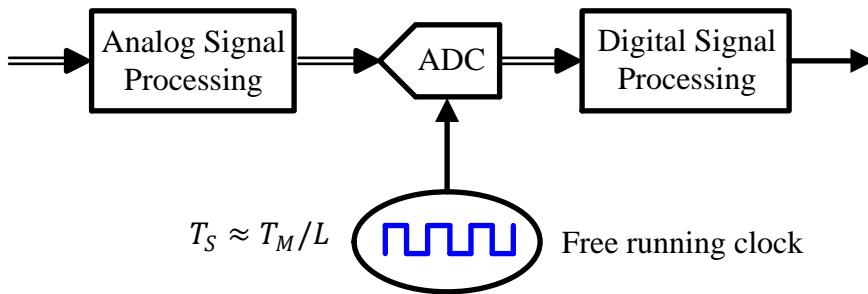
Analog timing recovery This was discussed in a little detail in Section 7.4.3.

Hybrid timing recovery In hybrid timing recovery, the timing information is derived through the samples of the Rx signal but the sampling instant is still controlled through adjusting the phase of a voltage controlled clock, see Figure 7.25a. The primary advantage is that the samples produced through this approach are aligned in both phase and frequency with the Tx clock, i.e., sampling time T_S and symbol time T_M are commensurate by nature. The main disadvantages are nicely detailed in Ref. [2] which we reproduce below.

1. A voltage controlled clock exhibits a higher level of phase noise as compared to a free running clock, and hence injects more timing jitter (variation of the clock transitions around its mean value) into the system.
2. A feedback path to the analog part of the Rx is required due to which there is a hardware overhead to the analog frontend of the Rx.



(a) Hybrid approach adjusts the sampling clock in analog domain through a digital signal processing mechanism



(b) Digital approach adjusts the sampling clock in digital domain through a digital signal processing mechanism

Figure 7.25: Timing synchronization advanced from a pure analog implementation through hybrid to an all-digital solution

3. The matched filter lies within the feedback path of the timing locked loop that causes additional delay and hence adversely alters the response of the loop.
4. This approach cannot be modified to handle the case where the input signal is multiplexed through different sources and hence contains independent clock sources.

Digital timing recovery Using this approach, the Rx signal $r(t)$ is sampled by a free running clock at a constant rate $1/T_S$ that is asynchronous to the symbol rate $1/T_M$ as shown in Figure 7.25b. These samples $r(nT_S)$ are matched filtered to produce $z(nT_S)$ at L samples/symbol, none of which lies at the symbol boundary, i.e., an integer multiple of T_M (except by luck). Here, the job of the timing synchronization loop is twofold.

1. Figure out an estimate of the timing offset $\hat{\epsilon}_\Delta$ (or an error signal $e_D[m]$ proportional to it).
2. Then, rather than shifting a physical clock (such as a voltage controlled clock), construct the 'missing samples' at optimal locations $z(mT_M + \hat{\epsilon}_\Delta)$ by an algorithm operating on asynchronous samples $z(nT_S)$, see Figure 7.26

for an understanding of the notation used. This process is known as *interpolation*. The economic advantage of digital processing and a free running clock has made this option the most viable in software defined radios.

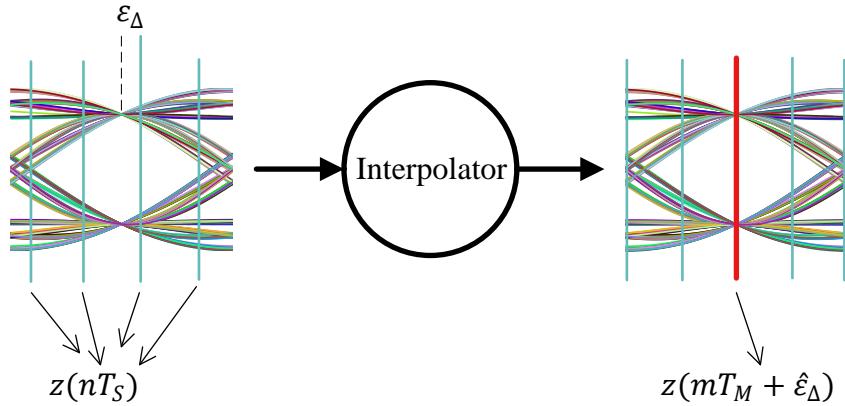


Figure 7.26: The input to the interpolator is raw matched filter output $z(nT_S)$ while its output is the samples aligned with symbol boundaries downsampled at symbol rate, $z(mT_M + \hat{\varepsilon}_\Delta)$

This all digital timing recovery option leads us to a Timing Locked Loop (TLL). Just like a PLL, a TLL consists of the following components, a block diagram of which is drawn in Figure 7.27.

Timing Error Detector (TED): In an ordinary PLL, the input is usually a sinusoid (even if it has zero frequency, as we saw during phase synchronization). In a TLL, the corresponding input is the matched filter output $z(nT_S)$. A Timing Error Detector (TED) generates an error signal $e_D[m]$ during each symbol interval using the matched filter outputs $z(nT_S)$ proportional to the timing phase difference ε_Δ between the actual and desired sampling instants. The TLL functions properly as long as the mean error has the same sign as the actual error.

$$\text{sign}(\text{Mean } e_D[m]) = \text{sign}(\varepsilon_\Delta), \quad -\frac{T_M}{2} \leq \varepsilon_\Delta \leq +\frac{T_M}{2}$$

The TEDs we cover operate at 1 or 2 samples/symbol.

Loop filter: A loop filter sets the dynamic performance limits of a TLL as well. Moreover, it helps filter out noise and irrelevant frequency components generated in the timing error detector. Its output signal is denoted as $e_F[n]$. The sample index n is the same as the symbol index m for a TED using $L = 1$ sample/symbol.

Interpolation control and interpolator: Taking $e_F[n]$ as an input, an interpolator control determines two values, the integer sample closest to the maximum eye opening (called the *basepoint index*) and the interval between this basepoint index and the optimal sampling instant (known as the *fractional interval*). Then, these two values are provided to the interpolator. The interpolator control also incorporates a symbol timing frequency difference that slowly traces the difference between the Tx and Rx clock ticks.

An interpolator computes the ‘missing samples’ $z(mT_M + \hat{\epsilon}_\Delta)$ by using the neighbouring samples $z(nT_S)$, see the input and output eye diagrams of Figure 7.27 for an understanding of this notation.

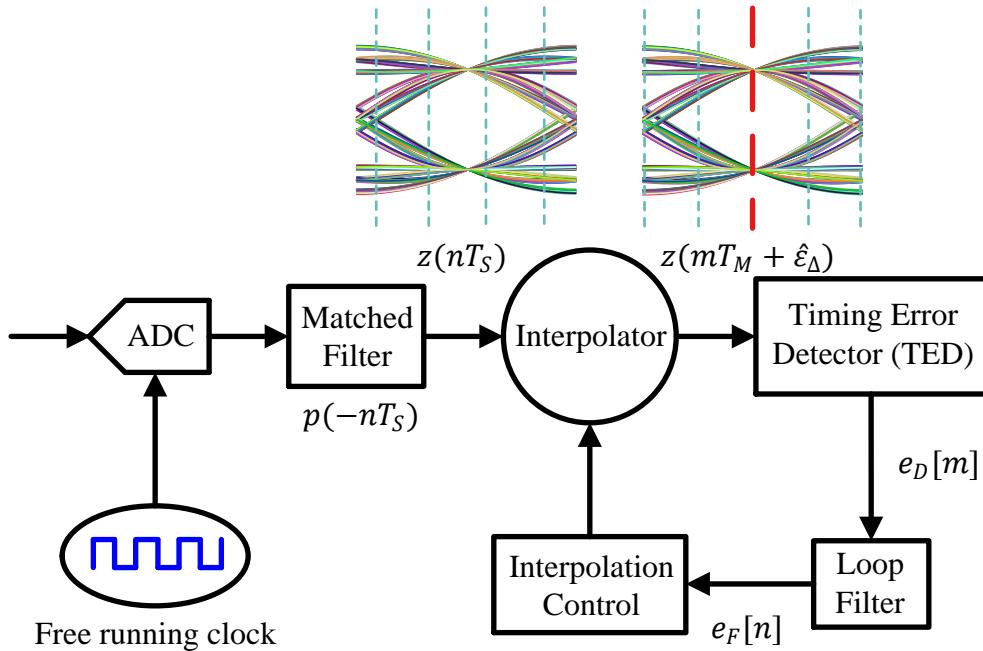


Figure 7.27: A Timing Locked Loop (TLL) for digital symbol timing recovery

We now move towards studying the role of timing error detectors in accomplishing this task. The mechanics of interpolation and interpolation control will be discussed later in this chapter.

7.6 Symbol Centric Timing Error Detectors

Just like a phase error detector, a Timing Error Detector (TED) sits at the heart of a feedback loop for timing correction. For exploring it further, we refer to the interesting interpretation of squaring operation in time domain from Section 7.4. There we claimed that perhaps generation of a timing line is not necessary for finding the optimal sampling instant. Instead, standard mathematical methods (such as a derivative) for $|z(nT_S)|^2$ can be used to approach its maximum. This route is also encouraged due to the square law being an approximation of $\ln \cosh(\cdot)$ which comes from starting at the log likelihood function itself.

Next, we employ this insight into forming reliable timing error detectors. From here onwards, I want you to remember two things.

- An eye diagram due to its symbol-periodic overlaps is an excellent summary of signal behaviour in time domain, very similar to a spectrum in frequency domain

(they both utilize all the available information). Keep in mind the average eye such as the one in Figure 7.16d for the point next.

- With average eye in hand, you can refer over and over again to the seesaw near my home shown in Figure 7.28 to understand how the timing error detectors work because the crux of timing synchronization is all about balancing a seesaw similar to the one shown here.



Figure 7.28: Remembering a seesaw perspective helps in understanding timing error detectors coming next

7.6.1 Feedback: Derivative Timing Error Detector

To generate the symbol rate sinusoid studied before, start with the lowest possible sample rate, i.e., 1 sample/symbol, for $|z(nT_S)|^2$ in a real PAM sequence and notice that,

$$\frac{\partial}{\partial \hat{\varepsilon}_\Delta} |z(mT_M + \hat{\varepsilon}_\Delta)|^2 = 2 z(mT_M + \hat{\varepsilon}_\Delta) \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \quad (7.20)$$

where the term $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$ is the derivative of the matched filter output with respect to $\hat{\varepsilon}_\Delta$. For a complex sequence like QAM or PSK, this output would have been the inphase component of this expression, i.e.,

$$2 \left\{ z(mT_M + \hat{\varepsilon}_\Delta) \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \right\}_I$$

which is similar to what we encountered in the case of carrier frequency synchronization, see Eq (6.27).

Now a derivative at a point was defined in Chapter 2 as the slope of the line tangent to the curve at that point. If you do not know about differentiation, *just remember the derivative as roughly the slope of the curve at a point*.

The objective is to maximize the correlation for which the derivative in Eq (7.20) must go to zero (when the slope reaches zero, it implies that the curve is at its maximum or minimum value). With 2 being an immaterial factor, an error signal $e_D[m]$ that

updates the estimate at time m can be set up as

$$e_D[m] = z(mT_M + \hat{\varepsilon}_\Delta) \cdot \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \quad (7.21)$$

which is our *Derivative Timing Error Detector (TED)*, commonly known as maximum likelihood TED. The name derivative TED arises due to utilizing the derivative of the matched filter output $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$.

Figure 7.29 illustrates from an eye diagram perspective why this is a valid error signal. For a pulse shaped and matched filtered PAM sequence $\pm A$ (no Q arm),

$z(mT_M + \hat{\varepsilon}_\Delta) \rightarrow$ Matched filtered output at $mT_M + \hat{\varepsilon}_\Delta$

$\dot{z}(mT_M + \hat{\varepsilon}_\Delta) \rightarrow$ Derivative of the matched filter output
(i.e., its slope) at $mT_M + \hat{\varepsilon}_\Delta$

In reference to this figure, we discuss the following four scenarios.

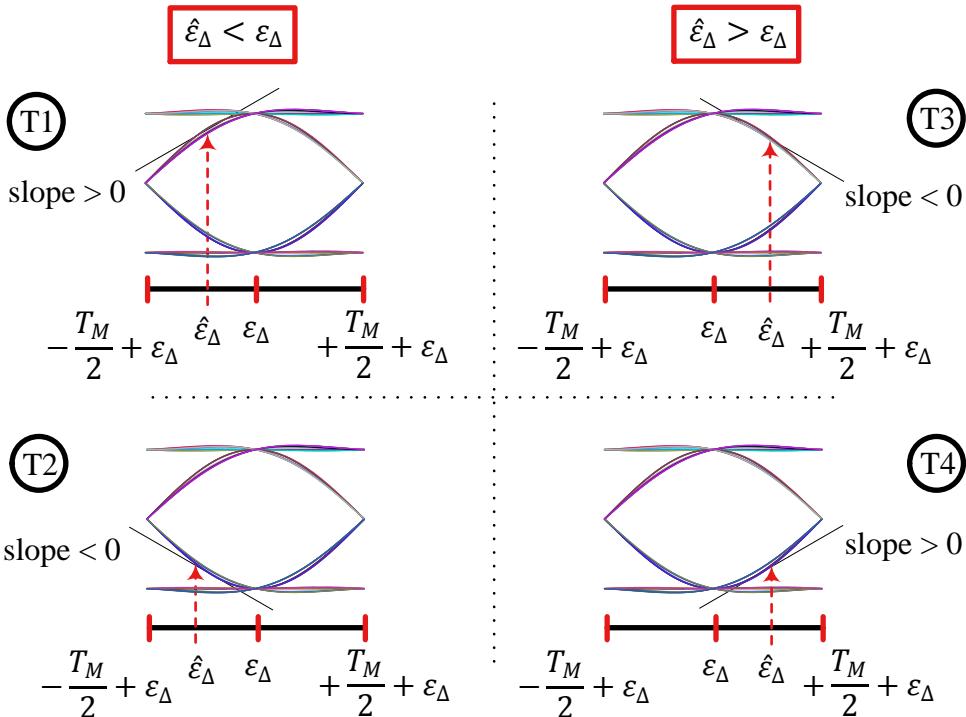


Figure 7.29: A PAM sequence eye diagram. —T1: $z(mT_M + \hat{\varepsilon}_\Delta) > 0$ and $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$. —T2: $z(mT_M + \hat{\varepsilon}_\Delta) < 0$ and $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$. —T3: $z(mT_M + \hat{\varepsilon}_\Delta) > 0$ and $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$. —T4: $z(mT_M + \hat{\varepsilon}_\Delta) < 0$ and $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$

T1: $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$ and $z(mT_M + \hat{\varepsilon}_\Delta) > 0$ When the current sampling instant $\hat{\varepsilon}_\Delta$ is earlier than the actual symbol delay ε_Δ , the value $z(mT_M + \hat{\varepsilon}_\Delta)$ in the top left of Figure 7.29 is naturally lesser than the peak $+A$. Hence, the derivative of the curve

is positive at that instant. A positive slope indicates that the estimate $\hat{\varepsilon}_\Delta$ needs to be increased at next update, i.e.,

$$\hat{\varepsilon}_\Delta(m+1) > \hat{\varepsilon}_\Delta(m)$$

So the product of $z(mT_M + \hat{\varepsilon}_\Delta)$ and the derivative yields the positive direction in which our symbol timing offset estimate $\hat{\varepsilon}_\Delta$ needs to move.

T2: $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$ and $z(mT_M + \hat{\varepsilon}_\Delta) < 0$ Again, the current value of $\hat{\varepsilon}_\Delta$ is earlier than the actual value but now the derivative at the sampling instant is negative. This is because the symbol value is $-A$ in this case. Consequently, a product of $z(mT_M + \hat{\varepsilon}_\Delta)$ with the derivative yields the positive direction to move $\hat{\varepsilon}_\Delta$.

T3: $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$ and $z(mT_M + \hat{\varepsilon}_\Delta) > 0$ Here, the symbol value is $+A$ while the derivative is negative. Thus, their product generates a negative error signal. Indeed, $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$ and it needs to be reduced in the next iteration.

$$\hat{\varepsilon}_\Delta[m+1] < \hat{\varepsilon}_\Delta[m]$$

T4: $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$ and $z(mT_M + \hat{\varepsilon}_\Delta) < 0$ Finally, the product of $z(mT_M + \hat{\varepsilon}_\Delta)$ and the positive slope produces a negative error term that correctly pulls the sampling instant back.

To check the validity of $e_D[m]$ as a valid error signal, observe that when $\hat{\varepsilon}_\Delta$ approaches ε_Δ closer to the peak sampling instant, the slope and consequently $e_D[m]$ goes to zero. This is exactly what maximizes the correlation in Eq (7.11).

In a data-aided version of the derivative TED, the matched filter output $z(mT_M + \hat{\varepsilon}_\Delta)$ is replaced by the actual symbol value $a[m]$.

$$e_D[m] = a[m] \cdot \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \quad (7.22)$$

Finally, during a steady state operation, the decisions from the detector can be deployed. Thus, the decision-directed derivative TED is constructed as

$$e_D[m] = \hat{a}[m] \cdot \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \quad (7.23)$$

For example, the symbol decision $\hat{a}[m]$ for a binary PAM case is

$$\hat{a}[m] = A \times \text{sign}\left\{ z(mT_M + \hat{\varepsilon}_\Delta) \right\}$$

Significance of Data Transitions

Recall from Section 3.8 that a single transition in an eye diagram gives information about 3 symbols:

NOW: at which the trace passes the current symbol (e.g., $+A$ or $-A$),

past: indicated by whether the trace is coming from an upward or a downward path, and

future: indicated by whether the trace is heading upward or downward.

With this information in mind, observe from Figure 7.29 the following cases where the triplet indicates {Past, NOW and Future}.

{+ - +}, or {- + -} When data transitions occur both before and after the current symbol, the slope of the pulses in the eye diagram is a reasonably large value at non-optimal instants. This instructs the proper direction to update the estimate $\hat{\varepsilon}_\Delta$ when multiplied with the data symbol.

{+++}, or {---} For no data transition, there is an approximately horizontal portion of the eye diagram at the top (case +) and at the bottom (case -). For this scenario, the actual value of the slope will be very small, no matter how far away the current sampling instant is from the optimal location. Consequently, there is no timing information that can be obtained.

The above discussion suggests that a sufficient density of data transitions is required for timing synchronization to operate.

The Two Filters

Just as in the case of derivative frequency error detector in Section 6.4.2, the derivative timing error detector needs not to compute the derivative of the Rx signal samples in real time. As in the case of that derivative frequency error detector, there can be two ways to proceed.

1. Pass the matched filtered output $z(nT_S)$ through a filter $h(nT_S)$ that computes the derivative of its input in time domain. Here, the matched filter and the derivative filter appear in series.
2. Create an alternative filter based on the *pre-computed derivative of the matched filter with respect to time*. To see why it works, assume that a filter $h(nT_S)$ such as $0.5\{+1, 0, -1\}$ from Section 2.6 computes the derivative of a signal in time domain.

$$\begin{aligned}\dot{z}(nT_S) &= z(nT_S) * h(nT_S) \\ &= \left\{ r(nT_S) * p(-nT_S) \right\} * h(nT_S) \\ &= r(nT_S) * \underbrace{\left\{ p(-nT_S) * h(nT_S) \right\}}_{h_{TMF}(nT_S)} = r(nT_S) * \left\{ -\dot{p}(-nT_S) \right\}\end{aligned}$$

Here, the left hand side is the derivative of the matched filter output while the right hand side is the convolution of the Rx signal with a filter $h_{TMF}(nT_S)$. Therefore, $h_{TMF}(nT_S)$ is known as a *Timing Matched Filter (TMF)* which is different than the frequency matched filter $h_{FMF}(nT_S)$ in Chapter 6.

$$h_{TMF}(nT_S) = -\dot{p}(-nT_S) \quad (7.24)$$

With this approach, the matched filter and the derivative matched filter appear in parallel because they both operate on the same input $r(nT_S)$.

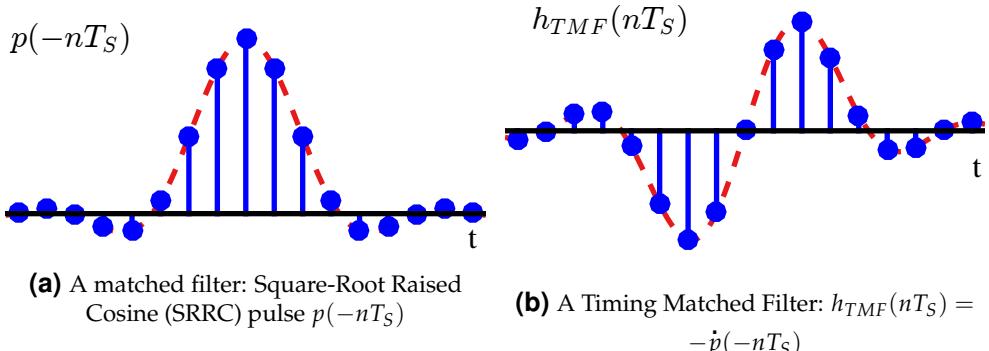


Figure 7.30: A discrete-time matched filter and the corresponding timing matched filter

As an example, let us construct a timing matched filter for a square-root Nyquist pulse, say a Square-Root Raised Cosine (SRRC), in discrete-time. When $p(-nT_S)$ is convolved with a derivative computing filter, the coefficients of a timing matched filter are generated which are plotted in Figure 7.30. The downsampled output of this filter is $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$ that is used in the derivative TED of Eq (7.22).

Although both approaches above require two filters, the difference is that the serial approach incurs an additional delay because the input for computing the derivative of the matched filter output is not available until the matched filter actually generates this output. A TLL or a PLL is an iterative solution in which any additional delay within the loop impacts its performance and hence the parallel approach of two filters operating on the same input samples at the same time proves more useful.

Unlike a frequency matched filter in which quarter cycle of the sine ends abruptly at the edges of the transition band and makes it difficult to realize, the timing matched filter is an easily realizable time series as shown in Figure 7.30b. Remember that an accurate representation of the derivative of a signal needs a sufficiently oversampled input. This makes the derivative TED a synchronization system where the input sample rate of the two filters is L samples/symbol and the output error signal is generated once per symbol.

Rx Structure for Derivative TED

Next, we look into the Rx structure that incorporates such a timing error detector. A block diagram for implementation of a Rx with a derivative TED in a decision-directed setting is shown in Figure 7.31.

The Rx signal $r(t)$ is sampled at a rate of $F_S = L/T_M$ to generate $r(nT_S)$ at approximately L samples/symbol. The matched filter and timing matched filter process $r(nT_S)$ in parallel at the same rate with their respective outputs being $z(nT_S)$ and $\dot{z}(nT_S)$. These two signals form the inputs to their respective interpolators operating in parallel that produce *one sample per symbol as commanded by interpolation control block* (we will learn about interpolation in Section 7.8 and interpolation control in Section 7.9). These two interpolator outputs are $z(mT_M + \hat{\varepsilon}_\Delta)$ and $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$ where the former is employed to make the decision $\hat{a}[m]$.

Subsequently, $\hat{a}[m]$ and $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$ combine into a timing error detector accord-

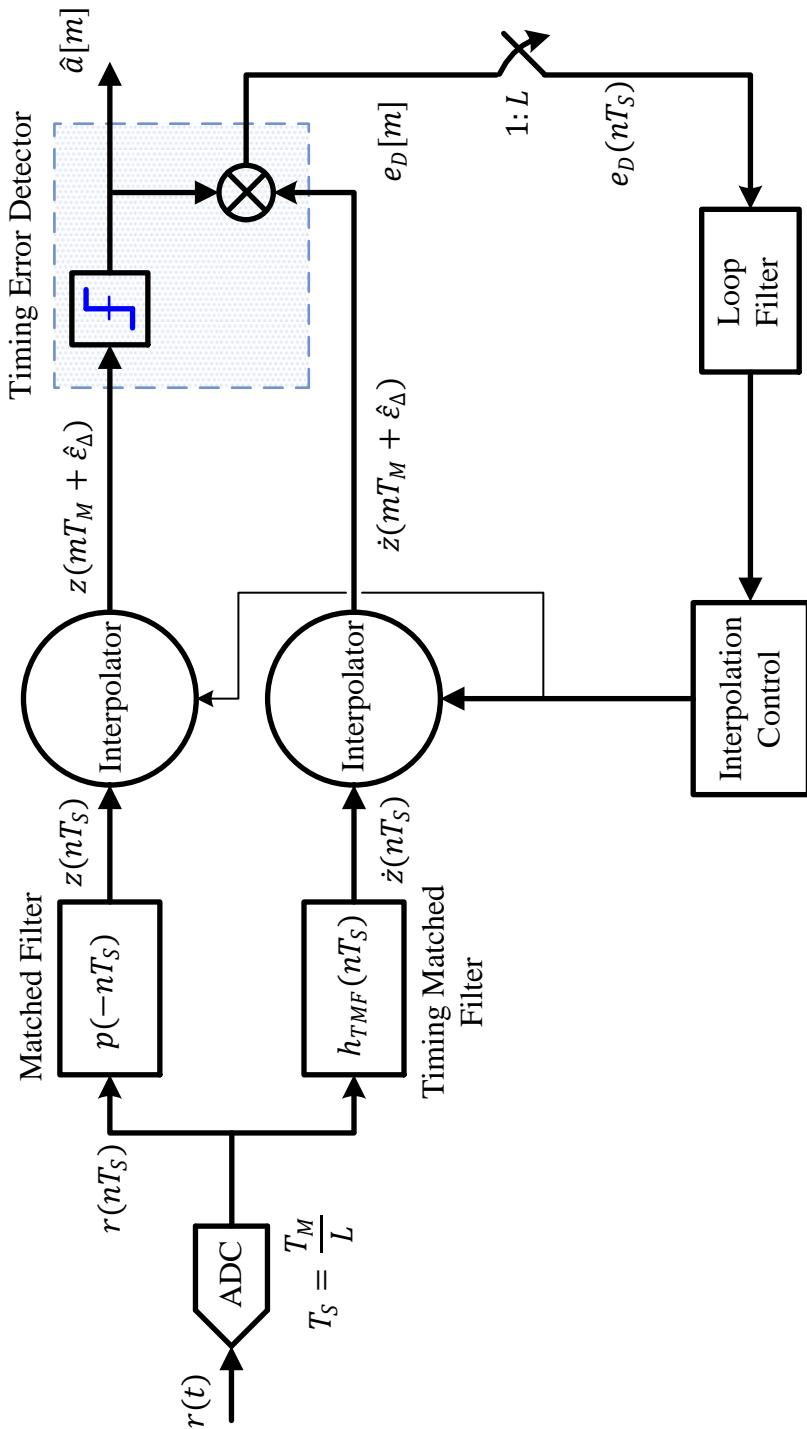


Figure 7.31: A timing locked loop with a derivative TED in a decision-directed setting ($L : 1$ downsampling not shown)

ing to Eq (7.23) and generate an error signal $e_D[m]$ once per symbol. In case of a non-data-aided operation, the sign block is removed and the matched filter output $z(mT_M + \hat{\varepsilon}_\Delta)$ is directly used instead of $\hat{a}[m]$ in forming the TED output. This signal $e_D[m]$ is then upsampled by L to create $e_D(nT_S)$ which matches the sample rate F_S of the loop filter and the interpolation control.

Mean Curve for Data-Aided Derivative TED

A mean curve is a key tool to investigate the timing acquisition behaviour of a TLL. It is formally known as an S-curve and was defined in Section 5.5.3 as the average of the phase detector output $e_D[m]$.

$$\text{S-curve : } \text{Mean}\{e_D[m]\} \equiv \bar{e}_D \quad \text{for each } \varepsilon_\Delta - \hat{\varepsilon}_\Delta \equiv \varepsilon_{\Delta:e} \quad (7.25)$$

As stated in that section, the ratio of its squared mean value to the variance around the origin is a quality metric for a TED. The mathematical expression for the mean curve of a derivative TED is derived in Appendix 7.15 and is given by

$$\bar{e}_D = -\gamma A^2 \cdot \dot{r}_p(\varepsilon_{\Delta:e}) \quad (7.26)$$

where γ is an amplitude scaling factor that arises from the gains and losses through the antennas, wireless channel and Tx/Rx frontend components and $\dot{r}_p(\cdot)$ is the time derivative of the overall Nyquist filter. This mean curve for a data-aided derivative TED is drawn in Figure 7.32 for $\alpha = 0.5$ within a span of $[-0.5T_M, +0.5T_M]$. A few comments are in order.

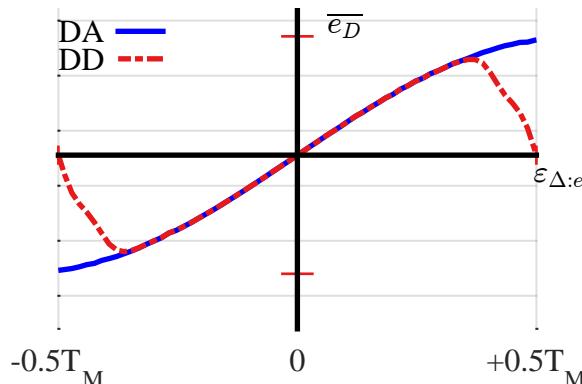


Figure 7.32: Mean curve for data-aided and decision-directed derivative TED for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidths $\alpha = 0.5$. Note the one symbol width span for a decision-directed TED

Positive slope As $r_p(nT_S)$ has a maximum at zero, $-\dot{r}_p(nT_S)$ has a null with positive slope at $\varepsilon_{\Delta:e} = 0$ thus leading to a stable loop operation.

Span of the mean curve From the expression, this mean curve is the derivative of a Nyquist pulse scaled by γA^2 , which was originally drawn in Figure 7.87. There, an arrow was marked from $-0.5T_M$ to $+0.5T_M$ as a hint towards the mean curve. As is evident, the data-aided mean curve does not converge towards zero at the symbol boundaries $[-0.5T_M, +0.5T_M]$. Instead, if we extend the same curve beyond $[-0.5T_M, +0.5T_M]$, it reaches zero even after $\pm T_M$.

This observation seems remarkable. Recall from the discussion on phase synchronizers in Chapter 5 that the phase mean curve is limited to the region $[-\pi, +\pi]$ because the phase itself is limited to the region $[-\pi, +\pi]$. However, the difference here is that while a timing offset is constrained to lie within $[-0.5T_M, +0.5T_M]$, the pulse shape $p(nT_S)$ and its auto-correlation $r_p(nT_S)$ extend to several symbols outside this region. Therefore, a TED with known data has some influence outside the primary duration spanning T_M seconds. On the other hand, a decision-directed TED or a non-data-aided TED behaves in a conventional manner within $[-0.5T_M, +0.5T_M]$ as we find next.

Decision-directed case: For the decision-directed scenario, the mean curve is the same as long as the decisions are correct which happens for $|\varepsilon_{\Delta:e}| < 0.35T_M$. For higher values of $\varepsilon_{\Delta:e}$, the assumption $\hat{a}[m] = a[m]$ used to derive the decision-directed mean curve is no longer valid. It is evident that the decision-directed TED results in positive slope of the mean curve occurring with a period of T_M independent of the underlying modulation scheme. This is in contrast with the mean curves for phase error detectors in Chapter 5 where the slope occurs at multiple locations within the interval $[-\pi, +\pi]$ for decision-directed case with the period depending on underlying modulation.

Next, we find out what significance this mean curve holds in the grand scheme of timing compensation.

Quality of a Timing Metric

In Section 7.5, we said that a TED converges to the proper value as long as the mean timing error has the same sign as the actual timing offset. The question now is to gauge how quickly the convergence takes place for a particular TED.

To answer this question, we construct an effective ‘SNR’ of the TED where the signal part and the noise part need to be defined. First, a mean curve for a non-data-aided scenario for a binary PAM sequence, $\gamma A^2 = 1$ and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.1, 0.5$ and 1 is drawn in Figure 7.33a.

Clearly, a higher excess bandwidth produces a larger slope at $\varepsilon_{\Delta:e} = 0$ because it provides more energy available for synchronization. This gives a hint towards the magnitude of the timing error around $\varepsilon_{\Delta:e} = 0$ as a measure of signal part during a steady state operation. So we define the signal part as the squared slope of mean curve at $\varepsilon_{\Delta:e} = 0$. Similarly, the self noise power generated within the TED can be measured through drawing the standard deviation (which is the square root of the variance) of the same timing function and shown in Figure 7.33b. As a result, an effective quality metric for a TED operation is the ratio of the above two terms.

$$\text{TED quality metric} = \frac{\text{squared slope of mean curve at } \varepsilon_{\Delta:e} = 0}{\text{error variance at } \varepsilon_{\Delta:e} = 0} \quad (7.27)$$

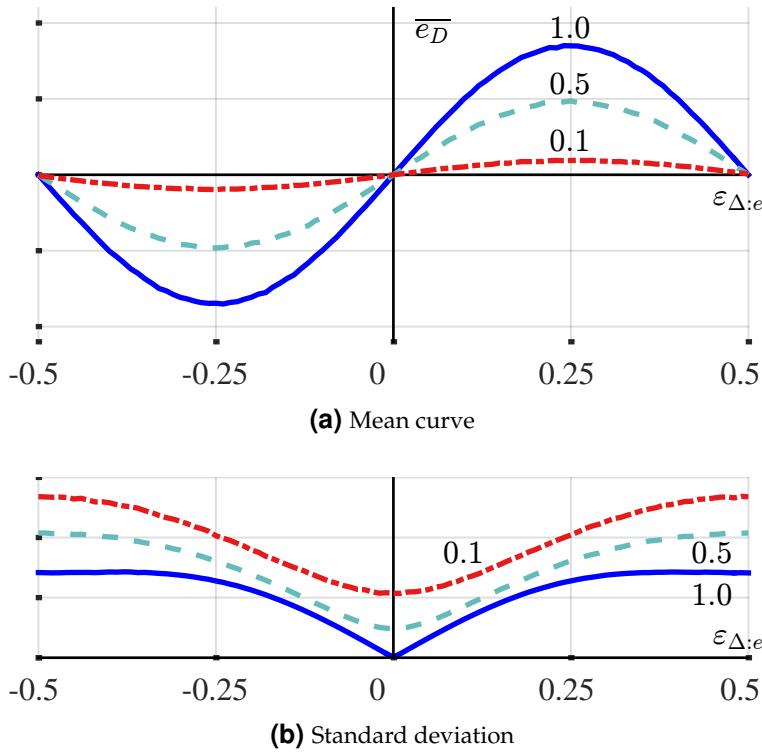


Figure 7.33: Non-data-aided derivative TED for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidths $\alpha = 0.1, 0.5$ and 1

Usually, the square-root of the above quantities is also used and this is why standard deviation, and not the variance, is drawn in Figure 7.33b. The larger this metric is, the faster the convergence. Also keep in mind that sometimes much of the self noise component can be filtered out through appropriate techniques (an example of which we will see later). Nevertheless, the channel noise still remains and plays its role in convergence and tracking performances.

Note 7.4 The curious case of a flipped mean curve

There are many resources which treat the mean curve for a timing error detector in a flipped form where the curve has a negative slope at $\epsilon_{\Delta e} = 0$. While the shape resembles a positive 'S' in the usual case, it is a negative 'S' in the latter. While both approaches are correct, this has been a source of a great deal of confusion and has sometimes lead to erroneous forms of some timing error detectors. Let us explore the root cause of this difference in understanding.

A timing error ϵ_{Δ} is introduced in a PAM waveform as

$$z(t) = \gamma \sum_i a[i] r_p(t - iT_M - \epsilon_{\Delta})$$

It is sampled at $t = nT_S + \hat{\varepsilon}_\Delta$ ($\hat{\varepsilon}_\Delta$ comes with a positive sign) which yields

$$\begin{aligned} z(nT_S + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p[nT_S + \hat{\varepsilon}_\Delta - iT_M - \varepsilon_\Delta] \\ &= \gamma \sum_i a[i] r_p[nT_S - iT_M - \varepsilon_{\Delta:e}] \end{aligned} \quad (7.28)$$

When $\varepsilon_{\Delta:e} > 0$, i.e., $\varepsilon_\Delta > \hat{\varepsilon}_\Delta$, our estimate $\hat{\varepsilon}_\Delta$ should increase. Similarly, when $\varepsilon_{\Delta:e} < 0$, i.e., $\varepsilon_\Delta < \hat{\varepsilon}_\Delta$, our estimate $\hat{\varepsilon}_\Delta$ should decrease. This is illustrated in Figure 7.34a and the resulting mean curve has a *positive slope* at the origin.

On the other hand, many scientists define $\varepsilon_\Delta = 0$ and assume as before that the Rx has the responsibility of properly adjusting the timing shift $\hat{\varepsilon}_\Delta$. In that case, the matched filter output is

$$z(t) = \gamma \sum_i a[i] r_p(t - iT_M)$$

and again it is sampled at $t = nT_S + \hat{\varepsilon}_\Delta$.

$$z(nT_S + \hat{\varepsilon}_\Delta) = \gamma \sum_i a[i] r_p[nT_S - iT_M + \hat{\varepsilon}_\Delta]$$

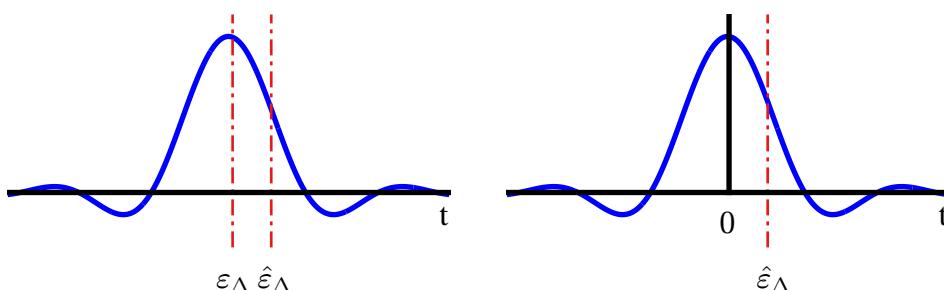
As opposed to an overall timing error with a negative sign in Eq (7.28), the overall timing error appears with a positive sign here. When our estimate $\hat{\varepsilon}_\Delta < 0$, it should increase. Similarly, when $\hat{\varepsilon}_\Delta > 0$, it should decrease. This is illustrated in Figure 7.34b and the resulting mean curve has a *negative slope* at the origin.

This is why the mean curve of a timing error detector is found flipped in many resources.

Interestingly, there is another aspect here as compared to carrier phase and frequency synchronizers that requires a further comment.

Phase synchronization First, recall from Eq (5.3) that a carrier phase rotates the signal in an anti-clockwise direction.

$$\begin{array}{ll} I \rightarrow & z_I(mT_M) = a_I[m] \cos \theta_\Delta - a_Q[m] \sin \theta_\Delta \\ Q \uparrow & z_Q(mT_M) = a_Q[m] \cos \theta_\Delta + a_I[m] \sin \theta_\Delta \end{array}$$



(a) Generating a mean curve with a positive slope (b) Generating a mean curve with a negative slope

Figure 7.34: A mean curve for a timing error detector can exhibit a negative slope at origin depending on how the timing error is defined

Next, recall how the phase synchronization block de-rotates the matched filter outputs by its updated estimate $\hat{\theta}_\Delta$ (see Eq (5.17), for example).

$$\begin{array}{ll} I & \rightarrow \quad \widehat{z}_I(mT_M) = z_I(mT_M) \cos \hat{\theta}_\Delta + z_Q(mT_M) \sin \hat{\theta}_\Delta \\ Q & \uparrow \quad \widehat{z}_Q(mT_M) = z_Q(mT_M) \cos \hat{\theta}_\Delta - z_I(mT_M) \sin \hat{\theta}_\Delta \end{array}$$

It is evident that the carrier phase error θ_Δ is introduced with a positive sign while the compensator introduces a phase correction term $\hat{\theta}_\Delta$ with a negative sign[†]. The overall effect is that the system experiences a *positive phase rotation* of

$$\theta_{\Delta:e} = \theta_\Delta - \hat{\theta}_\Delta$$

Timing synchronization The reverse is true in the case of timing synchronization. A timing error ε_Δ is introduced with a negative sign (as a delay) while the compensator samples the waveform with a positive sign as $nT_S + \hat{\varepsilon}_\Delta$. The overall effect is that the system experiences a *negative timing error* of

$$\varepsilon_{\Delta:e} = \varepsilon_\Delta - \hat{\varepsilon}_\Delta$$

However, as described previously, the derivative of the pulse auto-correlation $\dot{r}_p(nT_S)$ is

$$\dot{r}_p(nT_S) = p(nT_S) * \{-\dot{p}(-nT_S)\}$$

that balances the previous negative sign, again leading to a mean curve with a positive slope.

Note 7.5 Detector gain K_D

The detector gain K_D is defined as the slope of the mean curve $\overline{e_D}$ at $\varepsilon_{\Delta:e} = 0$. Assume $A = 1$ in the mean curve expression of Eq (7.26) for the derivative TED

$$\overline{e_D} = -\gamma \cdot \dot{r}_p(\varepsilon_{\Delta:e})$$

Thus, there are two dependencies for the gain K_D .

Rx signal amplitude γ Remember from Eq (4.7) that loop filter coefficients K_p and K_i involve K_D in their expressions. If the input signal level γ is not controlled, loop filter will have incorrect coefficients and the design will not perform according to assumed noise bandwidth and damping factor. In a wireless receiver, this amplitude is maintained at a reference value by using an Automatic Gain Control (AGC), instead of updating the loop coefficients with changing γ in a mobile environment.

Excess bandwidth α The TED gain also depends on the slope of the pulse auto-correlation, $\dot{r}_p(nT_S)$, at $\varepsilon_{\Delta:e} = 0$. Since $r_p(nT_S)$ varies according to the value of α , gain K_D is also a function of the excess bandwidth α . This relationship can also be traced back to the role of excess bandwidth in determining the strength of the timing line in Section 7.4.2.

[†]In complex notation, this is $\exp(j\theta_\Delta) \cdot \exp(-j\hat{\theta}_\Delta) = \exp(j(\theta_\Delta - \hat{\theta}_\Delta))$.

Next, we examine the implementation of a derivative TED for the following specifications.

Modulation \rightarrow 2 – PAM, $L = 8$ samples/symbol, $\alpha = 0.4$,

$$B_n T_M = 1/200, \quad \zeta = 1/\sqrt{2}, \quad \varepsilon_\Delta = 0.25T_M$$

With these specifications for a timing locked loop, the PI loop filter coefficients are computed through Eq (4.7). For this purpose, the NCO gain K_0 and TED gain K_D are also required. We will see in Section 7.9 that K_0 is unity while a simple routine is used to compute the derivative of the mean curve at $\varepsilon_\Delta = 0$ for K_D . The output error signal $e_D[m]$ for a derivative TED is shown in Figure 7.35a and the estimate of the timing offset $\varepsilon_\Delta = 0.25T_M$ is seen converging to its true value in Figure 7.35b. Later in Section 7.8, we will call this estimate as a fractional interval $\mu[m]$. The TLL is seen to converge in approximately 300 symbols as a result of these design parameters. Observe the abundance of self noise produced by a derivative TED, some of which is inherently present in this TED while the rest is contributed due to the non-data-aided version implemented here.

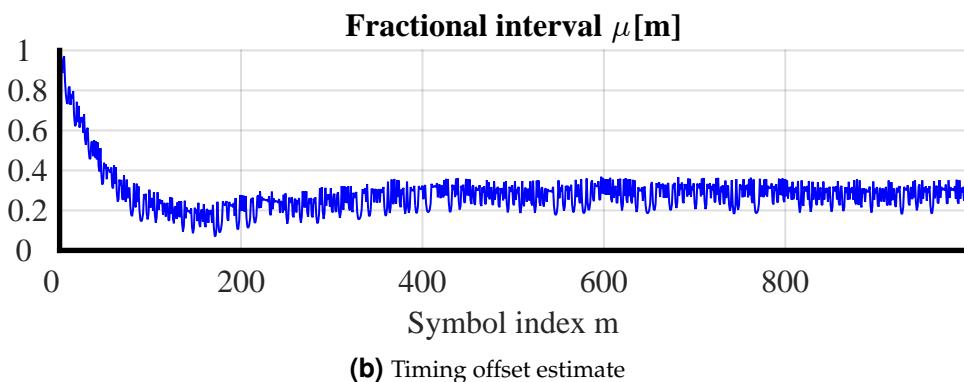
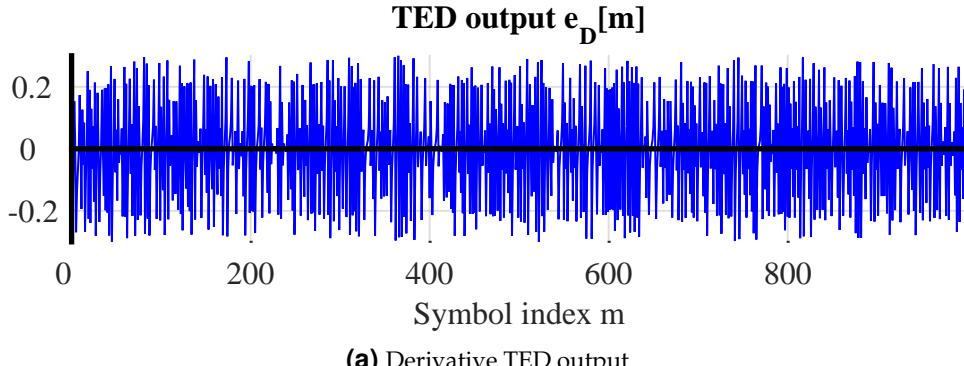


Figure 7.35: A timing locked loop with a derivative TED converging to the steady state value for an SRRC pulse with excess bandwidth $\alpha = 0.4$

Finally, the concept of a derivative TED carries over to a QAM scheme in the same manner but with the addition of inphase and quadrature components to form the TED output. From the complex rule of multiplication, this is equivalent to *multiplying*

the matched filter output with the conjugate of the derivative matched filter output and then taking the inphase part. For example, the data-aided TED from Eq (7.22) is given by

$$e_D[m] = a_I[m] \cdot \dot{z}_I(mT_M + \hat{\varepsilon}_\Delta) + a_Q[m] \dot{z}_Q(mT_M + \hat{\varepsilon}_\Delta) \quad (7.29)$$

Drawback of the Derivative TED

We saw above that the timing matched filter is constructed by computing the derivative of the matched filter and consequently its output is the derivative of the input signal. Naturally, this output is more fine-grained and hence accurate when the number of samples/symbol (denoted by L) is large. Here, L must be several times larger than the minimum limit set by the Nyquist theorem.

However, in most applications, reducing the complexity of the TLL is far more desirable than achieving the gains from the increased granularity. Instead of having a vastly oversampled Rx signal, a simpler form of a timing error detector is therefore attractive for system design operating at no more than 1 or 2 samples/symbol. Due to this reason, there are many approximations to the derivative TED that have been derived over the years. Some of those new TEDs were invented in an ad-hoc fashion that turned out to be just another of its approximations.

Next, we cover another popular timing error detector that is based on the derivative principle.

7.6.2 Feedback: Early-Late Timing Error Detector

Assume that the Rx signal is sampled at $L = 2$ samples/symbol. In this case, the matched filter output $z(nT_S)$ has every other sample that is a candidate of the symbol decision, i.e., $m = 2n$. *To keep the indexing in terms of symbols rather than samples*, a better notation is to use mT_M for the main sample as before and use $\pm T_M/2$ for the intermediate sample.

$$\dots, z((m-1)T_M + \hat{\varepsilon}_\Delta), z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right), z(mT_M + \hat{\varepsilon}_\Delta), \\ z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right), z((m+1)T_M + \hat{\varepsilon}_\Delta), \dots$$

In the above time series of the matched filter output,

- $z((m-1)T_M + \hat{\varepsilon}_\Delta)$ is a candidate for symbol decision for symbol $m-1$,
- $z(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$ is an *intermediate sample* only useful for TED operation,
- $z(mT_M + \hat{\varepsilon}_\Delta)$ is a candidate for symbol decision for symbol m ,
- $z(mT_M + T_M/2 + \hat{\varepsilon}_\Delta)$ is an *intermediate sample* only useful for TED operation, and
- $z((m+1)T_M + \hat{\varepsilon}_\Delta)$ is a candidate for symbol decision for symbol $m+1$.

Now recall that a popular approximation to compute the derivative of a waveform at a time n is the first central difference filter

$$h[n] = \{+1, 0, -1\}$$

where the scaling factor $1/2$ is removed for simplification. A reduced complexity solution is to just apply this filter to the matched filter output for finding its approximate slope. For this purpose, concentrate on the matched filter output $z(mT_M)$ and the two samples around it, that is to say

$$z\left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta\right), z(mT_M + \hat{\epsilon}_\Delta), z\left(mT_M + \frac{T_M}{2} + \hat{\epsilon}_\Delta\right)$$

These samples are drawn in Figure 7.36, where *the slope is being computed at time $mT_M + \hat{\epsilon}_\Delta$* . In this scenario,

- the symbol candidate $z(mT_M + \hat{\epsilon}_\Delta)$ corresponds to the symbol $a[m]$ (sometimes referred to as ‘prompt’),
- $z(mT_M - T_M/2 + \hat{\epsilon}_\Delta)$ is an early sample, and

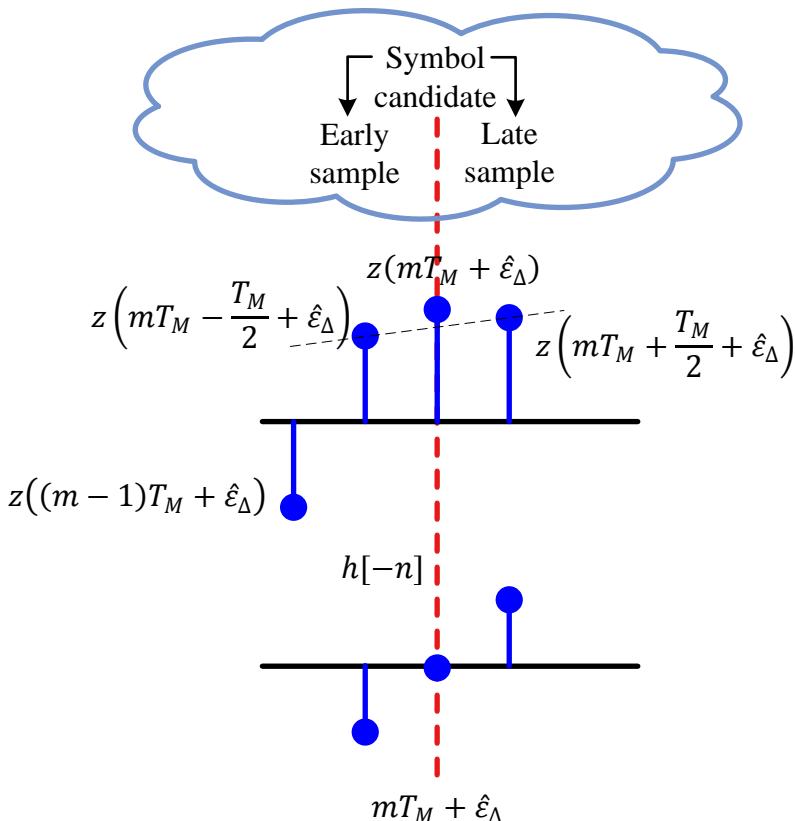


Figure 7.36: Convolution between the matched filter outputs $z(nT_S)$ taken at 2 samples per symbol and $h[n]$. Recall the seesaw perspective from Figure 7.28

- $z(mT_M + T_M/2 + \hat{\varepsilon}_\Delta)$ is a late sample.

These early and late samples, namely $z(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$ and $z(mT_M + T_M/2 + \hat{\varepsilon}_\Delta)$ respectively, can be employed for computing the derivative because they overlap with the coefficients -1 and $+1$ of $h[-n]$ during the convolution process, as shown in Figure 7.36 (remember one of the signals, $h[n]$ here, is reversed during convolution). Recall the seesaw perspective from Figure 7.28 to further clarify how an early-late TED tries to balance the input samples.

A timing error detector operates at the output of the interpolator. That is, a TED input is the samples interpolated at certain locations depending on $\hat{\varepsilon}_\Delta$. Thus, a non-data-aided *Early-Late Timing Error Detector (TED)* can be constructed by replacing the derivative term $\dot{z}(mT_M + \hat{\varepsilon}_\Delta)$ in the derivative TED of Eq (7.21) as

$$e_D[m] = z(mT_M + \hat{\varepsilon}_\Delta) \left\{ z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) - z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) \right\}$$

(7.30)

Naturally, its data-aided version can also be made by using the known symbol $a[m]$ in place of $z(mT_M + \hat{\varepsilon}_\Delta)$.

$$e_D[m] = a[m] \left\{ z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) - z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) \right\} \quad (7.31)$$

On the same note, a decision-directed early-late TED can be created as

$$e_D[m] = \hat{a}[m] \left\{ z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) - z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) \right\} \quad (7.32)$$

Rx Structure for Early-Late TED

We now look into the Rx structure for an early-late TED for which a block diagram in a decision-directed setting is shown in Figure 7.37. The Rx signal $r(t)$ is sampled at a rate of $F_S = 2/T_M$, or $T_S = T_M/2$, to generate $r(nT_S)$ at $L = 2$ samples/symbol. Next, the sampled signal $r(nT_S)$ is matched filtered at the same rate with its output being $z(nT_S)$. Under the command of an interpolation control block, an interpolator resamples these matched filter outputs at instants $\hat{\varepsilon}_\Delta$ to produce $z(nT_S + \hat{\varepsilon}_\Delta)$. Since this system runs at a rate of 2 samples/symbol, the delay block T_S (that represents a delay of 1 sample) supplies the samples at half a symbol duration, i.e., $z(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$ and $z(mT_M + T_M/2 + \hat{\varepsilon}_\Delta)$.

The interpolation control block also identifies the time instants mT_M at which the output $\hat{a}[m]$ is taken out of the decision block and the error signal $e_D[m]$ is formed once per symbol according to Eq (7.32). This is because the decisions and hence the early-late TED output should be constructed only every other sample, not each sample. In the case of non-data-aided operation, the matched filter output $z(mT_M + \hat{\varepsilon}_\Delta)$ is directly used instead of $\hat{a}[m]$ in forming the TED output. This signal $e_D[m]$ is then upsampled by 2 to create $e_D(nT_S)$ which matches the sample rate F_S of the loop filter and the interpolation control. The role of the interpolation control outputs will be explained shortly in Section 7.9.

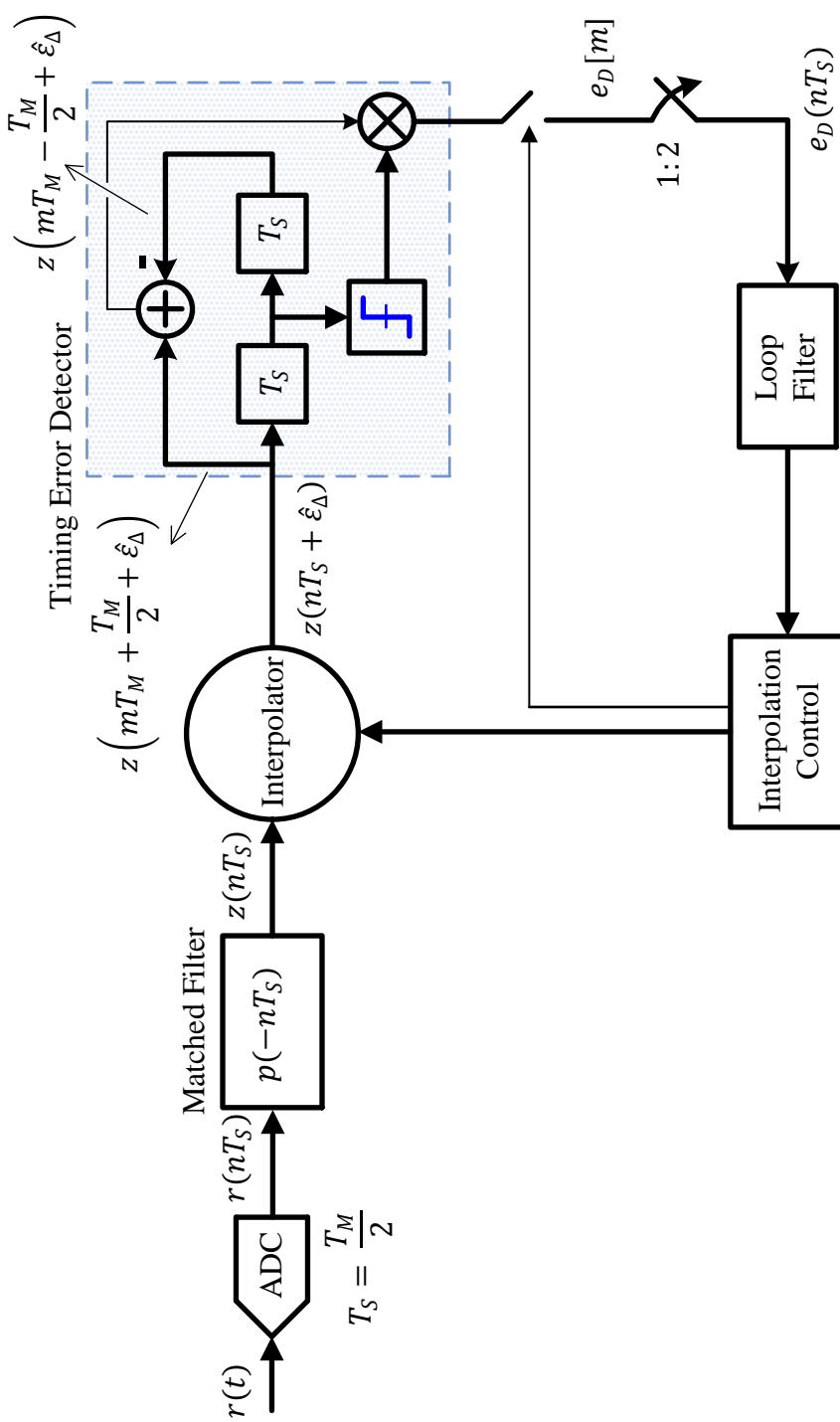


Figure 7.37: A timing locked loop with an early-late TED in a decision-directed setting

Mean curve for Data-Aided Early-Late TED

We consider the derivation of the mean curve for the data-aided version of early-late TED. While the mean curve for decision-directed scenario has only a minor difference as we shortly see, that for the non-data-aided type resembles the mean curve of the Gardner TED discussed later.

The mathematical expression for the mean curve of a data-aided early-late TED is derived in Appendix 7.15 and is given by

$$\overline{e_D} = \gamma A^2 \left\{ r_p \left(+\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) - r_p \left(-\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) \right\} \quad (7.33)$$

Compare this mean curve with that of the derivative TED in Eq (7.26). The only difference is that the derivative of the pulse auto-correlation $r_p(nT_S)$ is actually computed in the latter, while the derivative is approximated through $r_p(nT_S)$ half a symbol time before and after $-\varepsilon_{\Delta:e}$ in the former.

For this reason, as $r_p(nT_S)$ has a maximum at zero, $\overline{e_D}$ has a null with positive slope at $\varepsilon_{\Delta:e} = 0$ thus leading to a stable loop operation. It is drawn for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.1, 0.5$ and 1 in Figure 7.38.

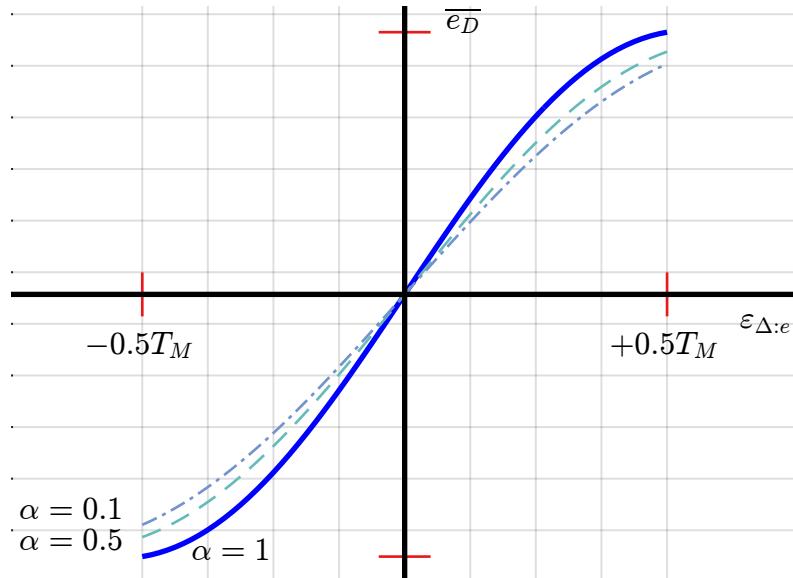


Figure 7.38: Mean curve for data-aided early-late TED for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.1, 0.5$ and 1

Also, a *higher excess bandwidth produces a larger slope at $\varepsilon_{\Delta:e} = 0$* leading to a TED that is more sensitive to timing variations. For the decision-directed scenario, the mean curve is the same as long as the decisions are correct which happens for $|\varepsilon_{\Delta:e}| < 0.35T_M$. For higher values of $\varepsilon_{\Delta:e}$, the assumption $\hat{a}[m] = a[m]$ used to derive the decision-directed mean curve is no longer valid.

As mentioned above, the mean curve for non-data-aided case is not pursued in this text. The comments on TED gain K_D mentioned in Note 7.5 in regards to derivative TED apply in this scenario as well. Finally, similar to the case in derivative TED, the error output for a QAM scheme can be written as a sum of inphase and quadrature parts which is the same as the product of the first signal with the conjugate of the other and then taking the inphase part. For example, from Eq (7.31),

$$e_D[m] = a_I[m] \left\{ z_I \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z_I \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\} + \\ a_Q[m] \left\{ z_Q \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z_Q \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\}$$

Alternative Forms of Early-Late TED

Another more familiar form of an early-late TED can now be understood starting from the fundamental Eq (7.20). Simply apply the definition of a derivative in discrete domain to get

$$\frac{\partial}{\partial \hat{\varepsilon}_\Delta} |z(mT_M + \hat{\varepsilon}_\Delta)|^2 \approx \frac{|z(mT_M + \hat{\varepsilon}_\Delta + \delta)|^2 - |z(mT_M + \hat{\varepsilon}_\Delta - \delta)|^2}{2\delta}$$

where δ is a fraction of the symbol time. Ignoring the constant 2δ in the denominator, an error signal can be formed as

$$e_D[m] = |z(mT_M + \hat{\varepsilon}_\Delta + \delta)|^2 - |z(mT_M + \hat{\varepsilon}_\Delta - \delta)|^2$$

Another version of this TED can be formed as

$$e_D[m] = |z(mT_M + \hat{\varepsilon}_\Delta + \delta)| - |z(mT_M + \hat{\varepsilon}_\Delta - \delta)|$$

It is obvious that the former version is called the *square law early-late TED* while the latter is called the *absolute value early-late TED*.

These are known as non-coherent version of the early-late TED. To understand their operation, consider Figure 7.16d plotting an average squared trajectory for three different excess bandwidths α and reproduced in Figure 7.39. Recalling the seesaw perspective from Figure 7.28, the ‘heavier’ side yields the error sign for subsequent correction. Computing difference in such a manner has the benefit that there is no need to use the matched filter output $z(mT_M + \hat{\varepsilon}_\Delta)$, or its decision $\hat{a}[m]$ for the TED output. The difference is zero when both the early and late samples reach the same value on the average squared curve, see Figure 7.39. The main disadvantage is the extra sample required at the peak of the matched filter output for detection purpose.

Yet other versions of an early-late TED also exist. One such type of early-late TED is described as

$$e_D[m] = z(mT_M + \hat{\varepsilon}_\Delta) \{ z(mT_M + T_S + \hat{\varepsilon}_\Delta) - z(mT_M - T_S + \hat{\varepsilon}_\Delta) \}$$

where T_S is the sample rate of the system, often chosen as $T_M/4$ for accessing the two sizeable slopes. At a sample rate of $L = 4$ samples/symbol, this is relatively an inefficient method emulated from the analog period to implement digital transceivers.

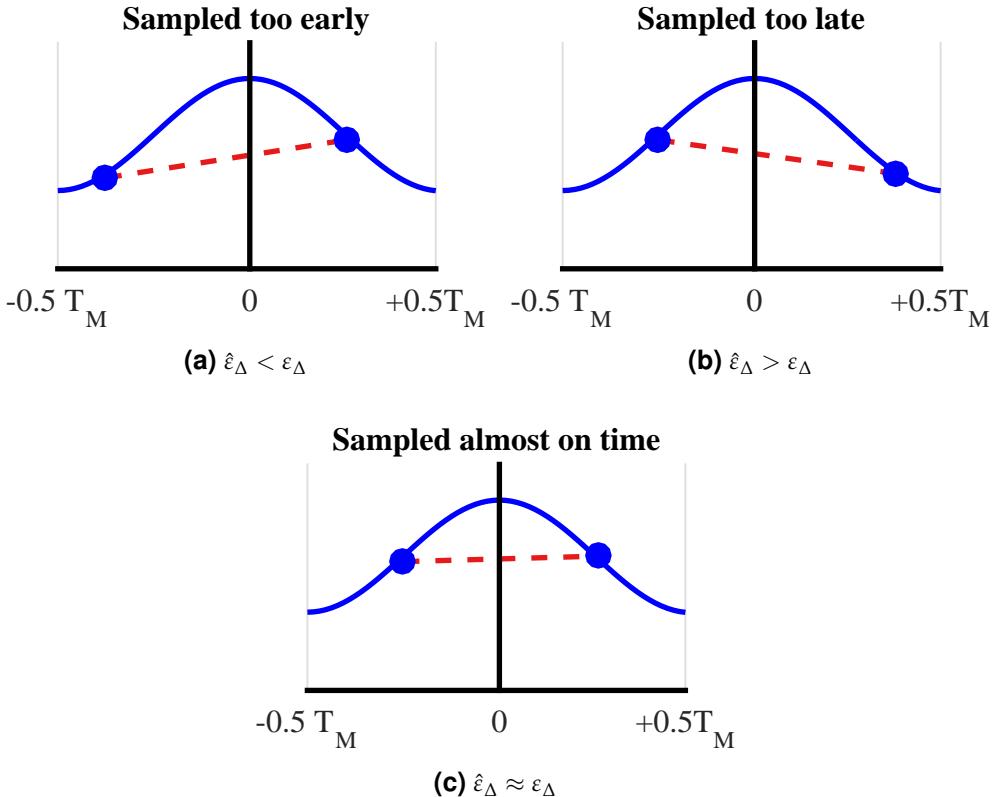


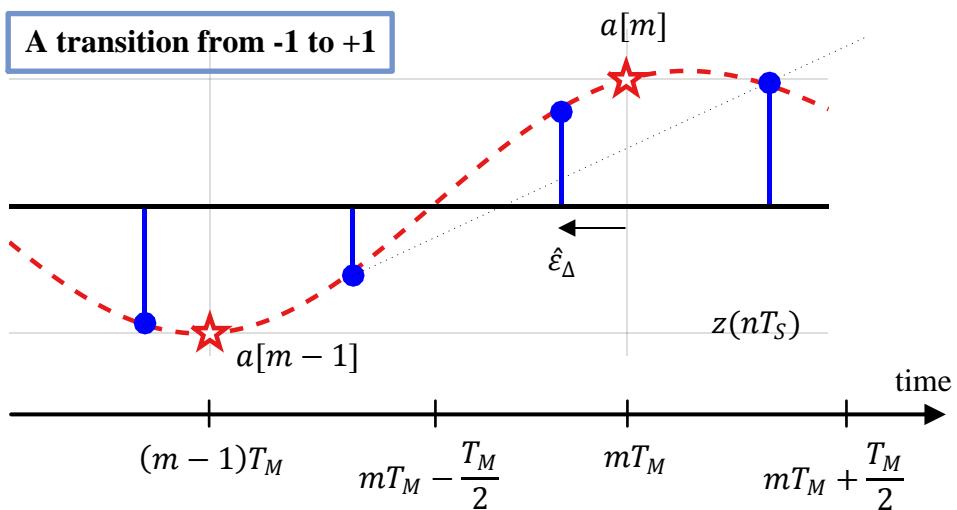
Figure 7.39: Average squared eye diagram for a binary PAM sequence of 400 symbols shaped with Raised Cosine with $\alpha = 1$ to demonstrate squared and absolute value forms of early-late TED. Note that there is no need to multiply with $z(mT_M + \hat{\varepsilon}_\Delta)$ or $\hat{a}[m]$

Early-late TED has been quite popular for timing recovery applications even before the digital era and shows continued interest during the subsequent evolution towards digital signal processing techniques. It has been widely used for timing recovery applications not only in digital communication systems but in global positioning systems as well. It is also reminiscent of the dual-window range gate scheme applied in Distance Measuring Equipment (DME) with an early and a late range gate aligned for pulse arrival.

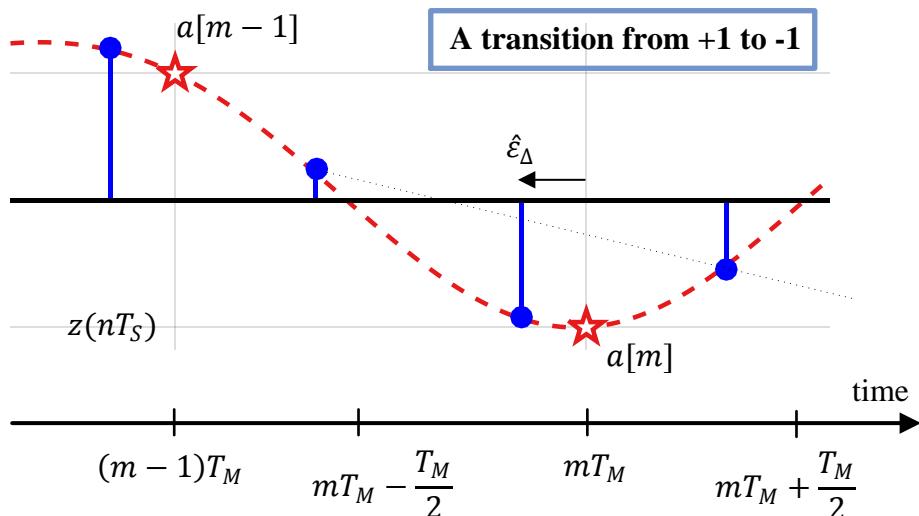
7.6.3 Why Symbol Centric?

The derivative TED, early-late TED and its other such approximations are symbol centric timing error detectors. To understand what this actually means, first consider how an early-late TED works. Start with Figure 7.40 of the matched filter output $z(nT_S)$ where the timing offset estimate is earlier than required, i.e., $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$. *Pay close attention to time axis in this figure.*

Transition from -1 to $+1$: Focusing first on Figure 7.40a, observe that during a trans-



(a) Both the slope and symbol $a[m]$ are positive



(b) Both the slope and symbol $a[m]$ are negative

Figure 7.40: Timing offset estimate is early, i.e., $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$. Early-late TED provides correct sign for timing offset to update in the next iteration

sition from a -1 to a $+1$, the slope is positive.

$$z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) - z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) > 0$$

Since the data symbol $a[m]$ is $+1$, $z(mT_M + \hat{\varepsilon}_\Delta)$ is positive and therefore,

$$e_D[m] = + \left\{ z \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\} > 0$$

Thus, the early-late TED provides the correct sign for the TLL to advance its timing estimate $\hat{\varepsilon}_\Delta$ at the next iteration.

Transition from $+1$ to -1 : Now from Figure 7.40b, observe that during a transition from a $+1$ to a -1 , the slope is negative.

$$z \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) < 0$$

Since the data symbol $a[m]$ is -1 , $z(mT_M + \hat{\varepsilon}_\Delta)$ is negative and therefore

$$e_D[m] = - \left\{ z \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\} > 0$$

Again, the early-late TED provides the correct sign for the TLL to advance its timing estimate $\hat{\varepsilon}_\Delta$ at the next iteration.

For the scenario where the timing offset estimate is late, i.e., $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$, the product of the slope and the symbol is negative to drive the sampling instant backwards. In a nutshell, the early-late TED seeks a time instant where the slope is zero, i.e., in this game of 3 samples, *the middle sample approaches the symbol center*.

Note 7.6 Symbol centric perspective

A derivative TED is a generalized version of this idea.

$$e_D[m] = z(mT_M + \hat{\varepsilon}_\Delta) \cdot \underbrace{\dot{z}(mT_M + \hat{\varepsilon}_\Delta)}_{\text{drive this term towards zero}} \quad (7.34)$$

The maximum correlation (which comes from maximum likelihood) theory is about finding the maximum of the correlation (on average). This maximum occurs where *the slope of the curve, and hence the derivative, approaches zero*. This approach is shown graphically in the eye diagram of Figure 7.41 where solid blue arrows indicate a positive or a negative symbol for early sampling while dashed red arrows indicate a positive or a negative symbol for the case of late sampling.

Next, we discuss an alternative approach to solving the timing synchronization problem.

7.7 Feedback: Zero Crossing (Gardner) Timing Error Detector

You might have noticed that we made a subtle assumption during the discussion on early-late TED as follows.

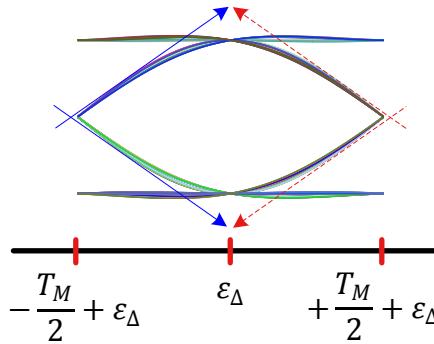


Figure 7.41: Symbol centric approach to solving the timing synchronization problem

A Subtle Assumption

The samples at the matched filter output are

$$\dots, z((n-2)T_S), z((n-1)T_S), z(nT_S), z((n+1)T_S), z((n+2)T_S), \dots$$

The question is: *how did we know which sample corresponds to $z(mT_M + \hat{\varepsilon}_\Delta)$?* At the startup time of the Rx, all we have is a series of samples spaced by half-symbol period $T_M/2$. Depending on the actual ε_Δ , we could have

$$\begin{aligned} z((n-2)T_S) &\rightarrow z((m-1)T_M + \hat{\varepsilon}_\Delta) \\ z((n-1)T_S) &\rightarrow z(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta) \\ z(nT_S) &\rightarrow z(mT_M + \hat{\varepsilon}_\Delta) \\ z((n+1)T_S) &\rightarrow z(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta) \\ z((n+2)T_S) &\rightarrow z((m+1)T_M + \hat{\varepsilon}_\Delta) \end{aligned}$$

or we could have either $z((n-1)T_S)$ or $z((n+1)T_S)$ correspond to $z(mT_M + \hat{\varepsilon}_\Delta)$ with other samples identified around accordingly[†].

To answer this question, consider again Figure 7.40a and notice that no matter which two samples we had chosen for finding the slopes, the algorithm would have brought the middle sample at the symbol center. It is redrawn as Figure 7.42 with samples denoted as b, c, d and e . Applying the early-late equation for three TED samples b, c and d (as we did before),

$$e_D[m] = c \cdot (b - d) > 0$$

However, if we had identified the three TED samples as c, d and e , then

$$e_D[m] = d \cdot (c - e) \quad (7.35)$$

Since $c - e > 0$ and d is negative,

$$d \cdot (c - e) < 0$$

[†]Frame synchronization can take us to the correct symbol level only, not the sample.

Instead of $\hat{\varepsilon}_\Delta < \varepsilon_\Delta$, the early-late TED then would treat this as a case of late sampling $\hat{\varepsilon}_\Delta > \varepsilon_\Delta$ (and hence the negative output in the above expression). The TLL would have brought the sampling instant earlier until the middle sample d reached the instant $(m - 1)T_M$ and hence identified as $z((m - 1)T_M)$. The left and right samples, namely e and c , approach zero in the meanwhile, as illustrated by Figure 7.42. This philosophy of symbol centric TEDs was first displayed in Figure 7.41.

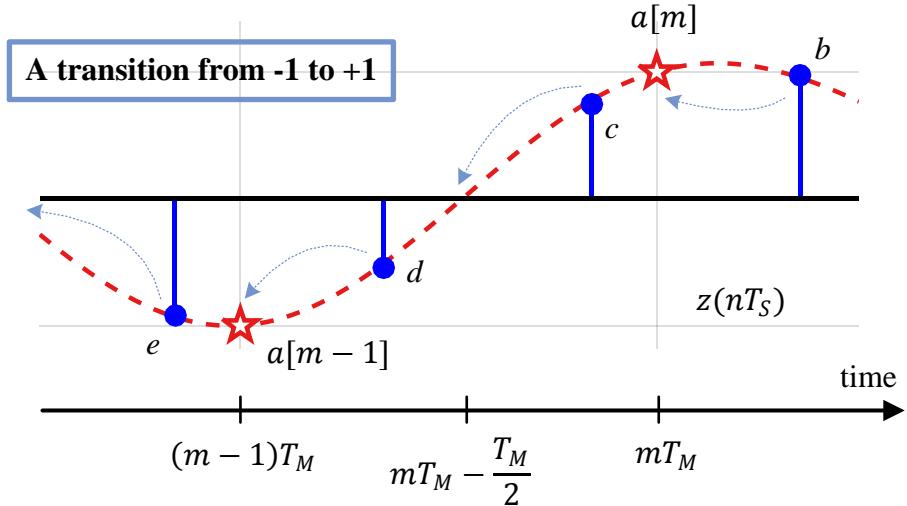


Figure 7.42: Observe how an early-late TED, even for a different set of samples, still pulls the middle sample towards the symbol center

Towards Zero Crossings

An alternative strategy is a zero crossing timing error detector where the algorithm targets the zero crossings of the waveform. This is done *by negating the slope* in a symbol centric TED. Let us find out what happens with negating the slope in an early-late expression, for which we again consider the same samples c, d and e considered above in Eq (7.35) with respect to Figure 7.42.

$$e_D[m] = d \cdot \{ - (c - e) \} = d \cdot (e - c)$$

Since d is negative while $e - c$ is also negative,

$$d \cdot (e - c) > 0$$

Thus, the sampling instant will be pushed forward until the middle sample d reaches $mT_M - T_M/2$. Then, the left neighbouring sample e will coincide with symbol $a[m - 1]$ while the right neighbouring sample c will coincide with $a[m]$. In this game of 3 samples, *the middle sample approaches the zero crossing*. This is the philosophy of the zero crossing TEDs.

The converging locations of zero crossing TED as well as early-late TED for the above example are shown in Figure 7.43. Notice that if we had chosen samples b, c and d in a zero crossing TED, the middle sample again would have converged towards zero.

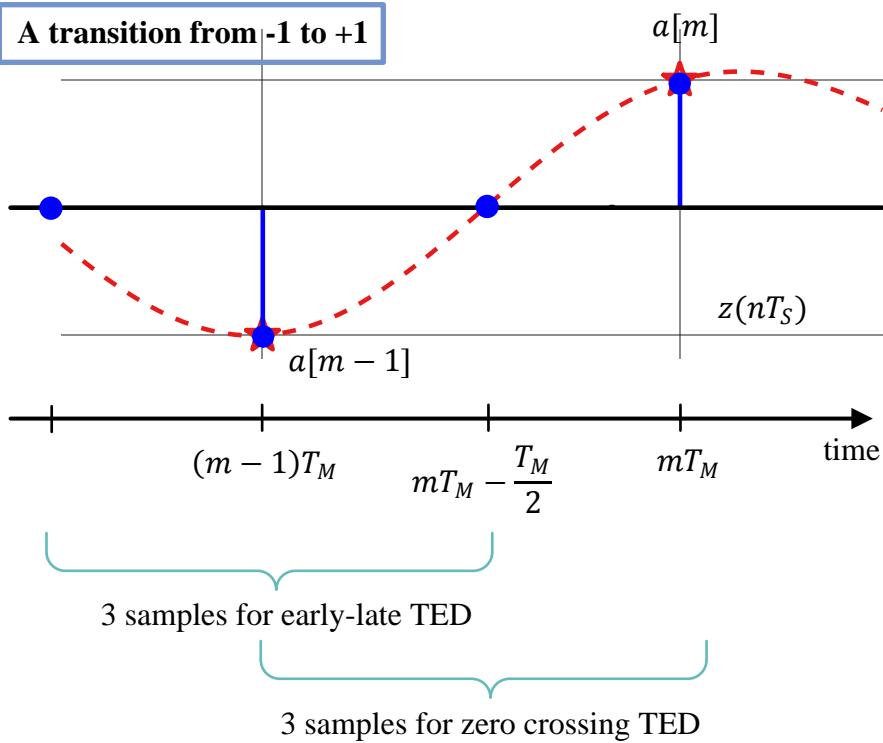


Figure 7.43: Final converging locations of zero crossing and early-late TEDs

Note 7.7 Zero crossing perspective

In light of the above discussion, we can say that as opposed to the symbol centric perspective in Eq (7.34), the zero crossing idea works as follows.

$$e_D[m] = \underbrace{z(mT_M + \hat{\epsilon}_\Delta)}_{\text{drive this term towards zero}} \cdot \{-\dot{z}(mT_M + \hat{\epsilon}_\Delta)\} \quad (7.36)$$

While one of the samples aligns with the zero crossings, the other sample automatically aligns with the maximum eye opening due to $T_M/2$ spacing between them. This approach is shown graphically in the eye diagram of Figure 7.44 where solid blue arrows indicate a positive or a negative symbol for late sampling while dashed red arrows indicate a positive or a negative symbol for the case of early sampling. Also, the right eye diagram is a shifted version of the left eye diagram to elaborate the zero crossing idea.

From the above analysis, we can form a timing error detector as

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ z((m-1)T_M + \hat{\epsilon}_\Delta) - z(mT_M + \hat{\epsilon}_\Delta) \right\} \quad (7.37)$$

This non-data-aided version is called the *Gardner Timing Error Detector (TED)* due to its inventor F. M. Gardner in Ref. [23]. Its data-aided and decision-directed versions

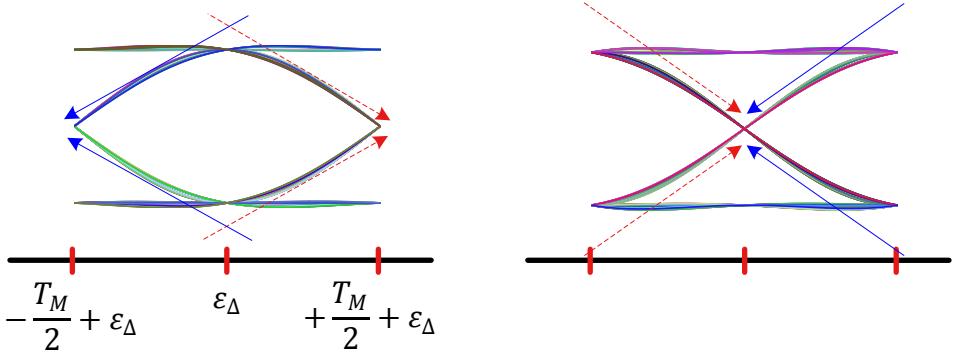


Figure 7.44: Zero crossing approach to solving the timing synchronization problem

are commonly known as *Zero Crossing Timing Error Detector (TED)*. By using the data symbols $a[m-1]$ and $a[m]$ in place of $z((m-1)T_M + \hat{\epsilon}_\Delta)$ and $z(mT_M + \hat{\epsilon}_\Delta)$, respectively, we get the data-aided variant.

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \{ a[m-1] - a[m] \} \quad (7.38)$$

On the same note, a decision-directed form can be created as

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \{ \hat{a}[m-1] - \hat{a}[m] \} \quad (7.39)$$

where the symbol decision $\hat{a}[m]$ for a binary PAM case is

$$\hat{a}[m] = A \times \text{sign} \{ z(mT_M + \hat{\epsilon}_\Delta) \}$$

Like symbol centric detectors, a zero crossing approach also depends on balancing the magnitudes of two samples taken midway from either side of the symbol. Consequently, its performance also suffers when the excess bandwidth α is small.

Gardner TED is based on the ideas from a wave difference method and a digital Data Transition Tracking Loop (DTTL) (used as a symbol synchronizer in the telemetry Rx of the Mariner Mars 1969 mission [24]). A description for his own derivation is given in Appendix 7.15.

Rx Structure for Zero Crossing/Gardner TED

The Rx structure for a zero crossing or Gardner TED is quite similar to that for early-late TED and is drawn in Figure 7.45 for a decision-directed setting.

The Rx signal $r(t)$ is sampled at a rate of $F_S = 2/T_M$, or $T_S = T_M/2$, to generate $r(nT_S)$ at $L = 2$ samples/symbol. Next, the sampled signal $r(nT_S)$ is matched filtered at the same rate with its output being $z(nT_S)$. Under the command of an interpolation control block, an interpolator resamples these matched filter outputs at instants $\hat{\epsilon}_\Delta$ to produce $z(nT_S + \hat{\epsilon}_\Delta)$. Since this system runs at a rate of 2 samples/symbol, the delay block T_S (that represents a delay of 1 sample) supplies the samples at half a symbol duration, i.e., $z(mT_M - T_M/2 + \hat{\epsilon}_\Delta)$.

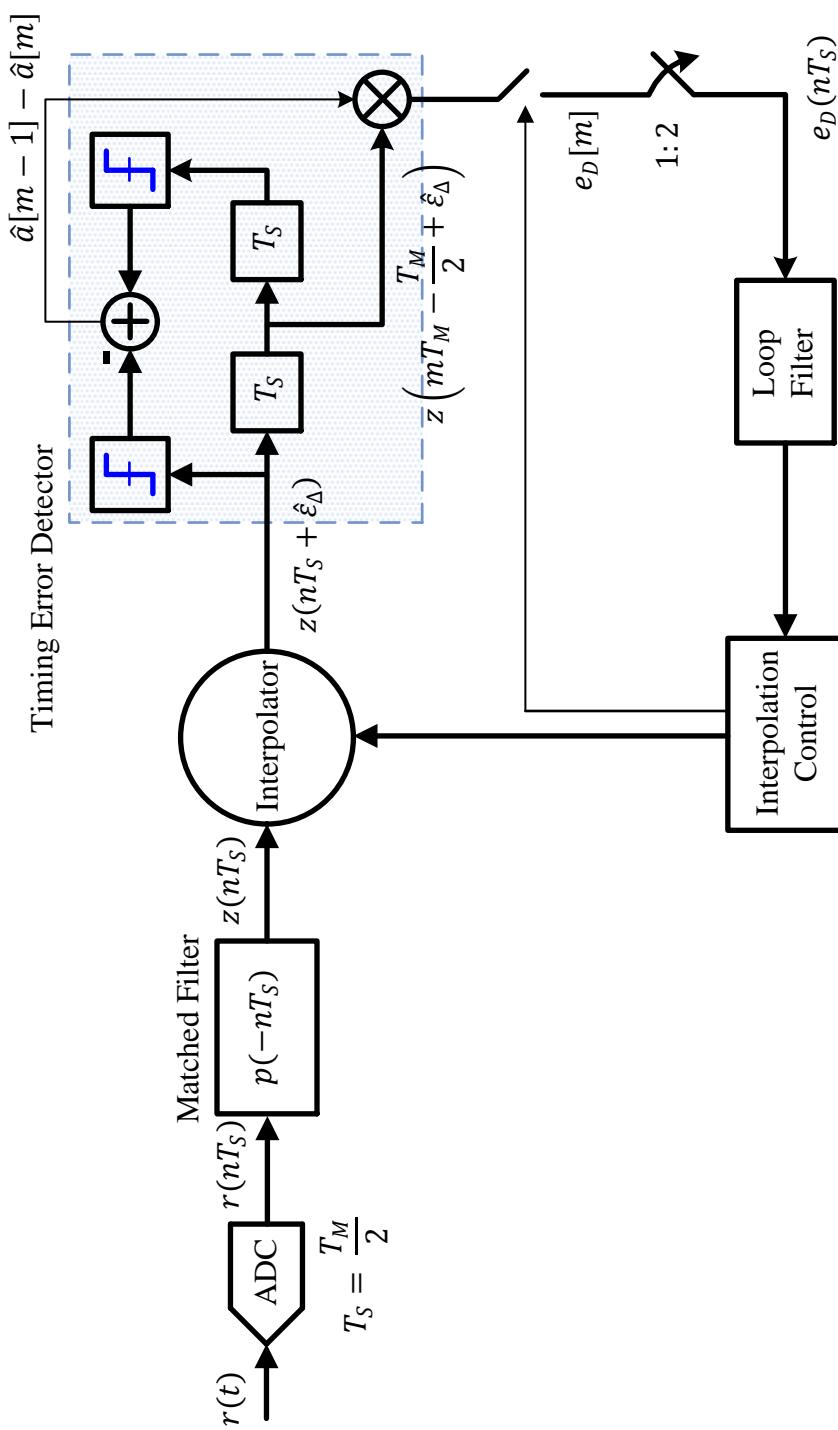


Figure 7.45: A timing locked loop with a zero crossing TED in a decision-directed setting

The interpolation control block also identifies the time instants $(m - 1)T_M$ and mT_M at which the outputs $\hat{a}[m - 1]$ and $\hat{a}[m]$ are taken out of the decision block and the error signal $e_D[m]$ is formed once per symbol according to Eq (7.32). In the case of Gardner TED, the matched filter outputs $z((m - 1)T_M + \hat{\varepsilon}_\Delta)$ and $z(mT_M + \hat{\varepsilon}_\Delta)$ are directly used instead of $\hat{a}[m - 1]$ and $\hat{a}[m]$ in forming the TED output. This signal $e_D[m]$ is then upsampled by 2 to create $e_D(nT_S)$ that then matches the sample rate F_S of the loop filter and the interpolation control. The role of interpolation control outputs will be shortly explained in Section 7.9.

Mean Curves for Zero Crossing and Gardner TEDs

The mathematical expression for the mean curve of a data-aided zero crossing TED is derived in Appendix 7.15 and is given by

$$\overline{e_D} = \gamma A^2 \left\{ r_p \left(+\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) - r_p \left(-\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) \right\} \quad (7.40)$$

Compare this mean curve with that of the early-late TED in Eq (7.33) and notice that they are identical. The comments on TED gain K_D mentioned in Note 7.5 in regards to a derivative TED apply in this scenario as well.

The mean curve for a Gardner TED, which is a non-data-aided version of a zero crossing TED, is a little complicated to derive. We just write the final expression here for reference.

$$\overline{e_D} = \frac{4\gamma^2 A^2}{T_M} \cdot \frac{1}{4\pi \left(1 - \frac{\alpha^2}{4} \right)} \sin \frac{\pi\alpha}{2} \cdot \sin \left(2\pi \frac{\varepsilon_{\Delta:e}}{T_M} \right) \quad (7.41)$$

It is drawn for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.5$ in Figure 7.46a. A similar quality metric as defined in the case of a derivative TED in Eq (7.27) can be constructed. For low excess bandwidth, mean and standard deviation of the Gardner TED suffer as shown in Figure 7.46 and the performance deteriorates. This is due to the increased scattering of the zero crossings after the matched filtering which are tracked by the Gardner TED.

Next, we simulate a Gardner TED for the same set of parameters as for a derivative TED in Section 7.6.1 except L which is 2 in this case, i.e.,

Modulation \rightarrow 2 – PAM, $L = 2$ samples/symbol, $\alpha = 0.4$,

$$B_n T_M = 1/200, \quad \zeta = 1/\sqrt{2}, \quad \varepsilon_\Delta = 0.25T_M$$

Again, the PI loop filter coefficients are computed through Eq (4.7) where K_0 is unity while a simple routine is used to compute the derivative of the mean curve at $\varepsilon_\Delta = 0$ for K_D . The output error signal $e_D[m]$ for a Gardner TED is shown in Figure 7.47a and the estimate of the timing offset $\varepsilon_\Delta = 0.25T_M$ is seen converging to its true value in Figure 7.47b. Later in Section 7.8, we will call this estimate as a fractional interval $\mu[m]$. The TLL is seen to converge in approximately 800 symbols.

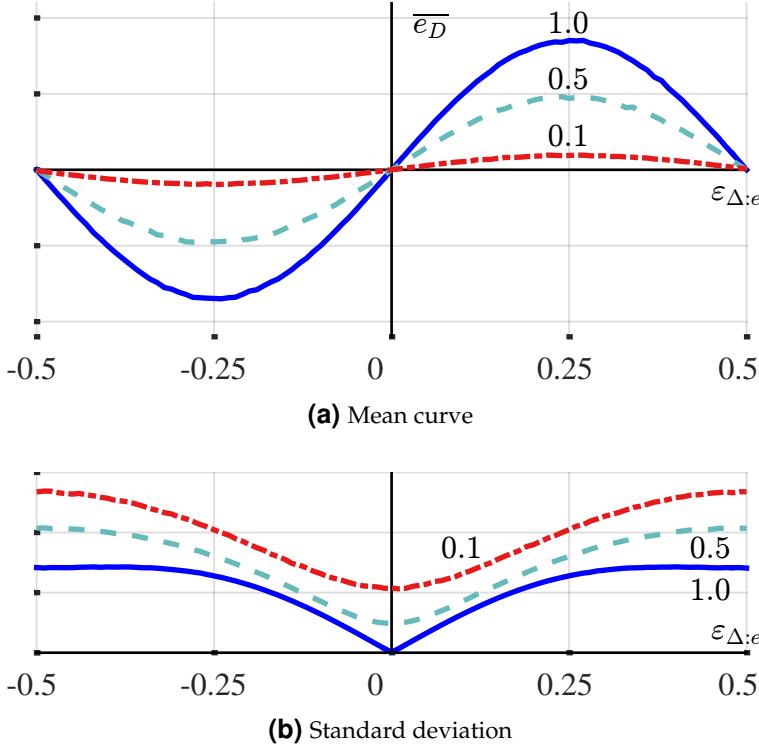


Figure 7.46: Gardner TED for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidths $\alpha = 0.1, 0.5$ and 1

Finally, the zero crossing TED for a QAM scheme is given by the sum of inphase and quadrature parts. For example, from Eq (7.38), the data-aided version can be written as

$$e_D[m] = z_I \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ a_I[m-1] - a_I[m] \right\} + \\ z_Q \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ a_Q[m-1] - a_Q[m] \right\}$$

Different Versions of Gardner TED?

In the communications research literature, you will observe that different versions of an early-late and zero crossing TEDs are prevalent. The Gardner TED is defined as

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ z((m-1)T_M + \hat{\epsilon}_\Delta) - z(mT_M + \hat{\epsilon}_\Delta) \right\} \quad (7.42)$$

On the other hand, some describe the same TED as

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ z(mT_M + \hat{\epsilon}_\Delta) - z((m-1)T_M + \hat{\epsilon}_\Delta) \right\} \quad (7.43)$$

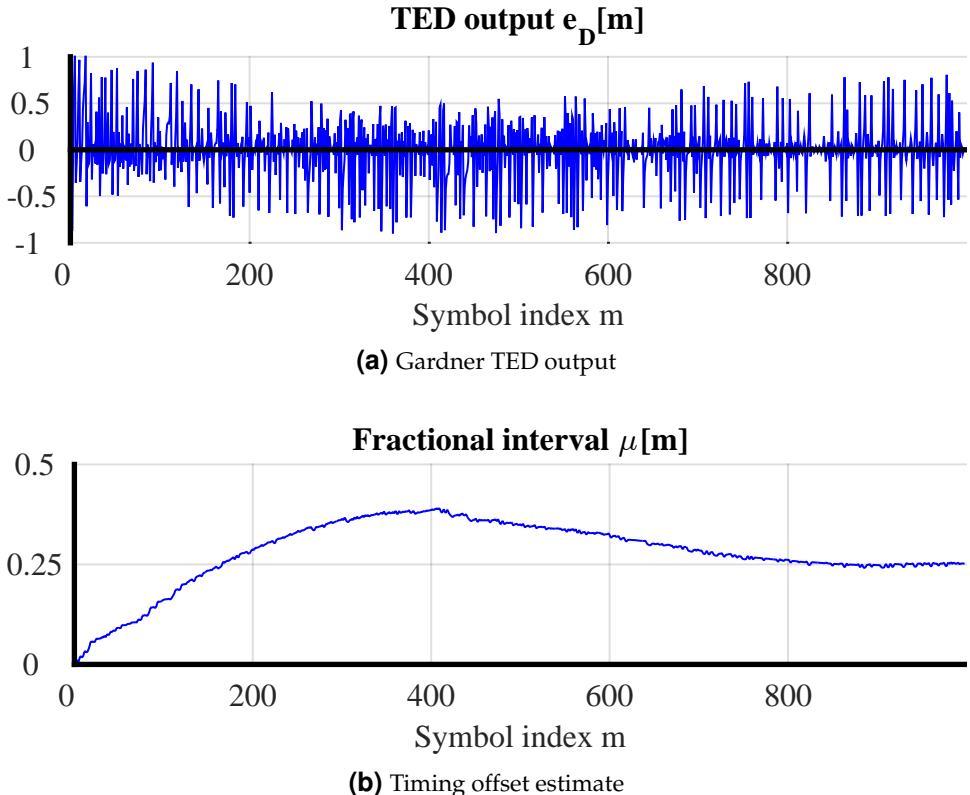


Figure 7.47: A timing locked loop with a Gardner TED converging to the steady state value for an SRRC pulse with excess bandwidth $\alpha = 0.4$

where it can be misidentified as an early-late TED. Both versions are correct and the difference arises according to the sign of the error that is employed towards timing compensation down the loop. To avoid this confusion, we explain this point below.

We explained in the zero crossing philosophy earlier that a formation such as Eq (7.43) converges to an early-late TED expression.

$$e_D[m] = z(mT_M + \hat{\varepsilon}_\Delta) \left\{ z \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\}$$

A mean curve for a carrier phase and carrier frequency error detector always has a positive slope at the origin. Referring back to Note 7.4, while many scientists also link the proper operation of a timing error detector with a positive slope at the origin, many others treat the mean curve as having a negative slope. The scientists following the former approach define Gardner TED as in Eq (7.42) and ELTED as in Eq (7.43). On the other hand, those following the latter approach define the Gardner TED as in Eq (7.43). This clarifies the confusion between a Gardner TED and an ELTED.

Interestingly, this is what lead to Gardner himself deriving the TED in [23] through the former approach but then reversing the sign of the error signal at the last moment so that the TED slope could become negative at the origin. According to him, “the reversal of sign has no significance in the formal manipulations or in the processor’s

computation burden, but assures negative slope at the tracking point of the detector output". As a final remark, this is why a description of the Gardner TED, but not the ELTED, is absent from one of the most comprehensive and delightful resources on timing synchronization [21].

A Comparison of Symbol Centric and Zero Crossing TEDs

From the discussion so far, particularly from Figure 7.43 which displays the final convergence of the three participating samples, it seems that the symbol centric and zero crossing approaches are similar to each other and there should be no reason to select one over another. As we find out next, this is not true.

The symbol centric approach is based on the maximum correlation (which comes from maximum likelihood) theory. The maximum likelihood technique is founded on the AWGN characteristic of the noise entering the Rx system. Therefore, a maximum likelihood TED achieves a better loop SNR than the Gardner TED. On the other hand, it does not care about the transitions within the symbol sequence in a short run. These transitions, on a short scale, have a significant impact on TED performance in terms of self noise produced at its output.

Consider Figure 7.48 for a depiction of the self noise generation process for $\alpha = 1$ which is drawn for a sequence $\{-1, +1, +1\}$ and the middle symbol corresponds to time $t = mT_M$. In this scenario, there is no data transition from mT_M to $(m+1)T_M$. Since the ideal timing has been acquired and the loop has converged, there should be zero update to the timing estimate $\hat{\varepsilon}_\Delta$ when we apply the TED relation.

However, it is evident that the computation of the derivative by a symbol centric TED (such as an early-late TED) through the two neighbouring samples around $t = mT_M$ generates a reasonably positive value.

$$z\left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) - z\left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta\right) \gg 0$$

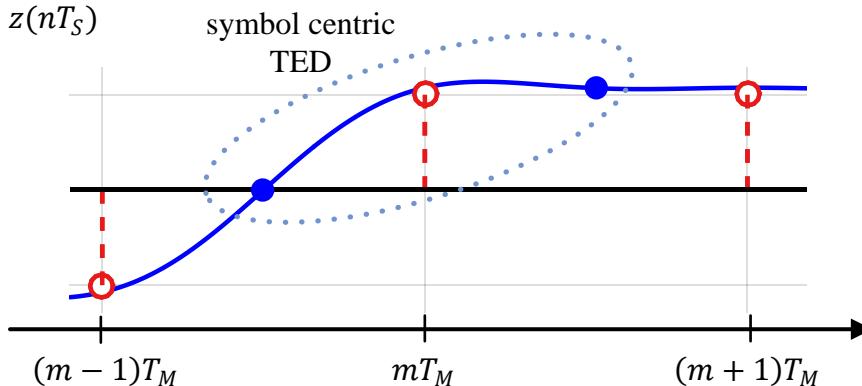


Figure 7.48: Self noise generation by early-late TED and avoidance by zero crossing TED during no data transition when $\alpha = 1$

As far as the Gardner TED is concerned, when the excess bandwidth α is less than 1, the zero crossings of data transitions are scattered around the midway point between the two symbol instants (instead of being at the dead center). While the

average over a long sequence is correct, individual trajectories pass from the right and left of this point, thus generating self noise. See the scattering of zero crossings in Figure 3.59b drawn for $\alpha = 0.5$ as an example. The spread of zero crossings has increased even more after the matched filtering.

Nonetheless, this is not true for excess bandwidth $\alpha = 1$. From Figure 7.48, we notice the following.

- At mT_M , the middle sample $z(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$ is zero.
- At $(m+1)T_M$, the difference $z(mT_M + \hat{\varepsilon}_\Delta) - z((m+1)T_M + \hat{\varepsilon}_\Delta)$ is zero.

And hence there is no self noise in this scenario. To cope with the self noise that arises due to less excess bandwidth α , simple 3 to 5 tap pre-filters prior to timing recovery are designed that reduce the self noise by a significant margin [25].

Carrier Independent Operation

One of the reasons for popularity of Gardner TED was the belief that its operation is independent of the carrier phase as well as a small frequency offset. Many sources still cite the Gardner TED as having an extra feature of rotationally invariant (carrier independent operation) and hence very suitable for timing acquisition when a significant carrier phase offset and possibly a small carrier frequency offset is present in the Rx signal.

As it turns out, this carrier independent operation is not exclusive to the Gardner TED. In fact, this is a feature of the non-data-aided fashion in which the TED processes the Rx samples. For passband modulation schemes, the Gardner TED is given from complex samples by the inphase part of a conjugate product instead of a simple product.

$$e_D[m] = \left[z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \left\{ z^* ((m-1)T_M + \hat{\varepsilon}_\Delta) - z^* (mT_M + \hat{\varepsilon}_\Delta) \right\} \right]_I$$

From the definition of a complex conjugate V^* , we know that

$$\begin{aligned} |V^*| &= |V| \\ \angle V^* &= -\angle V \end{aligned}$$

Consequently, the phase of the middle sample is canceled by the common phase of the two samples in the brackets above. However, the non-data-aided version of early-late TED exhibits exactly the same property for complex samples.

$$e_D[m] = \left[z(mT_M + \hat{\varepsilon}_\Delta) \left\{ z^* \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z^* \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\} \right]_I$$

In conclusion[†], as long as there is a conjugate product being taken between the complex samples having the same phase rotation, the TED is essentially carrier independent and Gardner TED is not unique in this regard. Having said that, the presence of a rotating phase in the constellation can perturb the timing error detectors to some extent in terms of the jitter and acquisition time.

[†]In terms of complex signals, $\exp(j\theta)$ is the phase rotation of the middle sample while $\exp(-j\theta)$ can be taken as common from the terms in the brackets.

Timing Lock Detectors

Similar to the carrier lock detectors described in Section 5.7.1, timing lock detectors can also be constructed based on some property of the modulated signal. These lock detectors operate in parallel to the TLL or PLL and aid the Rx state machine in executing necessary tasks according to each scenario. The expressions for two such detectors are as follows.

- The output of a timing lock detector should be at its peak for the correct timing. Therefore, when the matched filter output $z(mT_M)$ is at its peak, the second sample in an $L = 2$ oversampled signal should ideally cross through zero. This leads to a maximum amplitude difference between these two samples. Taking a long term average thus yields a non-data-aided strategy as

$$\text{LD}(\varepsilon_\Delta) = \sum_{n=0}^{2N_d-1} \left\{ |z(mT_M)|^2 - \left| z\left(mT_M - \frac{T_M}{2}\right) \right|^2 \right\}$$

- The lock detector mentioned above suffers from the dynamic range problem due to its dependency on the Rx signal amplitude. The solution then is to normalize the lock detector output by the signal magnitude as

$$\text{LD}(\varepsilon_\Delta) = \sum_{n=0}^{2N_d-1} \frac{|z(mT_M)|^2 - \left| z\left(mT_M - \frac{T_M}{2}\right) \right|^2}{|z(mT_M)|^2 + \left| z\left(mT_M - \frac{T_M}{2}\right) \right|^2}$$

Now we describe the operations of interpolation and interpolation control that play a similar role to a phase de-rotator and NCO, respectively, in the context of a PLL.

7.8 Interpolation

In general, interpolation is the process of reproducing a ‘missing’ sample at a desired location. In digital and wireless communications, the role of interpolation can be explained as follows.

Imagine a Tx signal constructed from the upsampled and pulse shaped modulation symbols. The job of the Rx is to sample this waveform at optimal intervals, i.e., exactly at the middle of the eye diagram. In other words, the Rx only needs one sample per symbol taken at ISI-free locations T_M seconds apart, or $nT_S = mT_M + \varepsilon_\Delta$ where ε_Δ is the original timing phase offset.

However, the sampling clock at the Rx that drives the Analog to Digital Converter (ADC) is often completely unrelated to the clock at the Tx that generated the Tx waveform. Therefore, it is not necessary (and in fact almost impossible) to have samples at the output of the Rx ADC which coincide exactly with the optimal instants.

While timing error detectors provide us with an estimate of timing offset $\hat{\varepsilon}_\Delta$, the loop itself is not capable of restoring the signal value at $mT_M + \hat{\varepsilon}_\Delta$. This is the job of the interpolator which in that sense is equivalent to a de-rotation block found in PLLs of Chapter 5. Restoring the phase was a straightforward task where one just needs to de-rotate (clockwise rotation) the signal samples by a given angle. Here, we need a re-computation of the signal values at optimal instants (maximum eye opening) shown as white circles in Figure 7.1b.

Next, we discuss how to accomplish the task of interpolating the matched filter output from a set of given samples (usually two or four). The discussion on interpolation and interpolation control is mainly based on Ref. [26]. We start with a formal problem definition.

Problem Definition

At the output of the matched filter, the samples are available at sample rate F_S . A new value needs to be computed between $z(nT_S)$ and $z((n+1)T_S)$ at a location $\mu_m T_S$, where μ_m is the *fractional interval during symbol number m* that in a TLL targets the maximum eye opening, i.e., at a location

$$t = nT_S + \mu_m T_S$$

Why not just denote this fraction interval as μ ? Because this fraction keeps on updating for each symbol m according to the TED output. And to be consistent with the notation that follows in interpolation control, we stick with μ_m throughout this discussion[†]. Keep in mind that μ_m is not the TED output $e_D[m]$ since a TED in general sets the direction and magnitude to shift the next sampling interval so as to recursively approach the actual timing phase offset ε_Δ . Moreover, the TED output is filtered by the loop filter to generate the signal $e_F[n]$. This signal is utilized by an interpolator control block to produce the fractional interval μ_m that informs the interpolator to compute an interpolant at instant $nT_S + \mu_m T_S$. Gradually, the fractional interval μ_m converges towards the actual timing offset ε_Δ .

7.8.1 Piecewise Polynomial Interpolation

In interpolation using a polynomial, the sample at $nT_S + \mu_m T_S$ is computed through the help of the neighbouring samples. Since this process is performed for each interpolant separately in a timing loop, it is actually known as *piecewise polynomial interpolation*.

First, consider that the matched filter output $z(nT_S)$ is assumed to have an underlying continuous-time waveform $z(t)$ as

$$z(t) \approx c_p t^p + c_{p-1} t^{p-1} + \cdots + c_1 t + c_0 \quad (7.44)$$

where p is the degree of the polynomial. Obviously, for the sample value at a time $nT_S + \mu_m T_S$, the above equation can be written as

$$\begin{aligned} z(nT_S + \mu_m T_S) \approx & c_p (nT_S + \mu_m T_S)^p + c_{p-1} (nT_S + \mu_m T_S)^{p-1} + \\ & \cdots + c_1 (nT_S + \mu_m T_S) + c_0 \end{aligned} \quad (7.45)$$

We consider an implementation through a Finite Impulse Response (FIR) filter for three special cases here, linear interpolation ($p = 1$), quadratic interpolation ($p = 2$) and cubic interpolation ($p = 3$).

[†]A subtle point here is that for perfect synchronization case, each symbol index m occurs after L indices of n . For imperfect synchronization, this is not necessarily true and a better notation for μ_m should be $\mu(n_m)$ where n_m is the sample after which a symbol candidate needs to be interpolated. For the sake of simplicity, we stick to notation μ_m with the understanding that the actual symbol time T_M can slightly deviate from LT_S owing to a sample clock offset between Tx and Rx.

Linear Interpolation, $p = 1$

Plugging $p = 1$ in Eq (7.44), we get

$$z(t) \approx c_1 t + c_0$$

To find the values of two coefficients c_1 and c_0 , we need two equations. Therefore, we employ two neighbouring samples $z(nT_S)$ and $z((n+1)T_S)$ such that

$$\begin{aligned} z(nT_S) &= c_1 n T_S + c_0 \\ z((n+1)T_S) &= c_1 (n+1) T_S + c_0 \end{aligned}$$

Subtracting the former equation from the latter,

$$\begin{aligned} z((n+1)T_S) - z(nT_S) &= c_1 (nT_S + T_S - nT_S) \\ c_1 &= \frac{z((n+1)T_S) - z(nT_S)}{T_S} \end{aligned}$$

The coefficient c_0 is thus given by

$$\begin{aligned} c_0 &= z(nT_S) - c_1 n T_S \\ &= z(nT_S) - \frac{z((n+1)T_S) - z(nT_S)}{T_S} n T_S \\ &= (1+n)z(nT_S) - nz((n+1)T_S) \end{aligned}$$

Having known c_0 and c_1 , we can now compute the interpolant $z(nT_S + \mu_m T_S)$. Plugging $p = 1$ in Eq (7.45),

$$z(nT_S + \mu_m T_S) \approx c_1 (nT_S + \mu_m T_S) + c_0$$

which yields

$$\begin{aligned} z(nT_S + \mu_m T_S) &= \frac{z((n+1)T_S) - z(nT_S)}{T_S} (nT_S + \mu_m T_S) + \\ &\quad (1+n)z(nT_S) - nz((n+1)T_S) \\ &= \mu_m z((n+1)T_S) - \mu_m z(nT_S) + z(nT_S) \end{aligned}$$

The expression for the linear interpolator is thus written as

$$z(nT_S + \mu_m T_S) = \mu_m z((n+1)T_S) + (1 - \mu_m) z(nT_S) \quad (7.46)$$

An example of linear interpolation is drawn in Figure 7.49. Observe how the interpolant is computed as a point at the desired location by connecting the two neighbouring samples through a straight line. Some comments are now in order.

- From Eq (7.46), observe that the interpolant $z(nT_S + \mu_m T_S)$ is a linear combination of $z(nT_S)$ and $z((n+1)T_S)$. Therefore, it can be thought of as their convolution with a filter $h_1[n]$.

$$\begin{aligned} z(nT_S + \mu_m T_S) &= \mu_m z((n+1)T_S) + (1 - \mu_m) z(nT_S) \\ &= h_1[-1]z((n+1)T_S) + h_1[0]z(nT_S) \\ &= \sum_{i=-1}^0 h_1[i]z((n-i)T_S) \end{aligned}$$

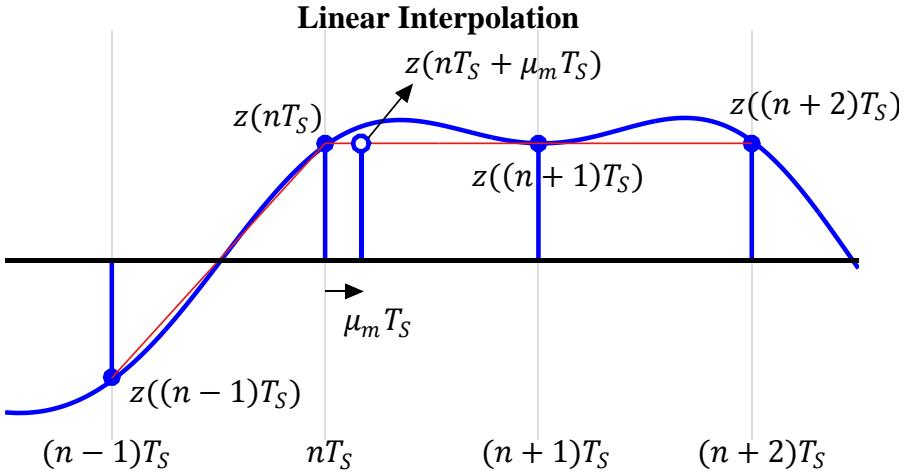


Figure 7.49: New sample value $z(nT_S + \mu_m T_S)$ at instant $nT_S + \mu_m T_S$ (shown as white circle) is computed by linear interpolation using two neighbouring samples

The filter coefficients are thus given by

$$\begin{aligned} h_1[-1] &= \mu_m \\ h_1[0] &= 1 - \mu_m \end{aligned} \quad (7.47)$$

where the subscript 1 denotes linear interpolation ($p = 1$). Furthermore, the index -1 in $h[-1]$ signifies the fact that we are employing $z((n+1)T_S)$ instead of $z((n-1)T_S)$. Later, we will generalize this concept to interpolate with filters $h_2[n]$ (quadratic) and $h_3[n]$ (cubic) as well.

- Notice that the sum of coefficients $h_1[0]$ and $h_1[-1]$ is

$$h_1[0] + h_1[-1] = 1 - \mu_m + \mu_m = 1$$

From the definition of Discrete Fourier Transform (DFT), this is the DC component at $k = 0$.

$$H_1[0] = \sum_{n=0}^{N-1} h_1[n] = 1$$

As a result, the interpolating filter does not modify the amplitude of the input waveform while generating the desired interpolant.

- In Section 1.9, we saw that a linear phase ensures no phase distortion in the waveform. By virtue of $h_1[0] = 1 - h_1[-1]$ and an even number of samples (i.e., two), the coefficients are symmetric about the center point $\mu_m = 1/2$. Consequently, it is a linear phase filter.

Cubic Interpolation, $p = 3$

Odd degree polynomials approximate the interpolant by using an even number of samples that ensures the linear phase property. After $p = 1$, the next odd degree

polynomial is with $p = 3$ and is known as cubic interpolation. Plugging $p = 3$ in Eq (7.44), we get

$$z(t) \approx c_3 t^3 + c_2 t^2 + c_1 t + c_0$$

Therefore,

$$\begin{aligned} z(nT_S + \mu_m T_S) &\approx c_3(nT_S + \mu_m T_S)^3 + c_2(nT_S + \mu_m T_S)^2 + \\ & \quad c_1(nT_S + \mu_m T_S) + c_0 \end{aligned} \quad (7.48)$$

Following the same derivation as in linear interpolation, the coefficients c_3 , c_2 , c_1 and c_0 need to be found every time μ_m changes. The four samples around the interpolant are given by

$$\begin{aligned} z((n-1)T_S) &= c_3((n-1)T_S)^3 + c_2((n-1)T_S)^2 + c_1(n-1)T_S + c_0 \\ z(nT_S) &= c_3(nT_S)^3 + c_2(nT_S)^2 + c_1 n T_S + c_0 \\ z((n+1)T_S) &= c_3((n+1)T_S)^3 + c_2((n+1)T_S)^2 + c_1(n+1)T_S + c_0 \\ z((n+2)T_S) &= c_3((n+2)T_S)^3 + c_2((n+2)T_S)^2 + c_1(n+2)T_S + c_0 \end{aligned}$$

These are four equations with four unknowns that can be easily solved to compute c_3 , c_2 , c_1 and c_0 . Substituting them back in Eq (7.48) yields

$$\begin{aligned} z(nT_S + \mu_m T_S) &= \left(\frac{\mu_m^3}{6} - \frac{\mu_m}{6} \right) z((n+2)T_S) + \\ & \quad \left(-\frac{\mu_m^3}{2} + \frac{\mu_m^2}{2} + \mu_m \right) z((n+1)T_S) + \\ & \quad \left(\frac{\mu_m^3}{2} - \mu_m^2 - \frac{\mu_m}{2} + 1 \right) z(nT_S) + \\ & \quad \left(-\frac{\mu_m^3}{6} + \frac{\mu_m^2}{2} - \frac{\mu_m}{3} \right) z((n-1)T_S) \end{aligned} \quad (7.49)$$

As before, the desired interpolant can again be written as the output of a filter as

$$\begin{aligned} z(nT_S + \mu_m T_S) &= h_3[-2]z((n+2)T_S) + h_3[-1]z((n+1)T_S) + \\ & \quad h_3[0]z(nT_S) + h_3[1]z((n-1)T_S) \\ &= \sum_{i=-2}^1 h_3[i]z((n-i)T_S) \end{aligned} \quad (7.50)$$

where the filter coefficients are given by

$$\begin{aligned} h_3[-2] &= \frac{\mu_m^3}{6} - \frac{\mu_m}{6} \\ h_3[-1] &= -\frac{\mu_m^3}{2} + \frac{\mu_m^2}{2} + \mu_m \\ h_3[0] &= \frac{\mu_m^3}{2} - \mu_m^2 - \frac{\mu_m}{2} + 1 \\ h_3[1] &= -\frac{\mu_m^3}{6} + \frac{\mu_m^2}{2} - \frac{\mu_m}{3} \end{aligned} \quad (7.51)$$

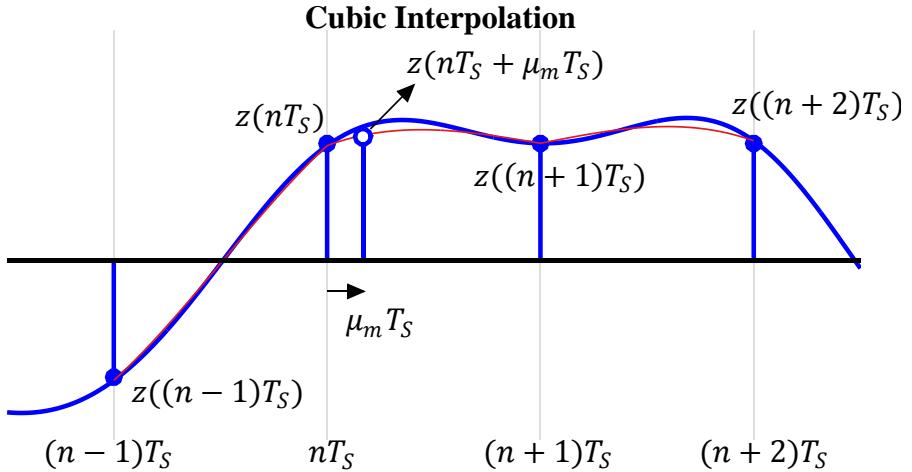


Figure 7.50: New sample value $z(nT_S + \mu_m T_S)$ at instant $nT_S + \mu_m T_S$ (shown as white circle) is computed by cubic interpolation using four neighbouring samples

In the above equations, the subscript 3 denotes cubic interpolation as $p = 3$. An example of cubic interpolation is drawn in Figure 7.50. Observe how the curve has been smoothed out as compared to linear interpolation in Figure 7.49. This gain comes at a price of higher computational complexity paid for cubic interpolation.

Quadratic Interpolation, $p = 2$

Plugging $p = 2$ in Eq (7.44), we get

$$z(t) \approx c_2 t^2 + c_1 t + c_0$$

Therefore,

$$z(nT_S + \mu_m T_S) \approx c_2(nT_S + \mu_m T_S)^2 + c_1(nT_S + \mu_m T_S) + c_0$$

To find the values of three coefficients c_2 , c_1 and c_0 , we need only three equations and hence just three samples out of the matched filter, namely $z((n-1)T_S)$, $z(nT_S)$ and $z((n+1)T_S)$. However, if we take that route, there will be no middle two samples between which the interpolant can be constructed. Then, the FIR filter will not be symmetric with respect to $\mu_m = 1/2$. As mentioned above, *an even number of samples are required to ensure a linear phase response of the FIR filter*. Consequently, an extra fourth sample is used for quadratic interpolation.

Skipping the derivation, the desired interpolant can again be written as the output of a filter as

$$\begin{aligned} z(nT_S + \mu_m T_S) &= h_2[-2]z((n+2)T_S) + h_2[-1]z((n+1)T_S) + \\ &\quad h_2[0]z(nT_S) + h_2[1]z((n-1)T_S) \\ &= \sum_{i=-2}^1 h_2[i]z((n-i)T_S) \end{aligned}$$

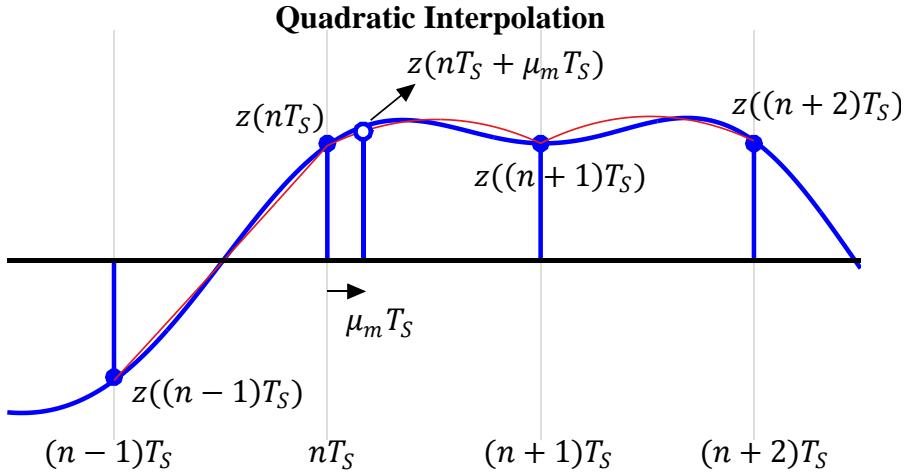


Figure 7.51: New sample value $z(nT_S + \mu_m T_S)$ at instant $nT_S + \mu_m T_S$ (shown as white circle) is computed by quadratic interpolation using four neighbouring samples

where the filter coefficients are given by

$$\begin{aligned}
 h_2[-2] &= \beta\mu_m^2 - \beta\mu_m \\
 h_2[-1] &= -\beta\mu_m^2 + (1 + \beta)\mu_m \\
 h_2[0] &= -\beta\mu_m^2 - (1 - \beta)\mu_m + 1 \\
 h_2[1] &= \beta\mu_m^2 - \beta\mu_m
 \end{aligned} \tag{7.52}$$

In the above equations, the subscript 2 denotes quadratic interpolation ($p = 2$) and β is an additional parameter to choose due to an extra sample being used. A value of $\beta = 0.5$ is usually used to reduce the hardware complexity as multiplication and division by a factor of 2 can be implemented by simple left and right shifts of the contents of a register. An example of quadratic interpolation is drawn in Figure 7.51. Notice the parabolic overshoots emerging from the sample values due to piecewise computation, generating vastly improved results without expending significant computational resources.

Ideal Interpolation, $p = \infty$

To unify the above findings, consider the general convolution equation out of an interpolating filter with degree p .

$$z(nT_S + \mu_m T_S) = \sum_{i=-2}^1 h_p[i]z((n-i)T_S)$$

The filter coefficients for $p = 1, 2$ and 3 were given in Eq (7.47), Eq (7.52) and Eq (7.51), respectively.

Something interesting happens when we vary μ_m from 0 to 1,

$$\mu_m : 0 \rightarrow 1$$

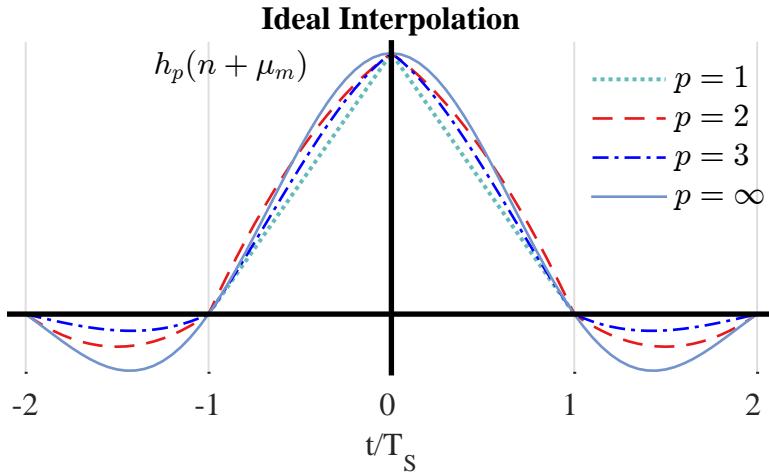


Figure 7.52: An increasing degree p of polynomial approximation takes the filter $h_p(n + \mu_m)$ closer and closer to an ideal sinc impulse response

and plot the filter coefficients as a function of μ_m . For example, for $p = 1$,

$$\begin{aligned} h_1[-1] &= \mu_m \\ h_1[0] &= 1 - \mu_m \end{aligned}$$

So for t/T_S between -1 to 0 (which is the same as t from $-T_S$ to 0), the plot of $h_1(-1 + \mu_m)$ is μ_m which goes from 0 to 1 .

$$h_1(-1 + \mu_m) : 0 \rightarrow 1$$

Here, $h_1(-1 + \mu_m)$ is the interpolated version of the discrete-time impulse response of the filter $h_1[n]$. Similarly, for t/T_S between 0 to 1 (which is the same as t from 0 to T_S), the plot of $h_1(0 + \mu_m)$ is $1 - \mu_m$ which goes from 1 to 0 .

$$h_1(0 + \mu_m) : 1 \rightarrow 0$$

In general, we can draw the impulse response $h_1(n + \mu_m)$ from -1 to 1 . This is because the filter coefficients are at locations -1 and 0 while μ_m ranges from 0 to 1 .

A similar procedure is followed for $h_2(n + \mu_m)$ and $h_3(n + \mu_m)$ between -2 and 2 [†]. The results are plotted in Figure 7.52 in an increasing order. Notice that as the degree p of the polynomial increases, the filter becomes closer in shape to a sinc signal. Why did the sinc signal appear here?

Recall from Section 2.7.2 that after inserting zeros during an upsampling process, a lowpass filter is employed to reject the spectral replicas leaving only the central spectrum. In time domain, this filtering has the effect of interpolating between the samples because multiplication with a rectangular filter in frequency domain is equivalent to convolving with a sinc function in time domain. This sinc function has integer spaced zero crossings due to which its convolution with a sampled signal creates the output exactly as the input at discrete-time intervals and a combination of those samples in between. That is why as we increase the degree p of our polynomial approximation towards $p = \infty$, our filter impulse response approaches a sinc signal.

[†]For a hint, look at the factor $+1$ in $h_2[0]$ and $h_3[0]$ in Eq (7.52) and Eq (7.51), respectively.

7.8.2 Farrow Structure

In the discussion on piecewise polynomial interpolation, we emphasized on the fact that the fractional interval μ_m needs to be updated for each symbol time mT_M and hence the subscript m in μ_m . For this reason, the interpolation process becomes a two-step procedure.

1. Update the filter coefficients $h_p[n]$.
2. Perform the convolution between $z(nT_S)$ and $h_p[n]$.

This process can be simplified if the two steps above can be combined in such a way that μ_m update is weaved into the convolution operation. In other words, instead of a two-input hardware multiplication with two variable quantities, complexity can be reduced by restructuring the problem such that one input in the two-input hardware multiplication remains fixed. This is accomplished by separating the constant factors from the powers of μ_m in filter equations, which is discussed in detail for cubic interpolation and can be easily generalized for other cases.

Begin with filter coefficients from cubic interpolation in Eq (7.51) and rewrite it as

$$\begin{aligned} h_3[-2] &= \frac{1}{6}\mu_m^3 + 0 \cdot \mu_m^2 - \frac{1}{6}\mu_m + 0 \\ &= b_3[-2]\mu_m^3 + b_2[-2]\mu_m^2 + b_1[-2]\mu_m^1 + b_0[-2]\mu_m^0 = \sum_{l=0}^3 b_l[-2]\mu_m^l \end{aligned}$$

where $b_3[-2] = 1/6$, $b_2[-2] = 0$, etc. Similarly, for other coefficients, we can write

$$\begin{aligned} h_3[-1] &= -\frac{1}{2}\mu_m^3 + \frac{1}{2}\mu_m^2 + 1 \cdot \mu_m + 0 = \sum_{l=0}^3 b_l[-1]\mu_m^l \\ h_3[0] &= \frac{1}{2}\mu_m^3 - 1 \cdot \mu_m^2 - \frac{1}{2}\mu_m + 1 = \sum_{l=0}^3 b_l[0]\mu_m^l \\ h_3[1] &= -\frac{1}{6}\mu_m^3 + \frac{1}{2}\mu_m^2 - \frac{1}{3}\mu_m + 0 = \sum_{l=0}^3 b_l[1]\mu_m^l \end{aligned}$$

In general,

$$h_3[i] = \sum_{l=0}^3 b_l[i]\mu_m^l$$

Observe that $b_l[i]$ are fixed coefficients independent of μ_m in the above expression and are given in Table 7.2.

With this understanding in place, the filtering process from Eq (7.50) can be rearranged such that convolution of the matched filter output is performed with the fixed

Table 7.2: Farrow coefficients $b_l[i]$ for cubic interpolator

i	$b_3[i]$	$b_2[i]$	$b_1[i]$	$b_0[i]$
-2	$\frac{1}{6}$	0	$-\frac{1}{6}$	0
-1	$-\frac{1}{2}$	$\frac{1}{2}$	1	0
0	$\frac{1}{2}$	-1	$-\frac{1}{2}$	1
1	$-\frac{1}{6}$	$\frac{1}{2}$	$-\frac{1}{3}$	0

coefficients $b_l[i]$.

$$\begin{aligned}
 z(nT_S + \mu_m T_S) &= \sum_{i=-2}^1 h_3[i] z((n-i)T_S) \\
 &= \sum_{i=-2}^1 \cdot \sum_{l=0}^3 b_l[i] \mu_m^l \cdot z((n-i)T_S) \\
 &= \sum_{l=0}^3 \mu_m^l \underbrace{\sum_{i=-2}^1 b_l[i] \cdot z((n-i)T_S)}_{y(l)} \\
 &= \sum_{l=0}^3 y(l) \mu_m^l
 \end{aligned}$$

To obtain a hardware efficient filter structure known as *Farrow structure*, the above expression can be opened as

$$\begin{aligned}
 z(nT_S + \mu_m T_S) &= y(3)\mu_m^3 + y(2)\mu_m^2 + y(1)\mu_m^1 + y(0)\mu_m^0 \\
 &= \{y(3)\mu_m^2 + y(2)\mu_m + y(1)\}\mu_m + y(0)
 \end{aligned}$$

One more such step produces the Farrow structure for a cubic interpolator.

$$z(nT_S + \mu_m T_S) = \left\{ \{y(3)\mu_m + y(2)\} \mu_m + y(1) \right\} \mu_m + y(0) \quad (7.53)$$

The Farrow structure for cubic interpolation is drawn in Figure 7.53 where T_S is a sample delay. Notice the fixed coefficients $b_l[i]$ operating on the matched filter output samples while a changing μ_m operates on the subsequent output $y(l)$. Although its block diagram looks complicated, the expression above is quite simple.

A similar strategy can be applied to devise Farrow structures for polynomial interpolation for the cases when $p \neq 3$. For example, for $p = 2$, it is given by

$$z(nT_S + \mu_m T_S) = \{y(2)\mu_m + y(1)\}\mu_m + y(0)$$

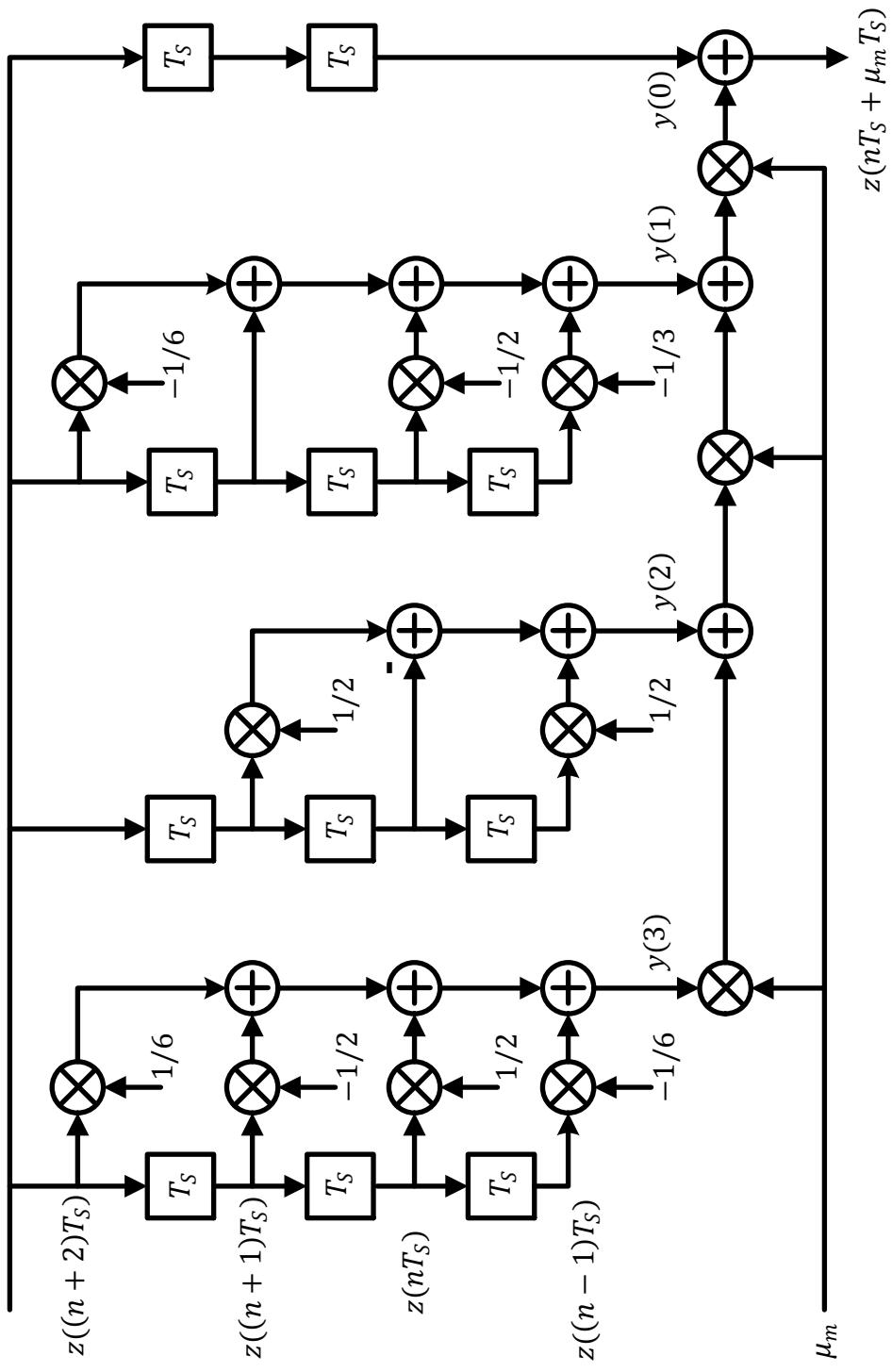


Figure 7.53: Farrow structure for cubic interpolation

7.9 Interpolator Control

Interpolator control is a seemingly mysterious topic in the area of symbol timing synchronization. There are different techniques available to implement an interpolator control block. However, the terminology used to understand all of them can easily cause confusion. For the purpose of clarity, I focus on a single method known as modulo-1 counter interpolator control. For interested readers, Ref. [21] and Ref. [18] are excellent resources for more information on this topic.

In the discussion on interpolation in Section 7.8, we learnt how to regenerate samples at desired locations $nT_S + \mu_m T_S$. However, we deliberately ignored two questions.

1. During each symbol time mT_M , how does the interpolation block know *the value of μ_m* ?
2. With L number of samples/symbol, how does the timing locked loop know *after which one sample out of L samples*, the interpolant corresponding to maximum eye opening should be taken? In other words, the available samples at the matched filter output are

$$\dots, z((n-2)T_S), z((n-1)T_S), z(nT_S), z((n+1)T_S), \dots$$

If the interpolant for m^{th} symbol lies between nT_S and $(n+1)T_S$, the sample index n is called the *m^{th} basepoint index* and will be denoted by n_m from now onwards. We did not use this notation in conjunction with introducing fractional interval μ_m to keep the interpolation discussion simple.

To summarize, the interpolator control block provides the interpolation block with (a) the fractional interval μ_m at which an interpolant needs to be computed, and (b) the basepoint index n_m after which the interpolant for symbol m decision should be computed[†].

[†]If the job of the interpolator control block is clear by now, you can ignore this footnote. In case it is still confusing, I wrote the following lines to refine the understanding.

- Taken at $L = 1$ sample/symbol, the Rx samples at the output of the matched filter are located at instants $n = mT_M + \hat{\varepsilon}_\Delta$. Thus, the available samples are

$$\dots, z((m-1)T_M + \hat{\varepsilon}_\Delta), z(mT_M + \hat{\varepsilon}_\Delta), z((m+1)T_M + \hat{\varepsilon}_\Delta), \dots$$

Here, based on its input being filtered TED output, the task of the interpolator control is to provide a fractional interval μ_m at symbol time m such that the impact of ε_Δ gradually vanishes.

- Taken at $L = 2$ samples/symbol, the samples at the matched filter output are located at instants

$$\dots, z(mT_M - T_M + \hat{\varepsilon}_\Delta), z(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta), z(mT_M + \hat{\varepsilon}_\Delta), \\ z(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta), z(mT_M + T_M + \hat{\varepsilon}_\Delta), \dots$$

Here, the task of the interpolator control is twofold as

1. to provide a fractional interval μ_m at time m such that the impact of ε_Δ gradually vanishes, and
2. to provide a basepoint index n_m , i.e., indicate the sample corresponding to mT_M after which a symbol decision needs to be made. For example, this mechanism should raise a flag at locations $(m-1)T_M + \hat{\varepsilon}_\Delta$, $mT_M + \hat{\varepsilon}_\Delta$ and $(m+1)T_M + \hat{\varepsilon}_\Delta$ while keeping the flag low at instants $mT_M - T_M/2 + \hat{\varepsilon}_\Delta$, $mT_M + T_M/2 + \hat{\varepsilon}_\Delta$, etc.

Once we have understood this problem for 2 samples/symbol, this concept can be easily extended to $L > 2$. The available samples out of the matched filter are

$$\dots, z((n-2)T_S), z((n-1)T_S), z(nT_S), z((n+1)T_S), z((n+2)T_S), \dots$$

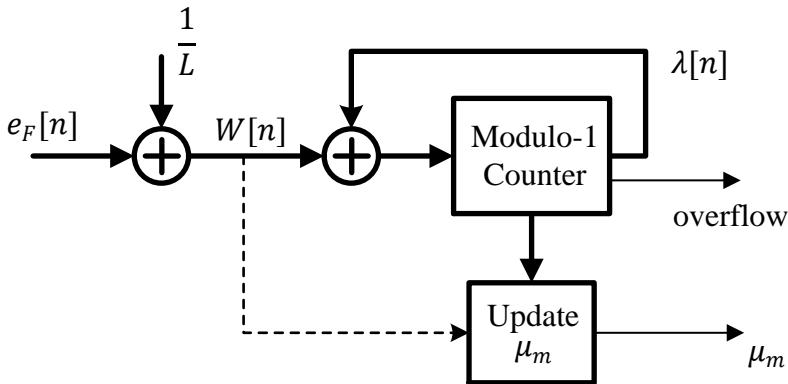


Figure 7.54: A block diagram of an interpolation control block based on a modulo-1 counter

7.9.1 Modulo-1 Counter

Imagine a situation where $\varepsilon_\Delta = 0$ and hence Tx and Rx clocks are exactly synchronized in timing phase and frequency. With L number of samples/symbol, the main task of the Rx is to find the first basepoint index that corresponds to the maximum eye opening. From there onwards, it just needs to count L samples each time before picking the next optimal sample. Or count by $1/L$ such that after L samples, the counter output is 1.

$$\underbrace{\frac{1}{L} + \frac{1}{L} + \cdots + \frac{1}{L}}_{L \text{ samples}} = 1$$

Well, this is not an exact method to implement the interpolator control but this is pretty close. Refer to Figure 7.54 to see how an incrementing modulo-1 counter sits at the heart of an interpolation control block. While the figure looks intimidating, we can understand it in a straightforward manner.

On average, a modulo-1 counter increments its previous value by $1/L$. Consequently, the outcome of the modulo-1 interpolator control block is

$$\lambda[n+1] = (\lambda[n] + W[n]) \bmod 1 \quad (7.54)$$

Two concepts, $W[n]$ and modulo-1 operation, need to be clarified here.

- When we say 'on average', it means that instead of incrementing by $1/L$ each time, its previous value is incremented by

$$W[n] = \frac{1}{L} + e_F[n] \quad (7.55)$$

The flag should be raised for 1 out of every L samples. Not any random sample, but the one after which the interpolant is our symbol m decision.

where $e_F[n]$ is the output of the loop filter in a timing locked loop. Recall that the loop filter input is the timing error detector output $e_D[m]$ upsampled by L . We can say that *a filtered TED output drives the interpolator control block*.

Now assume that at the start, a modulo-1 operation has not occurred yet. Then, inserting Eq (7.55) into Eq (7.54),

$$\begin{aligned}\lambda[n+1] &= \left(\cdots + \frac{1}{L} + e_F[n-2] + \frac{1}{L} + e_F[n-1] + \frac{1}{L} + e_F[n] \right) \bmod 1 \\ &= \underbrace{\left(\cdots + \frac{1}{L} + \frac{1}{L} + \frac{1}{L} + \right)}_{\text{Term 1}} + \\ &\quad \underbrace{\left. \cdots + e_F[n-2] + e_F[n-1] + e_F[n] \right)}_{\text{Term 2}} \bmod 1\end{aligned}$$

Observe that Term 1 above does not need to be composed of L samples now to add up to a value greater or equal to 1. Instead, it may consist of more than L samples if Term 2 is sufficiently negative, or less than L samples if Term 2 is sufficiently positive. However, due to a relatively small equivalent noise bandwidth of the loop, the error introduced is very small so as not to be excessively driven by random noise within the input signal. As a consequence, the output $\lambda[n]$ counts by L most of the times, obviously with a superimposed trajectory similar to loop filter output $e_F[n]$, see Eq (7.55). To further clarify this point, some examples and figures are following soon.

- Modulo-1 operation ensures that $\lambda[n]$ is kept within the range $[0, 1)$ when its value becomes greater than 1. This happens after L samples on average and this situation is called an *overflow*. An overflow indicates the TLL that the previous sample n is actually n_m , i.e., if interpolated by μ_m , it generates the decision candidate for symbol m . In other words, an overflow condition indicates the location of the basepoint index n_m .

To summarize, an incrementing modulo-1 counter increments by $W[n]$ (a value close to $1/L$) such that it overflows after every L samples on average. Its operation is illustrated in Figure 7.55 for $L = 2$ samples/symbol where the matched filter output $z(nT_S)$, desired interpolants $z(n_m T_S + \mu_m T_S)$ and modulo-1 counter output $\lambda[n]$ are drawn along with the terminology used for interpolation control. As we said earlier, this terminology in interpolation control can get confusing so it is best to spend some time with this figure to understand what the related terms are. Some important observations are the following.

- Understandably, the matched filter outputs $z(nT_S)$ do not coincide with the optimal ISI-free time instants.
- Modulo-1 counter output $\lambda[n]$ is the main parameter to focus on in this figure. At each sample time nT_S , the value of $W[n]$ gets added to its previous contents. Recall that $W[n] \approx 1/L$ due to loop filter output $e_F[n]$ being very small. In this example of $L = 2$, observe that $\lambda[n]$ is striding upwards in increments of approximately $1/2$ during each cycle.

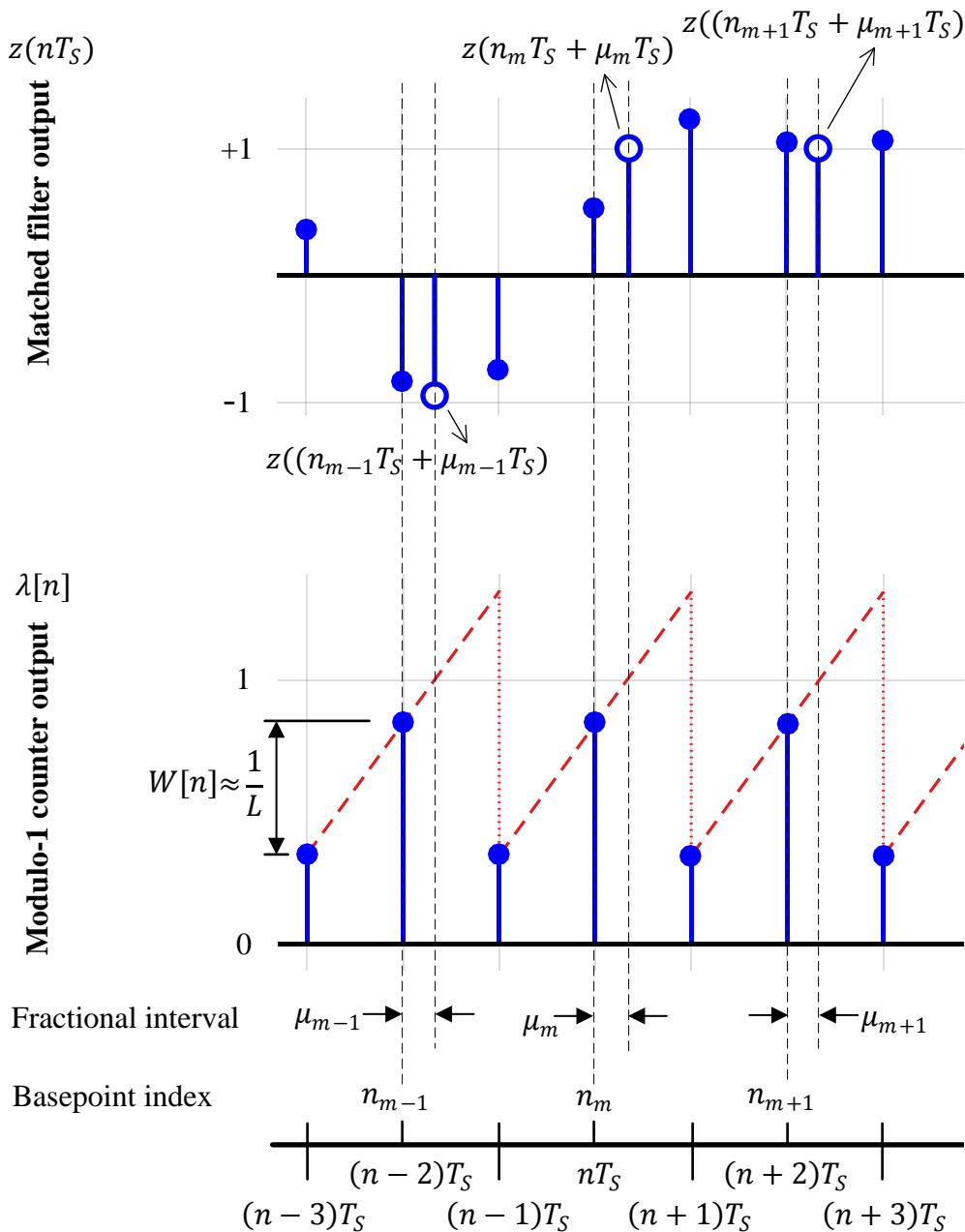


Figure 7.55: At $L = 2$ samples/symbol, relations between the available matched filtered samples based on nT_S and desired interpolants based on basepoint indices and fractional intervals

- As soon as $\lambda[n + 1]$ crosses 1, it is reduced by 1 by a modulo-1 operation.
- This boundary crossing of value 1 raises an overflow flag. So the overflow is equal to 1 at basepoint indices n_{m-1} , n_m and n_{m+1} , while it is equal to 0 for the rest of the sample times. Adding this 0101... series of overflow flag in the figure itself was avoided for the sake of simplification.
- Even though it is not evident from the figure, μ_{m-1} , μ_m and μ_{m+1} are all different values in general, unless
 - ◊ the TLL has acquired perfect timing phase lock,
 - ◊ there is zero timing or clock frequency offset, and
 - ◊ there is no noise or any other distortion present in the signal.

The subscript m indicates that a fractional interval μ_m is computed for each basepoint index n_m at the occurrence of the overflow flag. Next, we discuss how to compute the value of μ_m .

7.9.2 Computing Basepoint Indices and Fractional Intervals

Previously, we said that an overflow flag gets raised whenever the modulo-1 counter output $\lambda[n]$ crosses 1. This condition indicates the basepoint index n_m and triggers a command to the symbol detector that the coming interpolant is the desired symbol value.

To compute that interpolant, the fractional interval μ_m needs to be known. When an overflow occurs, a 1 is subtracted from $\lambda[n + 1]$ due to modulo-1 operation. So at $n_m + 1$, Eq (7.54) becomes

$$\lambda[n_m + 1] = \lambda[n_m] + W[n_m] - 1$$

where $n_m + 1$ must not be confused with n_{m+1} . We can write $W[n_m]$ as

$$W[n_m] = \lambda[n_m + 1] - \lambda[n_m] + 1 \quad (7.56)$$

Now consider Figure 7.56 where a zoomed in version of the samples around index n_m from Figure 7.55 is shown. Clearly, the two shaded triangles corresponding to $\lambda[n_m]$ and $1 + \lambda[n_m + 1]$ are similar triangles (similar triangles are those whose all three angles are equal).

This implies that the angle where both of these triangles are touching each other are equal. Or $\tan(\cdot)$ of these angles are also equal. The height of the triangle on the left is $1 - \lambda[n_m]$ while that of the triangle on the right is $(1 + \lambda[n_m + 1]) - 1 = \lambda[n_m + 1]$. Combining these facts and equating the tangents,

$$\frac{1 - \lambda[n_m]}{\mu_m} = \frac{\lambda[n_m + 1]}{1 - \lambda[n_m]}$$

which implies

$$(1 - \lambda[n_m] + \lambda[n_m + 1])\mu_m = 1 - \lambda[n_m]$$

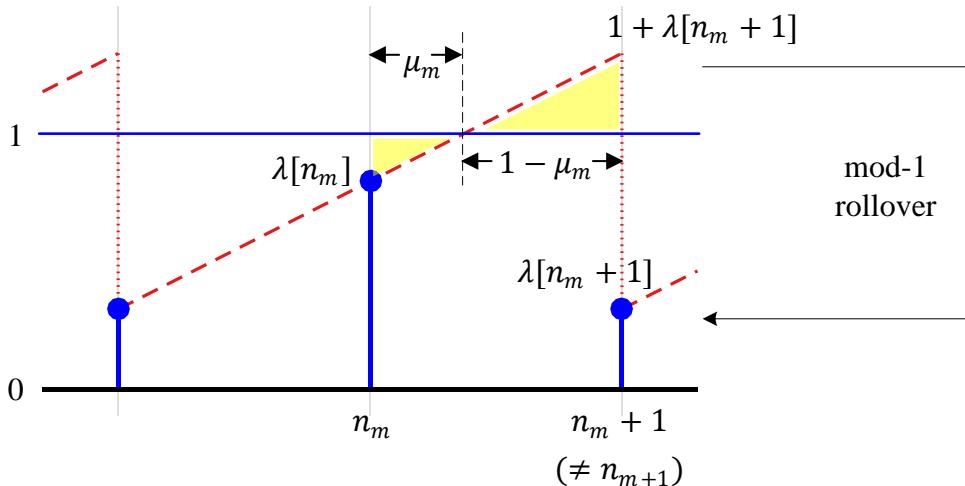


Figure 7.56: Formation of similar triangles to compute μ_m

Using Eq (7.56),

$$\mu_m = \frac{1 - \lambda[n_m]}{W[n_m]} \quad (7.57)$$

In summary, this is how the fractional interval μ_m is updated each time an overflow occurs. If the overflow flag stays low, the present value of μ_m is used again for interpolation. Finally, since the modulo-1 counter simply counts by 1 at the rate of error update, its gain K_0 in PLL calculations is unity.

In this respect, the role of modulo-1 counter in a timing locked loop is very similar to a numerically controlled oscillator (NCO) in a phase locked loop, where a modulo-1 operation (which in fact is a modulo- T_M operation) corresponds to a modulo- 2π operation of the NCO. This is a beautiful result, considering the analogy between symbol timing offset ε_Δ and carrier phase offset θ_Δ explored earlier in Table 7.1. With the details of interpolation and interpolator control in place, we can now draw a complete block diagram of a timing locked loop employing Gardner TED as an example. This is shown in Figure 7.57 which unites all the details involved in the implementation of a symbol timing synchronization system.

7.10 Sampling Clock Offset

In Section 7.2, we discussed the scenario of a clock frequency offset that is defined as the rate mismatch between the Tx and Rx clocks. This is commonly known as a *Sampling Clock Offset (SCO)*. For a Rx sampling the waveform at sampling intervals of \tilde{T}_S instead of T_S , an SCO and its normalized versions are defined as

$$\xi = \tilde{T}_S - T_S, \quad \xi_0 = \frac{\xi}{T_S} = \frac{\tilde{T}_S - T_S}{T_S} \quad (7.58)$$

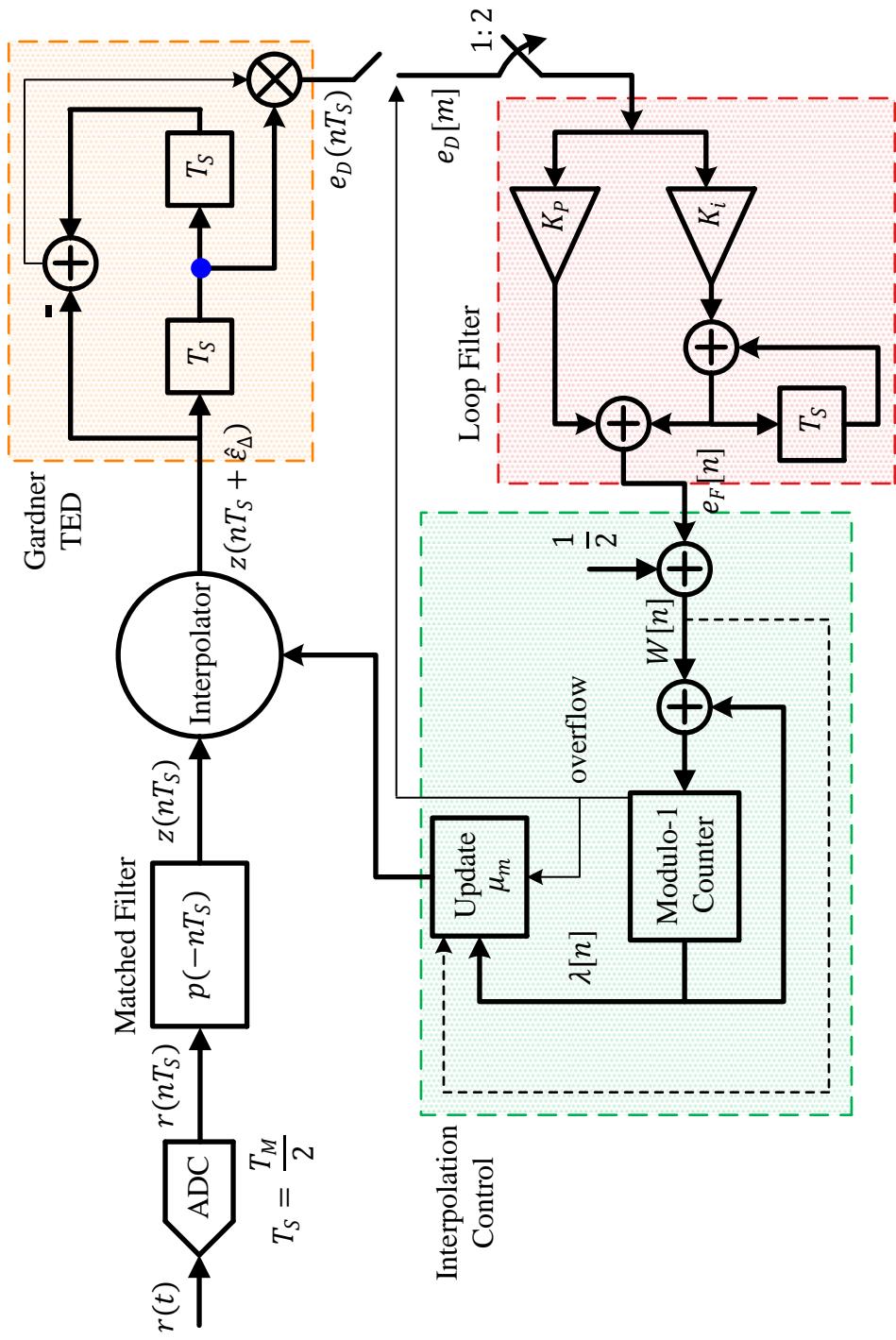


Figure 7.57: A timing locked loop with a Gardner TED, PI loop filter and interpolator control

Therefore, the Rx sample time is given by

$$\tilde{T}_S = T_S(1 + \xi_0)$$

A common analogy that captures this concept is driving on a highway next to a train line. Either the car is faster or slower than the train and you can see how a small initial speed difference culminates in one vehicle completely left behind. Our bodies also function on free-running biological clocks slightly unsynchronized with the 24-hour day and night cycle. Scientists have actually verified that a complete isolation from the outside world makes our sleep and wake-up patterns drift away.

The SCO ξ can be greater or lesser than the ideal value of 0 which determines whether the Rx clock is slower or faster than the Tx clock as described below.

$\xi > 0$: This implies that $\tilde{T}_S - T_S > 0$ or $\tilde{T}_S > T_S$, i.e., the Rx clock is slower than the Tx clock. Therefore, the Rx does not collect exactly L samples in a symbol period T_M as shown in Figure 7.58 for (an impractically large) $\xi = 1.1$. Ideally, the Rx should sample the waveform (shown by red vertical lines) at the locations of Tx samples shown as circles.

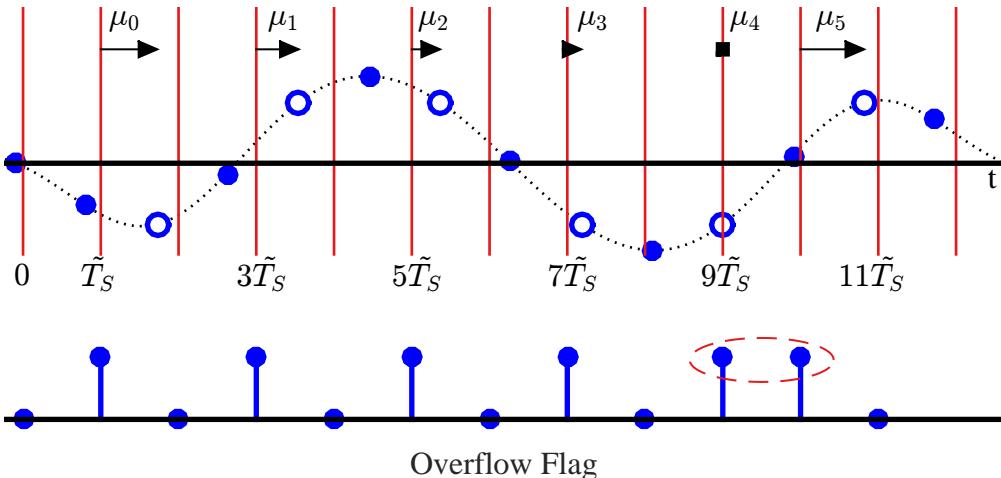


Figure 7.58: $\xi > 0$: Due to a Sampling Clock Offset (SCO), a slow Rx clock skips some samples

In Figure 7.58, assume an ideal scenario where the fractional interval μ_m is the same as the actual timing offset and let us focus on the fractional interval μ_m (represented by the time difference between a red dashed line and white circle) for $L = 2$. Observe that although the effect of $\xi > 0$ is initially small, the error accumulates with time and μ_0 at \tilde{T}_S is larger than μ_1 at $3\tilde{T}_S$. It keeps decreasing until it reaches zero around $9\tilde{T}_S$ and *wraps around zero at $10\tilde{T}_S$* ! The main point here is that the overflow that occurs every $L = 2$ samples now occurs at the very next sample – at both $9\tilde{T}_S$ and $10\tilde{T}_S$ – implying that the loop has missed a sample due to its slow rate^t. The wrap around of the fractional interval from 0 to 1 is an indication that a missing sample needs to be inserted in the TED registers, one

^tHere, we are using the term overflow as an indicator and not distinguishing between an overflow and an underflow according to the sign of ξ .

indication of which is that the overflow flag is high at two consecutive samples as shown in the figure.

$\xi < 0$: This implies that $\tilde{T}_S - T_S < 0$ or $\tilde{T}_S < T_S$, i.e., the Rx clock is faster than the Tx clock. In Figure 7.59, again assume an ideal scenario where the fractional interval μ_m is the same as the actual timing offset. Observe that the error accumulates with time and μ_1 at $3\tilde{T}_S$ is larger than μ_0 at \tilde{T}_S . It keeps increasing until it reaches close to one around $7\tilde{T}_S$ and then *wraps around one at $10\tilde{T}_S$* ! The main point here is that the overflow that occurs every two samples now occurs at a gap of 3 samples, implying that the Rx has inserted an extra sample due to its fast rate. The wrap around of the fractional interval from 1 to 0 is an indication that an extra sample needs to be deleted from the TED registers, one indication of which is that the overflow flag is low at two consecutive samples as shown in the figure.

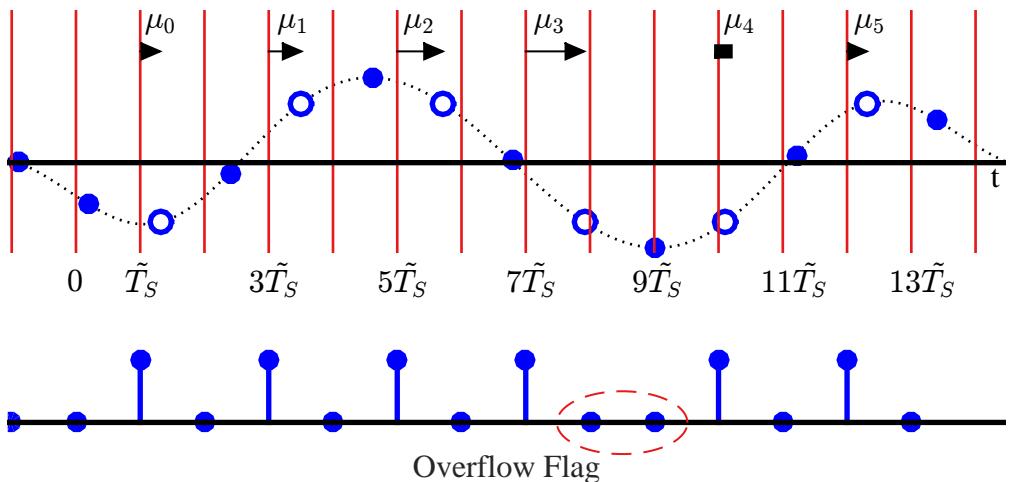


Figure 7.59: $\xi < 0$: Due to a Sampling Clock Offset (SCO), a fast Rx clock collects extra samples

Next, we describe the very useful M&M TED operating at 1 sample per symbol.

7.11 Feedback: Mueller and Muller (M&M) Timing Error Detector

In the previous sections, we saw that regeneration of a timing clock requires greater than 2 samples/symbol due to the bandwidth expansion and spectral line appearing at $\pm 1/T_M$ after squaring the input signal. We also showed why the clock itself is not important and the maximum eye opening can be found at just 2 samples per symbol through either driving the derivative to zero (symbol centric approach) or aligning one of the samples with zero crossings of the signal (zero crossing approach).

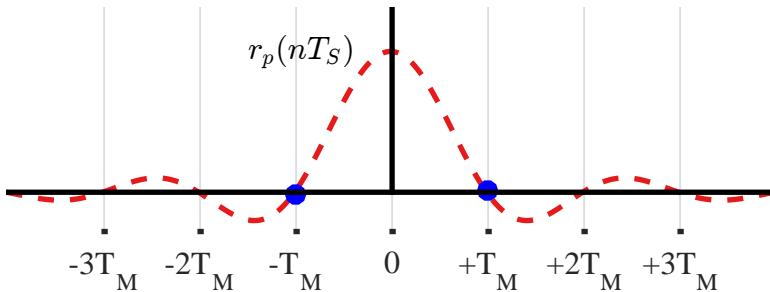
Until now, we have avoided the following question: the final decision device (threshold detector) operates at the symbol rate, i.e., T_M -spaced samples of the matched filter output are required to generate symbol decisions. However, we saw the effect of

symbol rate sampling at $\varepsilon_\Delta \neq 0$ in time and frequency domains in Section 7.1. So can there be a timing error detector that successfully operates at this minimum rate of just 1 sample/symbol? The widely popular Mueller and Muller TED[†] is one such example.

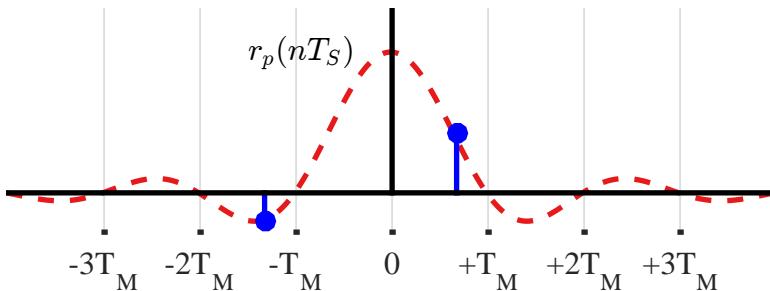
First, let us try to understand the thinking behind this timing error detector.

Utilizing the Pulse Symmetry

To understand the intuition behind M&M TED, Figure 7.60 draws the impulse response of a pulse auto-correlation function, a Raised Cosine in this case. When the timing is almost right, i.e., $\varepsilon_\Delta \approx 0$, then the two neighbouring samples taken before and after time T_M should be almost equal. This is because when a sample is taken close to the peak of the pulse auto-correlation, all other symbol-spaced samples are near the zero crossings. This is illustrated in Figure 7.60a.



(a) At $\varepsilon_\Delta = 0$, the TED output should be zero



(b) For $\varepsilon_\Delta \neq 0$, the TED output depends on the timing error

Figure 7.60: M&M TED utilizes the pulse symmetry to detect the error

Mathematically, this can be written as

$$r_p(+T_M - \varepsilon_\Delta) \approx r_p(-T_M - \varepsilon_\Delta)$$

and subsequently

$$r_p(+T_M - \varepsilon_\Delta) - r_p(-T_M - \varepsilon_\Delta) \rightarrow 0 \quad (7.59)$$

From here, it seems that we again need three samples to construct an expression for a TED, one for the symbol value to remove the modulation and the other two for finding the slope. This also makes sense due to the samples being $+T_M$ and $-T_M$

[†]Probably someone with a sweet tooth later named it as M&M TED.

away from the peak. However, Mueller and Muller devised a strategy that extracts the same information from two adjacent T_M -spaced samples without any help from an intermediate one. Their widely known algorithm is actually just a special case of their general formulation in Ref. [27].

The reasoning is quite simple: the relationship in Eq (7.59) is true only for timing offset $\varepsilon_\Delta \approx 0$. As soon as ε_Δ increases in magnitude, the asymmetry becomes visible. In Figure 7.60b for example, the right sample is larger than the left sample as ε_Δ becomes more negative. The difference expression in Eq (7.59) then generates a timing error signal that is proportional to the actual timing offset ε_Δ . This error signal can be incorporated into a TLL for the synchronization purpose. Recall the seesaw from Figure 7.28 to understand the working principle of the TED.

Note 7.8 Role of excess bandwidth α

From a time domain viewpoint, one can notice from Figure 7.60 that if the pulse autocorrelation has a large excess bandwidth α , its time sidelobes are quite small in magnitude and do not provide much timing information. On the other hand, a small excess bandwidth α gives rise to larger time sidelobes. We anticipate that an M&M TED performs well for a small excess bandwidth, as opposed to symbol centric and zero crossing TEDs encountered earlier that rely on the excess bandwidth to synchronize the signal.

From a frequency domain viewpoint, sampling signals with small excess bandwidth α at $L = 1$ sample/symbol implies less cancellation arising from the spectral null, see Figure 7.7 and therefore better extraction of timing information for the M&M TED. More of this spectral cancellation occurs for a large α .

The Two Terms

The question is how to obtain the two terms corresponding to the two seats of the seesaw, i.e., $r_p(+T_M - \varepsilon_\Delta)$ and $r_p(-T_M - \varepsilon_\Delta)$, that are employed for the error detection. We refer to Figure 7.61 for this purpose where reference 0 corresponds to symbol time m .

Generating $r_p(+T_M - \varepsilon_\Delta)$: Since the samples are symbol-spaced, their values are close to ± 1 in steady state. Any kind of summation tends *all* the samples towards zero. The matched filter output $z(mT_M)$ is made up of contributions from symbol m and all the neighbouring symbols. From symbol $m - 1$, the pulse autocorrelation $r_p(nT_S)$ brings a value $r_p(+T_M - \varepsilon_\Delta)$ at time m that is one symbol into the future. If we multiply $z(mT_M)$ with $a[m - 1]$, this removes the modulation of symbol $m - 1$ and only the term $r_p(+T_M - \varepsilon_\Delta)$ adds up coherently each time. This is drawn in Figure 7.61.

Generating $r_p(-T_M - \varepsilon_\Delta)$: From symbol m , the pulse autocorrelation $r_p(nT_S)$ sends a value $r_p(-T_M - \varepsilon_\Delta)$ back at symbol time $m - 1$ that is one symbol into the past. Multiplying the matched filter output $z((m - 1)T_M)$ with $a[m]$ removes the modulation of symbol m and produces only $r_p(-T_M - \varepsilon_\Delta)$.

Next, we translate the above findings into a more formal expression.

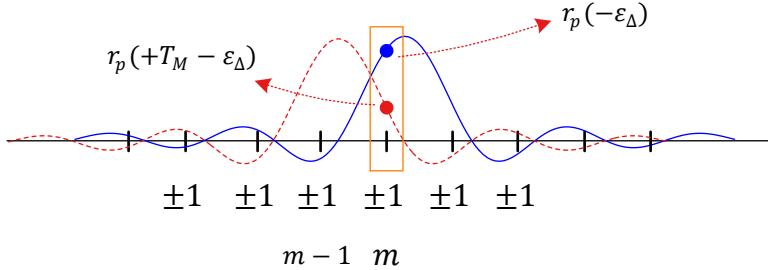


Figure 7.61: Logic behind extracting $r_p(+T_M - \varepsilon_\Delta)$

Derivation of M&M TED

First, the matched filter output is written as

$$\begin{aligned} z(nT_S) &= \left\{ \sum_i a[i] p(nT_S - iT_M - \varepsilon_\Delta) \right\} * p(-nT_S) \\ &= \sum_i a[i] r_p(nT_S - iT_M - \varepsilon_\Delta) \end{aligned}$$

For an initial timing estimate $\hat{\varepsilon}_\Delta = 0$, sampling at $n = mL = mT_M/T_S$ produces the output at $L = 1$ sample/symbol as

$$\begin{aligned} z(mT_M) &= \sum_i a[i] r_p(mT_M - iT_M - \varepsilon_\Delta) \\ &= \sum_i a[i] r_p\{(m-i)T_M - \varepsilon_\Delta\} \end{aligned}$$

Next, recall from Eq (7.77) (reproduced below) that for a binary PAM, the data symbols take values from a finite symbol set $\{-A, +A\}$. Each such option has a $1/2$ chance of occurring, hence whenever $i \neq m$, the average of $a[m] \cdot a[i]$ is zero, otherwise it is equal to A^2 for $i = m$.

$$\text{Mean}\{a[m] \cdot a[i]\} = \begin{cases} A^2 & i = m \\ 0 & \text{otherwise} \end{cases}$$

In the light of the above, let us evaluate the following expression.

$$\text{Mean}\{a[m-1] \cdot z(mT_M)\} = \text{Mean}\left\{a[m-1] \cdot \sum_i a[i] r_p\{(m-i)T_M - \varepsilon_\Delta\}\right\}$$

Only for $i = m-1$, the above average is non-zero. Therefore, plugging $i = m-1$ in the above expression,

$$\text{Mean}\{a[m-1] \cdot z(mT_M)\} = A^2 \cdot r_p(+T_M - \varepsilon_\Delta)$$

For a normalized constellation with $A^2 = 1$,

$$\text{Mean}\{a[m-1] \cdot z(mT_M)\} = r_p(+T_M - \varepsilon_\Delta) \quad (7.60)$$

Similarly,

$$\begin{aligned} \text{Mean}\{a[m] \cdot z((m-1)T_M)\} &= \\ \text{Mean}\left\{a[m] \cdot \sum_i a[i] r_p\{(m-1-i)T_M - \varepsilon_\Delta\}\right\} &= \\ &= r_p(-T_M - \varepsilon_\Delta) \end{aligned} \quad (7.61)$$

which is obtained through plugging $i = m$ above. Comparing the above two expressions with the basic principle behind balancing the matched filter output in Eq (7.59), it can be seen that

$$r_p(+T_M - \varepsilon_\Delta) - r_p(-T_M - \varepsilon_\Delta) = a[m-1]z(mT_M) - a[m]z((m-1)T_M)$$

With our usual notation of sampling at a phase $\hat{\varepsilon}_\Delta$, a timing error detector can be constructed as

$$e_D[m] = a[m-1]z(mT_M + \hat{\varepsilon}_\Delta) - a[m]z((m-1)T_M + \hat{\varepsilon}_\Delta)$$

(7.62)

This is the data-aided version of Mueller & Muller TED. For a decision-directed representation, the symbols $a[m]$ are replaced with their estimates $\hat{a}[m]$ as

$$e_D[m] = \hat{a}[m-1]z(mT_M + \hat{\varepsilon}_\Delta) - \hat{a}[m]z((m-1)T_M + \hat{\varepsilon}_\Delta) \quad (7.63)$$

A block diagram for the implementation of an M&M TED is shown in Figure 7.62 where the symbol T_M denotes a delay of a symbol time.

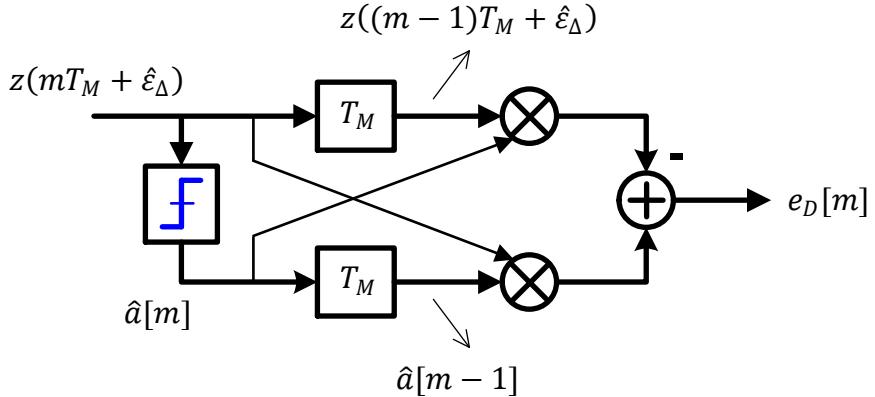


Figure 7.62: A block diagram for the implementation of M&M TED

Mean Curve for M&M TED

To derive the mean curve, first consider how a symbol timing offset ε_Δ and subsequent interpolation by $\hat{\varepsilon}_\Delta$ affect the matched filter output $z(mT_M)$, which was derived earlier in Eq (7.75) as

$$\begin{aligned} z(mT_M + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p(mT_M + \hat{\varepsilon}_\Delta - iT_M - \varepsilon_\Delta) \\ &= \gamma \sum_i a[i] r_p \left[(m-i)T_M - (\varepsilon_\Delta - \hat{\varepsilon}_\Delta) \right] \\ &= \gamma \sum_i a[i] r_p \left[(m-i)T_M - \varepsilon_{\Delta:e} \right] \end{aligned}$$

Here, $\varepsilon_{\Delta:e}$ was defined as

$$\varepsilon_{\Delta:e} = \varepsilon_\Delta - \hat{\varepsilon}_\Delta$$

Plugging in the expression for M&M TED,

$$\bar{e_D} = \text{Mean} \left\{ a[m-1]z(mT_M + \hat{\varepsilon}_\Delta) - a[m]z((m-1)T_M + \hat{\varepsilon}_\Delta) \right\}$$

Using the results from Eq (7.60) and Eq (7.61),

$$\bar{e_D} = \gamma A^2 \{ r_p(+T_M - \varepsilon_{\Delta:e}) - r_p(-T_M - \varepsilon_{\Delta:e}) \} \quad (7.64)$$

The above expression is the mean curve for a data-aided M&M TED. It is drawn for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.1, 0.5$ and 1 in Figure 7.63a. As described before, a *higher excess bandwidth produces a smaller slope at $\varepsilon_{\Delta:e} = 0$* here leading to a TED that is less sensitive to timing variations. Therefore, M&M TED works well with smaller excess bandwidths, the reasons for which were explained in Note 7.8 from time and frequency domain viewpoints. For the decision-directed scenario, the mean curve is the same as long as the decisions are correct which happens for $|\varepsilon_{\Delta:e}| < 0.35T_M$. For higher values of $\varepsilon_{\Delta:e}$, the assumption $\hat{a}[m] = a[m]$ used to derive the decision-directed mean curve is no longer valid. On the other hand, the mean curve and the standard deviation are drawn for a decision-directed scenario in Figure 7.63b and Figure 7.63, respectively. A quality metric as defined in Eq (7.27) can be constructed which demonstrates that the slope of the mean curve at $\varepsilon_{\Delta:e} = 0$ generates a larger numerator. However, for large timing errors, the variance for small excess bandwidth α is too high. We conclude that the M&M TED is a good choice for steady state tracking of the timing offset but not for the initial acquisition process.

From Eq (7.59), it is also evident that M&M TED has no self noise with Nyquist pulses. This is because when $\hat{\varepsilon}_\Delta \approx 0$, the Nyquist criterion ensures a zero TED output by making both $r_p(+T_M)$ and $r_p(-T_M)$ equal to zero. On the downside, the algorithm is very sensitive to carrier offsets. This enforces the earlier conclusion as M&M TED being a poor choice during the initial acquisition procedure.

The comments on TED gain K_D mentioned in Note 7.5 in regards to derivative TED apply in this scenario as well. Finally, in the case of a QAM scheme, the TED expression is given by the sum of inphase and quadrature parts. From Eq (7.62),

$$e_D[m] = a_I[m-1]z_I(mT_M + \hat{\varepsilon}_\Delta) - a_I[m]z_I((m-1)T_M + \hat{\varepsilon}_\Delta) + \\ a_Q[m-1]z_Q(mT_M + \hat{\varepsilon}_\Delta) - a_Q[m]z_Q((m-1)T_M + \hat{\varepsilon}_\Delta) \quad (7.65)$$

Let us explore its implementation in GNU Radio.

Exercise 7.1

The block ‘Clock Recovery MM’ implements an M&M TED with the following parameters.

Omega: This is our initial estimate of the number of samples/symbol L . In the presence of a Sampling Clock Offset (SCO), the actual value is slightly different and is unknown.

Gain Omega: This is the gain setting for the Integrator constant K_i of a Proportional +

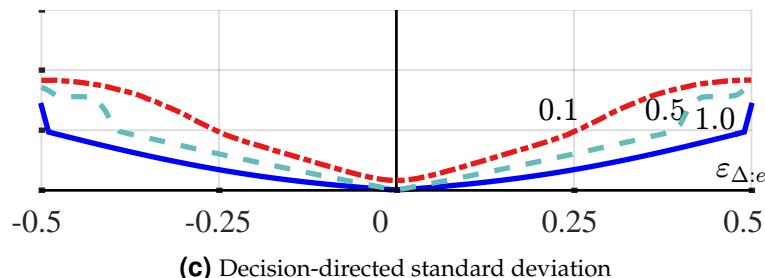
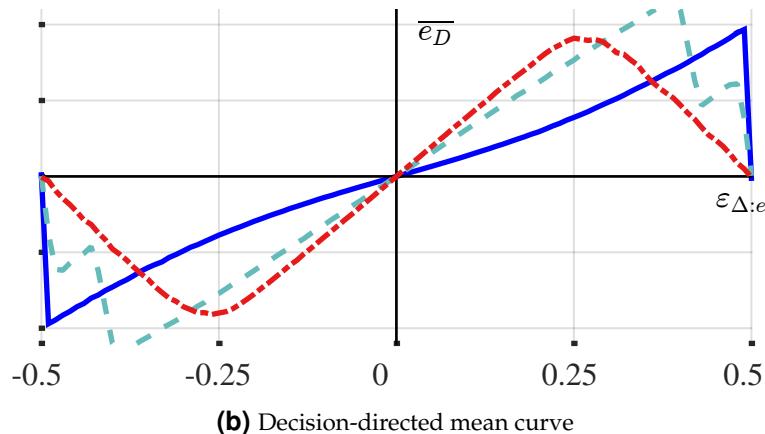
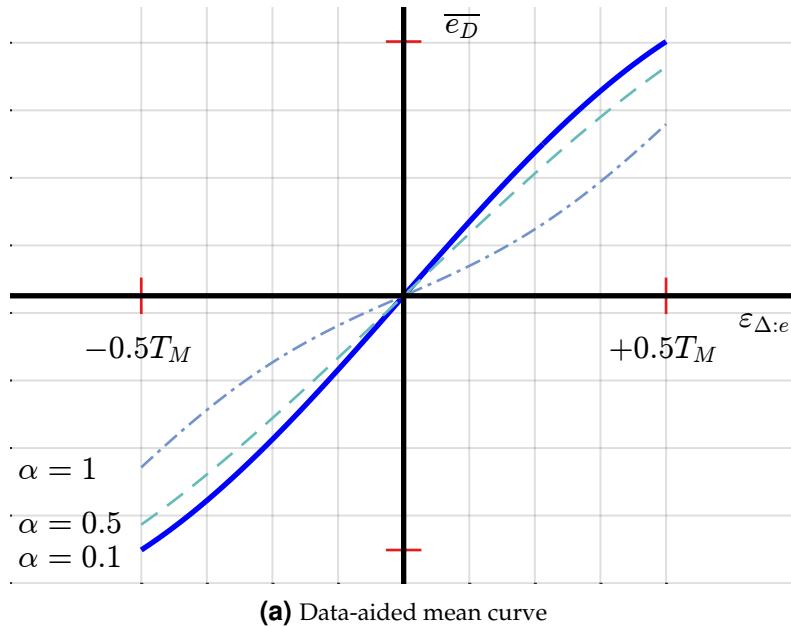


Figure 7.63: M&M TED for a binary PAM sequence and a Raised Cosine pulse as $r_p(nT_S)$ with excess bandwidth $\alpha = 0.1, 0.5$ and 1

Integrator (PI) loop filter. You should set this value from Eq (4.9) as

$$\text{'Gain Omega'} = \frac{\text{'Gain Mu'}^2}{4}$$

where 'Gain Mu' is the proportional gain K_p , see below.

Mu: This is our initial guess of the timing phase offset ε_Δ . While an initial estimate L of Omega above is reasonably close to the actual value, it is impossible to guess ε_Δ in advance. Therefore, it is best to set this value equal to 0.5 with the implicit assumption that ε_Δ is uniformly distributed between 0 and 1.

Gain Mu: This is the proportional part K_p of the PI loop filter. Now the reason M&M clock recovery is considered as a confusing block is that it is only the loop bandwidth that is required in many other synchronization blocks while here the input is directly K_p and K_i (also called 'alpha' and 'beta', respectively) of the loop filter. The general guideline to set the loop bandwidth has been mentioned before as starting it around $2\pi/100$ and tune it accordingly. When the user sets a normalized loop bandwidth $B_n T_S$, say 1/100, and assuming $\zeta = 1$, Eq (4.6) gives us the normalized natural frequency θ_n . From here, we can find this value K_p from Eq (4.8) as

$$\text{'Gain mu'} = K_p = 4\theta_n$$

Omega Relative Limit: This sets the maximum relative deviation from Omega above and depends on the clock specifications, i.e., ppm rating.

You can think of this implementation of M&M TED in GNU Radio as *explicitly* correcting both the timing phase offset ε_Δ as well as the sampling clock offset ξ . If a signal is sampled at time mT_M , then the subsequent sample time in an ideal case is $mT_M + T_M$. In current GNU Radio implementation, a slowly changing symbol time is injected to mitigate the sampling clock offset ξ as

$$\text{Updated symbol time} = \text{Previous symbol time} + (\text{'Gain Omega'})e_D[m]$$

$$\begin{aligned} \text{Updated symbol phase} = & \text{Previous symbol phase} + \text{Updated symbol time} + \\ & (\text{'Gain Mu'})e_D[m] \end{aligned}$$

where the expression 'symbol phase' refers to the timing phase and not a carrier phase. Now we simulate an M&M clock recovery scheme in GNU Radio choosing the default values for the above parameters as $K_p = 0.175$ and $K_i = K_p^2/4$. For a 16-QAM scheme with excess bandwidth $\alpha = 0.1$ and $L = 4$ samples/symbol, Figure 7.64 plots the input and output constellations. As an exercise for the reader, what is the loop bandwidth computed from the default parameters in this block?

Keep in mind that since the M&M TED does not involve any cancellation of phase through a conjugate product, it is not carrier independent and thus more suitable for timing synchronization after the settling of carrier loops.

7.12 Feedback: Band Edge Timing Error Detector

In Section 6.4.3, we discussed band edge filters where I said that this is one of the most delightful topics for a person seeking to understand the field of signal synchronization. While the discussion in that section was in the context of carrier frequency offset, now we will see how the band edge filters aid in timing synchronization as well.

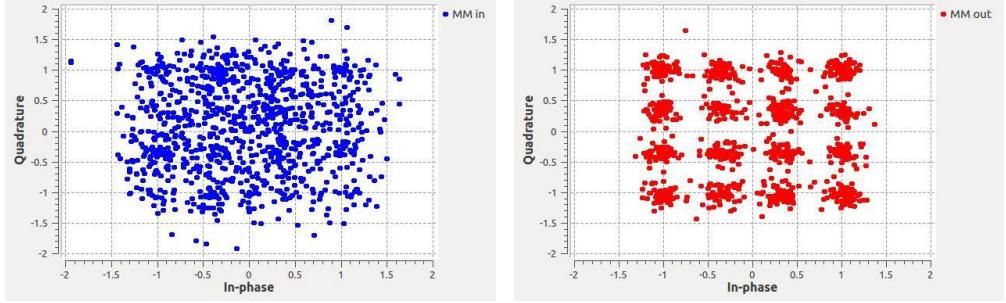


Figure 7.64: Constellation diagram for an M&M clock recovery scheme for 16-QAM

As a recap, we constructed an upper band edge filter $H_{BE+}(F)$ and a lower band edge filter $H_{BE-}(F)$ as shown in Figure 6.29. Then, we obtained two new filters by adding and subtracting $H_{BE+}(F)$ from $H_{BE-}(F)$: a sum band edge filter that has an even symmetric frequency response while a difference band edge filter that has an odd symmetric frequency response. As a consequence, they were named as *even band edge filter* $H_{BE,even}(F)$ and *odd band edge filter* $H_{BE,odd}(F)$, respectively, and were illustrated in Figure 6.30.

$$H_{BE,even}(F) = H_{BE-}(F) + H_{BE+}(F)$$

$$H_{BE,odd}(F) = H_{BE-}(F) - H_{BE+}(F)$$

Also, we introduced the outputs of even and odd band edge filters as

$$z_{BE,even}(nT_S) = z_{BE-}(nT_S) + z_{BE+}(nT_S)$$

$$z_{BE,odd}(nT_S) = z_{BE-}(nT_S) - z_{BE+}(nT_S)$$

Next, we formed an error signal with the available signals as

$$e_D[n] = \left\{ z_{BE,even}(nT_S) \cdot z_{BE,odd}^*(nT_S) \right\}_I$$

which helped us in acquiring the Carrier Frequency Offset (CFO). Let us now explore the Q component of the above expression. Plugging in the definitions of $z_{BE,even}(nT_S)$ and $z_{BE,odd}(nT_S)$,

$$\begin{aligned} e_D[n] &= \left\{ \left(z_{BE-}(nT_S) + z_{BE+}(nT_S) \right) \left(z_{BE-}(nT_S) - z_{BE+}(nT_S) \right)^* \right\}_Q \\ &= \left\{ z_{BE-}(nT_S) z_{BE-}^*(nT_S) - z_{BE+}(nT_S) z_{BE+}^*(nT_S) + \right. \\ &\quad \left. z_{BE+}(nT_S) z_{BE-}^*(nT_S) - z_{BE-}(nT_S) z_{BE+}^*(nT_S) \right\}_Q \\ &= \underbrace{\left| z_{BE-}(nT_S) \right|^2 - \left| z_{BE+}(nT_S) \right|^2}_{\text{Term 1}} + \\ &\quad \underbrace{\left. z_{BE+}(nT_S) z_{BE-}^*(nT_S) - \left(z_{BE+}(nT_S) z_{BE-}^*(nT_S) \right)^* \right\}_Q}_{\text{Term 2}} \end{aligned} \quad (7.66)$$

- Term 1 consists of the magnitude squared values and hence has no Q component. Actually, this is the I component of the error signal that computes the difference between the two band edge outputs as an indicator of a CFO, as explained in Section 6.4.3. Keeping only the Q part, the above error signal $e_D[n]$ becomes

$$e_D[n] = \left\{ z_{BE+}(nT_S)z_{BE-}^*(nT_S) - \left(z_{BE+}(nT_S)z_{BE-}^*(nT_S) \right)^* \right\}_Q$$

- Term 2 is a difference of two complex signals, one of which is a conjugate of the other shown above. The I component of both a complex signal and its conjugate are the same. Therefore, this I part cancels out in the above expression as derived before in Eq (1.24) and reproduced below for a complex signal V .

$$\begin{aligned} I &\rightarrow 0 = \frac{1}{2} \{ V - V^* \} \\ Q &\uparrow V_Q = \frac{1}{2} \{ V - V^* \} \end{aligned}$$

Consequently, we are left with the error signal $e_D[n]$ (ignoring an irrelevant factor of 2) as

$$e_D[n] = \left\{ z_{BE+}(nT_S)z_{BE-}^*(nT_S) \right\}_Q \quad (7.67)$$

It is interesting to find that the quadrature component of the product of upper and conjugate lower band edge filters is the same as that of the even and conjugate odd band edge filters (ignoring a scaling factor of 2), although their inphase components are different.

To see how this approach works for timing synchronization, we write

$$\begin{aligned} |\cdot| &= |z_{BE+}(nT_S)| \cdot |z_{BE-}(nT_S)| \\ \angle(\cdot) &= \angle z_{BE+}(nT_S) - \angle z_{BE-}(nT_S) \end{aligned}$$

where the minus sign is due to the conjugate operation. Now utilize the definition of the quadrature component of the complex signals as

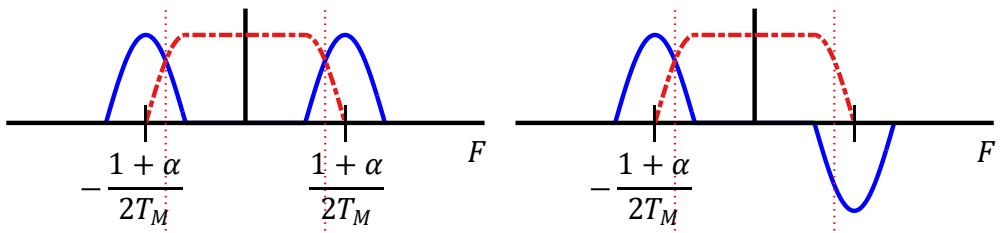
$$e_D[n] = |z_{BE+}(nT_S)| \cdot |z_{BE-}(nT_S)| \cdot \sin(\angle z_{BE+}(nT_S) - \angle z_{BE-}(nT_S)) \quad (7.68)$$

Next, we explore the implications of the sine wave above in the context of even and odd band edge filters – $H_{BE,even}(F)$ and $H_{BE,odd}(F)$ – respectively, since the discussion is easier due to their symmetry properties. The same concepts are built through upper and lower band edge filters ($H_{BE+}(F)$ and $H_{BE-}(F)$) later.

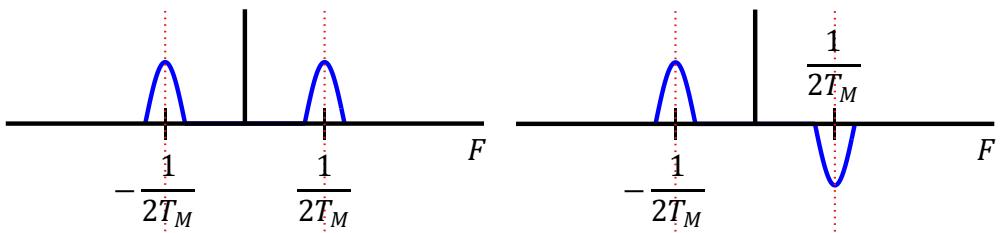
Sine Wave with a Zero CFO

While the error signal in Eq (7.68) is seen to be a sine wave, we are not clear what the frequency of that sinusoid is. For this purpose, we start with the example of a zero CFO in the Rx signal and trace the following steps.

- With a zero CFO, the incoming signal arrives right at the center of the spectrum for both even and odd band edge filters as shown in Figure 7.65a.
- The input square-root Nyquist spectrum has a quarter cycle of a cosine in its transition band, as described in Section 3.6. The band edge filter is a half cycle of a sine at $\pm 0.5/T_M$ as discussed in Section 6.4.2.
- At the lower band edge, the product of the quarter cycle cosine in the input spectrum and a quarter cycle sine in the band edge filter produces a half cycle sine at $-1/2T_M$ because $\sin 2A = 2 \cos A \sin A$. A similar product appears at $+1/2T_M$ for the upper band as drawn in Figure 7.65b. This completes the even band edge filter output $z_{\text{BE,even}}(nT_S)$.



(a) Input signal arriving at the middle of the two band edge filters



(b) Output from the filters is centered at $\pm 1/2T_M$

Figure 7.65: Symmetric output from even and odd band edge filters in the presence of a zero CFO

- The odd band edge filter output $z_{\text{BE,odd}}(nT_S)$ is computed in a similar manner as above and drawn in Figure 7.65. Considering $z_{\text{BE,odd}}^*(nT_S)$, recall from the DFT properties in Table 2.4 that the spectrum of a signal conjugate is given by

$$s^*[n] \xrightarrow{\mathcal{F}} S^*(-k) \bmod N$$

where $\bmod N$ is due to the discrete frequency domain. So for a regular scenario, both conjugation and reversal of the spectrum need to be performed. Since the spectrum of the odd band edge filter has only an I frequency component, see Figure 6.30b, its conjugate is the same and hence only a reversal of the spectrum is required.

- The underlying error signal (not just the quadrature part) is produced as a result of multiplication of the even output $z_{\text{BE,even}}(nT_S)$ with the conjugate of the odd output, i.e., $z_{\text{BE,odd}}^*(nT_S)$, in time domain.

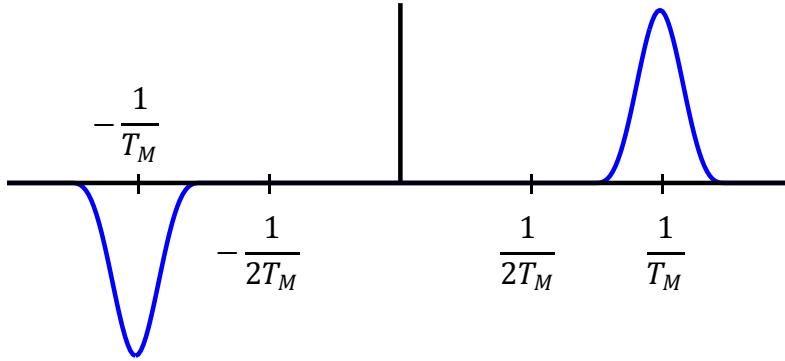


Figure 7.66: When there is no CFO, multiplication of the terms in $e_D[n]$ in time domain, i.e., convolution in frequency domain, generates an odd spectrum centered at $\pm 1/T_M$. Figure is scaled to show an enlarged convolution output

- Multiplication of $z_{\text{BE,even}}(nT_S)$ and $z_{\text{BE,odd}}^*(nT_S)$ in time domain launches a convolution of their spectra in frequency domain. When such a convolution is performed between the left spectrum shown in Figure 7.65b with the flipped version of the spectrum on the right of Figure 7.65b, the output is illustrated in Figure 7.66. To create this result, remember that flipped version of the even spectrum is the same and then imagine it sliding over the flipped odd spectrum. Notice the absence of a DC term due to the balanced positive and negative spectra since there is no CFO in the signal.

- Remember from Eq (2.9) that the convolution result starts at

$$\text{First sample of first signal} + \text{First sample of second signal}$$

which in this case leads to lower band spectrum centered at

$$-\frac{1}{2T_M} - \frac{1}{2T_M} = -\frac{1}{T_M}$$

Similarly, the upper band spectrum is centered at $+1/T_M$.

- Now consider an ideal case where the width of the resultant spectrum is really small and hence can be imagined as an impulse (we will see soon why this is actually true in Note 7.10). Then, we have two impulses centered at $\pm 1/T_M$ with opposite amplitudes, i.e., we have two complex sinusoids.

- Until now, we followed the steps towards revealing the spectrum of the basic error signal.

$$\text{spectrum of } z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S)$$

and we need to find a straightforward route to look at the spectrum of its time domain quadrature part from here.

$$\text{spectrum of } \left\{ z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S) \right\}_Q$$

- Compare Figure 7.66 with Figure 1.31 that draws the spectrum of a sine wave existing on time Q -plane. Hence, these two complex sinusoids centered at $\pm 1/T_M$ with opposite signs give rise to *a sine wave with frequency equal to the symbol rate $1/T_M$* , which is the frequency of the sinusoid discovered in Eq (7.68)[†].

With this approach in hand, the same result can be reached through considering the upper and lower band edge filters (this is true for Q component only; remember that the inphase part of the above two approaches is different). The time domain conjugation of the lower band edge filter output $z_{BE_even}^*(nT_S)$ flips the spectrum from $-1/2T_M$ to bring it to $+1/2T_M$. Next, this convolves with a similar spectrum of the even band edge filter output at $+1/2T_M$, thus producing a single lobe at $+1/T_M$. Considering it as a single complex sinusoid at $+1/T_M$, its Q component in time domain is a sine wave.

To summarize the above discussion, our *Band Edge Timing Error Detector (TED)* can be defined as

$$e_D[n] = \left\{ z_{BE,even}(nT_S) \cdot z_{BE,odd}^*(nT_S) \right\}_Q \quad (7.69)$$

To confirm our analysis, when this band edge TED is applied to an incoming QAM stream with excess bandwidth $\alpha = 0.25$ and $L = 4$ samples/symbol, the spectrum of its output is illustrated in Figure 7.67. Observe the two timing lines at symbol rate $\pm 1/T_M$ which are exactly the same as obtained through simply squaring the signal and drawn in Figure 7.17. The difference is that the band edge filters have eliminated

[†]For those who are familiar with Fourier Transform properties, another way to look at this is as follows. From the symmetry property of Fourier Transform, an odd spectrum with only I component has a time domain response that has only an odd Q component, see Section 2.9. An odd Q component of two complex sinusoids at $\pm 1/T_M$ is nothing but a sine wave of frequency $1/T_M$.

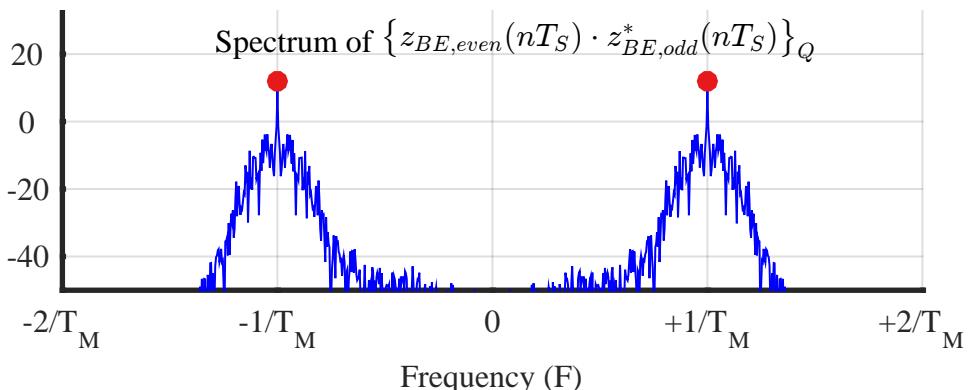


Figure 7.67: Spectrum of $e_D[n]$ as the *quadrature component* of the product between the even band edge filter output $z_{BE,even}(nT_S)$ and conjugate of odd band edge filter output $z_{BE,odd}^*(nT_S)$, generated for $L = 4$ samples/symbol and excess bandwidth $\alpha = 0.25$. Two timing lines at $\pm 1/T_M$ aid in timing acquisition while Figure 6.32 shows the inphase part of same $e_D[n]$ generating the CFO line at DC

much of the self noise entering the timing locked loop from the inclusion of the matched filter.

For obvious reasons, the magnitude squared spectrum conceals the Q component nature of the underlying sinusoid in this figure.

Note 7.9 Where is the timing offset ε_Δ ?

The *phase of the above symbol rate sinusoid*, whether generated through squaring the Rx signal or through band edge filters, is the timing offset ε_Δ . This is shown in Figure 7.68. The zero crossings of such a waveform establish the maximum eye openings for detection purpose.

There is a little caveat though. From Eq (7.68), we know that the frequency difference adds up due to the conjugate as

$$\frac{1}{2T_M} - \left(-\frac{1}{2T_M} \right) = +\frac{1}{T_M}$$

On the other hand, the phase difference between upper and lower band edge filters plays its part as well. Nevertheless, as long as the filters are even symmetric around the band edge frequencies, these two phases cancel out due to that symmetry.

In regards to how to use thus created sinusoid, it can be input to a regular PLL that generates its cleaner version at the same phase and rate.

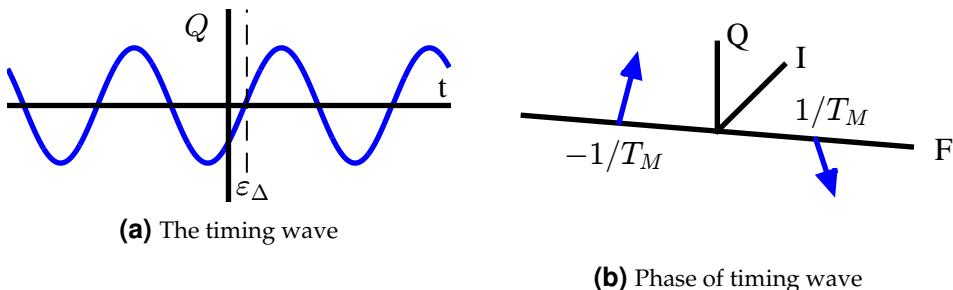
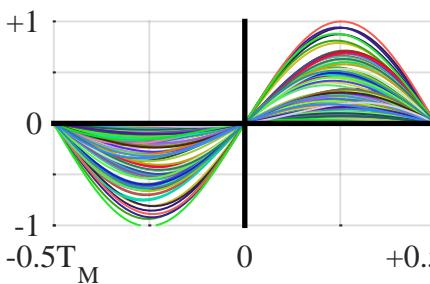


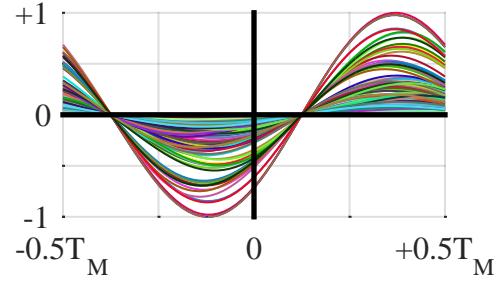
Figure 7.68: Phase of the sine wave at symbol rate produces the timing

Now we turn our attention towards the time domain waveform. For this purpose, Figure 7.69a generates the eye diagram for the band edge TED output for zero Carrier Frequency Offset (CFO), symbol timing offset $\varepsilon_\Delta = 0$ and excess bandwidth $\alpha = 0.5$. Clearly, the TED output waveform is a pure sinusoid and its zero crossings coincide with the maximum eye openings of the shaped data sequence according to the timing offset. The mean value of the TED output is also drawn in Figure 7.69c as an average of the eye diagram above and serves as an average characteristic of the band edge TED. The 'S' shape is evident while the TED output is fixed at zero.

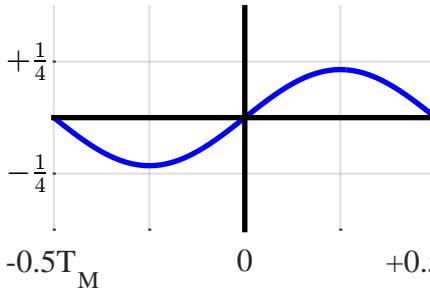
Next, the same figures are generated for $\varepsilon_\Delta = 0.125T_M$ and zero CFO. Figure 7.69b illustrates the zero crossings of the TED output coinciding with the maximum eye openings of the shaped data sequence in proportion to the timing offset. The mean value of the TED output is also drawn in Figure 7.69d as an average of the eye diagram above and demonstrates a timing shift of $\varepsilon_\Delta = 0.125T_M$.



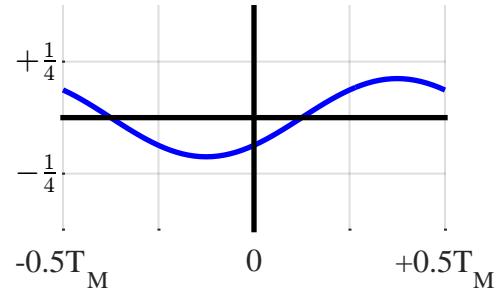
(a) Eye diagram for band edge TED for $\varepsilon_\Delta = 0$



(b) Eye diagram for band edge TED for $\varepsilon_\Delta = 0.125T_M$



(c) Mean curve generated from the eye diagram



(d) Mean curve generated from the eye diagram

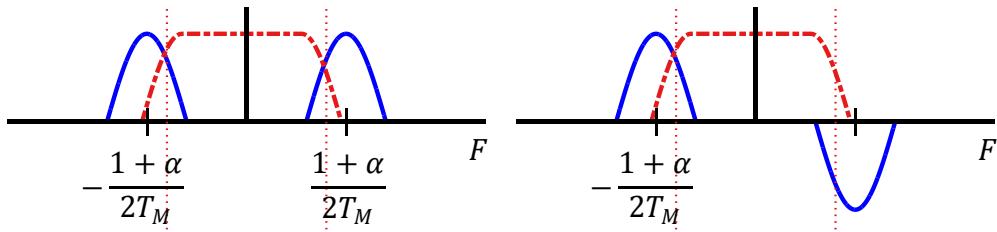
Figure 7.69: Eye diagrams and band edge TED mean curves for $\varepsilon_\Delta = 0$ and $\varepsilon_\Delta = 0.125T_M$. Zero crossings of the band edge TED output coincide with the maximum eye opening of the data sequence

Sine Wave with a Nonzero CFO

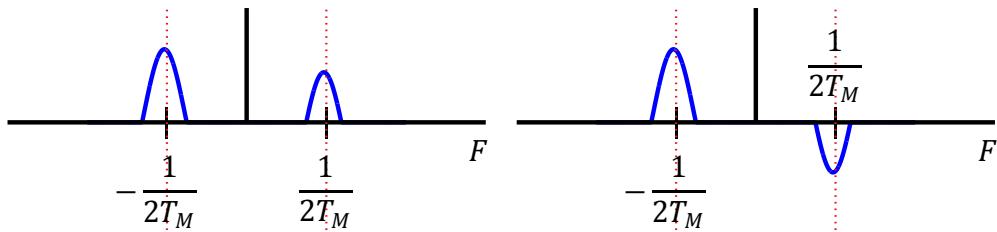
With a nonzero CFO, the incoming signal arrives shifted away from the center of the spectrum for both even and odd band edge filters. For example, a left shifted spectrum is shown in Figure 7.70a. At the lower band edge, the product of the quarter cycle cosine in the input spectrum and a quarter cycle sine in the band edge filter produces a half cycle sine close to (but not centered at) $-1/2T_M$. As compared to Figure 7.65b, this spectrum is wider and taller due to a larger overlap. On the other side, a similar product appears close to (but not centered at) $+1/2T_M$ for the upper band as drawn in Figure 7.70b which is narrower and shorter as compared to no CFO case in Figure 7.65b (imagining them as imbalanced impulses, I would have called it an 'ugly' sinusoid).

Following exactly the same steps as in the case of a zero CFO, we perform the convolution of the two spectra to reveal the output spectrum for $z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S)$ (not the Q component). This is illustrated in Figure 7.71 where again the spectral lobes can be imagined as impulses and we will shortly see in Note 7.10 why this is actually true. As opposed to Figure 7.66, the DC term here does not cancel out and produces an output corresponding to the CFO size in the incoming signal.

To see why, the spectrum of the odd band edge filter is flipped due to the conjugate operation. It is flipped again for the convolution purpose and slid over the even band



(a) Input signal arriving at the left of the spectrum



(b) Output from the filters is shifted a little away from $\pm 1/2T_M$

Figure 7.70: Asymmetric output from even and band edge filters in the presence of nonzero CFO

edge spectrum. The convolution starts with the small negative lobe on the bottom right of Figure 7.70b sliding over the large positive lobe on the bottom left. When this process moves forward and reaches 0 that generates the DC term, the two large positive lobes are overlapping each other while the two small lobes are overlapping each other (with one being negative). This instant in frequency domain, frozen at $F = 0$, is illustrated in Figure 7.72 and the area under this curve is an indicator of the amount of CFO present. This is another angle of looking at the energy difference perspective for CFO generation encountered before in Figure 6.37a.

During this procedure, the convolution generates an output from a larger lobe slid-

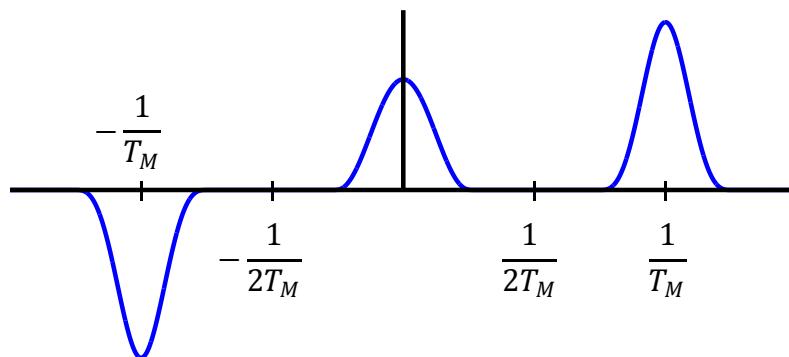


Figure 7.71: In the presence of a CFO, multiplication of the terms in $e_D[n]$ in time domain, i.e., convolution in frequency domain, generates a corresponding term at DC and an odd spectrum centered at $\pm 1/T_M$

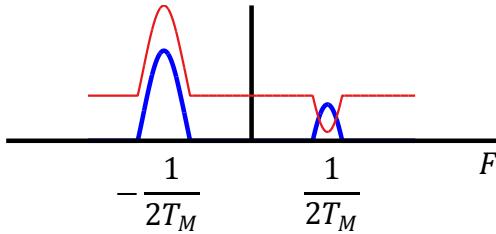


Figure 7.72: Spectral convolution generated for $F = 0$ that outputs a DC term proportional to the energy difference of the two overlapping regions which is the energy difference dependent on the CFO. Compare with Figure 6.37a

ing over a smaller lobe both at $-1/T_M$ and $+1/T_M$. As we explained in the zero CFO case before, the error signal $\{z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S)\}_Q$ generates a symbol rate sinusoid from the resulting impulses.

To confirm our findings, when this band edge TED is applied to an incoming QAM stream with excess bandwidth $\alpha = 0.25$ and $L = 4$ samples/symbol in the presence of a CFO, the spectrum of its output is illustrated in Figure 7.73.

- The inphase component, i.e., the spectrum of

$$\left\{ z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S) \right\}_I$$

reveals the CFO line at DC drawn in Figure 7.73a.

- The quadrature component, i.e., the spectrum of

$$\left\{ z_{\text{BE,even}}(nT_S) \cdot z_{\text{BE,odd}}^*(nT_S) \right\}_Q$$

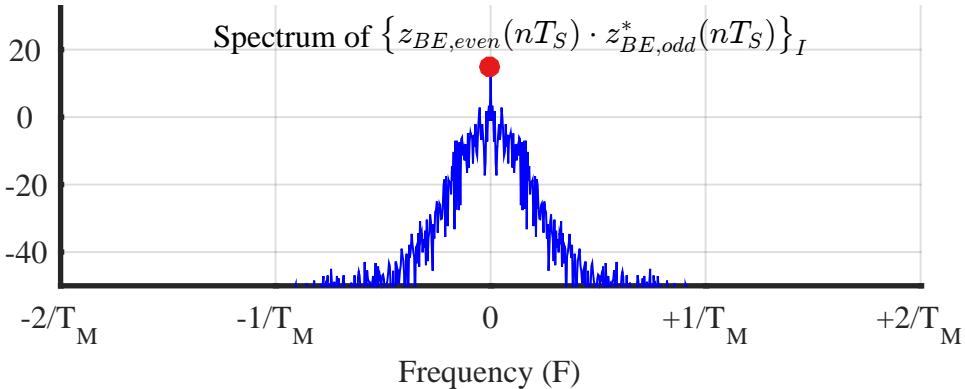
reveals the symbol rate timing lines at $\pm 1/T_M$ drawn in Figure 7.73b which are exactly the same as obtained through simply squaring the signal and drawn in Figure 7.17. The difference is that the band edge filters have eliminated much of the self noise entering the timing locked loop from the inclusion of the matched filter.

As far as the time domain waveform is concerned, the presence of a CFO makes no difference from the eye diagram and mean TED curve generated in Figure 7.69a with and without a symbol timing offset. The TED output waveform is still a pure sinusoid with zero crossings coinciding with the maximum eye openings of the shaped data sequence according to the timing offset ε_Δ .

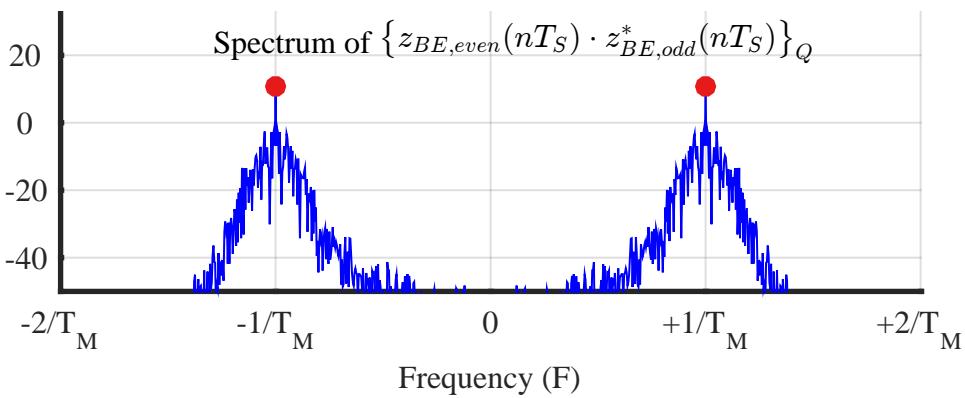
Note 7.10 From spectral lobes to impulses

As a final remark, we imagined the output lobes in Figure 7.66 and Figure 7.71 as impulses at $\pm 1/T_M$. In an actual plot of the spectrum in several figures, e.g., in Figure 7.73, we indeed encountered the impulses instead of the lobes.

The rationale behind this behaviour was explained in Section 7.4.1 that the filter outputs come from a shaped long binary random sequence. The spectrum therefore contains a summation of complex sinusoids with different frequencies and amplitudes scaled by the random sequence which add constructively at some points and destruc-



(a) Inphase component generates a DC line proportional to CFO



(b) Quadrature component generates timing lines at symbol rate $\pm 1/T_M$

Figure 7.73: Spectrum of inphase and quadrature components of $e_D[n]$ generated for $L = 4$ samples/symbol and excess bandwidth $\alpha = 0.25$

tively at others. As a consequence, it contains several short peaks and troughs in a random manner.

On the other hand, due to the odd symmetry of the pulse around $\pm 1/2T_M$, when the sliding spectrum reaches $-1/T_M$ or $+1/T_M$ during convolution, it overlaps with a similar curve but with lower values, an example of which is drawn in Figure 7.19b. Their addition after point by point product generates a structure within all this randomness that are the spectral lines at $\pm 1/T_M$.

A block diagram for the implementation of a band edge filter based timing and carrier recovery is drawn in Figure 7.74 where even and odd band edge filter outputs are constructed through addition and subtraction of the outputs from the upper and lower band edge filters, respectively. The inphase part of the error signal then drives the carrier frequency recovery loop while the quadrature part of the error signal drives the timing synchronization loop. In an actual implementation, a third loop drives the carrier phase towards zero around the matched filter as discussed in detail in Chapter 5.

One significant advantage a band edge TED has is the following. When we discuss

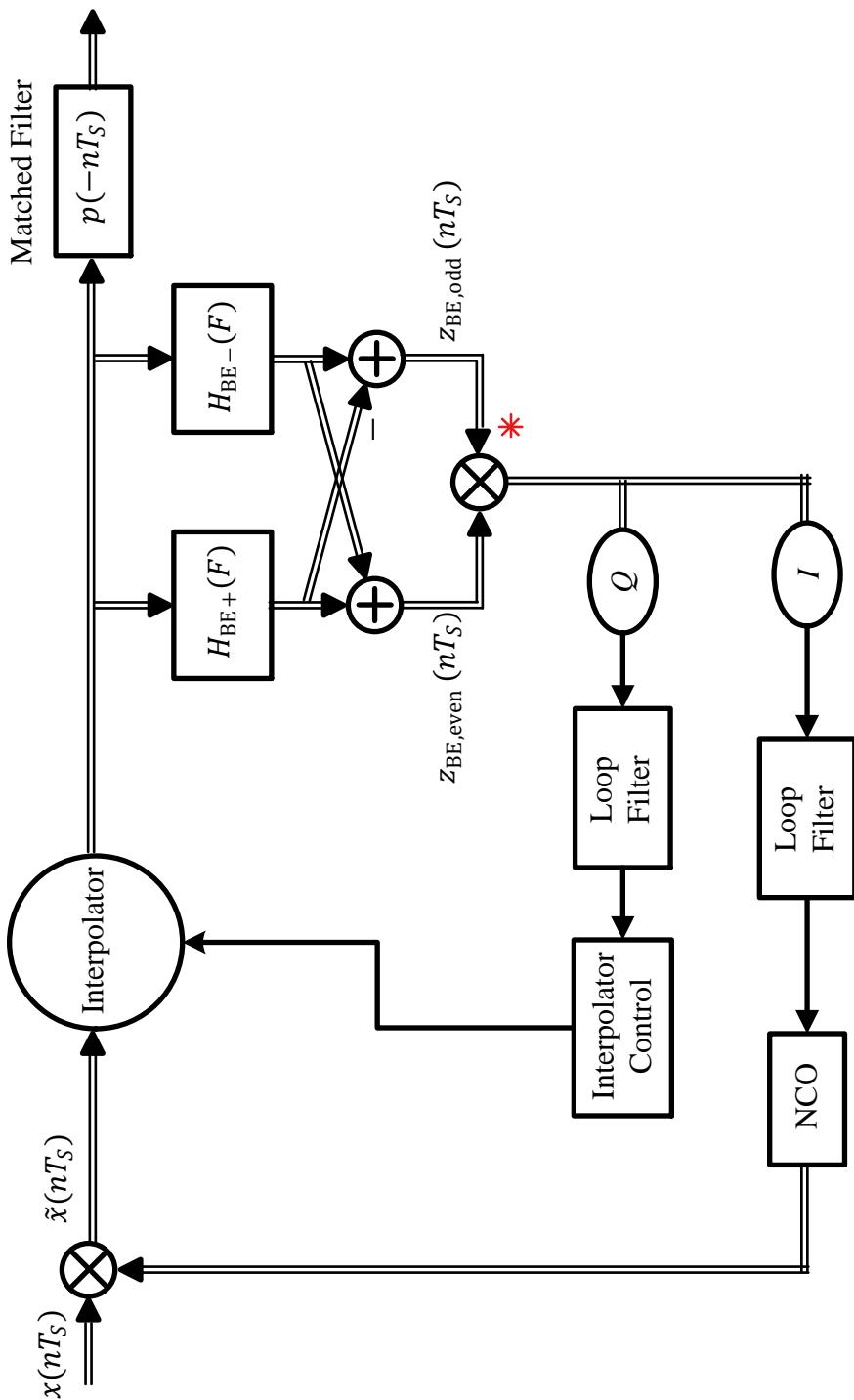


Figure 7.74: A block diagram for the implementation of band edge filters for carrier and timing recovery

wireless channels in Chapter 8, we will find that even a moderate degree of signal dispersion due to multipath destroys the symmetry properties of the waveforms exploited by many TEDs described before. In those situations, tracking the sampling clock frequency ξ becomes more important than the timing phase (which in many situations is not even recognizable due to the pulse distortion) while recovering the correct timing phase offset ε_Δ is left to the equalizer. The band edge timing recovery performs better in such a scenario with relatively faster convergence time [28]. This completes the picture of the band edge filters.

A Summary

What if someone just wants to understand a summary behind the mechanism of band edge filters? A very quick route to this understanding is through considering an alternating input sequence $(-1, +1, -1, +1, \dots)$ in both I and Q rails, which is the ultimate motivation of any timing synchronization scheme in any case. Then, the input completes a period in $2T_M$ and hence it is a complex sinusoid of frequency $1/2T_M$. Scaled versions of this complex sinusoid come out of both the even and odd band edge filters at $1/2T_M$. After conjugating the odd output, their product in time domain is their convolution with each other in frequency domain that generates a complex sinusoid at $1/T_M$, *the quadrature component of which is our timing sine wave* with a phase equal to the timing offset ε_Δ . Moreover, a carrier frequency offset F_Δ prevents the outputs from the two band edge filters to cancel out at DC during the convolution, thus producing a spectral line at DC proportional to F_Δ .

In hindsight, these band edge filters act similar to a squaring synchronizer but with an additional role of prefiltering the signal at the spectral edges to eliminate most of the self noise entering the synchronization system if a matched filter (manifesting full signal bandwidth) was employed. An analogous intuitive reasoning from the perspective of upper and lower band edge filters exists for timing synchronizer while that for the carrier frequency synchronizer was given in Figure 6.37.

7.13 Polyphase Clock Synchronization

While the title ‘polyphase clock synchronization’ casts an intimidating image, the fundamental concept is very simple as we shortly see. A more interested reader can refer to Ref. [29] for an in-depth coverage of this topic.

As stated before, the job of the Rx is to sample the matched filtered waveform at optimal intervals, i.e., exactly at the middle of the eye diagram, to send one sample per symbol taken at ISI-free locations T_M seconds apart to a symbol detector. These ideal timing instants should be located at $nT_S = mT_M + \varepsilon_\Delta$ (ε_Δ is the original timing phase offset) but naturally they are missing in the actual waveform.

Interpolation is the process of reproducing a ‘missing’ sample at these desired locations by computing the signal value at $mT_M + \hat{\varepsilon}_\Delta$, where $\hat{\varepsilon}_\Delta$ is the estimate of timing offset provided by a timing error detector. I highly recommend reading Section 2.7.2 before proceeding further where we learned that the sample rate of a signal can be increased by a factor of P through inserting $P - 1$ zero-valued samples between the original input samples. In the new time axis, the original samples lie at indices 0, P , $2P$, \dots which can be interpolated to obtain a smooth curve with an increased sample rate. That interpolation was accomplished through lowpass filtering the $P - 1$ spectral replicas arising in frequency domain.

What we do here is a little different. For this purpose, we explore it in both time and frequency domains.

Note 7.11 Main idea

Polyphase clock synchronization is *exactly* the same as timing or clock synchronization methodologies discussed before. In particular, it implements a scheme similar to the brute force estimator of Section 7.3.1 in a parallel setup and a feedback manner. There are four blocks in a timing locked loop.

- Timing error detector (TED)
- Loop filter
- Interpolator control
- Interpolator

In polyphase clock synchronization, the timing error detector, the loop filter, and the interpolator control block remain the same. Only the interpolator is combined with the matched filter to perform both Rx filtering and interpolation in one operation. Let us find out how.

Time Domain View

For a usual scenario, a TED operates at L samples/symbol and the interpolator delivers one matched filter output computed as $z(mT_M + \hat{\varepsilon}_\Delta)$ to the symbol detector after throwing the remaining $L - 1$ samples. Here, the idea is to entirely eliminate the interpolator through upsampling the matched filter output $z(nT_S)$ by a large factor P via inserting $P - 1$ zeros between every two samples and then filtering it with the aid of an ideal lowpass filter. A block diagram of such a procedure is illustrated in Figure 7.75a where the sample rates of different signals are as below ($T_S = T_M/L$ or $F_S = L/T_M$ where T_S and F_S are sample time and rate, respectively, while T_M is the symbol time).

- Matched filter input $r(nT_S) \rightarrow F_S = 1/T_S$
- Matched filter impulse response $h_{MF}(nT_S) \rightarrow 1/T_S$
- Lowpass filter output $z(nT_S/P) \rightarrow P/T_S$

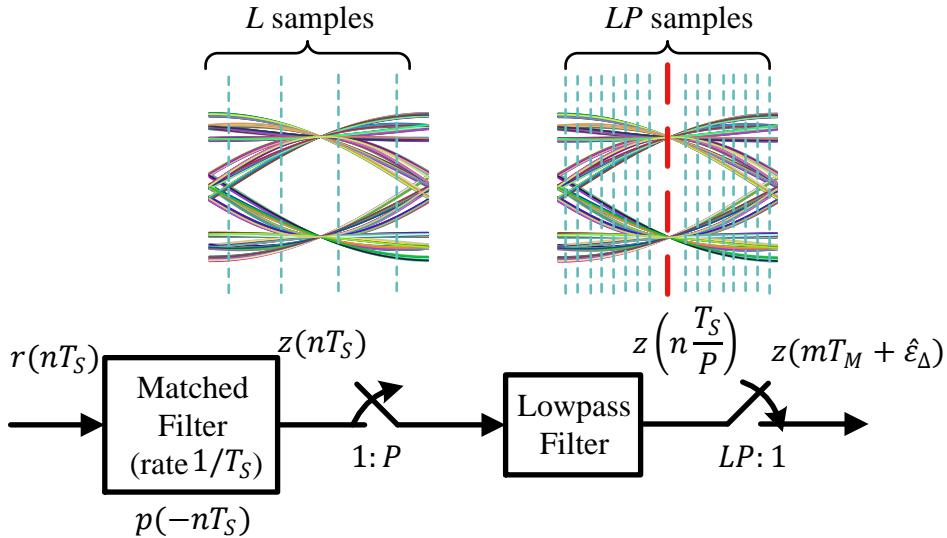
In summary, the output of this process is a heavily upsampled matched filter output

$$z\left(n \frac{T_S}{P}\right) = z\left(n \frac{T_M}{LP}\right)$$

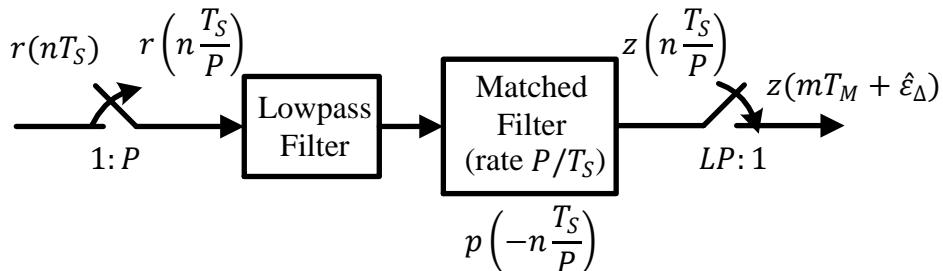
at a rate of LP to be precise. Instead of L samples/symbol, the new waveform now contains LP samples/symbol. Naturally, one of these samples is quite close to the ideal instant $mT_M + \hat{\varepsilon}_\Delta$. Such a sample can be selected via downsampling the filter output by LP with the corresponding offset. We can say that the resolution of the output is controlled by the upsampling factor P .

Notice that the system suffers from two losses in this approach:

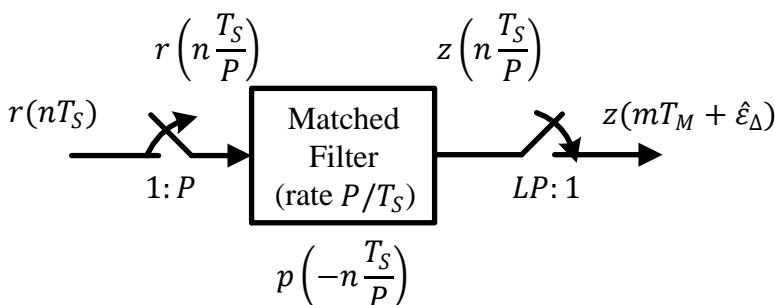
1. additional computational complexity due to an extra lowpass filter, and
2. distortion effects, even if little, that arise in the signal from non-ideal filtering.



(a) Matched filter output $z(nT_S)$ is upsampled and lowpass filtered



(b) Upsampling and filtering can be applied to matched filter input $r(nT_S)$



(c) An upsampled matched filter is employed for both shaping and interpolation

Figure 7.75: Upsampling the matched filter eliminates the need of an extra lowpass filter

Alternatively, upsample and filter operations can be applied to the matched filter input $r(nT_S)$ instead of the output $z(nT_S)$ because it is also a lowpass filter. Such a

modification causes no change in the final output $z(mT_M + \hat{\epsilon}_\Delta)$ if the matched filter is designed such that now it operates at a higher rate of P/T_S instead of $1/T_S$ as shown in Figure 7.75b. In this case, sample rates of different signals are the following.

- Matched filter input $r(nT_S/P) \rightarrow P/T_S$
- Matched filter impulse response $h_{MF}(nT_S/P) \rightarrow P/T_S$
- Lowpass filter output $z(nT_S/P) \rightarrow P/T_S$

The impulse response of the matched filter resides in the memory as a set of rate P/T_S sampled coefficients of the flipped pulse shape $p(t)$.

$$h_{MF}\left(n \frac{T_S}{P}\right) = p\left(-n \frac{T_S}{P}\right)$$

Here, all three signals, namely the matched filter input, its impulse response and its output, all operate at rate P/T_S .

Following this alternative route in Figure 7.75b, we realize that the interpolating filter and the matched filter are both lowpass filters. Filtering the received signal twice is thus not required and the lowpass filter can be removed. Such an arrangement is drawn in Figure 7.75c.

Previously, the job of the matched filter was just to shape the incoming spectrum to maximize the SNR. Here, matched filter is performing an additional task of removing the spectral replicas that arise due to $1 : P$ upsampling. In time domain, such an exercise raises from dead $P - 1$ zeros inserted between every two samples of $r(nT_S)$.

Intuition Towards a Polyphase Architecture

Continuing from Figure 7.75c, let us imagine what happens when a convolution between upsampled Rx signal $r(nT_S/P)$ and a rate P/T_S matched filter impulse response $h_{MF}(nT_S/P)$ takes place. For the example that follows, I use $L = 2$ original samples/symbol and a small upsampling factor $P = 4$ for illustration purpose (in reality, a higher value of P will be required for better interpolation). Therefore, the system processes $LP = 8$ samples during each symbol time T_M .

The original rate $F_S = 1/T_S$ matched filter has a support from $-LG$ to $+LG$ samples where G is the group delay of the filter (number of symbols from the center to either end). When redesigned at a rate of P/T_S , its support now ranges from $-PLG$ to $+PLG$ samples and its convolution with $r(nT_S/P)$ occurs at the higher rate P/T_S . This high rate convolution can be written as

$$z\left(n \frac{T_S}{P}\right) = \sum_{i=-PLG}^{+PLG} r\left(i \frac{T_S}{P}\right) h_{MF}\left(n \frac{T_S}{P} - i \frac{T_S}{P}\right) \quad (7.70)$$

which is just a regular convolution but at a rate $P/T_S = PF_S$. The interesting point to notice is that *$P - 1$ out of P samples in upsampled Rx signal $r(nT_S/P)$ are zero*, therefore only $1/P$ of the multiplications at each convolution step are required.

Note 7.12 Discarding the multiplications with zero

During each convolution step, the zeros of the upsampled Rx signal $r(nT_S/P)$ do not contribute anything at the filter output. Since their contribution is zero, the filtering operation can be performed at a reduced complexity if we can track the locations of nonzero (i.e., original) samples of the input $r(nT_S)$ and perform the weighted sum only from the register locations containing those samples.

Convolution at time nT_S : To elaborate on this convolution, consider this process at any time nT_S and for simplicity assume $n = 0$. The top plot in Figure 7.76 shows a flipped matched filter $h_{MF}(-iT_S/P)$ designed at a rate of P/T_S for $P = 4$ and $L = 2$. It is a Square-Root Raised Cosine (SRRC) pulse with excess bandwidth $\alpha = 0.5$ and the flipped matched filter is the same as the matched filter due to pulse shape symmetry. The dashed red lines mark the boundary for each symbol time T_M while green dotted lines mark the second out of $L = 2$ samples for each symbol. So in total, $LP = 8$ samples correspond to one symbol time.

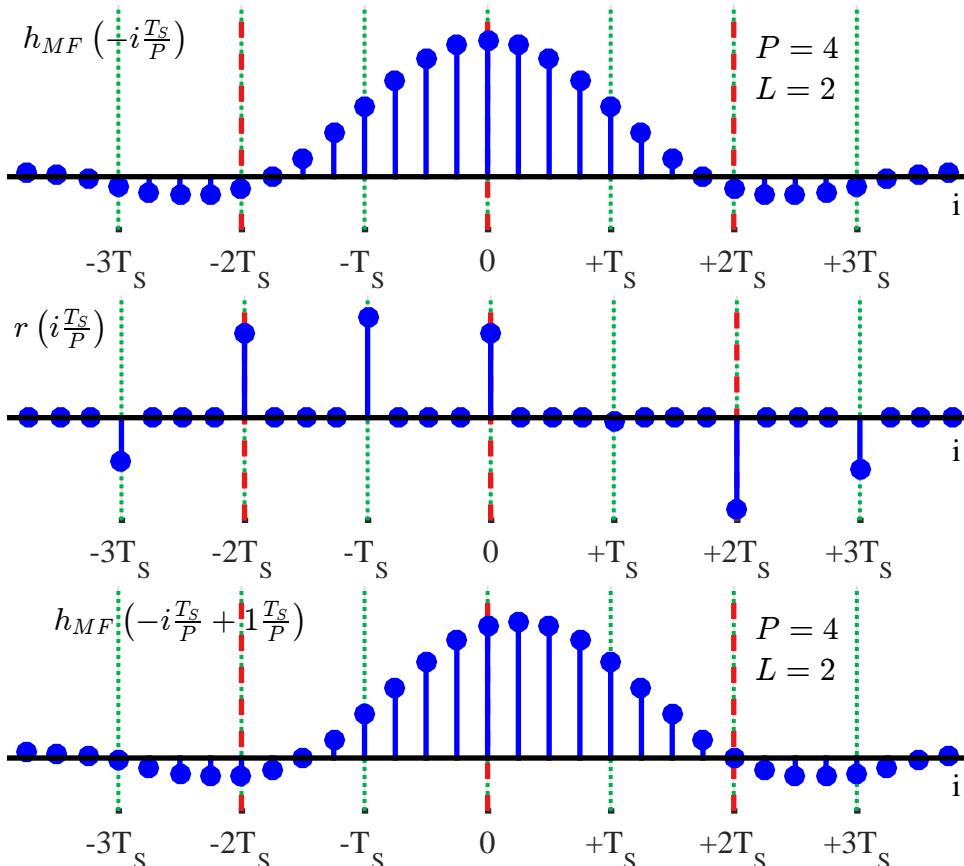


Figure 7.76: (Top) Matched filter designed at rate P/T_S with $L = 2$ and $P = 4$. (Middle) Upsampled Rx signal with $P - 1$ zeros between every L samples. (Bottom) Matched filter shifted by 1 during rate P/T_S convolution

The middle plot illustrates a binary PAM sequence shaped by an SRRC pulse with excess bandwidth $\alpha = 0.5$ at $L = 2$ samples/symbol. Since only every P^{th} value of $r(nT_S/P)$ is nonzero, samples marked by the red dashed and green dotted lines (shown in the middle plot) are multiplied with the corresponding values of $h(-iT_S/P)$ (shown in the top plot) before summation. So the convolution at this particular time nT_S involves only these T_S -spaced samples and thus given by

$$\sum_{i=-LG}^{+LG} r(iT_S) h_{MF}(nT_S - iT_S) = z(nT_S)$$

Convolution at time $nT_S + 1 \cdot T_S/P$: Since the flipped matched filter impulse response is the input sliding during convolution, $h_{MF}(-iT_S/P)$ needs to be shifted by one sample at the high rate P/T_S to yield $h_{MF}(-iT_S/P + 1 \cdot T_S/P)$. This is drawn in the bottom plot of Figure 7.76 which leads to computation of the next convolution output at $nT_S + 1 \cdot T_S/P$. For this output, the red dashed and green dotted lines in the middle and bottom plots get multiplied with each other before summation.

$$\sum_{i=-LG}^{+LG} r(iT_S) h_{MF}\left(nT_S - iT_S + \frac{1 \cdot T_S}{P}\right) = z\left(nT_S + \frac{1 \cdot T_S}{P}\right)$$

A plus sign in a regular signal $s(nT_S + 1 \cdot T_S/P)$ indicates a left shift by $1 \cdot T_S/P$ but in the above expression, we have the summation index i as the variable instead of nT_S , so $h_{MF}(nT_S - iT_S + 1 \cdot T_S/P)$ is a time reversed signal. Hence, the signal shifts by $1 \cdot T_S/P$ to the right.

Continuing the above convolution procedure in a similar manner, a general expression can be written as

$$\sum_{i=-LG}^{+LG} r(iT_S) h_{MF}\left(nT_S + \frac{\delta \cdot T_S}{P} - iT_S\right) = z\left(nT_S + \frac{\delta \cdot T_S}{P}\right)$$

$\delta = 0, 1, \dots, P-1$

(7.71)

We can conclude that at each instant T_S/P , *a convolution of T_S -spaced Rx signal $r(nT_S)$ is performed with a $PT_S/P = T_S$ -spaced version of the matched filter* (and not with high rate T_S/P -spaced version). From Eq (7.71) above, the total number of such filters is P and each of them starts with a different time offset indexed by δ .

$$h_{MF,\delta}(nT_S) = h_{MF}\left(nT_S + \frac{\delta \cdot T_S}{P}\right)$$

$\delta = 0, 1, \dots, P-1$

(7.72)

To avoid confusion, there is a slight abuse of notation here since a different index should be used for the T_S -spaced filter. Next, we see how such a reasoning can lead towards a significantly more efficient architecture for interpolating the signal in a timing locked loop.

Polyphase Partition

Having established the intuition behind discarding the zero valued products within a convolution sum, we want to formulate these savings in an interpolator architecture. For this purpose, the high rate matched filter can be partitioned in the following manner.

First, consider the rate P/T_S samples of the matched filter below, which we call a *mother filter*. While the filter indices are from $-PLG$ to $+PLG$, it helps in what follows next if we keep them from 0 to $2PLG$.

$$h_{MF} \left(0 \frac{T_S}{P} \right), h_{MF} \left(1 \frac{T_S}{P} \right), \dots, h_{MF} \left(P \frac{T_S}{P} \right), h_{MF} \left((P+1) \frac{T_S}{P} \right), \dots, \\ h_{MF} \left(2P \frac{T_S}{P} \right), h_{MF} \left((2P+1) \frac{T_S}{P} \right), \dots$$

Instead of a 1-dimensional array, this can also be written as a 2-dimensional matrix such that the first P elements are written into the 1st column, the next P elements are written into the 2nd column, and so on. This mapping is illustrated in Table 7.3 for a total filter length of $3P$. Then, looking into each horizontal row, we have one child filter in which the spacing between two samples is equal to that between P samples of the mother filter. Let us call them *polyphase partitions* on which the expression *polyphase clock synchronization* is based. We will soon learn the logic behind the term ‘polyphase’.

Table 7.3: Polyphase partitions of the mother filter $h_{MF}(nT_S/P)$ forming a polyphase filterbank

Polyphase Partition ↓	Filter Samples		
	→ → → →		
$h_{MF,0}(nT_S) \rightarrow$	$h_{MF} \left[0 \frac{T_S}{P} \right]$	$h_{MF} \left[P \frac{T_S}{P} \right]$	$h_{MF} \left[2P \frac{T_S}{P} \right]$
$h_{MF,1}(nT_S) \rightarrow$	$h_{MF} \left[1 \frac{T_S}{P} \right]$	$h_{MF} \left[(P+1) \frac{T_S}{P} \right]$	$h_{MF} \left[(2P+1) \frac{T_S}{P} \right]$
\vdots
$h_{MF,\delta}(nT_S) \rightarrow$	$h_{MF} \left[\delta \frac{T_S}{P} \right]$	$h_{MF} \left[(P+\delta) \frac{T_S}{P} \right]$	$h_{MF} \left[(2P+\delta) \frac{T_S}{P} \right]$
\vdots
$h_{MF,P-1}(nT_S) \rightarrow$	$h_{MF} \left[(P-1) \frac{T_S}{P} \right]$	$h_{MF} \left[(2P-1) \frac{T_S}{P} \right]$	$h_{MF} \left[(3P-1) \frac{T_S}{P} \right]$

From Table 7.3, notice that the sample rate of the mother filter is P/T_S and such a set forms a bank of P parallel filters, known as a *polyphase filterbank*. Each partition is a $P : 1$ downsampled version of the mother filter and operates at a lower rate of $1/T_S$. For this reason, the length of each partition is $1/P$ of the mother filter (e.g., a length of 3 each for a total length of $3P$ in the table). If the length of the mother filter

is not an integer multiple of P , then zeros need to be added to make it so. Or when a Nyquist filter of length $2PLG + 1$ is modified, there are two options available [30]:

- (a) the last sample can be discarded, or
- (b) the filter can be designed for twice the number of stages and then even indexed samples can be discarded.

Looking at the first column, the starting index – and hence the timing offset – of each of these child filters is unique taking values from the set $0, 1, \dots, P - 1$. The central portion of their impulse responses are drawn in Figure 7.77 where a timing offset can be observed in each of these square-root Nyquist filters. If the input was a simple square-root Nyquist filter, i.e., just like $h_{MF,0}(nT_S)$, the output of each polyphase arm is a Nyquist filter due to the convolution between two square-root Nyquist filters, each with a different timing offset. I am not drawing the result of such convolutions due to their close similarity in shape to those in Figure 7.77.

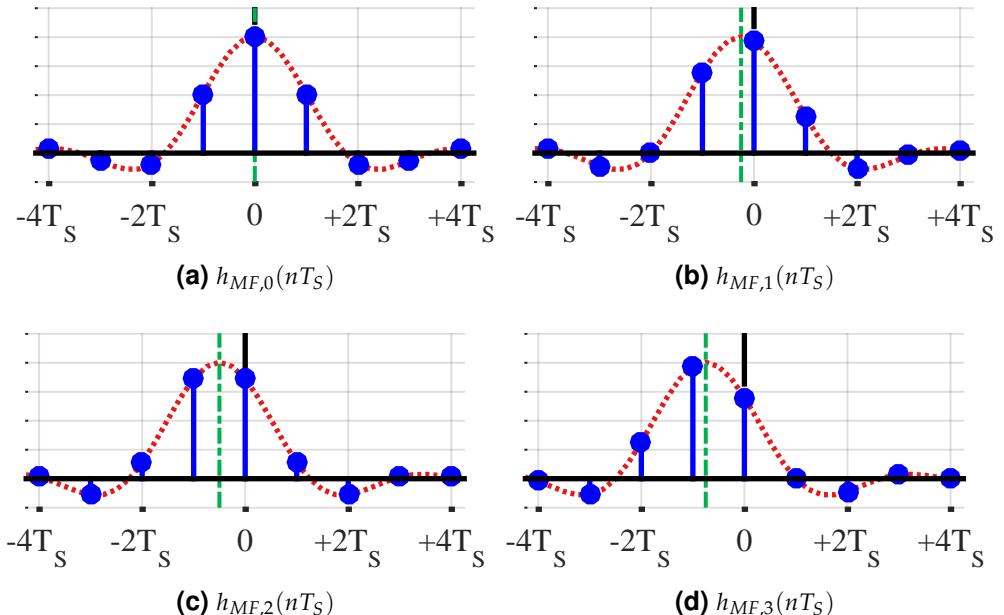


Figure 7.77: Impulse responses of the polyphase partitions

Clearly, the zeroth partition $h_{MF,0}(nT_S)$ produces a sample that is exactly at the peak of the Nyquist filter. For each successive filter, there is a shift in the peak towards the left equal to $\delta T_S / P$ *with respect to the time origin of that partition*, as shown by dashed green lines in Figure 7.77. The concept of a polyphase filterbank is mainly about these time shifts[†] and their impact on the incoming signal. This is why these polyphase filters are said to provide linear phase shifts producing the time delays that perform a *time-delay beamforming* task. We study more about this in Section 10.2.5 where I think it is more appropriate to introduce the origins of this concept in the context of downsampling.

[†]At the time of this writing, GNU Radio tutorial describes a bank of derivative filters with different delays which are actually the derivatives of the polyphase children shown in this figure.

Coming back to the polyphase partitions, the output for their time 0 would have been an interpolated sample between the peak (the sample corresponding to the symbol) and the second sample (since $L = 2$) and hence gradually decreasing. This sample needs to be picked up at the right time which is accomplished through a commutator as we see next.

Understanding the Operation of Polyphase Filterbank

The operation of such a polyphase filterbank drawn in Table 7.3 is usually described starting from Eq (7.71) for a mathematical representation and then taking the z-Transform route[†]. In this text, we will take the simpler route of a visual understanding.

Linking this equation to Figure 7.76, recall that the T_S -spaced samples represented by vertical lines in the top plot of Figure 7.76 are nothing but the polyphase partition $h_{MF,0}(nT_S)$ in Figure 7.77. Since all the remaining partitions produce an output zero, there is no use in including those outputs in the convolution sum. At the next high rate sampling instant, i.e., $1T_S/P$, the convolution only happens between the input and polyphase partition 1, i.e., the bottom plot of Figure 7.76 which is the polyphase partition $h_{MF,1}(nT_S)$ in Figure 7.77. A similar argument holds for the next output and other polyphase children.

With this insight in hand, we can utilize the polyphase partitions already performed for the mother filter to construct a parallel architecture such that the convolution with each such partition happens at the low sample rate $1/T_S$. Follow the sequence of operations in Figure 7.78 for $P = 4$ which illustrates the high rate P/T_S operations from the perspective of polyphase arms. This is somewhat similar to my son's favourite CONNECT4 game.

- With the notations for an empty register, a zero valued sample and nonzero valued samples clear, the first sample of a zero-packed time series $r(nT_S/4)$ arrives from the doorstep of partition 0 at time 0 (this is from the convolution Eq (7.71) and the further steps can be verified by plugging in value of $n = 0, 1, \dots$). From here onwards, we assume that any overlap implies a multiplication operation between the filter coefficient and the input sample present at the same location. For example, the first input sample gets multiplied with $h_{MF}[0] = h_{MF,0}(0T_S)$ in the top left box at instant $0T_S$.

[†]The z-Transform of the filter impulse response naturally leads to a polyphase partition of the mother filter $h_{MF}(nT_S) = h_{MF}[n]$ as follows.

$$H_{MF}(z) = \sum_n h_{MF}[n]z^{-n} = h_{MF}[0] + h_{MF}[1]z^{-1} + h_{MF}[2]z^{-2} + \dots + h_{MF}[P]z^{-P} + \dots$$

This can be rearranged in a 2D array through loading the filter taps by columns.

$$\begin{aligned} H_{MF}(z) &= h_{MF}[0] &+ h_{MF}[P+0]z^{-(P+0)} &+ h_{MF}[2P+0]z^{-(2P+0)} &+ \dots \\ h_{MF}[1]z^{-1} & &+ h_{MF}[P+1]z^{-(P+1)} &+ h_{MF}[2P+1]z^{-(2P+1)} &+ \dots \\ \vdots & &\vdots & &\vdots \\ h_{MF}[P-1]z^{-(P-1)} & &+ h_{MF}[2P-1]z^{-(2P-1)} &+ h_{MF}[3P-1]z^{-(3P-1)} &+ \dots \end{aligned}$$

Now summing row-wise and taking z^{-i} common from each row, we can write it as

$$H_{MF}(z) = \sum_{i=0}^{P-1} \sum_n h_{MF}(nP+i)z^{-(nP+i)} = \sum_{i=0}^{P-1} z^{-i} \sum_n h_{MF}(nP+i)z^{-nP} \quad (7.73)$$

where the inner summation is each a polyphase partition of the mother filter.

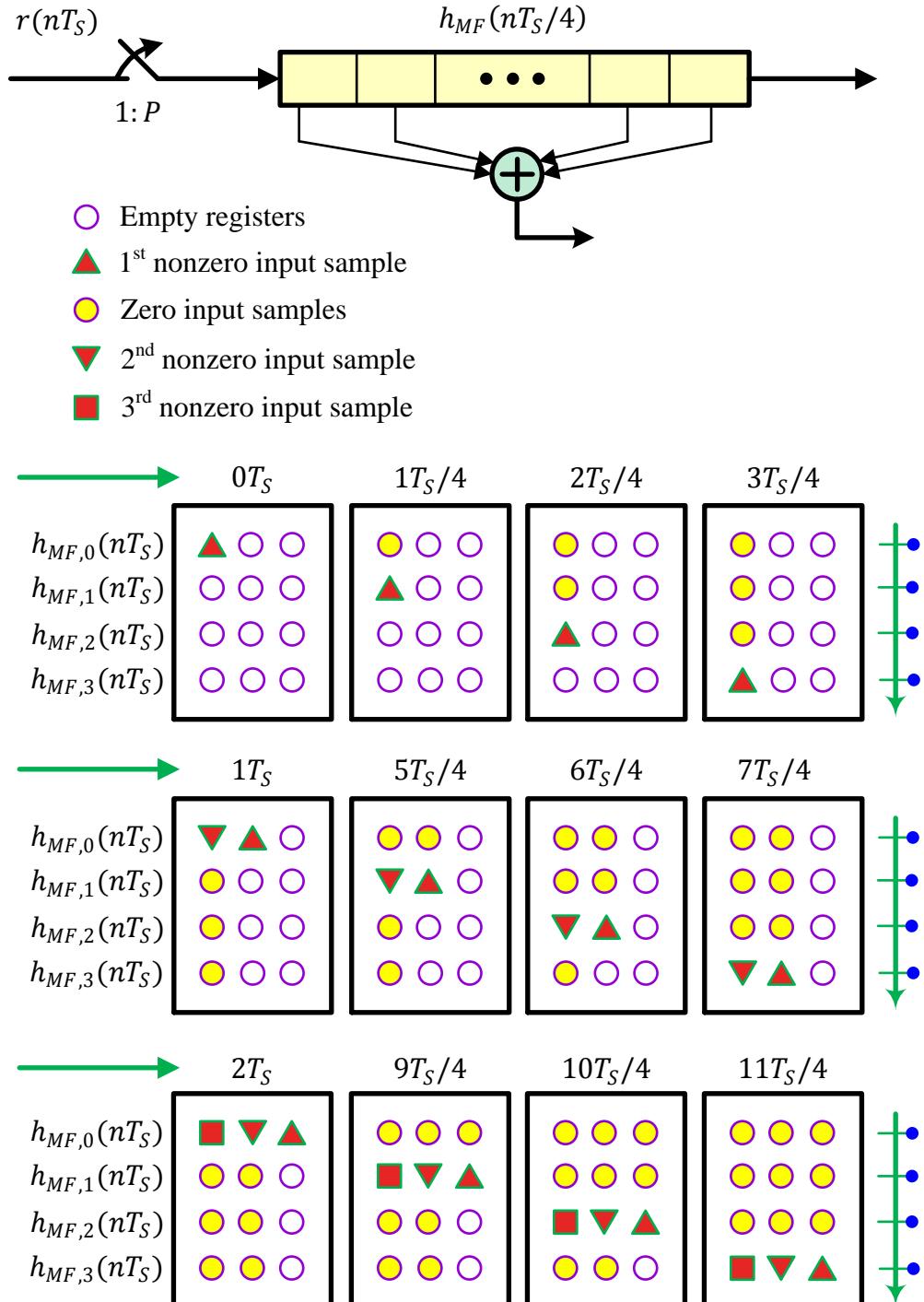


Figure 7.78: During interpolation, a polyphase architecture loads the input samples by columns at a rate $1/T_S$ and reads the output samples by rows at a rate P/T_S where $P = 4$. The convolution at each partition happens at the low rate $1/T_S$

At the next instant $1T_S/4$, this sample moves down to partition 1 and a zero valued sample moves in. This process repeats until time $3T_S/4$.

- At instant $1T_S$, the second nonzero sample arrives at partition 0 while the first nonzero sample has also moved to partition 0. In the form of products between inputs and overlapping filter coefficients, the convolution output is being taken at a high rate P/T_S from the right edges of the partitions.
- Following this process, *the last row corresponding to time $2T_S$ onwards is analogous to Figure 7.76* where *all the samples are first residing in partition 0*, then in partition 1 and so on. At each such instant, the rest of the partitions contain zero samples only. Therefore, instead of forming the final summation from all the polyphase partitions as required by convolution, we can simply point to the polyphase arm supplying the output sample at each instant and shown by green arrows on the right. It is important to understand that the summation is still there but with zero valued samples from other partitions. This point is supremely important when we cover the polyphase filterbanks in the context of downsampling in Section 10.2.5.

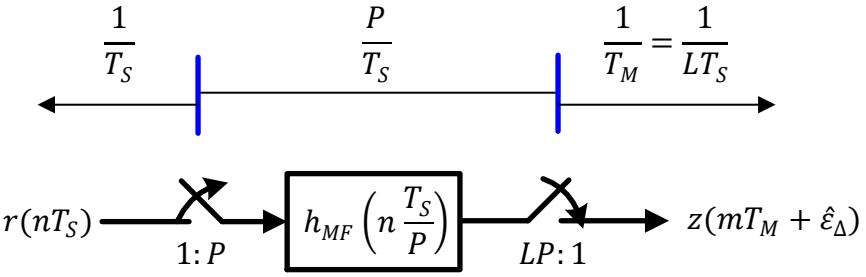
Ideally, I should have drawn this figure as a 12×1 column instead of a 3×4 matrix but then this book would have been 2 pages longer. The last row (i.e., the last 4 time instants from $2T_S$ to $11T_S/4$) clearly demonstrates that this $1/T_S$ rate convolution is equivalent to the original data samples $r(nT_S)$ – and not zero-packed $r(nT_S/4)$ – forming the input simultaneously to all the filters in the filterbank and the output at rate $4/T_S$ simply taken from the arm that sequentially supplies the nonzero output sample. Such successive selection is said to be performed through a commutator shown as a *downwards* moving pointer in an efficient implementation at the right of Figure 7.79b. This architecture can be further optimized through eliminating the unnecessary memory elements occupied by the zero valued samples. In Ref. [1], fred harris draws the minimum resource versions of a polyphase filterbank for upsampling and downsampling operations.

In Figure 7.79a, our original upsampler + matched filter architecture of Figure 7.75c is redrawn where the Rx signal $r(nT_S)$ is convolved with the mother filter $h(nT_S/P)$ at a rate P/T_S in a conventional serial manner. This is to highlight the fact that it is an inefficient architecture due to incorporating $P - 1$ extra multiplications with zero valued samples.

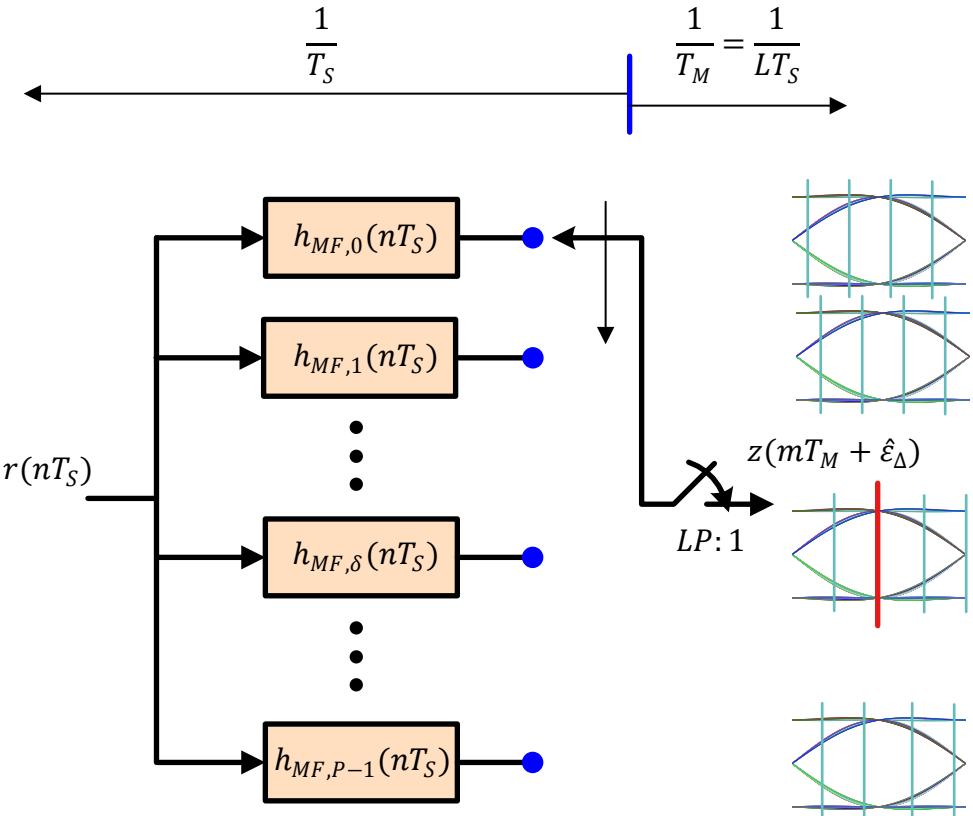
As far as the timing or clock synchronization is concerned, the most interesting part happens in the eye diagrams of Figure 7.79b. Instead of a simple square-root Nyquist filter, if the input was a PAM data stream shaped with such a square-root Nyquist filter, the output of each polyphase arm is a sample of the same data stream shaped with a Nyquist filter, each with a different timing offset. One such partition, say δ , produces a sample that is right at the maximum eye opening. This maximum can be tracked through any TED within a timing locked loop where the polyphase part replaces the combination of matched filter and interpolator. Rest of the details of the loop operation remain the same as we see next.

Where is the Timing Error Detector (TED) in a Polyphase Synchronizer?

A question that arises at this stage is the following. Where is the Timing Error Detector (TED) in a polyphase clock synchronizer?



(a) Inefficient interpolation at a rate P/T_S through the mother filter $h_{MF}\left(n \frac{T_S}{P}\right)$



(b) Efficient interpolation at a rate $1/T_S$ through the polyphase children

Figure 7.79: Two architectures to interpolate the matched filter output

From Eq (7.71), the ratio of the polyphase filter index δ to the total number of partitions P plays the same role as the fractional interval μ_m in the interpolation filter described in Section 7.9.

$$\mu_m = \frac{\delta}{P}$$

We can say that the polyphase architecture implements the interpolation with a quantized fractional interval. The number of partitions in the mother filter – which is equal to the original upsampling factor P – controls the degree of this quantization.

Out of P such options, a Timing Locked Loop (TLL) selects one output based on the maximum eye opening. Since a polyphase partition is nothing but only an interpolation technique, it cannot direct the loop towards such an optimal point. A Timing Error Detector (TED) must be a part of such a TLL that drives the indexing of the polyphase arms in a manner that the index divided by P is a quantized version of the fractional interval μ_m .

The nature of this TED does not matter. In their original paper Ref. [29], fred harris and Michael Rice demonstrated the derivative TED (the maximum likelihood TED) that we have explained in Section 7.6.1. This caused a confusion that a polyphase clock synchronizer inherently incorporates such a TED. This is not true and any TED – such a zero crossing or Gardner – can be employed for this task as stated in their paper as well.

One can start the loop with any filter partition while the middle filter $P/2$ is a natural choice. For example, in the Gardner TED with $L = 2$ samples/symbol, the loop can start with the middle filter output as the zero crossing sample while the outputs of its previous and next filters can form the difference expression required by the TED.

Exercise 7.2

In GNU Radio, the block ‘Polyphase Clock Sync’ implements the concepts discussed earlier in which a derivative (ML) TED is embedded. This TED operates by multiplying the outputs of two filters, an upsampled matched filter $h_{MF}(nT_S)$ and an upsampled timing matched filter $h_{TMF}(nT_S)$, also known as a derivative matched filter, see Section 7.6.1 (keep in mind that a series of derivative filters shown in GNU Radio tutorials are the derivatives of downsampled matched filters drawn in Figure 7.77).

Samples/Symbol: While the input samples/symbol L depends on the TED in general, the derivative (ML) TED in this block works better with a larger number of samples/symbol.

Loop Bandwidth: The loop bandwidth is the equivalent noise bandwidth of a PLL described in Section 4.6 and should be adjusted accordingly (the damping factor ζ in this block is chosen as 1, i.e., it is a critically damped system).

Taps: These are the matched filter taps that are generated from a Square-Root Raised Cosine pulse for which the routine `firdes.root_raised_cosine()` can be employed.

Filter Size: Filter size refers to the number of polyphase partitions P that should be defined according to the desired granularity $1/P$.

Initial Phase: Initial phase is the timing phase set at the beginning of the run. Usually this value is set equal to $P/2$ so that the filter can converge towards any timing offset that is assumed to be uniformly distributed between 0 and 1, or 0 and $P - 1$ in terms of the filterbank index.

Maximum Rate Deviation: The timing offset here is not the timing phase offset ε_Δ but instead a sampling clock offset, i.e. the clock frequency difference between the Tx and Rx systems. This parameter sets the maximum departure from the ideal rate.

Output Samples/Symbol: In general, an equalizer is employed after the timing recovery as we will see in Chapter 8 where a symbol-spaced equalizer works at 1 sample/symbol while a fractionally-spaced equalizer works at greater than 1 (usually

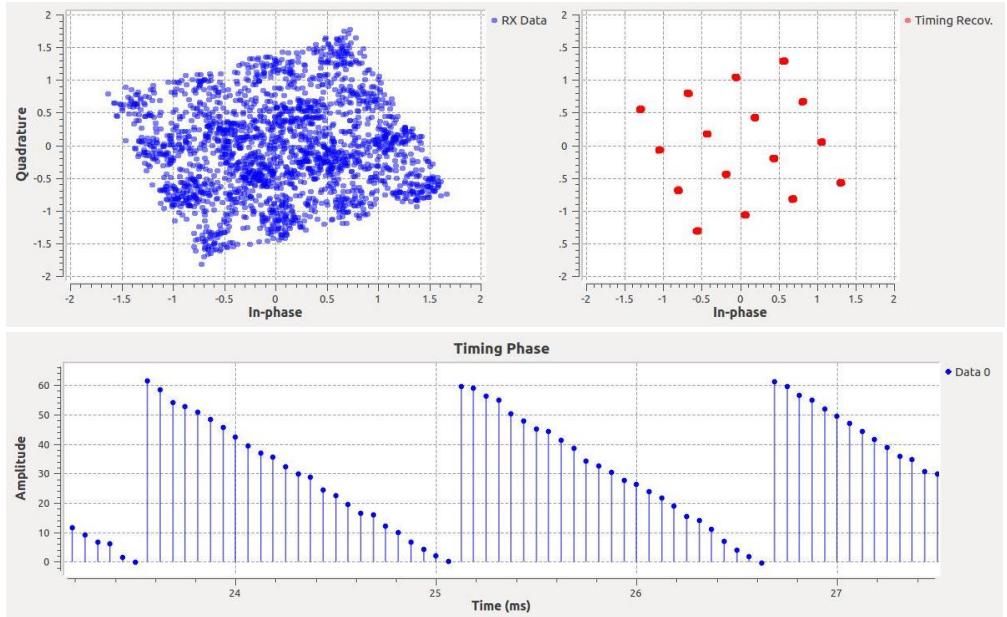


Figure 7.80: Output from the Polyphase Clock Sync block after timing convergence

2) samples/symbol. Therefore, the output number of samples/symbol depends on the type of the equalizer used for the system.

For a 16-QAM scheme, an example GNU Radio plot is shown in Figure 7.80 for a sampling clock offset of 1% (timing offset 1.01), a loop bandwidth of $2\pi/150$, input samples/symbol $L = 4$, an SRRC filter with excess bandwidth $\alpha = 0.5$ and $P = 64$ polyphase partitions. The rotation of the constellation in Figure 7.80 just signifies the fact that timing can be locked on any random phase which needs to be recovered at a later stage through a PLL.

Notice from the figure that there are $P = 64$ filter indices that act as the fractional interval and provide the interpolated sequence value at any instant. Furthermore, to measure the rollover period on the time axis, consider from Eq (7.58) that a single sample is added or skipped from the signal waveform when ξ/T_S reaches 1. As discussed during the interpolator control, a full rollover from 0 to 1 or vice versa of the fractional interval μ_m (shown as timing phase) occurs after a complete symbol period T_M , i.e., L samples. Then, we get the period as

$$\frac{\xi}{T_S} \rightarrow L, \quad \text{or} \quad \frac{T_S}{\xi \cdot L} = 1.5625 \text{ ms}$$

where a sample rate of 16 kHz, $\xi = 0.01$ and $L = 4$ are used for this purpose. This is verified from the period of timing phase in Figure 7.80.

Frequency Domain View[†]

We start the frequency domain description[†] keeping in mind Figure 7.79 that illustrates two architectures for the interpolator. A mother filter at a high rate P/T_S is employed in Figure 7.79a. To view what happens in frequency domain, we trace the following steps assuming that $L = 2$ samples/symbol and upsampling factor $P = 4$. In practice, the value of P is significantly higher.

- Drawn as dashed green curves in Figure 7.81a, the spectrum of the Rx signal $r(nT_S)$ contains replicas at integer multiples of sampling frequency $F_S = L/T_M$. In this example with $L = 2$ samples/symbol,

$$F_S = \frac{1}{T_S} = \frac{1}{T_M/2} = \frac{2}{T_M}$$

Therefore, the baseband is $L/T_M = 2/T_M$ wide. Closely notice the dotted red lines marking the baseband here and the shaded replicas.

- By periodically inserting $P - 1$ zeros between every two samples, an increase in sample rate $1 : P$ causes the baseband to take in $P - 1$ spectral copies at multiples of $2/T_M$. Specifically note the change in the dotted red lines in Figure 7.81b since our baseband is now changed to incorporate a spectral width of LP/T_M . Also, the shaded replicas are shown as solid blue spectrum due to their inclusion in baseband.
- Acting both as a lowpass filter and spectral shaping filter, a mother filter – which is a high rate P/T_S matched filter $h_{MF}(nT_S/P)$ – is now employed. Shown in Figure 7.81c, rate P/T_S implies a spectral width of $P/(T_M/L) = LP/T_M$.
- The mother filter rejects the extra $P - 1$ spectral replicas in the baseband leaving only the central spectrum. As illustrated in Figure 7.81d, this filtering restores the original spectrum alone in the now wider baseband of width LP/T_M .

Note 7.13 The role of phase

The interpolation is performed at a rate P/T_S above, although we eventually needed the samples for symbol detection at a much lower rate $1/T_M = 1/(LT_S)$. In frequency domain, the source of this inefficiency can be traced back to our second step illustrated in Figure 7.81b where $P - 1$ replicas were incorporated into the primary spectrum only to be cleaned up by the oncoming mother filter. We should not have taken them in the first place. No cleaning required in frequency domain implies computational saving.

Can we exploit the fact that frequency domain is not all about magnitude and manifests the phase as well? As we see now, interpolation can be accomplished at a low rate $1/T_S$ in a very interesting way through looking into the phase spectrum.

Moving towards the polyphase architecture of Figure 7.79b, we follow the mechanism behind this rate $1/T_S$ interpolation next. We start with the frequency domain view of the polyphase partition of the mother filter $h_{MF}(nT_S/P)$ where each row of Table 7.3 is its downsampled version with a different timing offset.

[†]The frequency domain view is described here for the sake of completion and to develop different viewpoints for looking at the same process. If you find it a bit complicated and your purpose is to just understand what polyphase clock synchronization is about, feel free to skip this section. On the other hand, if you need to see why a polyphase filterbank is actually such a powerful mental construct, keep reading.

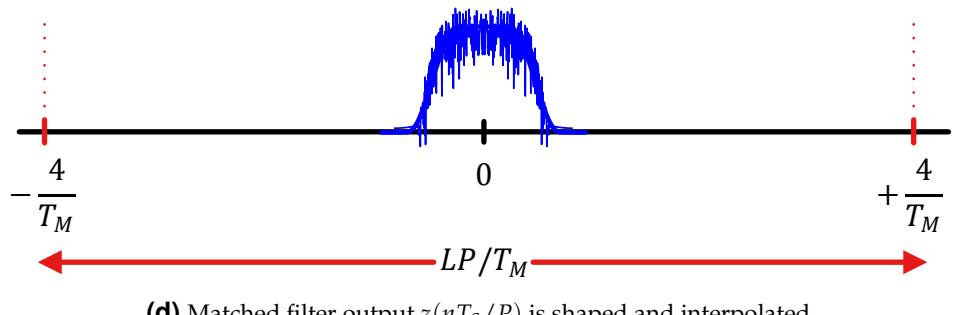
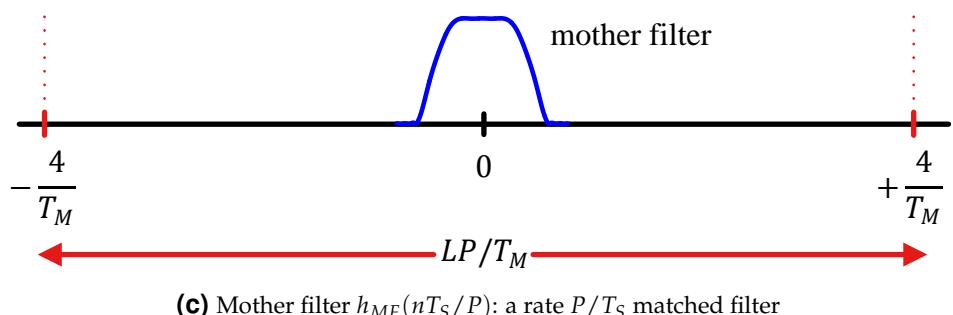
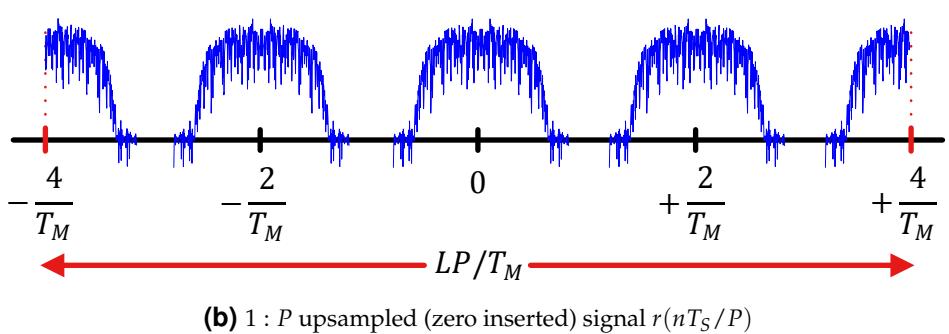
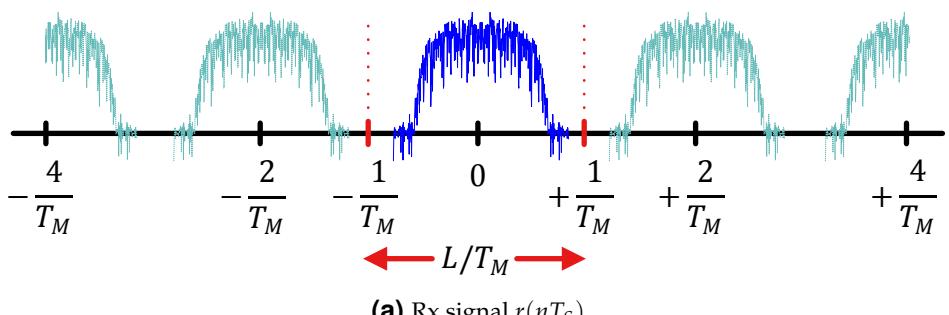


Figure 7.81: Spectra of the signals at various stages of interpolation corresponding to Figure 7.79a for $L = 2$ samples/symbol and upsampling factor $P = 4$

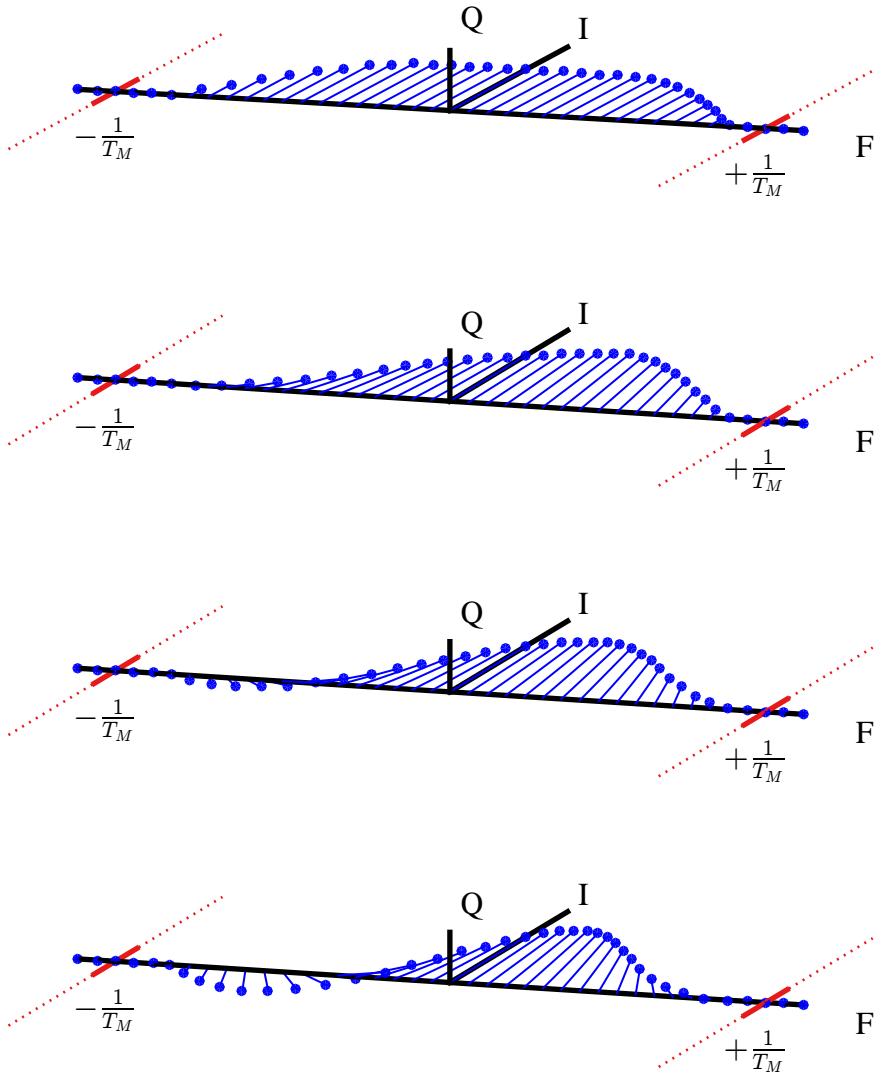


Figure 7.82: Spectra of the polyphase partitions corresponding to Figure 7.79b for excess bandwidth $\alpha = 0.5$, $L = 2$ samples/symbol and upsampling factor $P = 4$ (rotation is a little exaggerated to show the effect of multiplication with frequency domain complex sinusoids)

Child filter 0: $h_{MF,0}(nT_S)$ The partition $h_{MF,0}(nT_S)$ starts with the first sample and hence no timing offset. Consequently, it is a simple downsampled by P version of the mother filter and drawn in Figure 7.82 as $H_{MF,0}(F)$. This partition then just shapes the incoming spectrum by acting as a rate $1/T_S$ matched filter or a square-root Nyquist pulse.

Child filter 1: $h_{MF,1}(nT_S)$ The next partition $h_{MF,1}(nT_S)$ starts with the second sample of the mother filter and takes in further values at a stride of P . Therefore, it is also a downsampled by P version of the mother filter but with a time offset equal to $\delta \cdot T_S/P = 1 \cdot T_S/P$. Now recall from Section 1.9 that a shift in time of a signal induces a linear phase shift in frequency, i.e., a shift of $+\delta \cdot T_S/P$ in a time series changes the phase of its DFT as

$$\text{Time shift : } s \left(t + \delta \frac{T_S}{P} \right) \longrightarrow \text{Phase shift : } +2\pi F \cdot \delta \frac{T_S}{P}$$

which was shown in Eq (1.61) as

Shift in one domain \longrightarrow Multiplication by a complex sinusoid
in the other domain

This phase spectrum as a function of F can be seen as the multiplication of the original spectrum with a complex sinusoid of inverse period $\delta \cdot T_S/P$. Then, $H_{MF,1}(F)$ not only shapes the incoming spectrum by acting as a rate $1/T_S$ matched filter but also mixes the cumulative Nyquist spectrum with that inverse period $1 \cdot T_S/P = 1/(F_S P)$ sinusoid, i.e., the processing filter is

$$\begin{aligned} I &\rightarrow H_{MF,1:I}(F) = H_{MF,0}(F) \cos 2\pi \frac{F}{F_S} \cdot \frac{1}{P} \\ Q &\uparrow H_{MF,1:Q}(F) = H_{MF,0}(F) \sin 2\pi \frac{F}{F_S} \cdot \frac{1}{P} \end{aligned} \quad (7.74)$$

This partition is drawn in Figure 7.82 as $H_{MF,1}(F)$. Observe how this is the same downsampled by P spectrum of the mother filter but multiplied with a slowly varying sinusoid of inverse period $1 \cdot T_S/P$. The rotation is anticlockwise because the shift is to the left and positive. For a 2D plot, these I and Q plots of polyphase filters for $\delta = 0, 1, 2, 3$ are drawn in Figure 7.83 and Figure 7.84, respectively. Keep in mind that the smooth sinusoidal shape at the band edges is due to the shaping filter having a sinusoidal transition band comprising of the excess bandwidth.

We conclude that just like in time domain, a signal mixed (multiplied) with a complex sinusoid of a certain frequency causes a spectral shift in frequency domain, owing to time frequency duality, *a shift in time domain implies multiplication with a complex sinusoid in frequency domain*.

When taking an iDFT of this result, each constituent complex sinusoid of frequency domain corresponding to samples $n = -N/2, \dots, -1, 0, +1, \dots, N/2 - 1$ represents a shift to $n + 1/P$ as follows. Considering only the I part of the iDFT and using the fact that $F/F_S = k/N$,

$$I \rightarrow \frac{1}{N} \sum_{k=-N/2}^{N/2-1} H_{MF,\delta:I}[k] \cos 2\pi \frac{k}{N} n - H_{MF,\delta:Q}[k] \sin 2\pi \frac{k}{N} n$$

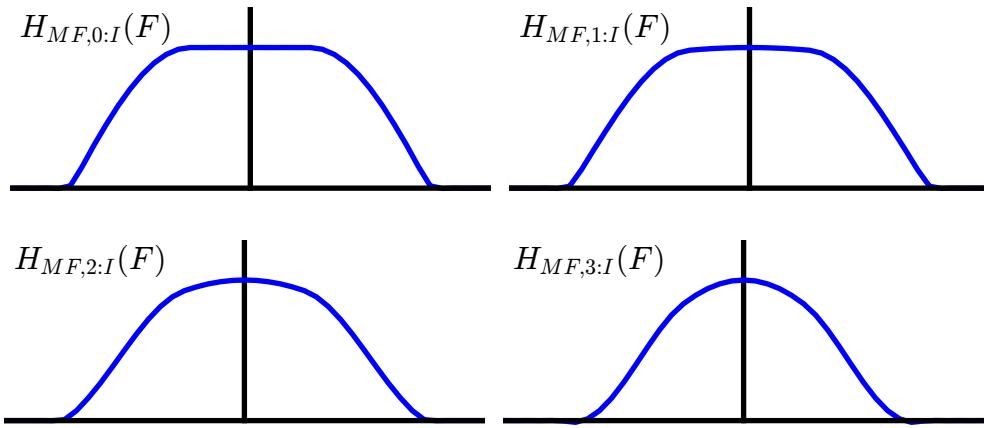


Figure 7.83: I spectra of the polyphase partitions corresponding to Figure 7.82 for excess bandwidth $\alpha = 0.5$, $L = 2$ samples/symbol and upsampling factor $P = 4$

From Eq (7.74) for a general δ , this can be written as

$$I \rightarrow \frac{1}{N} \sum_{k=-N/2}^{N/2-1} H_{MF,0}[k] \cos 2\pi \frac{k}{N} \frac{\delta}{P} \cos 2\pi \frac{k}{N} n - H_{MF,0}[k] \sin 2\pi \frac{k}{N} \frac{\delta}{P} \sin 2\pi \frac{k}{N} n$$

Using the identity $\cos A \cos B - \sin A \sin B = \cos(A + B)$, the above equation can be rearranged as

$$I \rightarrow \frac{1}{N} \sum_{k=-N/2}^{N/2-1} H_{MF,0}[k] \cos 2\pi \frac{k}{N} \left(n + \frac{\delta}{P} \right)$$

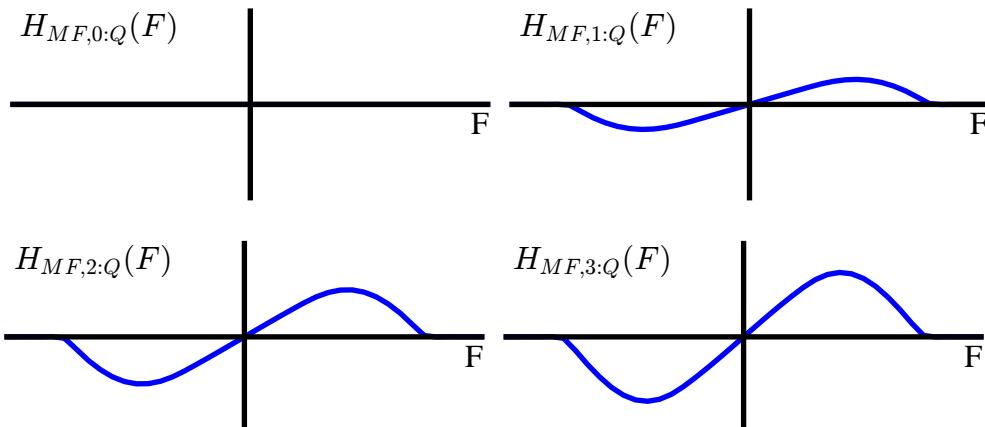


Figure 7.84: Q spectra of the polyphase partitions corresponding to Figure 7.82 for excess bandwidth $\alpha = 0.5$, $L = 2$ samples/symbol and upsampling factor $P = 4$

Similarly, the Q part is given by

$$I \rightarrow \frac{1}{N} \sum_{k=-N/2}^{N/2-1} H_{MF,0}[k] \sin 2\pi \frac{k}{N} \left(n + \frac{\delta}{P} \right)$$

As a result, we can see that the peak in time domain occurs at $n + \delta/P = 0$ or $n = -\delta/P$. For $\delta = 1$, the sinusoid arising from the above phase shift moves the time domain samples to the left by $1 \cdot T_S/P$ (which was shown earlier in Figure 7.77). On the other hand, at time 0 according to its own reference, the interpolated sample is available at actual instant $1 \cdot T_S/P$. The commutator, operating at a rate of P/T_S , *plucks this sample just at the right time*, i.e., at $1 \cdot T_S/P$! And this is why the commutator direction is shown coming from top to bottom in all the figures about upsampling and interpolation.

Child filters 2: $h_{MF,2}(nT_S)$ and onwards The remaining polyphase partitions are also drawn in Figure 7.82 where the inverse period of the mixing complex sinusoid can be seen increasing. To be exact, their inverse periods are

$$\begin{aligned} 2\pi F \cdot 2 \frac{T_S}{P} &= 2\pi \frac{F}{F_S} \cdot \frac{2}{4} \\ 2\pi F \cdot 3 \frac{T_S}{P} &= 2\pi \frac{F}{F_S} \cdot \frac{3}{4} \end{aligned}$$

These filters, after shaping the incoming spectrum, also play a similar role as $H_{MF,1}(F)$ by contributing their respective time outputs at $2T_S/P$ and $3T_S/P$, respectively. This is the concept of *time-delay beamforming* associated with a polyphase filterbank. Note that these inverse periods have been exaggerated a little in figures above to demonstrate the effect of downsampled matched filters mixing with a complex sinusoid.

The Origin of the Expression ‘Polyphase’

Now let us denote

$$\begin{aligned} h_{MF,0}(nT_S) &\xrightarrow{F} H_{MF,0}(F) \\ h_{MF,\delta}(nT_S) &\xrightarrow{F} H_{MF,\delta}(F) \end{aligned}$$

and from Eq (7.72), we had

$$h_{MF,\delta}(nT_S) = h_{MF} \left(nT_S + \delta \frac{T_S}{P} \right) = h_{MF} \left\{ \left(n + \frac{\delta}{P} \right) T_S \right\}, \quad \delta = 0, 1, \dots, P-1$$

Just a difference of a time shift implies that magnitudes of all child filters are the same but their phases are a function of this time shift.

$$\begin{aligned} |H_{MF,\delta}(F)| &= |H_{MF,0}(F)| \\ \angle H_{MF,\delta}(F) &= 2\pi \frac{F}{F_S} \cdot \frac{\delta}{P} \end{aligned}$$

where $\angle H_{MF,0}(F)$ is zero due to the pulse shape being real and even. We draw

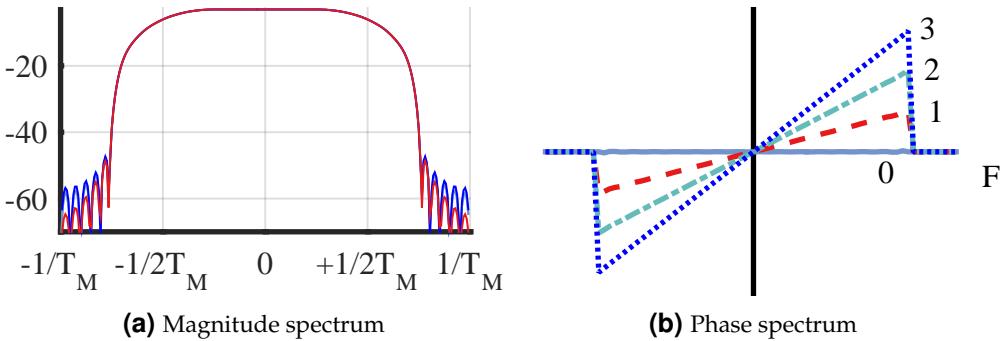


Figure 7.85: While the magnitude spectra of the polyphase partitions are similar, their phase spectra demonstrate a linear phase profile with different slopes due to their respective time origins

the magnitude and phase spectra of the polyphase partitions in Figure 7.85 for a Square-Root Raised Cosine mother filter with excess bandwidth $\alpha = 0.5$ and $L = 2$ samples/symbol. Notice how their magnitude spectra are the same while their phase spectra demonstrate a linear phase profile with different slopes arising due to their respective time origins. These slopes differ by a scaling factor of $1/P$ only, see the equation above. It is due to these distinct phase profiles that these filters combined are known as a *polyphase filterbank*!

In my opinion, appreciating the operation of polyphase filters is easier in *IQ* plots than magnitude phase plots. For this reason, the *IQ* spectra of Figure 7.82 better explain this concept than the phase spectrum of Figure 7.85. Later in Section 10.2.5, we will go deeper into their phase aspects in the context of downsampling the signal.

A block diagram for implementing a timing locked loop with a Gardner TED, a PI loop filter and interpolation control is drawn in Figure 7.86 where both the matched filtering and interpolation are combined into one operation performed by a polyphase matched filter. Since there is no interpolation, there is no need for a fractional index μ_m . Instead, the ‘compute filterbank index’ block simply points to the polyphase partition supplying the closest fractional delay at each instant.

$$\mu_m = \frac{\delta}{P}$$

This architecture is particularly attractive for radios that require high speed baseband signal processing on an embedded platform. Later, when we will describe a QAM Tx architecture in Section 10.1, we will see how polyphase clock synchronization at the Rx is just a simple dual to the pulse shaping operations at the Tx.

7.14 The Small Picture

During acquisition, the timing synchronization block works in the presence of a carrier phase offset and mostly a carrier frequency offset. Therefore, the luxury of downsampling the signal at $L = 1$ sample/symbol is not yet available. This invites the energy maximization perspective of the matched filtered signal $z(nT_S)$ for which we process the signal either before or after matched filtering in a prescribed manner. Recalling the *seesaw perspective* of Figure 7.28, we proceed as follows.

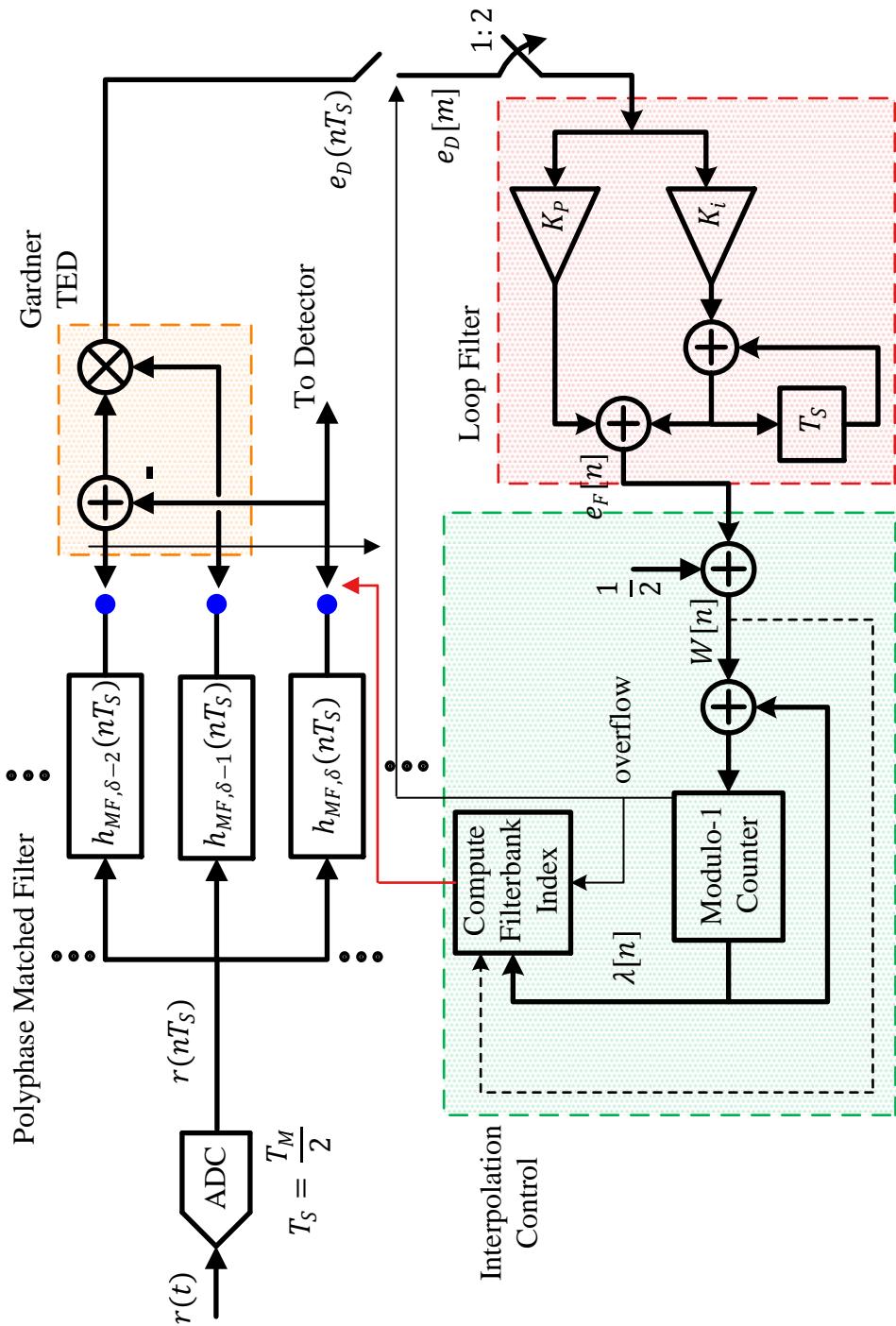


Figure 7.86: A timing locked loop with a Gardner TED, PIloop filter and interpolator control

Squaring: Since energy is just a sum of magnitude squared values of a signal, some schemes explicitly operate on this principle to generate spectral lines at $\pm 1/T_M$ and track them through a simple PLL (or a bandpass filter).

$$\sum_n |z(nT_S)|^2$$

A squaring loop as well as digital filter and square estimator are examples of such approaches.

Peak search: Other schemes implicitly maximize this energy by searching for the peak through taking its derivative, which comes out to be the product between the signal and its derivative.

$$z(mT_M + \hat{\varepsilon}_\Delta) \dot{z}(mT_M + \hat{\varepsilon}_\Delta)$$

For example, a derivative, early-late or a zero crossing TED (implicitly through tracking zero crossings) as well as band edge filters generate an error signal based on this principle which guides the timing locked loop.

The workings of the algorithms operating at $L = 1$ sample/symbol is quite similar.

This brings an end to our account of theory and practice on synchronization. The common strategy you would have noticed in carrier and timing synchronization is the following:

- Summarize the signal in time domain through an eye diagram or frequency domain through a spectrum.
- Balance it around a fulcrum which is point zero whether in time or frequency domain.

This balance nicely corresponds to our aesthetic sense of imparting symmetry to many of our designed products. If you look around, most of the man-made objects you will find are quite symmetric, even those that do not have simple geometrical shapes like shoes. Symmetry brings about a closure to design by eliminating complexity. A word of caution: it also blocks some routes to creativity.

7.15 Appendix

Here, we derive some relations utilized in the main text.

Mean Curve for Data-Aided Derivative TED

To derive the mean curve, first consider how a symbol timing offset ε_Δ delays the matched filter output $z(mT_M)$. When the Rx ADC samples the signal $r(t)$ at a rate of $F_S = 1/T_S$ samples/second, we get

$$r(nT_S) = \gamma \sum_i a[i] p(nT_S - iT_M - \varepsilon_\Delta)$$

where γ is an amplitude scaling factor that arises from the gains and losses through the antennas, wireless channel and Tx/Rx frontend components. This signal is input to a matched filter

$h(nT_S) = p(-nT_S)$ and the output is written as

$$\begin{aligned} z(nT_S) &= \gamma \left(\sum_i a[i] p(nT_S - iT_M - \varepsilon_\Delta) \right) * p(-nT_S) \\ &= \gamma \sum_i a[i] r_p(nT_S - iT_M - \varepsilon_\Delta) \end{aligned}$$

Combining the two operations of interpolating the matched filter output at $\hat{\varepsilon}_\Delta$ (which we will cover later in the chapter) and downsampling at $n = mL$, we can write

$$nT_S = mT_M + \hat{\varepsilon}_\Delta$$

which yields

$$\begin{aligned} z(mT_M + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p(mT_M + \hat{\varepsilon}_\Delta - iT_M - \varepsilon_\Delta) \\ &= \gamma \sum_i a[i] r_p[(m-i)T_M + \hat{\varepsilon}_\Delta - \varepsilon_\Delta] \\ &= \gamma \sum_i a[i] r_p[(m-i)T_M - \varepsilon_{\Delta:e}] \end{aligned} \quad (7.75)$$

Here, $\varepsilon_{\Delta:e}$ is defined as

$$\varepsilon_{\Delta:e} = \varepsilon_\Delta - \hat{\varepsilon}_\Delta$$

Similarly, the derivative of $z(mT_M + \hat{\varepsilon}_\Delta)$ is given by

$$\dot{z}(mT_M + \hat{\varepsilon}_\Delta) = \gamma \sum_i a[i] \cdot \dot{r}_p[(m-i)T_M - \varepsilon_{\Delta:e}]$$

where $\dot{r}_p[(m-i)T_M - \varepsilon_{\Delta:e}]$ is the time derivative of the overall Nyquist filter[†].

Next, we plug in this value in the expression for the derivative TED as in Eq (7.22) for mean curve calculation.

$$\begin{aligned} \overline{e_D} &= \text{Mean}\{e_D[m]\} = \text{Mean}\left\{ a[m] \cdot \dot{z}(mT_M + \hat{\varepsilon}_\Delta) \right\} \\ &= \text{Mean}\left\{ a[m] \cdot \gamma \sum_i a[i] \cdot \dot{r}_p[(m-i)T_M - \varepsilon_{\Delta:e}] \right\} \\ &= \gamma \sum_i \text{Mean}\left\{ a[m] \cdot a[i] \right\} \cdot \dot{r}_p[(m-i)T_M - \varepsilon_{\Delta:e}] \end{aligned} \quad (7.76)$$

For a binary PAM, the data symbols take values from a finite symbol set $\{-A, +A\}$. Each such option has a $1/2$ chance of occurring, hence whenever $i \neq m$, the average of $a[m] \cdot a[i]$ is zero.

$$\text{Mean}\{a[m] \cdot a[i]\} = 0$$

Only when $i = m$, its average value is A^2 .

$$\begin{aligned} \text{Mean}\{a[m] \cdot a[m]\} &= \frac{1}{2}(a[m] = +A) \cdot (a[m] = +A) + \\ &\quad \frac{1}{2}(a[m] = -A) \cdot (a[m] = -A) = A^2 \end{aligned}$$

[†]To check where the negative sign arising due to the presence of $-\varepsilon_{\Delta:e} = -(\varepsilon_\Delta - \hat{\varepsilon}_\Delta)$ inside $r_p[(m-i)T_M - \varepsilon_{\Delta:e}]$ has gone after taking the derivative, remember that

$$r_p(nT_S) = p(nT_S) * p(-nT_S)$$

$$\dot{r}_p(nT_S) = p(nT_S) * \{-\dot{p}(-nT_S)\}$$

which can be proved through the definition of convolution, see Ref. [18].

We will use this result for deriving the mean curves of other TEDs as well.

$$\text{Mean}\{a[m] \cdot a[i]\} = \begin{cases} A^2 & i = m \\ 0 & \text{otherwise} \end{cases} \quad (7.77)$$

As a result, plugging $i = m$ in Eq (7.76) produces $\overline{e_D}$ as[†]

$$\overline{e_D} = \gamma A^2 \cdot \dot{r}_p(-\varepsilon_{\Delta:e})$$

Remember that $r_p(nT_S)$, being a Nyquist pulse auto-correlation such as a Raised Cosine, is an even time domain signal. Without proof, we state that the derivative of an even signal is an odd signal and consequently $\dot{r}_p(nT_S)$ is an odd time domain signal. This is drawn in Figure 7.87 for a Raised Cosine pulse with excess bandwidth $\alpha = 0.5$. The reason behind marking the region from $-0.5T_M$ to $+0.5T_M$ with an arrow will be explained soon.

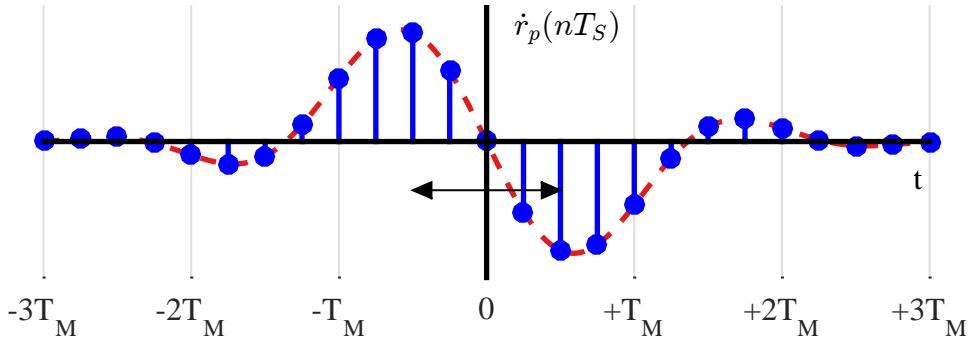


Figure 7.87: Derivative of a Raised Cosine pulse, $\dot{r}_p(nT_S)$, with excess bandwidth $\alpha = 0.5$

Now $\dot{r}_p(-\varepsilon_{\Delta:e})$ being an odd signal, we can write

$$\dot{r}_p(-\varepsilon_{\Delta:e}) = -\dot{r}_p(\varepsilon_{\Delta:e})$$

Plugging in the expression for $\overline{e_D}$ above generates

$$\overline{e_D} = -\gamma A^2 \cdot \dot{r}_p(\varepsilon_{\Delta:e})$$

[†]This can be seen by expanding the summation over i as

$$\begin{aligned} \overline{e_D} &= -\gamma \cdot \text{Mean} \left\{ a[0] \left(a[0] \cdot \dot{r}_p[(m-0)T_M - \varepsilon_{\Delta:e}] \right. \right. \\ &\quad \left. \left. + a[1] \cdot \dot{r}_p[(m-1)T_M - \varepsilon_{\Delta:e}] + \dots + a[m] \cdot \dot{r}_p[-\varepsilon_{\Delta:e}] + \dots \right) \right\} \\ &= -\gamma \cdot \text{Mean} \left\{ a[m] \right\} \dot{r}_p(-\varepsilon_{\Delta:e}) = -\gamma A^2 \cdot \dot{r}_p(-\varepsilon_{\Delta:e}) \end{aligned}$$

Mean Curve for Data-Aided Early-Late TED

To derive the mean curve, first consider how a symbol timing offset ε_Δ and subsequent interpolation by $\hat{\varepsilon}_\Delta$ affect the matched filter output $z(mT_M)$, which was derived earlier as

$$\begin{aligned} z(mT_M + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p(mT_M + \hat{\varepsilon}_\Delta - iT_M - \varepsilon_\Delta) \\ &= \gamma \sum_i a[i] r_p \left[(m-i)T_M - \varepsilon_{\Delta:e} \right] \end{aligned}$$

where

$$\varepsilon_{\Delta:e} = \varepsilon_\Delta - \hat{\varepsilon}_\Delta$$

From the above equation, we can write

$$z \left(mT_M \pm \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) = \gamma \sum_i a[i] r_p \left[(m-i)T_M \pm \frac{T_M}{2} - \varepsilon_{\Delta:e} \right]$$

Plugging in the expression for early-late TED,

$$\begin{aligned} \overline{e_D} &= \text{Mean} \left\{ a[m] \cdot \left\{ z \left(mT_M + \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \right\} \right\} \\ &= \text{Mean} \left\{ a[m] \cdot \left\{ \gamma \sum_i a[i] r_p \left[(m-i)T_M + \frac{T_M}{2} - \varepsilon_{\Delta:e} \right] - \right. \right. \\ &\quad \left. \left. \gamma \sum_i a[i] r_p \left[(m-i)T_M - \frac{T_M}{2} - \varepsilon_{\Delta:e} \right] \right\} \right\} \end{aligned}$$

Now we use the result derived in Eq (7.77) regarding the independence of data symbols.

$$\text{Mean}\{a[m] \cdot a[i]\} = \begin{cases} A^2 & i = m \\ 0 & \text{otherwise} \end{cases}$$

Equating $i = m$ in the above expression generates

$$\overline{e_D} = \gamma A^2 \left\{ r_p \left(+\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) - r_p \left(-\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) \right\}$$

Mean Curve for Data-Aided Zero Crossing TED

To derive the mean curve of a data-aided zero crossing TED, first consider how a symbol timing offset ε_Δ and subsequent interpolation by $\hat{\varepsilon}_\Delta$ affect the matched filter output $z(mT_M)$, which was derived in Eq (7.75) as

$$\begin{aligned} z(mT_M + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p \left[(m-i)T_M - \varepsilon_{\Delta:e} \right] \\ z((m-1)T_M + \hat{\varepsilon}_\Delta) &= \gamma \sum_i a[i] r_p \left[(m-1-i)T_M - \varepsilon_{\Delta:e} \right] \end{aligned}$$

Here, $\varepsilon_{\Delta:e}$ was defined as

$$\varepsilon_{\Delta:e} = \varepsilon_\Delta - \hat{\varepsilon}_\Delta$$

From the above equation, we can write

$$z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) = \gamma \sum_i a[i] r_p \left[(m-i)T_M - \frac{T_M}{2} - \varepsilon_{\Delta:e} \right]$$

Plugging in the expression for zero crossing TED,

$$\begin{aligned}\overline{e_D} &= \text{Mean} \left\{ z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \cdot \left\{ a[m-1] - a[m] \right\} \right\} \\ &= \text{Mean} \left\{ \gamma \sum_i a[i] r_p \left[(m-i)T_M - \frac{T_M}{2} - \varepsilon_{\Delta:e} \right] \cdot \left\{ a[m-1] - a[m] \right\} \right\}\end{aligned}$$

Now we use the result derived in Eq (7.77) regarding the independence of data symbols.

$$\text{Mean}\{a[m] \cdot a[i]\} = \begin{cases} A^2 & i = m \\ 0 & \text{otherwise} \end{cases}$$

Equating $i = m-1$ and $i = m$, respectively, in the above expression generates the desired result.

$$\overline{e_D} = \gamma A^2 \left\{ r_p \left(+\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) - r_p \left(-\frac{T_M}{2} - \varepsilon_{\Delta:e} \right) \right\}$$

Gardner's Derivation of Gardner TED

In his seminal paper Ref. [23], Gardner arrived at the expression for this TED through an interesting route. He noticed that although the timing can be extracted from the slope of the squared average eye as drawn in Figure 7.16d, it contains excessive self noise during no data transitions. Applying the derivative first to the waveform generates a zero output at no data transitions. This signal can then be squared and its derivative can be computed. In place of a derivative, Gardner chose a delay difference methodology as follows.

First, take the difference between two matched filter outputs $T_M/2$ seconds apart.

$$y(mT_M + \hat{\varepsilon}_\Delta) = z(mT_M + \hat{\varepsilon}_\Delta) - z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \quad (7.78)$$

Squaring the above expression, we get

$$\begin{aligned}y^2(mT_M + \hat{\varepsilon}_\Delta) &= z^2(mT_M + \hat{\varepsilon}_\Delta) + z^2 \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - \\ &\quad 2z(mT_M + \hat{\varepsilon}_\Delta) \cdot z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right)\end{aligned} \quad (7.79)$$

Now consider the difference signal $y(nT_S)$ at a time $T_M/2$ seconds earlier, i.e., plug in $mT_M - T_M/2$ in Eq (7.78) above.

$$y \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) = z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) - z((m-1)T_M + \hat{\varepsilon}_\Delta)$$

Squaring this expression produces

$$\begin{aligned}y^2 \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) &= z^2 \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) + z^2((m-1)T_M + \hat{\varepsilon}_\Delta) - \\ &\quad 2z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \cdot z((m-1)T_M + \hat{\varepsilon}_\Delta)\end{aligned} \quad (7.80)$$

Finally, we can subtract $y^2(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$ from $y^2(mT_M + \hat{\varepsilon}_\Delta)$ that cancels the common term $z^2(mT_M - T_M/2 + \hat{\varepsilon}_\Delta)$. Subtracting Eq (7.80) from Eq (7.79),

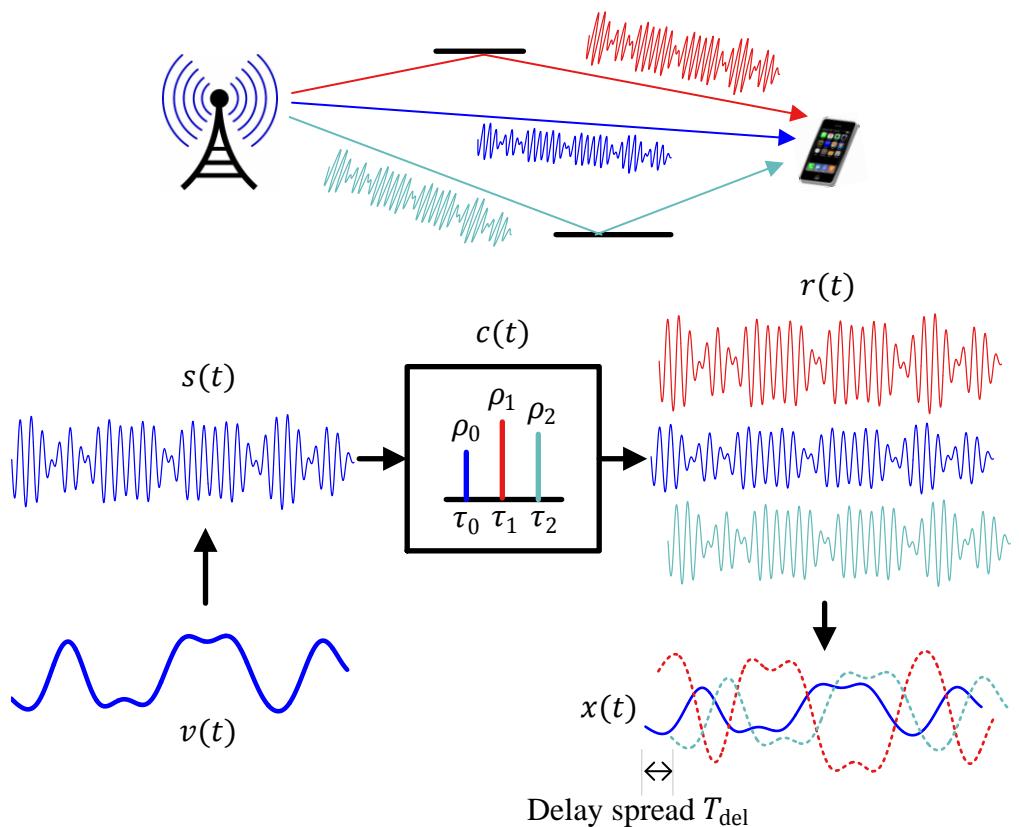
$$\begin{aligned}y^2(mT_M + \hat{\varepsilon}_\Delta) - y^2 \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) &= \\ z^2(mT_M + \hat{\varepsilon}_\Delta) - z^2((m-1)T_M + \hat{\varepsilon}_\Delta) &+ \\ 2z \left(mT_M - \frac{T_M}{2} + \hat{\varepsilon}_\Delta \right) \left\{ z((m-1)T_M + \hat{\varepsilon}_\Delta) - z(mT_M + \hat{\varepsilon}_\Delta) \right\}\end{aligned}$$

Looking at the above expression, the first two terms on the right hand side statistically generate the same average for a long enough period. Therefore, they must be equal and cancel out. In fact, they were potential sources of self noise and hence should be eliminated. A timing error detector can thus be formed from the third term as

$$e_D[m] = z \left(mT_M - \frac{T_M}{2} + \hat{\epsilon}_\Delta \right) \left\{ z((m-1)T_M + \hat{\epsilon}_\Delta) - z(mT_M + \hat{\epsilon}_\Delta) \right\}$$

Chapter 8

Wireless Channel and Equalization



“If your mind carries a heavy burden of past, you will experience more of the same.... The quality of your consciousness at this moment is what shapes the future.”

Eckhart Tolle

We started the departure from the ideal linearly modulated signal with the description of AWGN. Then, we explored the process of signal recovery in the presence of carrier phase, carrier frequency and symbol timing errors. In this chapter, we deal with the wireless channel and how it distorts the received signal. The solution is the equalizer which in the Rx design is a block of paramount importance. As an example, the equalization method is the defining feature of the winning technology for each generation of our cellular networks. Although equalization on its own requires a complete book for a proper justice, my intention in this chapter is to examine the subject to a reasonable level. The interested reader can delve deeper through the excellent analysis in Ref. [3].

Note 8.1 Brain of a digital Rx

In a wireless communications Rx, while the timing synchronization plays a similar role as that of a heart in a human body, the equalizer plays a similar role as that of the brain by making sense out of the jumbled up data received by the system.

Before we move towards the effects of this multipath distortion, we touch on how we have already encountered different channels before. A wireless channel is the most general phenomenon that acts on the Tx signal as it arrives at the Rx. Any impairment thus caused can be considered as a special case of this general development. For example, we saw the special cases of a wireless channel with a single path in the previous chapters on synchronization.

Carrier phase offset: A carrier phase offset is a channel with a direct path rotated in phase as compared to the perfect symbol phases at the Tx constellation. Estimation and compensation of this phase is necessary to have an identical constellation at the Rx.

Carrier frequency offset: A carrier frequency offset can be considered as a single path that impinges on the Rx antenna of a device that is moving directly opposite to or in the direction of the Tx antenna, thus changing the frequency of the Rx signal. Obviously, this is physically not true but we will explore the idea of a Doppler shift in Section 8.1.4. There, we will find that the Doppler shift of a single path is like a carrier frequency offset but the Doppler shifts of multiple paths is what plays a central role in defining the complexity of the wireless channel.

Symbol timing offset: A symbol timing offset is again a channel with a direct path that arrives with a delay at the Rx. This delay (according to the Rx time scale) needs to be figured out so that the waveform can be sampled at the appropriate times.

8.1 A Wireless Channel

Recall that we have worked with a unit Rx power in the previous chapters to focus on the main concept under consideration. In reality, the signal is transmitted with a sufficient power to satisfy a certain link budget to ‘close’ the link. For this purpose, we need to understand how the wireless channel acts on the Tx signal. Through extensive measurement campaigns, it has been found that the Tx signal undergoes two major kinds of fading in a wireless channel.

Large scale fading: Wave propagation in an ideal free space is the starting point where the effect of a wireless channel enters our signal equations which implies the following.

- The region between Tx and Rx is considered free of all objects that might absorb or reflect RF energy.
- The atmosphere behaves as a uniform and nonabsorbing medium.
- The earth is treated as being far away from the signal path.

In such a setup, we assume that a Tx transmits P_{Tx} watts of power uniformly radiating in all spatial directions. Taking the Tx as a point source, the power radiates outwards in every dimension, i.e., in a sphere. Then, the power density at a distance d from the Tx point source is the ratio of the Tx power P_{Tx} to the surface area of the sphere $4\pi d^2$.

$$\text{Power density} = \frac{P_{Tx}}{4\pi d^2} \quad \text{Watts}/m^2$$

Therefore, *the Tx power decays with inverse of squared distance in free space* and the corresponding loss is termed as path loss. This is drawn in Figure 8.1 for a sphere with radius d .

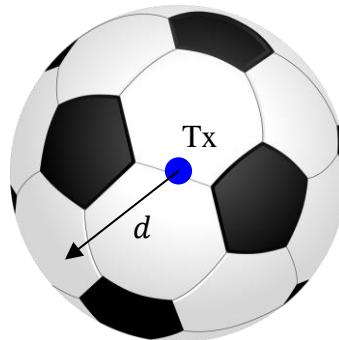


Figure 8.1: Free space path loss at a distance d from the Tx antenna

For most practical channels, the free space path loss is an idealized simplification because a signal can reach the Rx after getting reflected, diffracted or scattered from multiple objects in the surroundings as well as the ground. For example, a single reflection from the ground can degrade the Rx power as much

as the fourth power of the inverse distance – as opposed to the squared distance in free space. Different environments exhibit different such coefficients appearing as a power to the inverse distance. As a general rule,

$$P_{Rx} \propto \frac{1}{d^\beta}$$

with the value of β dictated by the environment and empirically determined through an experimental campaign.

In addition to this path loss, the large scale fading includes *shadowing* caused by buildings and other obstacles affecting the wave propagation (the term is borrowed from the sunlight shadowed by the clouds). Combined path loss and shadowing represent the changes in average signal power over a large area.

Since large scale fading is useful for estimating average Rx power over a certain distance, it is mostly utilized in configuring the power budget of a communication link and cell site planning in mobile radio systems. To a DSP practitioner, small scale fading is more relevant as explained next.

Small scale fading: Dramatic variations in signal amplitude occur at the Rx from constructive and destructive interference of multipath components originating from the surrounding environment that give rise to small scale fading. This is the main challenge for designing efficient communication systems and hence we focus on solving this problem in the rest of this chapter.

Next, we move towards a simple abstraction of multipath distortion assuming an absence of carrier waves.

8.1.1 Multipath Distortion Assuming No Carrier Wave

In this section, we describe the effect of small scale fading without including the effect of carrier wave (we address that in the next section).

An example of these multipath components is shown in Figure 8.2 where many multipath components arrive at the Rx of a hiker after being reflected from the nearby surfaces such as an aeroplane, houses, trees and the mountains.

Here, the direct path to the hiker is shown by the solid bold line. Assume that the second path arrives after a delay of $0.6 \mu\text{s}$ of the direct path (say, from the aeroplane) and the third path arrives after a delay of $1.1 \mu\text{s}$ as compared to the direct path. We will use these numbers in our examples below.

In the discussion that follows, *we ignore the RF carrier* to highlight the important relevant concepts.

A Low Data Rate Signal

Suppose that the year is early 2000s and our hiker in Figure 8.2 only wants to check his email and/or messages (probably driven by what is available) and requires a data rate of only $1/T_M = 100 \text{ kbps}$. This translates to a symbol time T_M of

$$T_M = \frac{1}{100,000} = 10 \mu\text{s}$$

as drawn in Figure 8.3a.

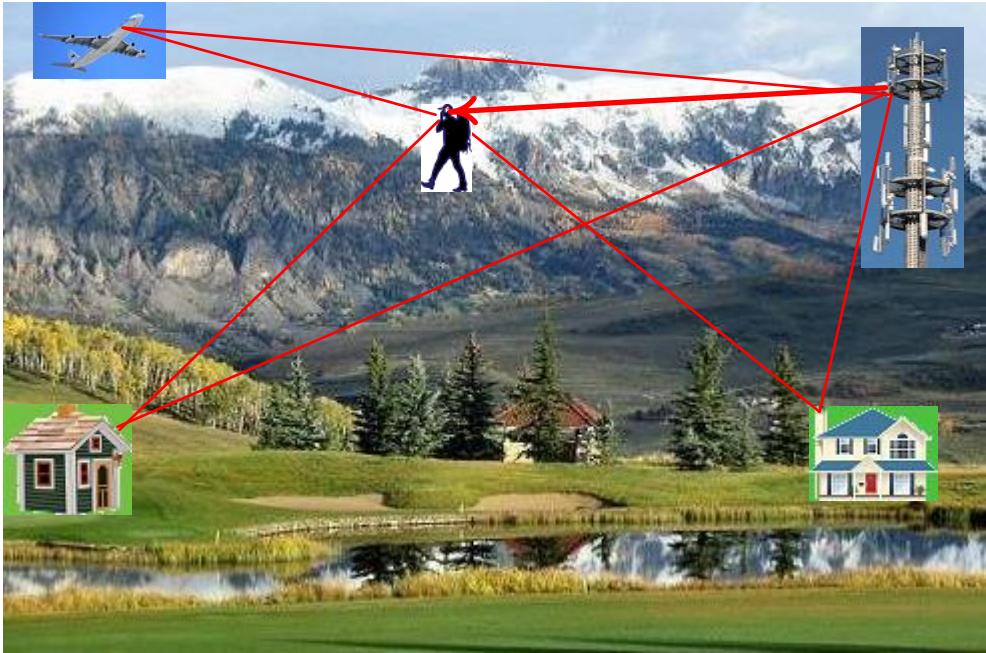


Figure 8.2: Multipath components arriving at the Rx

As described above, the first and second multipath components arrive $0.6 \mu\text{s}$ and $1.1 \mu\text{s}$ after the direct path, respectively (ignoring the carrier). This is shown in Figure 8.3b.

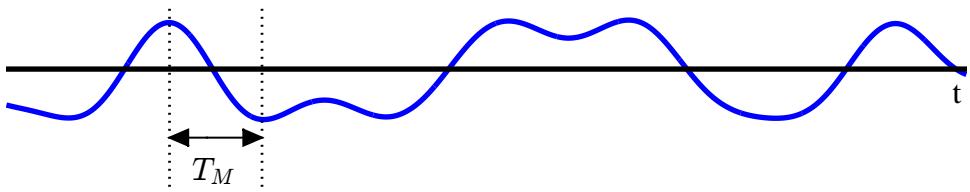
Since nature adds the signals at the antenna, the Rx will have a summation of these three paths, effectively the same signal delayed by different amounts but with different attenuation and phase shifts of the carrier waves (not drawn in the figure). As we find out later, those phase shifts depend on the path delay and the carrier frequency. Observe that *the maximum delay of $1.1 \mu\text{s}$ is around 10% of the symbol duration $T_M = 10 \mu\text{s}$* , an implication of which is that multipath components will distort the Rx signal but there will a limited interference among the symbols themselves. That is to say that symbol 1 through its last path will interfere with symbol 2 to a little extent but not with any other symbol farther than that.

This is a situation that can be handled without much effort in terms of the computational resources. We claim that the wireless channel does not pose a significant problem to Rx processing time in this low data rate scenario[†].

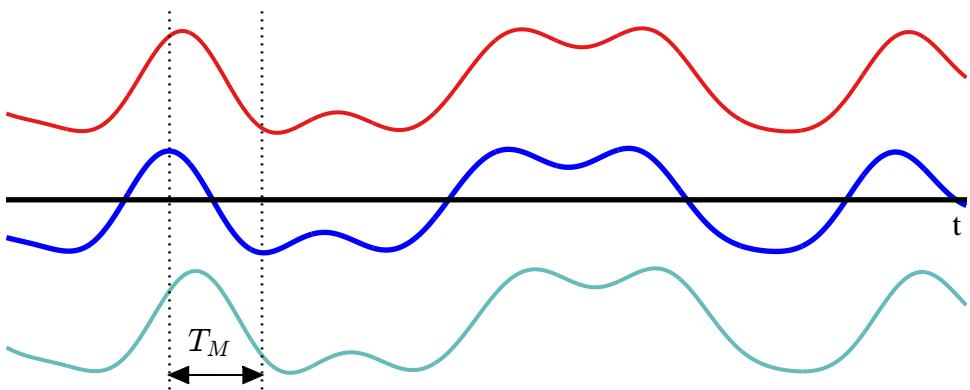
Towards a High Data Rate

Let us return to Figure 8.2 and fast forward to a decade, say early 2010s. Escaping the fast pace of life, our hiker wants to sit on top of that peaceful mountain to watch YouTube videos. In fact, he considers having access to it anywhere as a basic necessity of life like water (and to a modern human, electricity). Assume that a data

[†]It is not necessary to understand this in the current context but the disadvantage here is that in the case of highly destructive interference, there is no other way to recover except providing diversity to the Tx signal. Diversity is a signal replica in some form whether in time, frequency, space, etc.



(a) A low data rate $1/T_M = 100$ kbps signal with a symbol duration $T_M = 10 \mu\text{s}$



(b) First and second multipath components arriving 0.6 and $1.1 \mu\text{s}$ after the direct path, respectively (carrier wave not shown)

Figure 8.3: For low data rate, multipath components will distort the Rx signal but there will not be much interference among the symbols themselves

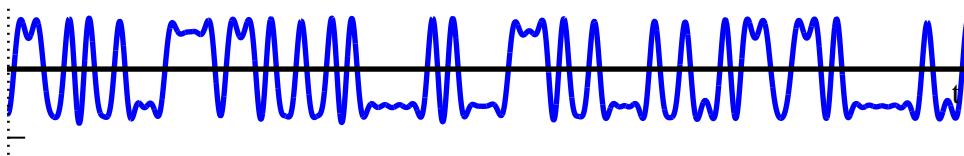
rate of $1/T_M = 10$ Mbps is needed for this purpose, which translates to a symbol duration of $T_M = 0.1 \mu\text{s}$ as shown in Figure 8.4a.

Since *the environment is still the same*, the multipath components previously arriving 0.6 and $1.1 \mu\text{s}$ after the direct path will still arrive 0.6 and $1.1 \mu\text{s}$ after the direct path. This is illustrated in Figure 8.4b (again ignoring the carrier). The different path lengths will translate into different attenuations and phase shifts resulting in constructive and destructive interference throughout the signal span, as we shortly see next. Notice that for such a high data rate signal, the distortion effects from our first symbol will reach dozens of symbols in the future. For even higher data rates, hundreds of the future symbols can be affected.

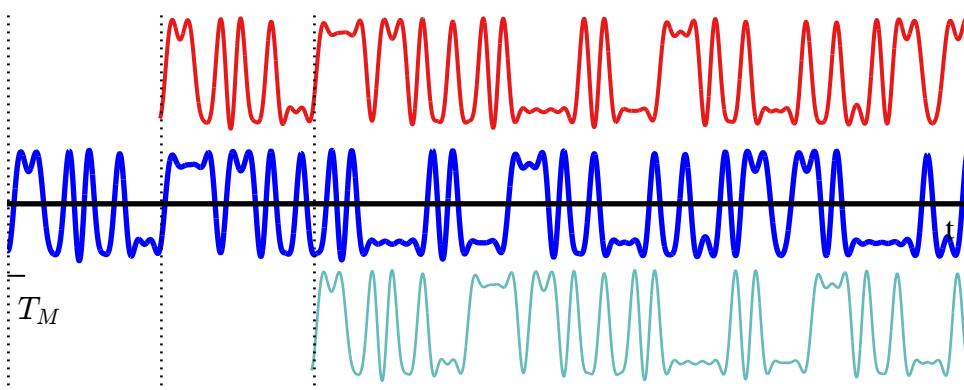
Since each symbol is interfering with many dozens of symbols in the future through the late arriving paths, this phenomenon known as Inter-Symbol Interference (ISI). We can conclude that *the same harmless channel for low rate communication has become harsh for high rate communication!* In fact, operators of the first transatlantic telegraph cable in 1858 onwards had to transmit Morse code dots and dashes at a very slow rate to be understood at the other end [31] due to this reason.

Note 8.2 The Cold Equations

In the context of a wireless channel, the universe does not care about our data rates! The laws of the universe manifest themselves as emotionless masters and we humans



(a) A high data rate $1/T_M = 10$ Mbps signal with a symbol duration $T_M = 0.1 \mu s$



(b) First and second multipath components arriving 0.6 and 1.1 μs after the direct path, respectively (carrier wave not shown)

Figure 8.4: Effect of multipath on a Rx signal. For such a high data rate signal, the interference effects from our first symbol can reach dozens of symbols in the future

have no option but to comply. This reminds me of “The Cold Equations”, a science fiction short story by American writer Tom Godwin [32]. In Godwin’s words.

“Existence required order, and there was order; the laws of nature, irrevocable and immutable. Men could learn to use them, but men could not change them. The circumference of a circle was always pi times the diameter, and no science of man would ever make it otherwise. The combination of chemical A with chemical B under condition C invariably produced reaction D. The law of gravitation was a rigid equation, and it made no distinction between the fall of a leaf and the ponderous circling of a binary star system. ... to the laws of nature she was x, the unwanted factor in a cold equation.”

The Two ISIs

In Section 3.6, we have discussed the ISI in terms of designing spectrally efficient pulse shapes where the sampling instant has to be exactly at integer multiples of T_M to avoid ISI from the neighbouring signals. One example of such ISI arising from non-optimal sampling was drawn in Figure 7.3 as well. This is the first type of ISI on which

we have some kind of control through careful system design. For example, a Nyquist pulse ensures zero crossings and hence zero interference at symbol locations which are detected by a symbol timing synchronization block. Furthermore, scientists have come up with schemes such as duobinary signaling to inject a controlled amount of ISI in the communication system to further carve out a desired spectrum.

The second type of ISI arises from the interactions of the signal with the wireless channel, examples of which we have seen in Figure 8.3b and Figure 8.4b. Here, we have no control over the environment and the channel is not known with sufficient precision to design a Rx with zero ISI all the time. Moreover, this ISI cannot be overcome by simply increasing the Tx power since the multipath power rises as well setting an irreducible error floor at the Rx. The solution is to design a suitable equalizer which is described in detail later.

8.1.2 What Happens When a Carrier Wave Enters the Picture

In this section, we will analyze how a wireless channel affects the passband signal sent over the air. In general, a channel comprises of the response created by the multipath copies of the signal as well as the analog and digital frontends employed, e.g., a power amplifier at the Tx and a low noise amplifier at the Rx, various filters for meeting the spectral mask requirements as well as eliminating noise and interference outside the desired band, the mixers, and so on. However, we will neglect these contributions from the frontend and treat the wireless channel as originating from the multipath copies of the Tx signal only. This is a reasonable assumption as long as a sufficient level of linearity is maintained in designing those components.

We start with Figure 8.5 to clarify the signal notations used at various stages of the transceiver. In Section 3.7 on Quadrature Amplitude Modulation (QAM), we represented a QAM waveform as a passband signal consisting of two Pulse Amplitude Modulated (PAM) waveforms in I and Q rails:

$$\begin{aligned} s(t) &= \sum_m a_I[m] p(t - mT_M) \sqrt{2} \cos 2\pi F_C t - \\ &\quad \sum_m a_Q[m] p(t - mT_M) \sqrt{2} \sin 2\pi F_C t \\ &= v_I(t) \sqrt{2} \cos 2\pi F_C t - v_Q(t) \sqrt{2} \sin 2\pi F_C t \end{aligned} \tag{8.1}$$

with $v_I(t)$ and $v_Q(t)$ defined as continuous-time versions of the corresponding PAM waveforms.

$$\begin{array}{ll} I & \rightarrow & v_I(nT_S) = \sum_m a_I[m] p(nT_S - mT_M) \\ Q & \uparrow & v_Q(nT_S) = \sum_m a_Q[m] p(nT_S - mT_M) \end{array}$$

Particularly, *keep in mind that $v(t)$ is the signal at the Tx before the upconversion and $x(t)$ is the signal at the Rx after the downconversion*. We take the time domain view of the problem first and then move towards a frequency domain explanation.

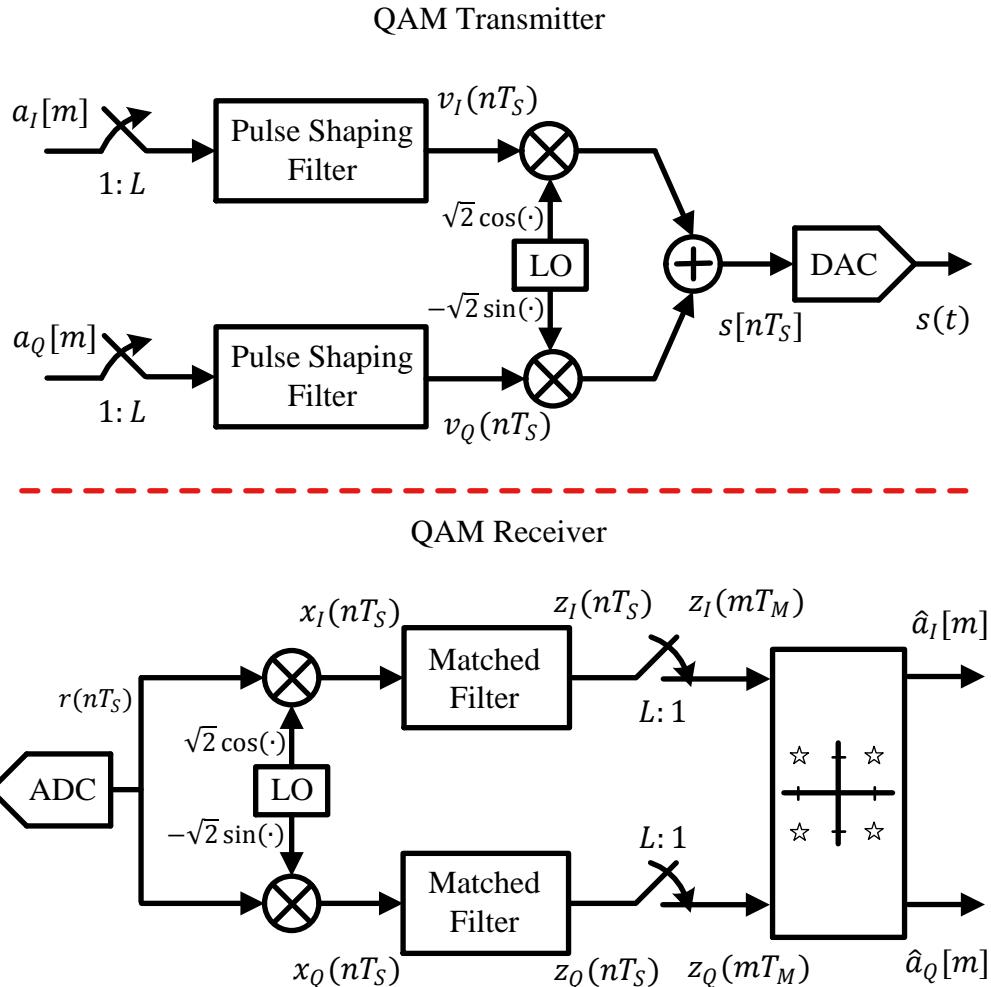


Figure 8.5: A block diagram of a QAM system to explain the signal notations at various blocks

Time Domain: Channel Impulse Response

Before the Rx antenna sums all the arriving paths, refer to Figure 8.3b and notice that each multipath arrives with two distinguishing features as follows.

Amplitude: Clearly, each multipath traverses a different route on its way to the Rx. Large scale fading (i.e., path loss and shadowing) then dictates the respective amplitude of each such path arising through reflection, diffraction and scattering of the electromagnetic wave. Owing to the large scale fading, there is not much change in these amplitudes over shorter distances (as opposed to delays and phases). This is a key point which we exploit in the coming discussion. For a total number of N_{MP} multipath, we denote the amplitude of the i^{th} path by ρ_i .

Delay: On the same note, different routes imply different path lengths which in turn

cause a different delay for each multipath. Let us denote the delay of i^{th} path by τ_i .

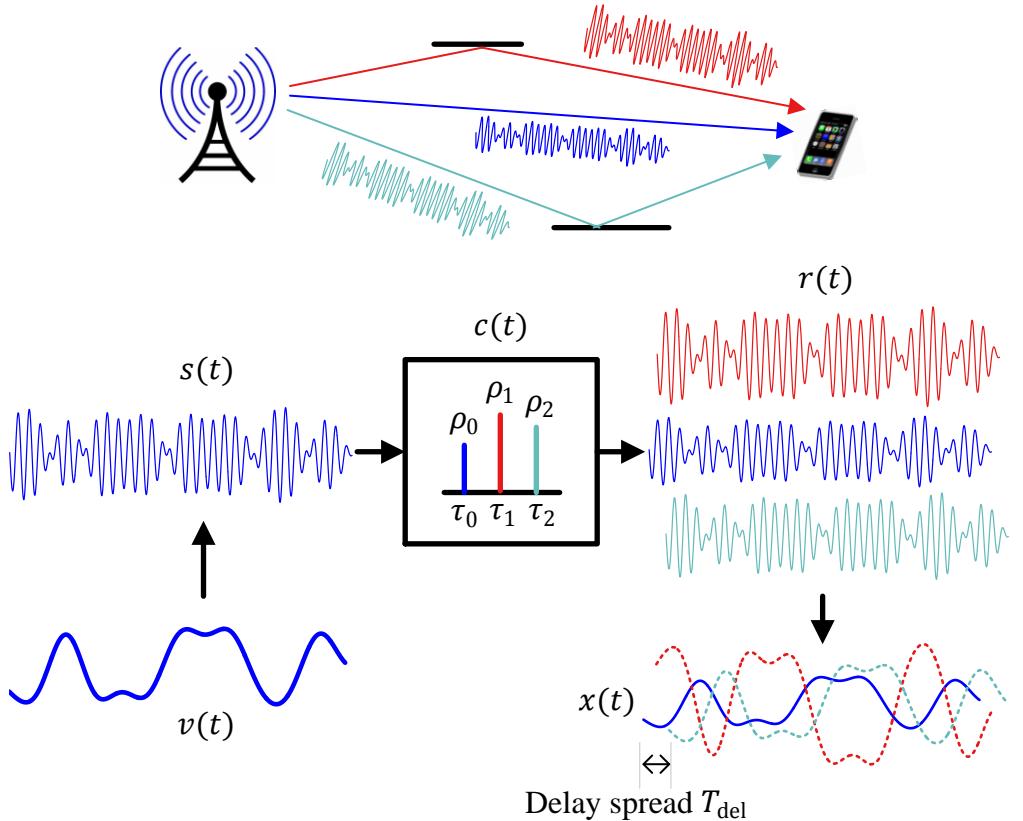


Figure 8.6: Impulse response $c(t)$ of the passband channel consists of N_{MP} impulses with different amplitudes ρ_i and delays τ_i

With the above notation, employing the fact that all impinging signals on the Rx antenna are summed by nature and assuming no motion within the channel, the cumulative waveform is expressed as

$$r(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot s(t - \tau_i) \quad (8.2)$$

Imagine the passband channel as a Linear Time-Invariant (LTI) system with an impulse response $c(t)$. If we write the above equation as

$$r(t) = s(t) * c(t),$$

which expression for $c(t)$ would satisfy Eq (8.2)? Recall that the convolution of a signal with an impulse is the signal itself while a delayed version of a signal is its convolution with a delayed impulse.

$$s(t) * \delta(t) = s(t), \quad s(t) * \delta(t - \tau) = s(t - \tau)$$

Therefore, the passband channel $c(t)$ in Eq (8.2) is discovered to be

$$c(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \delta(t - \tau_i) \quad (8.3)$$

It is evident that convolving $s(t)$ with $c(t)$ yields our Rx signal $r(t)$. The channel impulse response $c(t)$ is drawn in Figure 8.6 for 3 paths with different delays and amplitudes. Notice from the Rx waveforms that there are some similarities and some differences as compared to the scenario when no carrier wave was shown in Figure 8.3b or Figure 8.4b.

Delays: The delay τ_i of each multipath is the same as before. The blue solid line shows the direct path with the shortest delay τ_0 . While this delay is of central significance for ranging and position finding applications, it is immaterial for the purpose of information transfer and hence τ_0 is usually treated as reference time 0 in a wireless receiver system. The other two paths arrive after delays of τ_1 and τ_2 , respectively, from the reference time.

Amplitudes: The amplitudes of all three paths are different as one should expect as a typical case. Their power decays according to the traveled distance and the objects encountered within the route. On the other hand, all copies of similar amplitude were shown in Figure 8.3b or Figure 8.4b to focus on the concept of multipath arriving at the Rx antenna, and not the nature of those multipath. Consequently, keep in mind that this difference here is not due to the carrier wave and previously I drew the two multipath above and below the direct path just for visibility.

Also, the direct path is not necessarily, but only often, received with the highest est power. There can be numerous situations in a wireless channel that a later path has more power such as the Line of Sight (LoS) being blocked or other paths from nearby sources adding constructively at a later time.

Phases: The crucial difference arising from the carrier wave is the change in phase, an example of which is shown in the second multipath, i.e., the one arriving at τ_1 (see the difference in red dotted waveform in $x(t)$ at bottom right of Figure 8.6). This is what makes the wireless channel so interesting and we are going to investigate next how these phases $2\pi F_C \tau_i$ emerge.

Now let us insert the expression for a QAM signal from Eq (8.1) into the Rx waveform in Eq (8.2).

$$\begin{aligned} r(t) &= \sum_{i=0}^{N_{MP}-1} \rho_i \cdot s(t - \tau_i) \\ &= \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \left\{ v_I(t - \tau_i) \sqrt{2} \cos 2\pi F_C (t - \tau_i) - \right. \\ &\quad \left. v_Q(t - \tau_i) \sqrt{2} \sin 2\pi F_C (t - \tau_i) \right\} \end{aligned}$$

Using the identities $\cos A \cos B + \sin A \sin B = \cos(A - B)$ and $\sin A \cos B - \cos A \sin B = \sin(A - B)$,

$$r(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot v_I(t - \tau_i) \sqrt{2} \left(\cos 2\pi F_C t \cos 2\pi F_C \tau_i + \sin 2\pi F_C t \sin 2\pi F_C \tau_i \right) - \\ \sum_{i=0}^{N_{MP}-1} \rho_i \cdot v_Q(t - \tau_i) \sqrt{2} \left(\sin 2\pi F_C t \cos 2\pi F_C \tau_i - \cos 2\pi F_C t \sin 2\pi F_C \tau_i \right)$$

The above equation can be simplified by gathering the common terms.

$$r(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \left\{ v_I(t - \tau_i) \cos 2\pi F_C \tau_i + v_Q(t - \tau_i) \sin 2\pi F_C \tau_i \right\} \sqrt{2} \cos 2\pi F_C t - \\ \sum_{i=0}^{N_{MP}-1} \rho_i \left\{ v_Q(t - \tau_i) \cos 2\pi F_C \tau_i - v_I(t - \tau_i) \sin 2\pi F_C \tau_i \right\} \sqrt{2} \sin 2\pi F_C t \\ = \tilde{v}_I(t) \sqrt{(2)} \cos 2\pi F_C t - \tilde{v}_Q(t) \sqrt{(2)} \sin 2\pi F_C t \quad (8.4)$$

where $\tilde{v}_I(t)$ and $\tilde{v}_Q(t)$ are defined as

$$\begin{aligned} I \rightarrow \quad & \tilde{v}_I(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cos 2\pi F_C \tau_i \cdot v_I(t - \tau_i) + \\ Q \uparrow \quad & \sum_{i=0}^{N_{MP}-1} \rho_i \sin 2\pi F_C \tau_i \cdot v_Q(t - \tau_i) \\ & \tilde{v}_Q(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cos 2\pi F_C \tau_i \cdot v_Q(t - \tau_i) - \\ & \sum_{i=0}^{N_{MP}-1} \rho_i \sin 2\pi F_C \tau_i \cdot v_I(t - \tau_i) \end{aligned}$$

Compared with the QAM expression in Eq (8.1), it is evident that the Tx signal has changed because we have alternative expressions $\tilde{v}_I(t)$ and $\tilde{v}_Q(t)$ in place of $v_I(t)$ and $v_Q(t)$. When the Rx downconverts the above signal by a frequency F_C , the double frequency terms get filtered out. Without applying further trigonometric identities, we can see that the downconverted signal $x(t)$ is the same as $\tilde{v}(t)$ above[†]. In terms of complex signals,

$$x(t) = \tilde{v}(t)$$

with $\tilde{v}_I(t)$ and $\tilde{v}_Q(t)$ as inphase and quadrature components of $\tilde{v}(t)$. Let us rewrite $\tilde{v}_I(t)$ and $\tilde{v}_Q(t)$ as

$$\begin{aligned} I \rightarrow \quad & \tilde{v}_I(t) = \sum_{i=0}^{N_{MP}-1} \gamma_{i,I} \cdot v_I(t - \tau_i) - \sum_{i=0}^{N_{MP}-1} \gamma_{i,Q} \cdot v_Q(t - \tau_i) \\ Q \uparrow \quad & \tilde{v}_Q(t) = \sum_{i=0}^{N_{MP}-1} \gamma_{i,I} \cdot v_Q(t - \tau_i) + \sum_{i=0}^{N_{MP}-1} \gamma_{i,Q} \cdot v_I(t - \tau_i) \end{aligned}$$

Here, γ_i are defined as

$$\begin{aligned} I \rightarrow \quad & \gamma_{i,I} = \rho_i \cdot \cos 2\pi F_C \tau_i \\ Q \uparrow \quad & \gamma_{i,Q} = -\rho_i \cdot \sin 2\pi F_C \tau_i \end{aligned} \quad (8.5)$$

[†]This is because $r(t) = \Re\{\tilde{v}(t) \exp(j2\pi F_C t)\}$ from Eq (8.4) and $x(t) = \text{LPF}\{r(t) \exp(-j2\pi F_C t)\}$.

Applying the multiplication rule of complex signals $I \cdot I - Q \cdot Q$ and $I \cdot Q + Q \cdot I$, we can say that the downconverted signal is equivalent to a summation of N_{MP} complex shaped symbol streams $v(t)$, each delayed by an amount τ_i **and weighted by a complex gain γ_i** . For complex signals, this is expressed as

$$x(t) = \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot v(t - \tau_i) \\ = v(t) * c_B(t) \quad (8.6)$$

From the above equation, it is evident that the baseband (complex) channel response $c_B(t)$ can be expressed as

$$c_B(t) = \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot \delta(t - \tau_i) \quad (8.7)$$

Note 8.3 Delay spread

With the first path chosen as the reference $\tau_0 = 0$, the delay of the last significant path $\tau_{N_{MP}-1}$ is commonly known as **Delay Spread** of the channel T_{Del} . It signifies the amount of time over which the channel is stretched out in time domain (ignoring the paths with amplitudes below the noise floor).

$$T_{\text{Del}} = \tau_{N_{MP}-1} - \tau_0 \quad (8.8)$$

For our example three path channel examined earlier, the delay spread turns out to be τ_2 seconds and shown in Figure 8.6. An alternative definition using root mean square value of the delays is also commonly used. Of particular note is the fact that T_{Del} has nothing to do with the Tx signal or our data rates. Instead, it depends on the particular environment and is the first out of four fundamental quantities through which a wireless channel is usually categorized.

From the definition, the delay spread strongly depends on the propagation environment. For earlier generations of cellular systems, the typical values were large particularly for urban channels. With the increasing traffic demand, the cell sizes keep shrinking thus giving rise to smaller delay spreads. These smaller cell sizes also lead to antennas installed down from the radio towers to building rooftops or street level that reduces the possibility of reflections returning from far away buildings. Furthermore, as we shortly see, the multipath delays vary with time thus leading to delay spread exhibiting a random behaviour.

Next, we have a look at this process in frequency domain.

Frequency Domain: Channel Frequency Response

Let us take the expression for the impulse response of the wireless channel in Eq (8.3) into frequency domain. For this purpose, first recall from Section 1.9 that a right shift of τ_i in a signal changes the phase of its DFT by $-2\pi(k/N)\tau_i$ for each $k = -N/2, \dots, -1, 0, 1, \dots, N/2 - 1$. For a continuous-time case, this is given by

$$\text{Time shift : } s(t - \tau_i) \longrightarrow \text{Phase shift : } -2\pi F \tau_i \quad (8.9)$$

This phase shift being a function of frequency F in $-2\pi F \tau_i$ represents *multiplication of the signal spectrum with a complex sinusoid of inverse period τ_i* (inverse period by definition is frequency; however, we are already in frequency domain and hence I had to devise this expression). This was shown in Eq (1.61) as

$$\text{Shift in one domain} \longrightarrow \text{Multiplication by a complex sinusoid in the other domain}$$

In the context of the wireless channel, the Fourier Transform of

$$c(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \delta(t - \tau_i)$$

consists of N_{MP} complex sinusoids with the expression

$$\begin{aligned} I &\rightarrow C_I(F) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \cos(-2\pi F \tau_i) \\ Q &\uparrow C_Q(F) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \sin(-2\pi F \tau_i) \end{aligned}$$

which comes from the Eq (1.72) that relates a shifted unit impulse to its DFT, a complex sinusoid. Utilizing the identities $\cos(-A) = \cos A$ and $\sin(-A) = -\sin A$,

$$\begin{aligned} I &\rightarrow C_I(F) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \cos 2\pi F \tau_i \\ Q &\uparrow C_Q(F) = - \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \sin 2\pi F \tau_i \end{aligned} \tag{8.10}$$

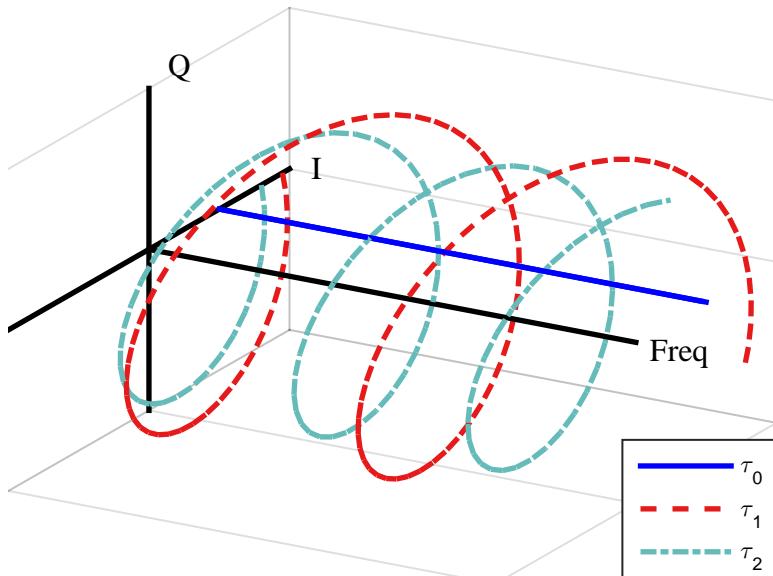


Figure 8.7: Frequency response $C_P(F)$ of the channel consists of N_{MP} complex sinusoids with different amplitudes ρ_i and inverse periods τ_i

This channel frequency response is plotted in Figure 8.7 where only the positive portion of the spectrum is shown for clarity. The inverse period of each complex sinusoid is given by the delay τ_i . Observe that the delay of the direct path τ_0 has been assumed equal to zero and hence the straight line (infinite inverse period). According to the channel impulse response in Figure 8.6, the direct path with delay τ_0 has the lowest magnitude while the second path with delay τ_1 has the largest magnitude.

With the shape of the frequency response in hand, we redraw the time domain Figure 8.6 with frequency domain signals in Figure 8.8. Due to a shift in time domain of these paths, there is a complex sinusoidal multiplication in frequency domain between the signal spectrum and each such complex sinusoid that is the cause of the ISI. Otherwise, when the spectra of the multipath copies get coherently added to the spectrum of the direct path, all replicas arriving at the same frequency boost the signal power.

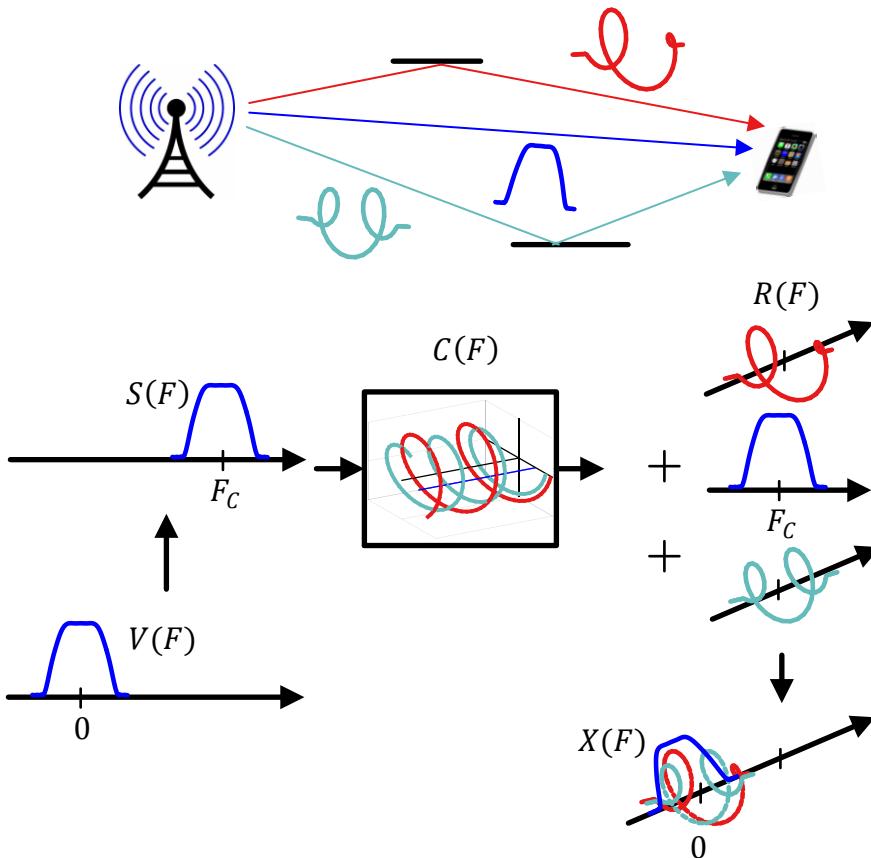


Figure 8.8: Frequency response $C(F)$ of the channel consists of N_{MP} spectra with different amplitudes ρ_i and rotations $2\pi F_C \tau_i$

To see the actual cumulative frequency response, we need to add all of them up as in Eq (8.10). However, we are not interested in how the channel behaves for the huge spectrum ranging from 0 to 100 GHz onwards. Instead, we focus on a few MHz of the channel frequency response around an arbitrary carrier frequency F_C . When

we do that, we can either see the inphase and quadrature components of the channel frequency response or its magnitude and phase spectra. Both of these options are plotted in Figure 8.9.

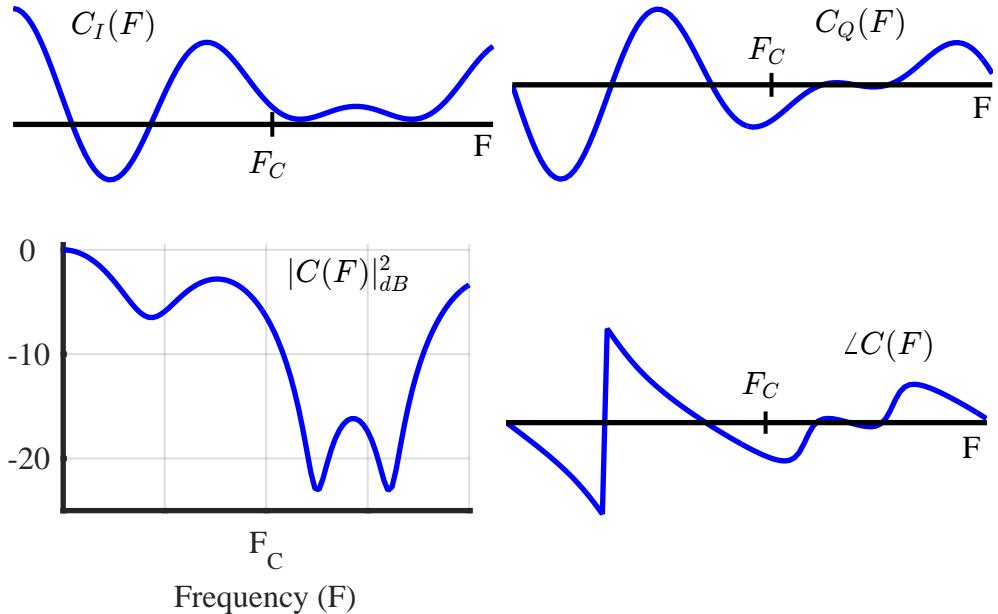


Figure 8.9: Inphase, quadrature, magnitude and phase components of the *cumulative* channel frequency response in Eq (8.10) around a carrier frequency F_C

It is the magnitude response which generates most of the interest since it determines the fate of our Rx signal spectrum, as we shortly see. We can see that the magnitude spectrum of our example channel discriminates among different spectral regions. For some regions, it welcomes the incoming signal and allows them to pass through almost unattenuated while for some other regions, it enforces huge attenuation and any signal within that zone experiences a bad fate. Similar magnitude responses appear in most figures on a wireless channel and *this addition of frequency domain complex sinusoids* is the intuition behind their crests and troughs.

How does this channel treat the incoming signal riding on a carrier wave of frequency F_C ? If the spectrum of the shaped pulse stream $v(t)$ is $V(F)$, then the spectrum $S(F)$ of the upconverted signal is located at frequencies $\pm F_C$ because the spectra of a cosine and sine that are employed by the Tx for upconversion are two impulses at $\pm F_C$. Convolution of $V(F)$ with these two impulses shifts its spectrum at $\pm F_C$.

Next, this signal $s(t)$ gets convolved with the wireless channel with impulse response $c(t)$ consisting of multipath components. This convolution in time domain gives rise to a multiplication of the signal spectrum $S(F)$ with the channel frequency response $C(F)$ of Eq (8.10). With the channel output $R(F)$ being the Rx signal,

$$R(F) = S(F) \cdot C(F) \quad (8.11)$$

The channel output is then downconverted through mixing with a complex sinusoid of frequency F_C . Shifting each term in Eq (8.11) by F_C ,

$$X(F) = R(F + F_C) = S(F + F_C) \cdot C(F + F_C)$$

With the double frequency term filtered out and referring to Figure 8.5, the downconverted signal spectrum is given by

$$X(F) = V(F) \cdot C(F + F_C) = V(F) \cdot C_B(F) \quad (8.12)$$

where $C_B(F)$ is the frequency domain representation of the baseband channel $c_B(t)$.

$$C_B(F) = C(F + F_C)$$

Compare Eq (8.12) of frequency domain multiplication with Eq (8.6) of time domain convolution. Finally, plug in the expression for $C(F)$ from Eq (8.10),

$$\begin{aligned} I &\rightarrow C_{B,I}(F) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \cos 2\pi(F + F_C)\tau_i \\ Q &\uparrow C_{B,Q}(F) = - \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \sin 2\pi(F + F_C)\tau_i \end{aligned} \quad (8.13)$$

The spectrum of the modulated signal along with the channel constituents are plotted in Figure 8.10. A few comments are in order regarding this figure.

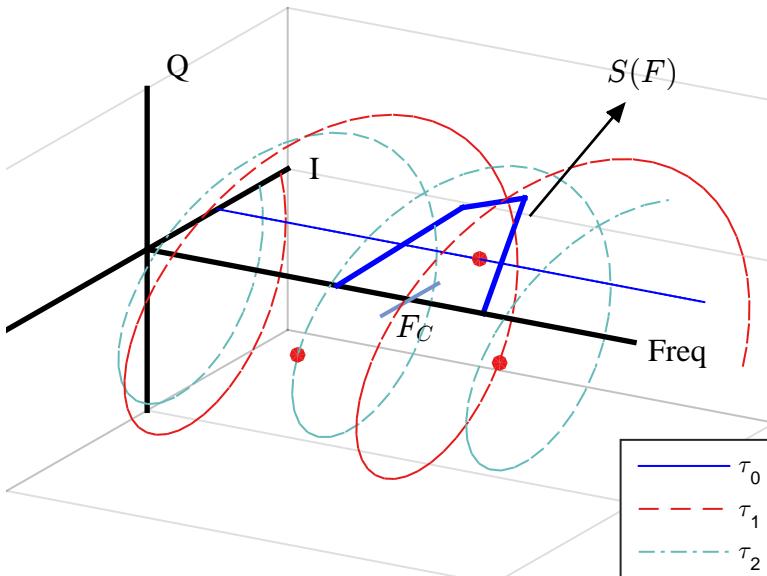


Figure 8.10: Complex multiplication of the channel frequency response components with the modulated signal spectrum. Red dots show the complex channel gain for each path

- The gain component of the first multipath contains no Q part due to making τ_0 our reference time 0, as $\sin 2\pi F \cdot 0 = 0$.
- The points shown as red dots are *the values of the complex sinusoids exactly at F_C forming the complex channel gains* denoted by γ_i as before in Eq (8.5). The

I and Q components of these complex channel gains are

$$\begin{array}{ll} I \rightarrow & \gamma_{i,I} = \rho_i \cdot \cos 2\pi F_C \tau_i \\ Q \uparrow & \gamma_{i,Q} = -\rho_i \cdot \sin 2\pi F_C \tau_i \end{array} \quad (8.14)$$

where the negative sign comes from the clockwise rotation of the complex sinusoids, see Eq (8.10).

- Beware that when the frequency domain product is taken, it is not a simple product of the I and Q spectra of the Tx signal with the I and Q spectra of the channel, respectively. Instead, it is a complex multiplication, i.e., the I component is a difference between products of two aligned-axes terms (i.e., $I \cdot I - Q \cdot Q$), while Q is a sum of products of two cross-axes terms (i.e., $Q \cdot I + I \cdot Q$).

Note 8.4 Channel gain

In the above analysis, we have treated the channel as a linear Finite Impulse Response (FIR) filter. The gain of an FIR filter is its time average or DC value which is represented by the value at $F = 0$. Inserting $F = 0$ in Eq (8.12),

$$X(0) = V(0) \cdot C(F_C) = V(0)C_B(0)$$

For the cumulative channel gain, either plug $F = F_C$ in Eq (8.10) or $F = 0$ in Eq (8.13).

$$\begin{array}{ll} I \rightarrow & C_I(F) \Big|_{F=F_C} = C_I(F_C) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \cos 2\pi F_C \tau_i \\ Q \uparrow & C_Q(F) \Big|_{F=F_C} = C_Q(F_C) = - \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \sin 2\pi F_C \tau_i \end{array} \quad (8.15)$$

So when the channel output in Figure 8.10 is taken from frequency F_C to baseband, the channel gain also shifts from F_C to frequency 0 and hence manifests itself as a function of F_C in the baseband signal. This has been one out of the two most pleasing features exhibited by a wireless channel. We will encounter the other in a time-varying scenario, i.e., when there is any kind of movement within the channel.

To get an idea of what kind of cold equations we are dealing with, let us consider an example.

Example 8.1

A Tx needs to communicate to a Rx at a carrier frequency of 1.9 GHz. It is up against a multipath channel that has two paths with the following gains and delays.

$$\begin{aligned} \rho_0 &= 0.9, & \tau_0 &= 0 \mu\text{s} \\ \rho_1 &= 0.6, & \tau_1 &= 5 \mu\text{s} \end{aligned}$$

Then, the channel impulse response is

$$\begin{aligned} c(t) &= \sum_{i=0}^1 \rho_i \cdot \delta(t - \tau_i) \\ &= 0.9\delta(t) + 0.6\delta(t - 5 \cdot 10^{-6}) \end{aligned}$$

The channel gain from the first path has only an inphase component with an amplitude of $\rho_0 = 0.9$ due to the delay being zero.

$$\gamma_0 = 0.9$$

From Eq (8.14), the channel gain for the second path is given by

$$\begin{aligned} I \rightarrow \quad \gamma_{1,I} &= \rho_1 \cdot \cos(2\pi F_C \cdot \tau_1) \\ &= 0.6 \cdot \cos(2\pi \cdot 1.9 \cdot 10^9 \cdot 5 \cdot 10^{-6}) = 0.6 \end{aligned}$$

$$\begin{aligned} Q \uparrow \quad \gamma_{1,Q} &= -\rho_1 \cdot \sin(2\pi F_C \cdot \tau_1) \\ &= -0.6 \cdot \sin(2\pi \cdot 1.9 \cdot 10^9 \cdot 5 \cdot 10^{-6}) = 0 \end{aligned}$$

Therefore, this path has only an inphase component equal to 0.6 and zero Q component. This happened because $F_C \tau_1$ is, incidentally, equal to a multiple of 2π . Since the Q part of both paths is zero, there is no Q arm of the channel gain.

What happens if the direct path delay τ_0 stays the same but the first path delay τ_1 is *slightly* different by 2 nanoseconds, i.e., $5.002 \mu\text{s}$? Employing the same method as above, the channel gain γ_1 is given by

$$\begin{aligned} I \rightarrow \quad \gamma_{1,I} &= 0.1854 \\ Q \uparrow \quad \gamma_{1,Q} &= 0.5706 \end{aligned}$$

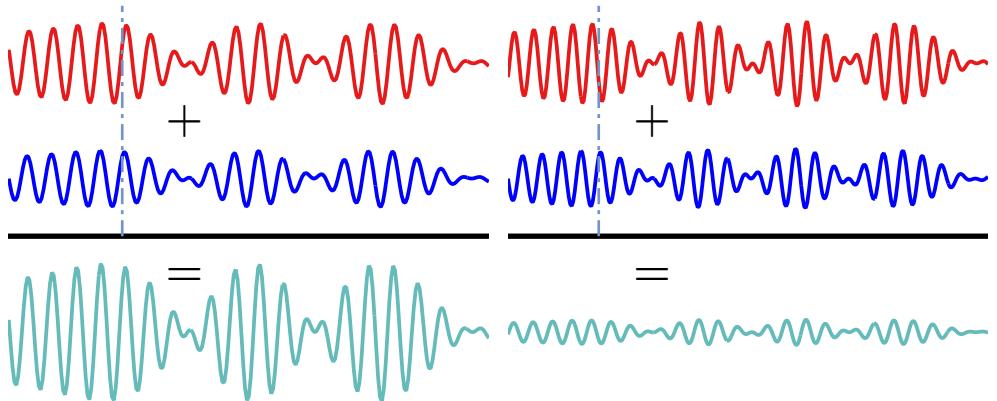
As is usually the case, the energy in the Q rail is not zero! If you draw the inphase/quadrature or magnitude/phase plots of the two channel responses above, you would see how drastic a difference has appeared by a variation of these 2 ns in the path length.

We can infer the following from the above example.

- Since $\sin 2\pi l = 0$ for any integer l , the channel response has a zero Q arm whenever $F_C \tau_i$ is an integer for all paths which is a rare case.
- Even a slight change in the environment causes a significant change in the channel response. Concentrating on the term $2\pi F_C \tau_i$, the phase of each path completes a 2π cycle for each $\tau_i \cdot F_C = 1$.

$$\text{variation in } \tau_i = \frac{1}{F_C} \quad \rightarrow \quad \text{Phase change} = 2\pi$$

In the example above, $1/F_C = 0.5263 \text{ ns}$ which at the speed of electromagnetic wave is a distance of 15.8 cm. In this channel consisting of just two paths, if the second path length is different by a mere 15.8 cm, the two waves add constructively to boost the signal amplitude as shown in Figure 8.11a. The vertical line is drawn to identify the similar phases in the two waveforms.



(a) Constructive interference $\tau F_C \approx \text{integer}$ **(b)** Destructive interference $\tau F_C \approx 0.5 \cdot \text{integer}$

Figure 8.11: Constructive and destructive interference arising from the different delays of multipath

On the other hand, the phase of each path undergoes a shift of π for each $\tau_i \cdot F_C = 1/2$.

$$\text{variation in } \tau_i = \frac{1}{2F_C} \quad \rightarrow \quad \text{Phase change} = \pi$$

In the example above, if the second path length is different by just 7.9 cm, the two waves add destructively to plummet the signal amplitude as shown in Figure 8.11b. The vertical line is drawn to identify the opposite phases in the two waveforms.

Many readers find it easier to understand the interference patterns by drawing the path amplitudes and phases as vectors and then doing vector addition and subtraction. One such example is shown in Figure 8.12 for a relatively mild channel. Later, we will find that the Doppler shift makes these vectors rotate in a random manner.

- The example also shows that the channel response is in general a complex function of frequency and time, even if an inphase only modulation such as BPSK is employed for transmission. Therefore, as we also found out in the phase synchronization case in Chapter 5, a BPSK Rx has to employ a Q arm as well to scavenge energy lost from the I arm.

Note 8.5 Coherence bandwidth

While the magnitude response varies with frequency, the range of frequencies over which it can be assumed approximately constant is termed as *Coherence Bandwidth* B_C of the channel. It is evident that the faster the frequency domain complex sinusoids rotate in Figure 8.7, the smaller the coherence bandwidth would be. Now the inverse periods of such sinusoids are the path delays τ_i . A larger τ_i produces a faster rotation than a smaller τ_i . Coherence bandwidth therefore is inversely proportional to the delay spread

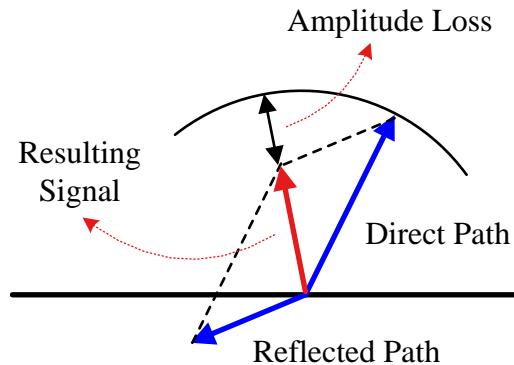


Figure 8.12: A vector sum representation of multipath interference

T_{Del} :

$$B_C \propto \frac{1}{T_{\text{Del}}} \quad (8.16)$$

The coherence bandwidth B_C for our example channel is redrawn as shaded regions in Figure 8.13. If we split the channel magnitude response in discrete components, then the portions within a B_C exhibit a strong magnitude correlation, i.e., their magnitudes cannot attain a value far off from each other. Two such instances are drawn in the figure.

1. When the channel is good, it is good for almost all spectral components within B_C .
2. When a deep fade occurs, it drowns almost the complete region within B_C .

Coherence bandwidth B_C is the second out of four fundamental quantities through which a wireless channel is mainly categorized.

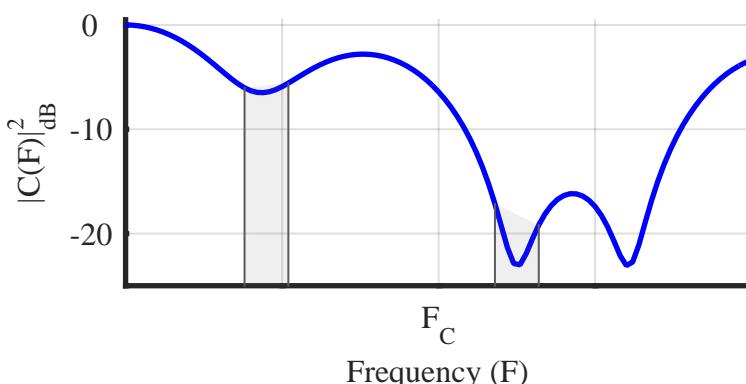


Figure 8.13: Coherence bandwidth B_C of a channel

8.1.3 Frequency Flat and Frequency Selective Fading

To understand different types of fading, again refer to Figure 8.6 that depicts a multi-path channel.

Frequency Flat Fading in Time Domain

A low data rate scenario depicted in Figure 8.3b is redrawn with the inclusion of the carrier wave in Figure 8.14 where the first and second multipath are shown above and below the actual signal for clarity. Also, the vertical line is drawn to identify the phases in the waveforms at the same point in time.

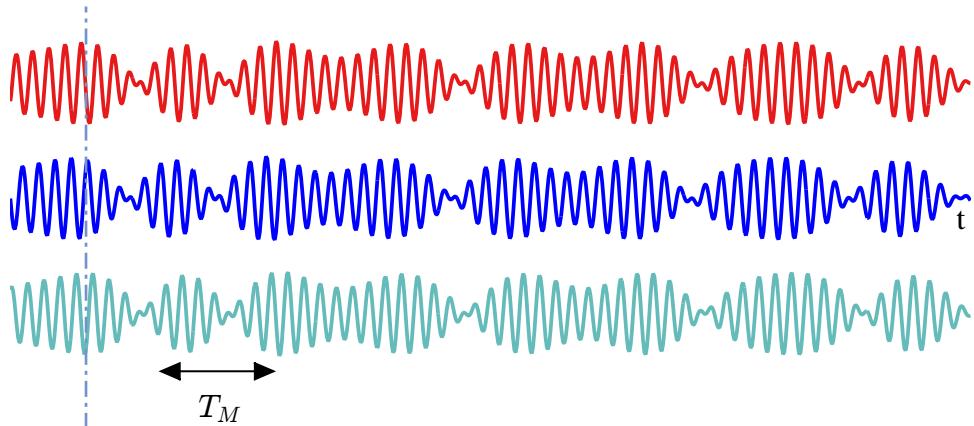


Figure 8.14: First and second multipath components arriving 0.6 and 1.1 μs after the direct path, respectively. With the carrier wave present, the phases are different but the data symbol values are almost the same

As plotted in Figure 8.11 before, the phases of these paths might add constructively or destructively or somewhere in between. However, due to the short delay of the second and third paths with respect to the first one, the data symbol values of these paths are still (almost) the same and hence *there is very low Inter-Symbol Interference (ISI) that shows up on top of those path phases*.

In this sense, the data symbols are all in this together. Depending on the phases of the paths, all the data symbols either experience the same boost or suffer a similar attenuation. The fate of the whole signal is the same.

Before we find in the frequency domain description the rationale behind the terminology 'frequency flat', we explore the mathematics behind it. Recall from Eq (8.6) that the time domain description of the downconverted signal is given by

$$x(t) = \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot v(t - \tau_i)$$

When τ_i are fairly close to each other, the Rx signal replicas can be taken out as a common factor due to their similarity $v(t - \tau_i) \approx v(t)$.

$$x(t) \approx \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot v(t) = \gamma \cdot v(t) \quad (8.17)$$

where γ is a complex constant[†] expressed as

$$\gamma = \sum_{i=0}^{N_{MP}-1} \gamma_i \quad (8.18)$$

and the channel is termed as a frequency flat fading channel. Equation (8.17) reinforces the concept of the whole signal experiencing the same fate decided by γ . Accordingly, the channel impulse response mainly consists of a single impulse as drawn in Figure 8.15.

A question at this stage is that how late the delays can be from the first path for a channel to be categorized as frequency flat. For this purpose, Figure 8.14 reveals that this is directly connected with the duration of a symbol, i.e., the symbol time T_M . As long as the last arriving path delay $\tau_{N_{MP}-1}$ is still a fraction of a symbol time T_M , the ISI effects are minimal and the signal undergoes flat fading.

Recall from Note 8.3 that the delay of the last significant path $\tau_{N_{MP}-1}$ with respect to the first path is the channel delay spread T_{Del} . An exact number for the delay spread to ensure flat fading is obviously chosen by system designers according to the maximum tolerable deterioration. As an example, 10% of the symbol duration T_M is commonly taken as a reasonable number. Hence, a frequency flat fading implies

Time domain condition for frequency flat fading

$$T_{\text{Del}} < 0.1T_M \quad (8.19)$$



Figure 8.15: The impulse response for a frequency flat fading channel consists of a single impulse

Frequency Flat Fading in Frequency Domain

In frequency domain, recognizing the frequency flat fading and determining the channel condition to ensure it is a straightforward task. We start with having another look at Figure 8.8 which plots the frequency domain representation of the signal multipath components. The spectra of all incoming signal paths are the same but arrive with different delays that gives rise to complex sinusoids with different inverse periods. Clearly, when the delays τ_i are small, or which is essentially the same, the delay spread T_{Del} is small, not much rotation occurs across the signal bandwidth thus leading to a relatively flat gain. Although the channel response in this case is not a function of frequency F , their exact cumulative sum depends on the carrier frequency F_C , that can be large or small.

A constant function with respect to frequency F implies that from Eq (8.12), we can write the downconverted signal $X(F) = R(F + F_C)$ after lowpass filtering as

$$X(F) = C(F_C) \cdot V(F)$$

[†]A reminder: there is no movement anywhere within the channel as of now.

Interestingly, taking the Fourier Transform of $x(t)$ in Eq (8.17),

$$X(F) = \gamma \cdot V(F)$$

which leads to

$$\gamma = C(F_C)$$

i.e., the channel reduces to almost a single complex constant, the same channel gain encountered in Eq (8.15). In a regular multipath scenario, each complex path gain gets multiplied with a corresponding copy of the delayed signal before adding into the final expression. On the other hand, in frequency flat fading, all complex path gains are summed to form a single multiplicative factor.

A frequency flat fading scenario is drawn in Figure 8.16 as an example. Instead of the modulated data, a simple square-root Nyquist pulse with excess bandwidth $\alpha = 0.5$ is plotted for visual clarity. Notice that the whole signal bandwidth almost fits within the coherence bandwidth of the channel and hence the Rx magnitude depends on the channel attenuation at that frequency which is shown as a blue dot at F_C . Clearly, this blue dot representing the channel attenuation is nothing but $C(F_C)$.

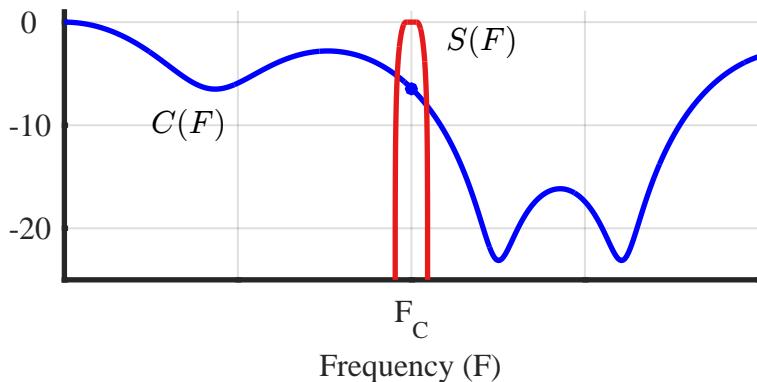


Figure 8.16: Signal bandwidth B_{QAM} within the channel coherence bandwidth B_C gives rise to frequency flat fading. The term flat implies multiplication of the spectrum with (almost) a single attenuation factor shown as the blue dot here

Assuming a symbol rate $1/T_M$ and a Nyquist filter with excess bandwidth α , the bandwidth of a QAM modulated signal is expressed as

$$B_{\text{QAM}} = \frac{1 + \alpha}{T_M}$$

From the discussion above, we can write the frequency domain condition for frequency flat fading as

Frequency domain condition for frequency flat fading

$$B_C > \frac{1}{T_M} \approx B_{\text{QAM}} \quad (8.20)$$

Interestingly, inverting the time domain condition in Eq (8.19) and the definition of coherence bandwidth $B_C \propto 1/T_{\text{Del}}$ yields

$$B_C > \frac{10}{T_M}$$

This is what creates ambiguity and the constants are ignored due to the absence of an exact such number linking T_{Del} and B_C . Next, we explore the topic of frequency selective fading.

Frequency Selective Fading in Time Domain

In this scenario, a high data rate signal depicted in Figure 8.4b is redrawn with the inclusion of the carrier wave in Figure 8.17 where the first and second multipath are shown above and below the actual signal for clarity.

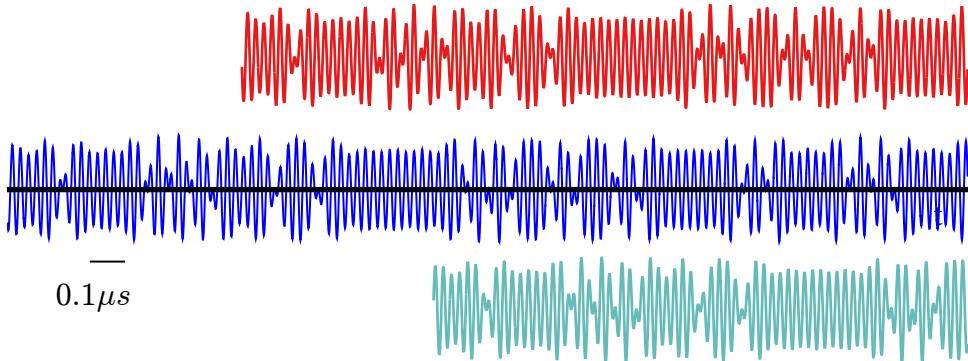


Figure 8.17: First and second multipath components arriving 0.6 and 1.1 μs after the direct path, respectively. With the carrier wave present, the phases are different and the data symbol values randomly add to form the cumulative signal as well

As plotted in Figure 8.11, the phases of these paths might add constructively or destructively or somewhere in between. Moreover, due to the long delay of the second and third paths with respect to the first one, the data symbol values of these paths adding up at random instants generate *a tremendous amount of Inter-Symbol Interference (ISI) that shows up on top of those path phases*. This happens because the random data symbols get summed up into other symbols long after they were actually transmitted. In this sense, there are two phenomenon, and not one, that contribute towards the constructive and destructive interference:

1. path delays τ_i , and
2. symbol values $a[m]$

From Eq (8.3), the time domain description of such a signal is given by

$$c(t) = \sum_{i=0}^{N_{MP}-1} \rho_i \cdot \delta(t - \tau_i)$$

and the channel is termed as a frequency selective fading channel. Accordingly, the channel impulse response is composed of several impulses as drawn in Figure 8.18.

Before we find in the frequency domain description the rationale behind the terminology 'frequency selective', a natural progression of the above description is that as long as the last arriving path delay $\tau_{N_{MP}-1}$ is received after a symbol time T_M , the data symbols add up in a random fashion to produce ISI and the signal undergoes selective fading.

Hence, frequency selective fading implies

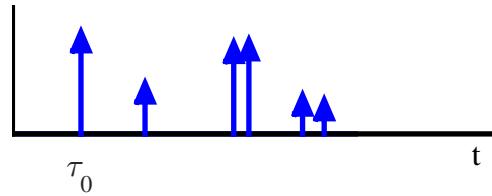


Figure 8.18: The impulse response for a frequency selective fading channel consists of several impulses

Time domain condition for frequency selective fading

$$T_{\text{Del}} > T_M \quad (8.21)$$

Frequency Selective Fading in Frequency Domain

As before, recognizing the frequency selective fading and determining the channel condition to ensure it is a straightforward task in frequency domain. The spectra of all incoming signal paths are the same but with different delays that give rise to complex sinusoids with different inverse periods. When these inverse periods are large (due to long time delays), a faster rotation occurs across the signal bandwidth thus leading to spectral selectivity. A larger delay spread implies increased frequency selectivity and a smaller coherence bandwidth.

We can exploit the relation between the delay spread T_{Del} and coherence bandwidth B_C . Inverting Eq (8.21) yields

Frequency domain condition for frequency selective fading

$$B_C < \frac{1}{T_M} \approx B_{\text{QAM}}$$

where the constants are ignored due to the absence of an exact such number linking T_{Del} and B_C . A frequency selective fading scenario is drawn in Figure 8.19 as an example. Instead of the modulated data, a simple square-root Nyquist pulse with excess bandwidth $\alpha = 0.5$ is plotted for visual clarity and the shaded box represents the coherence bandwidth B_C .

Notice that due to the high data rate and hence large signal bandwidth, the whole bandwidth extends beyond the coherence bandwidth of the channel and hence the Rx magnitude is distributed across a wide spectral region. As the term selectivity implies, when some part of the signal spectrum is in a deep fade, the rest of the portion is not and hence we are able to achieve *frequency diversity* through this selective behaviour if we employ an equalizer at the Rx[†].

[†]Finally, when there are many objects in the surroundings that cause several paths to reach the Rx from different directions and there is no single dominant component, then the envelope of the channel is said to have a Rayleigh distribution giving rise to the term *Rayleigh fading*. In the presence of a line of sight component, it turns into a *Rician fading* that represents a non-zero mean.

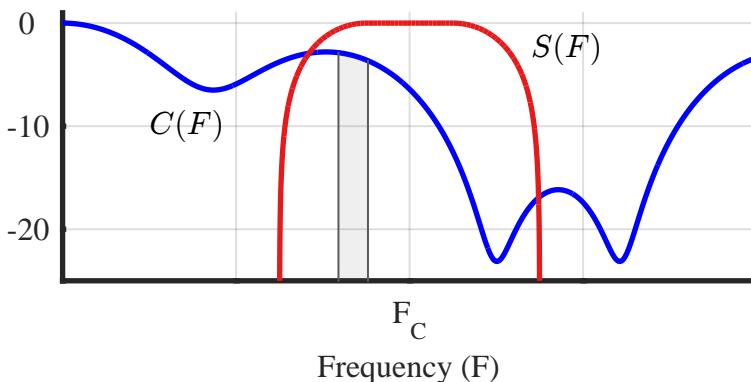


Figure 8.19: Signal bandwidth B_{QAM} extending beyond the channel coherence bandwidth B_C gives rise to frequency selective fading. The term selective implies discriminatory treatment of different portions of the signal spectrum by the channel

8.1.4 Doppler Shift: A Deceptive Villain

Until now, the Tx, the Rx and everything else in the environment was stationary. With the introduction of motion in the channel, interesting things happen due to a phenomenon known as *Doppler shift*, a concept taught in high school physics. Some people have actually paid 300 dollars to learn this concept from the sirens of the police car approaching behind them for a ticket. We first examine what the Doppler shift is and analyze later how it affects the reception of the signal.

A Single Path

Imagine an unmodulated carrier wave $s(t) = \cos 2\pi F_C t$ impinging on the antenna of a mobile in a parked car as drawn in Figure 8.20. The signal arrives at the Rx τ_0 seconds later when the electromagnetic wave travels a distance of d_0 m at a speed of approximately $c = 3 \times 10^8$ m/s. From the time reference of the Tx, this signal is a delay (a right shift) and can be written as

$$\rho_0 \cos 2\pi F_C (t - \tau_0) = \rho_0 \cos 2\pi F_C \left(t - \frac{d_0}{c} \right)$$

where $\tau_0 = d_0/c$.

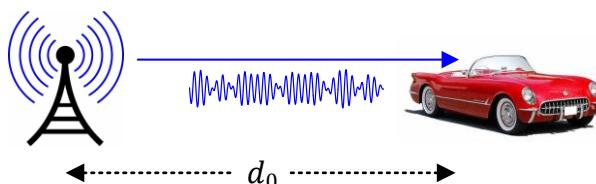


Figure 8.20: Movement of either the Tx, the Rx or any object within the channel causes a Doppler shift in the signal frequency

Now imagine that the car is started and is driven at a constant velocity of v m/s directly opposite to the antenna. For all further times t , the electromagnetic wave that had to travel d_0 m now has to cover a distance of $d_0 + vt$.

$$\begin{aligned}\rho_0 \cos 2\pi F_C \left(t - \frac{d_0 + vt}{c} \right) &= \rho_0 \cos 2\pi F_C \left(t - \frac{v}{c}t - \frac{d_0}{c} \right) \\ &= \rho_0 \cos 2\pi \left\{ \left(F_C - \frac{v}{c}F_C \right) t - F_C \tau_0 \right\}\end{aligned}\quad (8.22)$$

Some comments are now in order.

- As encountered before, the term $-2\pi F_C \tau_0$ is a constant phase shift that arises due to the delay between the Tx and the Rx.
- The frequency of a signal is the term that appears with variable t , except the factor 2π . We conclude that the new frequency of the signal is

$$F_C^{\text{new}} = F_C - \frac{v}{c}F_C$$

Consequently, there is a shift in frequency which is known as *Doppler shift*. Here, the Doppler shift, also commonly known as Doppler frequency F_D , is

$$F_D = -\frac{v}{c}F_C \quad (8.23)$$

- From the above expression, the Doppler shift depends on the carrier frequency F_C : the higher the frequency, the higher the Doppler shift and vice versa.
- The Doppler shift also depends on the velocity of the Rx antenna v . In fact, rewriting the relation as

$$\frac{F_D}{F_C} = -\frac{v}{c}$$

reveals that as a fraction of carrier frequency F_C , it is just a ratio of the velocities between the mobile and the electromagnetic wave.

- Recall that the expression $d_0 + vt$, i.e., an increasing distance, implies that the car in Figure 8.20 started moving away from the Tx antenna at an angle of π radians. Therefore, the wave has to travel a larger distance and hence the minus sign with the Doppler shift. We can write the actual shift in frequency as

$$-\frac{v}{c}F_C = \frac{v \cos \pi}{c}F_C$$

Subsequently, for a general angle ϕ with respect to the axis of propagation of the wave, the Doppler shift is written as

$$F_D = \frac{v \cos \phi}{c}F_C = F_{D,\max} \cos \phi \quad (8.24)$$

In other words, we only take into account the velocity component in the direction of arrival of the wave[†].

As a sanity check, let us run some quick numbers now. If the car is being driven at a speed close to the speed of light c , then the Doppler frequency is $-F_C$ from the above expression and the signal frequency at the Rx is $F_C - F_C = 0$. Of course, it is evident from Figure 8.20 that an electromagnetic wave cannot catch the vehicle traveling at the same speed.

For a realistic example, if the car is moving at a speed of 60 km/hr (16.67 m/s) exactly in a line away from the Tx (so $\phi = \pi$) and the transmission is taking place at a carrier frequency of $F_C = 1.9$ GHz, then the Doppler frequency turns out to be

$$F_D = -\frac{16.67}{3 \cdot 10^8} \cdot 1.9 \cdot 10^9 \approx -105 \text{ Hz}$$

The Doppler shift F_D is so small that even a simple Phase Locked Loop (PLL) would easily track it down! As a comparison, assume the rating for the transceiver crystal as ± 20 ppm. The maximum carrier frequency offset at the Tx or Rx was calculated in Section 6.1 to be

$$\pm \frac{20}{10^6} \times 1.9 \times 10^9 = \pm 38000 \text{ Hz}$$

with the worst case scenario being a carrier frequency offset of 76000 Hz. This value is much larger than a mere 105 Hz Doppler shift found above. Then, the question is why the Doppler shift has such a detrimental effect on the Rx signal bit error rate. We seek the answer to this question next.

Two Paths

In Figure 8.20, the car was moving at a speed of v m/s in a straight line away from the Tx antenna and receiving only the direct path. Now in addition to the direct path, there is a multipath copy reflected from the pole of a birdhouse as in Figure 8.21. The birdhouse is situated directly opposite to the tower along the edge of the road where the car is heading. At a time when the distance of the mobile from the Tx is d_0 m, its distance from the birdhouse is d_{RB} m, where the subscript RB denotes Rx to birdhouse.

Within a time duration t , the direct path signal travels a distance equal to $d_0 + vt$ meters. However, the multipath arrives at the Rx after covering a total of

$$\underbrace{d_0 + vt + d_{RB} - vt}_{\text{Tx to birdhouse}} + \underbrace{d_{RB} - vt}_{\text{birdhouse to Rx}} = d_0 + 2d_{RB} - vt = d_1 - vt$$

where $d_1 = d_0 + 2d_{RB}$ that is covered in a time duration $\tau_1 = d_1/c$. As a consequence,

[†]Many scientists write the relation using $c = F_C \lambda_C$ where λ_C is the carrier wavelength. Then, the Doppler shift becomes

$$F_D = \frac{v \cos \phi}{c} F_C = \frac{v \cos \phi}{F_C \lambda_C} F_C = \frac{v \cos \phi}{\lambda_C}$$

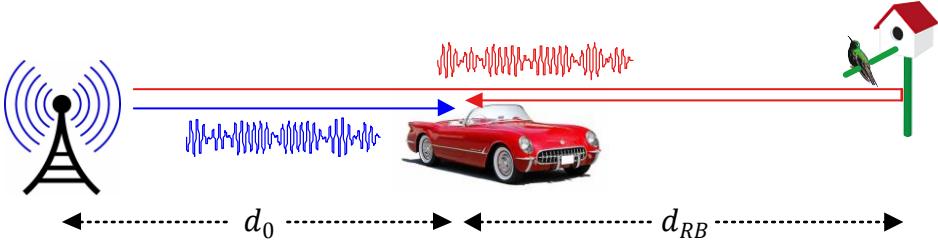


Figure 8.21: A reflection within a non-stationary channel causes the addition of two Doppler shifted signals

the multipath signal can be expressed as

$$\begin{aligned} \rho_1 \cos 2\pi F_C \left(t - \frac{d_1 - vt}{c} \right) &= \rho_1 \cos 2\pi \left\{ \left(F_C + \frac{\nu}{c} F_C \right) t - F_C \frac{d_1}{c} \right\} \\ &= \rho_1 \cos 2\pi \left\{ \left(F_C + \frac{\nu}{c} F_C \right) t - F_C \tau_1 \right\} \end{aligned} \quad (8.25)$$

As a result, the Doppler shift of this multipath component is

$$F_D = +\frac{\nu}{c} F_C$$

Given that the transmitted signal $s(t)$ was an unmodulated carrier wave and there are two paths within the channel in our example, the overall received signal $r(t)$ can be written as

$$\begin{aligned} r(t) &= \sum_{i=0}^{N_{MP}-1} \rho_i \cdot s(t - \tau_i) \\ &= \rho_0 \cos 2\pi F_C (t - \tau_0) + \rho_1 \cos 2\pi F_C (t - \tau_1) \end{aligned}$$

Plugging in the values from Eq (8.22) and Eq (8.25) arising from the motion,

$$\begin{aligned} r(t) &= \rho_0 \cos 2\pi \left\{ \left(F_C - \frac{\nu}{c} F_C \right) t - F_C \tau_0 \right\} + \\ &\quad \rho_1 \cos 2\pi \left\{ \left(F_C + \frac{\nu}{c} F_C \right) t - F_C \tau_1 \right\} \end{aligned} \quad (8.26)$$

We can infer that the Rx signal is composed of two unmodulated carrier waves, one with a frequency of $F_C(1 - \nu/c)$ and the other with a frequency of $F_C(1 + \nu/c)$.

To realize what the shape of the Rx signal looks like, imagine the car approaching really close to the birdhouse. In this case, the delays as well as the large scale attenuation of the direct and the multipath component are almost the same, i.e.,

$$\tau_0 \approx \tau_1 \quad \text{and} \quad \rho_0 \approx \rho_1$$

Then, the Rx signal in Eq (8.26) takes the form

$$\begin{aligned} r(t) &= \rho_0 \cos 2\pi \left\{ \underbrace{\left(F_C - \frac{\nu}{c} F_C \right) t - F_C \tau_0}_{F_1} \right\} + \\ &\quad \rho_0 \cos 2\pi \left\{ \underbrace{\left(F_C + \frac{\nu}{c} F_C \right) t - F_C \tau_0}_{F_2} \right\} \end{aligned}$$

Using the identity

$$\cos A + \cos B = 2 \cos\left(\frac{A+B}{2}\right) \cos\left(\frac{A-B}{2}\right)$$

the Rx signal becomes

$$\begin{aligned} r(t) &= 2\rho_0 \cos 2\pi \left(\frac{F_2 + F_1}{2} t - F_C \tau_0 \right) \cos 2\pi \left(\frac{F_2 - F_1}{2} \right) t \\ &= 2\rho_0 \underbrace{\cos 2\pi (F_C t - F_C \tau_0)}_{\approx \text{GHz}} \underbrace{\cos 2\pi \left(+ \frac{v}{c} F_C t \right)}_{\approx \text{Hz}} \end{aligned} \quad (8.27)$$

We conclude that the summation of the multipath component at frequency $F_C + F_D$ to the direct path at frequency $F_C - F_D$ generates *a product of two sinusoids*, one at the order of the carrier frequency (GHz in typical wireless communications) and the other at a few tens of Hz (around 200 Hz for highway speeds and much less otherwise). This low frequency component rides on the high frequency component and creates crests and troughs as drawn in Figure 8.22.

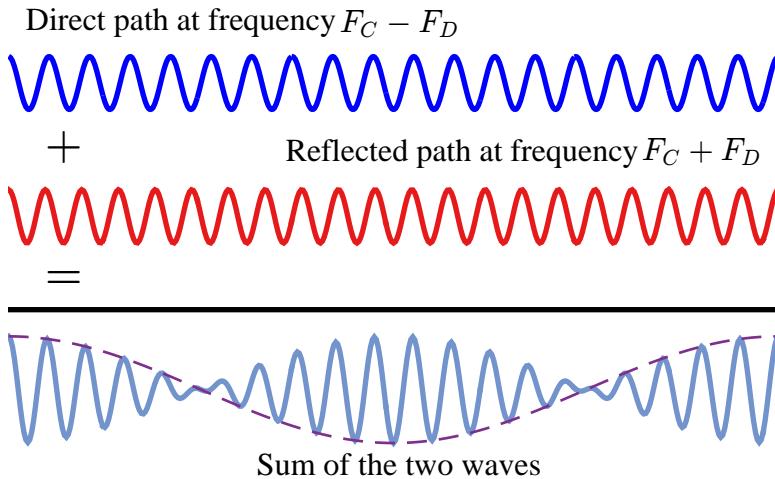


Figure 8.22: The multipath component at frequency $F_C + F_D$ added to the direct path at frequency $F_C - F_D$

In the chosen example, the ratio of the carrier frequency F_C to the Doppler frequency F_D is chosen as 20 which produces 20 cycles of the high frequency component in the sum waveform for a full period of the Doppler frequency waveform. This sum waveform, instead of having a constant envelope, now exhibits fading dips that can drown any amplitude modulation riding the carrier wave.

This is very similar to the *beating effect* in acoustics where it is defined as an interference pattern between two sounds of *slightly* different frequencies. The frequency of this beating envelope is the frequency difference between the two carrier waves which is equal to the difference between the two Doppler shifts. Musicians utilize this concept to periodically alternate the sound between soft and loud.

Intuitive Understanding of the Doppler Shift

Figure 8.22 was drawn by using a mathematical derivation. For an intuitive understanding of why this Doppler shift occurs, it is redrawn in Figure 8.23 to highlight this perspective.

First, recall that the ratio of the carrier frequency F_C to the Doppler frequency F_D is chosen as 20. Now observe the peaks of the two participating carrier waves, particularly during the second and third cycles and notice a slow shift in their phases represented by the shaded boxes. These shaded boxes – depicting the slowly varying envelope – expand until the phase difference reaches π where the envelope becomes zero at cycle 5.

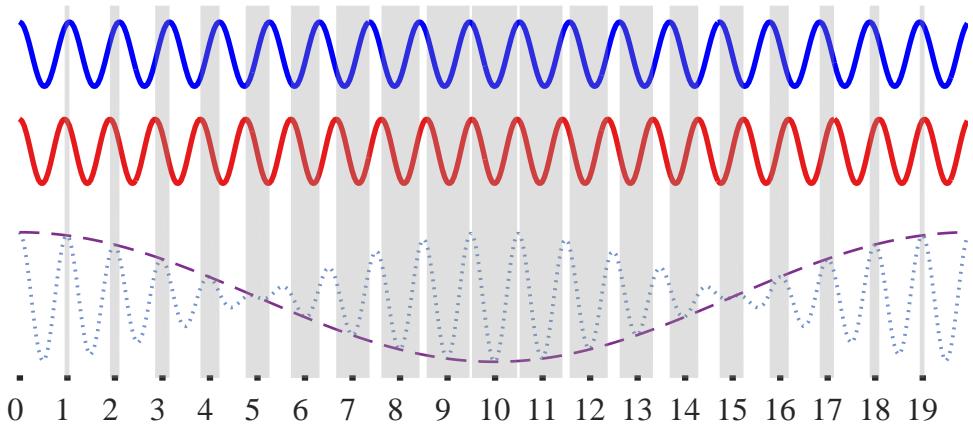


Figure 8.23: An intuitive understanding of why Doppler shift occurs. Look at the peaks of the two carrier waves particularly during the second and third cycles and notice the slow change in their phases (represented by the shaded boxes) corresponding to the envelope

Subsequently, it reaches its minimum at cycle 10 when the two carrier waves are completely in phase but at their negative peaks. The Doppler frequency in the form of this envelope then approaches an out of phase instant at cycle 15 and completes a full rotation at cycle 20. The high frequency waveform is drawn as dotted line in Figure 8.23 for this purpose. In general, $2\pi t(F_2 - F_1)/2$ is the waveform riding on the carrier, see Eq (8.27). Consequently, the envelope completes half a cycle (from a zero to the next zero) in

$$\frac{1}{2} \cdot \frac{2}{F_2 - F_1} = \frac{1}{F_2 - F_1} \text{ seconds} \quad (8.28)$$

where

$$\frac{F_2 - F_1}{2} = \frac{F_C + F_D - (F_C - F_D)}{2} = F_D$$

in this example and hence confirmed.

The important point to understand is that this kind of peaks and valleys exhibited by the envelope is a direct result of the Rx moving through the environment where these carrier waves are summing up. Therefore, the dynamics of this phenomenon are governed by the Rx velocity (i.e., speed and direction) itself.

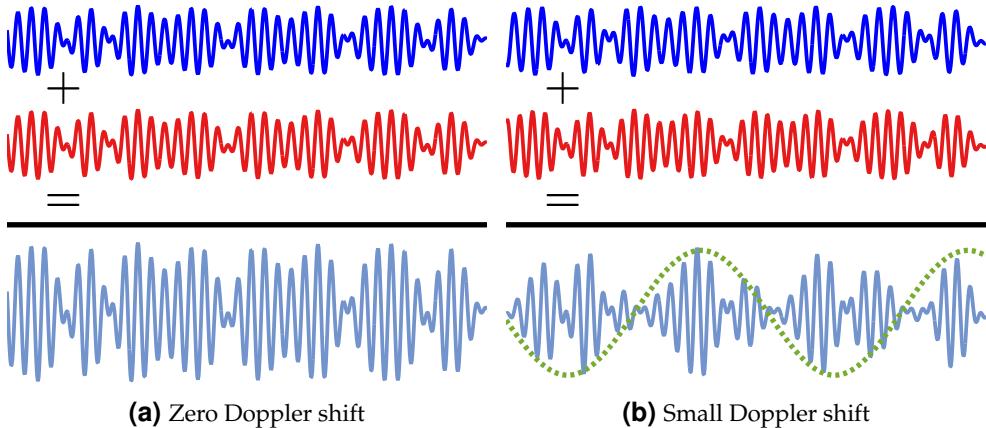


Figure 8.24: Effect of the Doppler shift on the modulated data

Multiple Paths

Now we are ready to encounter the question we asked earlier: why is the Doppler shift such a deceptive villain when it is so small that even a simple Phase Locked Loop (PLL) can easily track it down?

To see the effect of this Doppler shift on the actual modulated data, consider Figure 8.24 where both zero and a small Doppler affected multipath are drawn for a comparison. Notice the fading dips in the sum waveform whose impact is more pronounced in the regions where no data transition occurs.

As far a single path is concerned, the Doppler shift F_D acts on the data in the same way as a carrier frequency offset F_Δ , i.e., it gets added to or subtracted from the carrier frequency F_C for a Rx waveform to modify the frequency as $F_C \pm F_D$. As a result, there is no loss of energy for a small F_D or F_Δ since any part of energy lost in the I arm can be recovered from the Q arm and vice versa.

Here in the case of two paths, the envelope with Doppler frequency F_D is being superimposed on the carrier wave at frequency F_C in the form of a product. For this reason, the Doppler envelope from the two paths actually drowns the signal and there is no way to recover except providing the system with another copy of the signal through some kind of diversity.

We have seen above that with two multipath components, there is a low frequency wave riding on the carrier wave and creating periodic peaks and dips. When a large number of such multipath components – each having its own unique Doppler shift due to arriving from a different angle – superimpose on each other at the Rx antenna, a random pattern of peaks and valleys is formed as drawn in Figure 8.25. While it does not look so, the figure is very similar to Figure 8.22 in 3 dimensions and multiple contributing carrier waves.

When the car heads into such a channel, it encounters fading peaks and valleys throughout the journey. You can consider it as a blanket on the top of the carrier oscillations. Unbeknownst to the driver, she is drowning and ascending into these Rx power curves within an ocean of RF waves. Techniques must be devised to mitigate this effect of time-varying channel on the Rx signal. To be precise, this represents a time-varying gain in a frequency flat fading channel and a time-varying impulse response in a frequency selective fading channel.

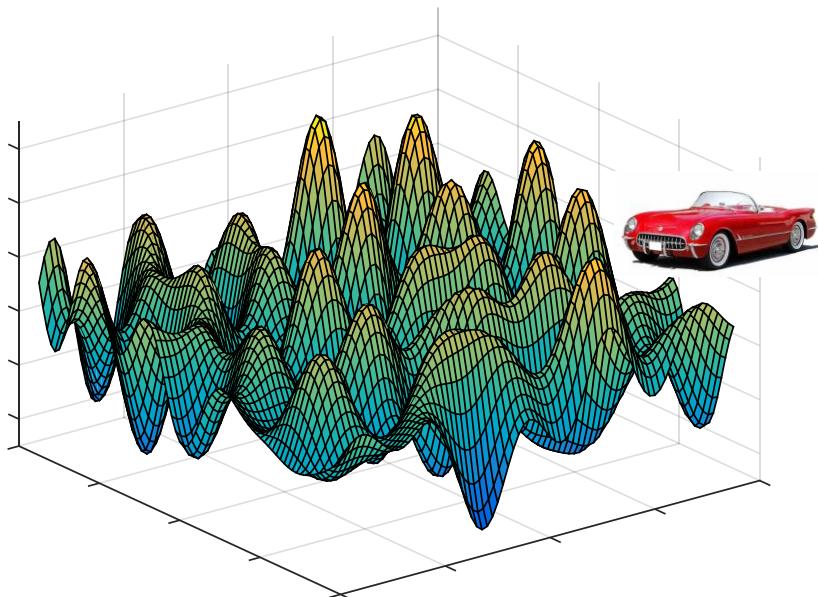


Figure 8.25: Peaks and valleys formed due to the summation of a large number of multipath components arriving at the Rx, each with its own unique Doppler shift

Note 8.6 Movement is good

With so many peaks and valleys, the channel in Figure 8.25 looks intimidating. This is a natural human tendency to resist new or uneven phenomena. However, many a times, such scenarios bring benefits of their own. In the context of wireless communications, multipath copies of the signal were considered disastrous until we figured out that nature is providing us with free copies of the same signal. This frequency selective channel can be exploited with the help of an equalizer to achieve frequency diversity. Similarly, multiple antenna (MIMO: Multiple Input Multiple Output) systems benefit from a rich multipath environment to exploit spatial diversity and turn one channel between the Tx and the Rx into many parallel independent channels in the form of spatial multiplexing.

On the same note, channel variations due to the Doppler shift are advantageous to a mobile Rx. If the person is located at a point of a deep fade due to the positions of the surrounding (and stationary) objects, this fading dip can be infinitely long unless the Rx moves to come out of it. A mobile device traverses these peaks and valleys to average out the effects of the fading and keep connected all the time.

It seems that a better strategy is to choose a peak point to get the best signal strength forever. Nonetheless, remember that the channel changes not only due to the Rx (or Tx) motion but also due to the motion of the surrounding objects. Then, the chosen peak is only as good as the rate at which the objects are moving around. In fact, this has happened to many big corporations in the past. They climbed to a peak in a particular market to become a giant and thought that they would stay there forever. However, there was movement sometimes from seemingly unrelated events and sometimes from other small companies that changed the ‘channel’ casting a deep fade on their fates.

Ref. [33] summarizes the above findings through the following two equivalent approaches for obtaining the fading rate by superimposing two incident waves.

1. Plot the resulting interference pattern depicting the mountains and valleys and count the number of fading dips per second that a Rx sees when moving through the pattern.
2. Determine the fading rate from the beat frequency, i.e., the difference of the Doppler shifts of the participating carrier waves.

8.1.5 What Happens When Motion Enters the Picture

After understanding the effect of Doppler shift arising from multiple paths, we consider the overall effect of motion in the channel on the Rx signal. However, motion introduces time variations and hence our previous concept of a Linear Time-Invariant (LTI) system introduced in Chapter 2 – which gives rise to convolution in time domain and multiplication in frequency domain – does not apply here. The question is how to proceed in this scenario?

For this purpose, we first assume that the Doppler frequencies within the channel stay the same, i.e., all the motions within the channel are occurring at a uniform rate. For example, the car driven away from the Tx antenna has a constant velocity with respect to the direction of wave propagation. Obviously, this is true only on a small enough time scale, just like a channel with zero motion in the time-invariant case was true for a small enough duration. Otherwise, we would have introduced a concept of uniform acceleration and the carrier waves would have a quadratic (chirp) component in addition to the linear (frequency) part[†].

Note 8.7 Isolating the motion

Each multipath arrives at the Rx with a unique amplitude and a phase shift owing to a distinct path length. To isolate the effects of motion on the Rx signal, we assume here that the differences between the path lengths are negligible, even at GHz frequencies. Otherwise, we will have to include time related distortions such as ISI and corresponding rotations in frequency domain here.

This assumption is similar to a stationary channel in the previous section. Simultaneous time and motion induced distortions on the Rx signal can be seen by combining the delay and Doppler effects together.

Most of what follows is already understood in the concept of time domain as explained in Section 8.1.2. Therefore, we will quickly review some concepts and reinterpret some figures without going into deep technical or mathematical details. I recommend you to again look at Figure 8.5 depicting a QAM Tx and Rx for connecting the relevant notation used for various signals. In particular, $v(t)$ is the baseband pulse shaped symbol stream, $s(t)$ is the carrier modulated Tx waveform, $r(t)$ is the Rx waveform and $x(t)$ is the downconverted baseband signal which ideally should be the same as $v(t)$ ignoring the double frequency terms.

[†]Remember the equation of motion for a particle moving in a straight line with constant acceleration.

$$d = d_0 + vt + \frac{1}{2}at^2$$

Frequency Domain: Channel Frequency Response

Refer to Figure 8.26 where the differences between all the path lengths are assumed negligible to isolate the motion induced effects. Before the Rx antenna sums all the arriving paths, notice that each multipath arrives with a distinct amplitude and a distinct Doppler shift. Clearly, channel $c(F)$ is seen to be a combination of impulses $\delta(F - F_C - F_{D,i})$ where each $F_{D,i}$ can be positive or negative depending on the direction of motion. In Figure 8.26, $F_{D,1}$ is a positive Doppler frequency while $F_{D,2}$ is negative.

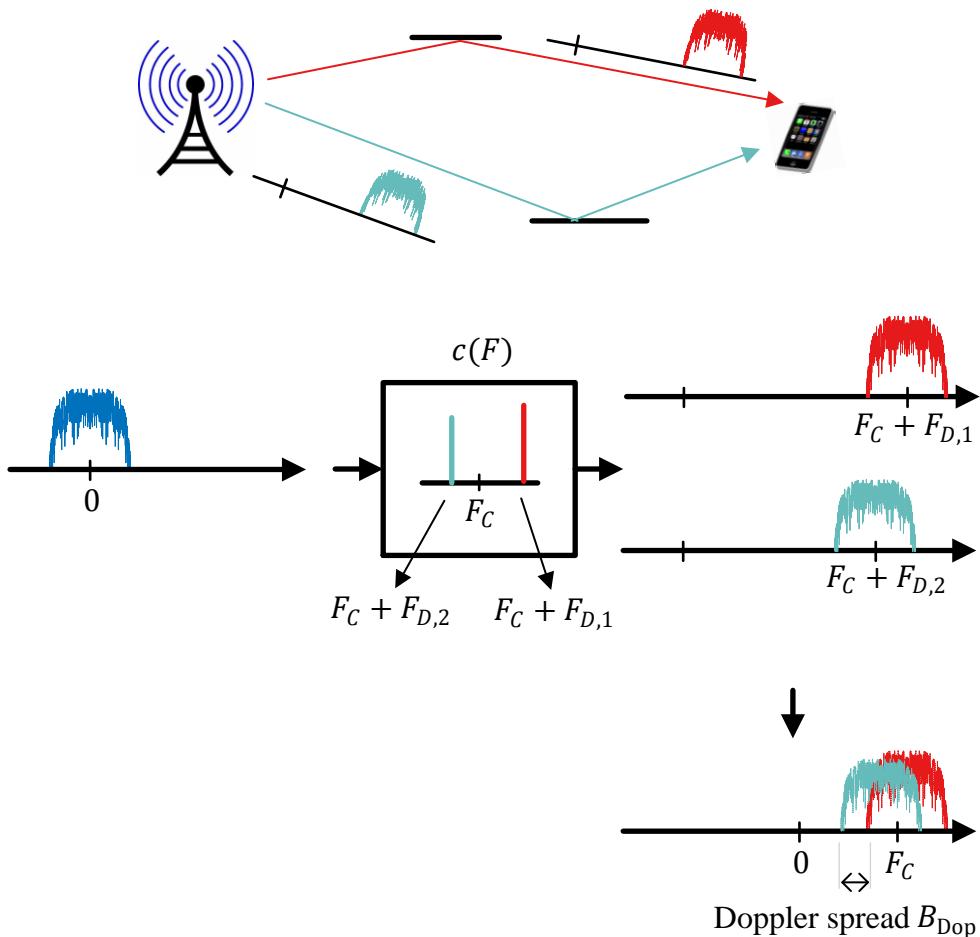


Figure 8.26: Frequency response $c(F)$ of the channel consists of some impulses with different amplitudes and frequency shifts (here we assume approximately equal path lengths to isolate motion induced distortion) and a corresponding convolution.

Compare with Figure 8.6

The analogy in the stationary channel are the delays τ_i with a dissimilarity in that τ_i can only attain a positive value. On the other hand, a Doppler shift acts on the real signal

$$s(t) = v_I(t)\sqrt{2} \cos 2\pi F_C t - v_Q(t)\sqrt{2} \sin 2\pi F_C t$$

impacting the real sinusoids. A real sinusoid consists of both the positive and negative frequency components of the spectrum. Therefore, in addition to impulses $\delta(F - F_C - F_{D,i})$ on the right side of the spectrum, we have impulses on the left side of the spectrum as well. However, we focus here on the positive side of the spectrum and implicitly take a parallel channel acting on the negative half of the spectrum in a mirror image.

Owing to the large scale fading, the amplitudes of both paths are different as before (although the actual path delays are assumed nearly equal here to avoid bringing delay distortions in this figure). A crucial disparity arises from the fact that the channel impulse response in Section 8.1.2 was one sided in time, i.e., the delays τ_i could only be positive. Therefore, its frequency domain representation was not conjugate symmetric. In the current scenario, the addition of two channel outputs, one acting on positive half of the spectrum and the other on the negative half, maintains the symmetry properties of the real signal sent into the air.

Note 8.8 Doppler spread

As a dual to delay spread, the difference between the largest Doppler frequency and the smallest Doppler frequency among the incoming multipath components is commonly known as *Doppler Spread* of the channel B_{Dop} . It signifies the width of the spectrum over which the channel is stretched out in frequency domain ignoring the paths with amplitudes below the noise floor. Denoting the largest Doppler frequency by $F_{D,\text{max}}$ and the smallest by $F_{D,\text{min}}$,

$$B_{\text{Dop}} = |F_{D,\text{max}} - F_{D,\text{min}}| \quad (8.29)$$

Like the delay spread, B_{Dop} has nothing to do with the Tx signal or our data rates. Instead, it depends on the carrier frequency F_C and the rate of change of the channel in terms of the movement. Doppler spread is the third out of four fundamental quantities through which a wireless channel is usually categorized.

Its typical values are a few Hz for walking speeds to a few hundreds of Hz for highway traffic. Since the movements within the environment vary with time, Doppler spread is also a random variable usually analyzed in a statistical manner.

Doppler Spectrum

To develop the statistical characterization of the received signal as a result of arriving paths, Clarke proposed a model which assumes that the field incident at the mobile antenna comes from contributions of waves with equal amplitudes, random angles of arrival from all directions and random carrier phases. Then, the Doppler power spectral density of the fading channel describes how much magnitude is contributed by the wave at each frequency. In reference to Figure 8.26, we can ask how much spectral broadening is caused in a scenario by random waves arriving from all directions.

Since the Doppler frequency of an unmodulated wave arriving from an angle ϕ is $F_{D,\text{max}} \cos \phi$, we can utilize the fact that the cosine amplitudes are biased towards ± 1 instead of being uniformly distributed. For this purpose, we draw a simple histogram of $F_{D,\text{max}} \cos \phi$ in Figure 8.27 where more power is observed for frequencies near $\pm F_{D,\text{max}}$ and very little for frequencies away from these values. This 'bathtub' shape is known as the classical *Doppler spectrum*.

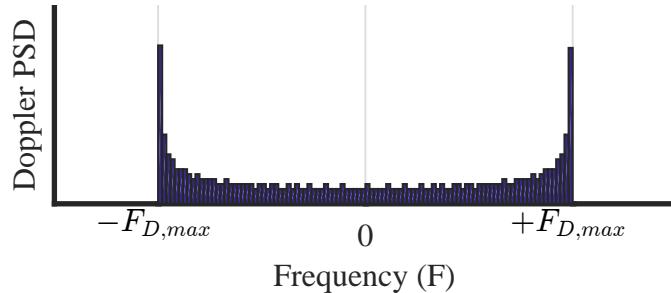


Figure 8.27: Doppler spectrum

Next, we look at the motion impact on the signal in time domain.

Time Domain: Channel Impulse Response

Consider Figure 8.28 for the time domain signal which has a spectrum consisting of both positive and negative sides. There is a complex multiplication in time domain between the Tx signal and the two channel sinusoids with frequencies $F_C + F_{D,i}$.

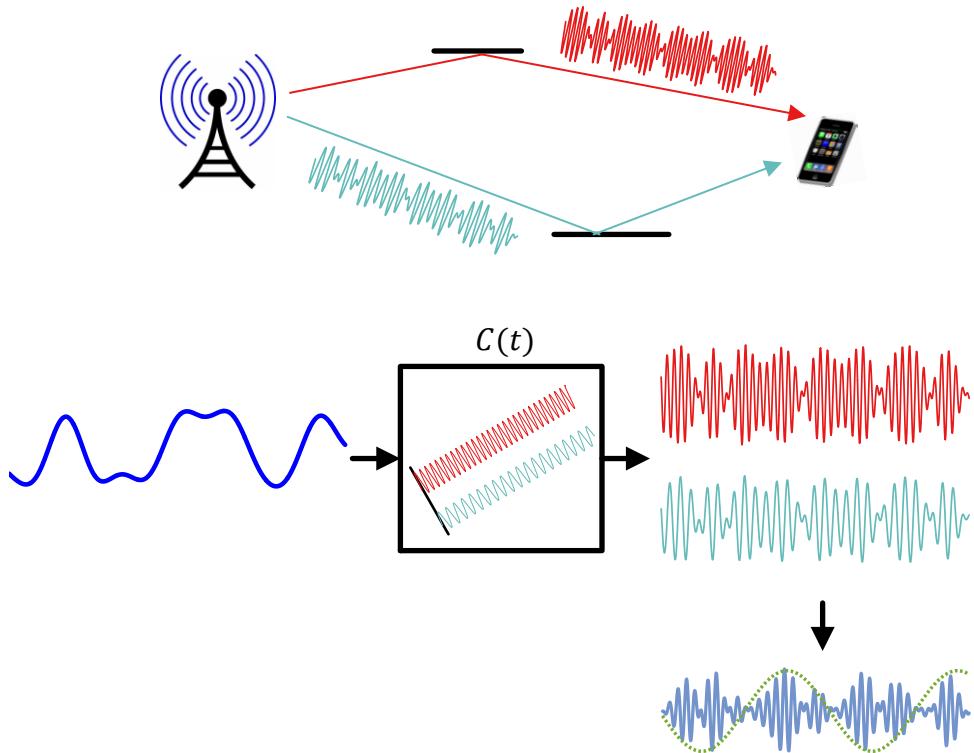


Figure 8.28: Impulse response $C(t)$ of the channel consists of different sinusoids at different amplitudes and frequencies $F_C + F_{D,i}$ and results in complex multiplication in time domain. Compare with Figure 8.8

Owing to the high frequencies, it is not quite clear from the figure but the two sinusoids in the channel $C(t)$ are shown to possess increasingly higher frequencies. This is because $F_{D,1}$ is a positive Doppler frequency and $F_{D,2}$ is negative. When the output is formed by summing multiple copies of the modulated signal riding at such slightly different carrier waves, the peaks and valleys emerge in proportion to their frequency differences.

Note 8.9 Coherence time

While the amplitude of the Rx signal varies with time, the duration over which it can be assumed approximately constant is termed as *Coherence Time* T_C of the channel. It is evident that the higher the Doppler frequencies, the faster the changes in the signal riding such sinusoids and hence smaller the coherence time. Now the periods of the mountains and valleys formed in the resultant signal are defined by the differences in the Doppler frequencies $F_{D,i}$, see Eq (8.28). The shortest such period is given by the difference between the maximum and the minimum Doppler frequency $|F_{D,\max} - F_{D,\min}|$.

Consequently, the coherence time is inversely proportional to the Doppler spread B_{Dop} defined in Eq (8.29).

$$T_C \propto \frac{1}{B_{\text{Dop}}} \quad (8.30)$$

which is very similar to the relationship between the coherence bandwidth B_C and the delay spread T_{Del} . Coherence time T_C is the fourth out of four fundamental quantities through which a wireless channel is mainly categorized. We can say that the signal envelope within the time span of a T_C more or less survives the effect of Doppler induced peaks and valleys in time domain. Just like the signal bandwidth within the frequency span of a B_C survives the effect of delay induced distortion in frequency domain.

8.1.6 Time Flat (Slow) and Time Selective (Fast) Fading

To understand different types of fading in the context of time variations, refer again to Figure 8.26 showing a multipath channel.

Time Flat (Slow) Fading

Recall that a low data rate scenario ($T_{\text{Del}} < 0.1T_M$) depicted in Figure 8.14 presented a case of frequency flat fading in time domain. Similarly, a slow motion scenario is illustrated in Figure 8.29 where three multipath components are arriving with Doppler shifts $F_{D,i}$ from the carrier frequency[†]. In this scenario, the magnitudes of $F_{D,i}$ are small and hence observe very little spreading of the cumulative spectrum.

This can be understood by recalling Eq (8.28) where half a cycle of the envelope was shown to occur in $1/(F_2 - F_1)$ seconds. When $F_2 - F_1$ is small, it takes a long time for the envelope to go from one zero to the next and the channel amplitude can be taken as a constant for a large number of symbols. Figure 8.29 reveals that the signal bandwidth B_{QAM} plays a central role in determining how much effect the Doppler spread B_{Dop} has on the Rx output. As long as the largest Doppler frequency difference

[†]Resemblance of the spectrum in Figure 8.29 with a bat and its wings is quite amusing.

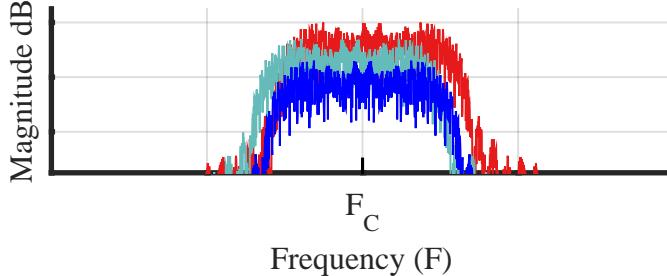


Figure 8.29: Multipath components with small Doppler shifts $F_{D,i}$ compared to the signal bandwidth (which is proportional to the symbol rate $1/T_M$). As a consequence, there is very little spreading of the cumulative spectrum

is still a fraction of the signal bandwidth,

$$|F_{D,\max} - F_{D,\min}| \ll B_{\text{QAM}}$$

as illustrated in Figure 8.29, the motion effects are minimal and the signal undergoes time flat fading, i.e., the signal envelope undergoes slow changes with time with respect to the symbol time T_M .

Then, time flat fading – commonly known as slow fading – implies

Frequency domain condition for slow fading

$$B_{\text{Dop}} \ll \frac{1}{T_M} \approx B_{\text{QAM}} \quad (8.31)$$

In time domain, a channel is time flat if it is changing very slowly with respect to the symbol rate $1/T_M$, i.e., the coherence time T_C is much larger than the symbol time T_M . As a consequence, the Rx sees an almost constant channel response without any significant variations for a large number of symbols as illustrated in Figure 8.30. Mathematically, we can write

Time domain condition for slow fading

$$T_C \gg T_M \quad (8.32)$$

We can exploit the relation between the Doppler spread and coherence time $T_C \propto 1/B_{\text{Dop}}$ to check that the above relation is just an inverted version of the frequency domain condition for slow (or time flat) fading in Eq (8.31).

Next, we explore the topic of time selective (or fast) fading.

Time Selective (Fast) Fading

To complete the analogy, a high data rate scenario ($T_{\text{Del}} > T_M$) depicted in Figure 8.17 presented a case of frequency selective fading in time domain. Similarly, a fast motion scenario is illustrated in Figure 8.31 where three multipath components are arriving

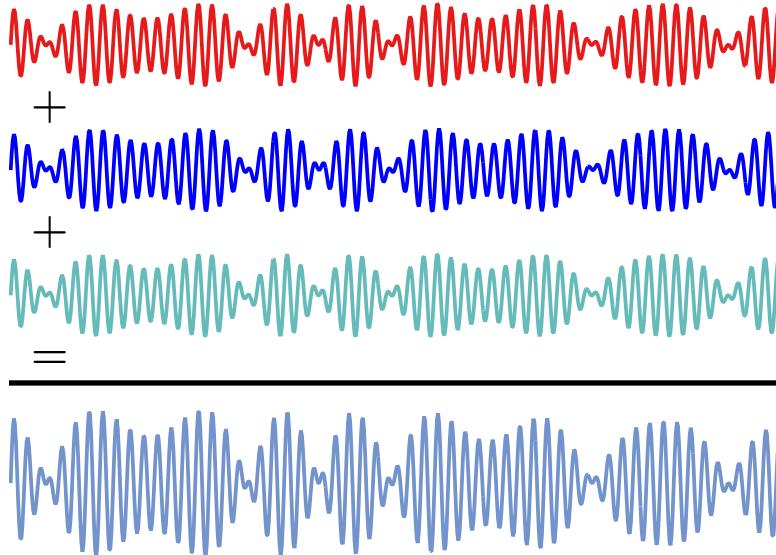


Figure 8.30: A small Doppler spread implies that the envelope variations for a stream of symbols stay the same

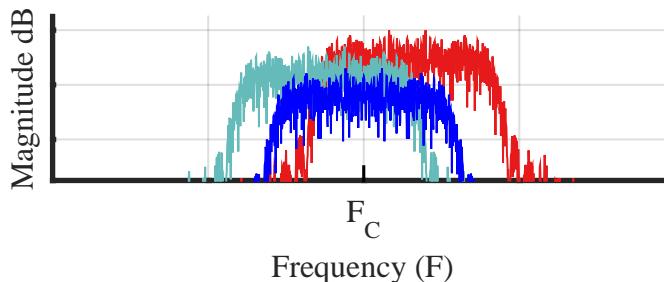


Figure 8.31: Multipath components with large Doppler shifts $F_{D,i}$ compared with the signal bandwidth (proportional to the symbol rate $1/T_M$). There is a significant spreading of the cumulative spectrum

with Doppler shifts $F_{D,i}$ around the carrier frequency. In this scenario, the magnitudes of $F_{D,i}$ are large causing a significant spreading of the cumulative spectrum.

Recalling Eq (8.28), when $|F_{D,\max} - F_{D,\min}|$ is large, the signal envelope exhibiting a short period quickly complete its repetitions and the channel amplitude varies in proportion, even within a symbol time T_M in some cases. We say that the fading rate is fast and the time selective (or fast) fading can be expressed as

Frequency domain condition for fast fading

$$B_{\text{Dop}} > \frac{1}{T_M} \approx B_{\text{QAM}} \quad (8.33)$$

Notice that the signal envelope, being inversely proportional to the significant Doppler

spread $|F_{D,\max} - F_{D,\min}|$, either changes within a symbol time T_M or very few symbols experience the same fade as illustrated in Figure 8.32. We say that the coherence time is less or on the order of a symbol time T_M .

Time domain condition for fast fading

(8.34)

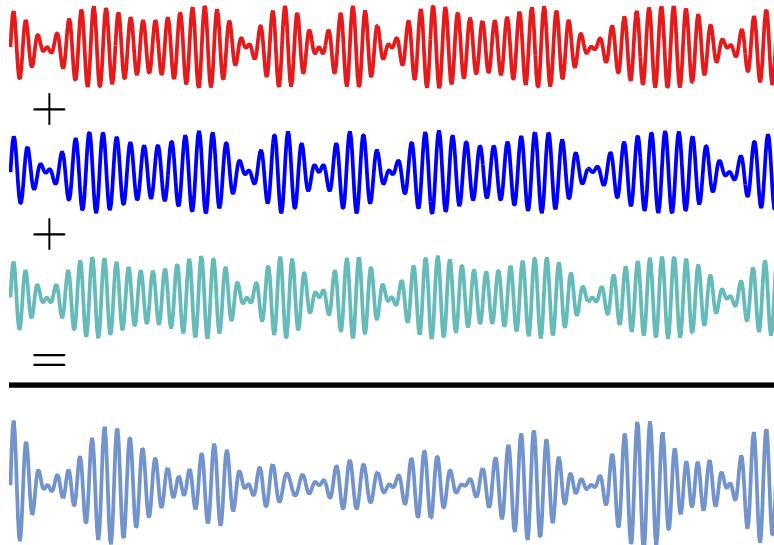


Figure 8.32: A large Doppler spread implies that fading envelope varies quickly, either during a symbol time T_M or within a few symbols

As the term selectivity implies, when some symbols of the signal stream are in a deep fade, the rest of the symbols are not and hence we are able to achieve *time diversity* through this selective behaviour[†]. Sometimes when we experience a deep fade in a voice call, we successfully overcome it by moving slightly from that place. Motion is a kind of diversity!

Now we can exploit the relation between the Doppler spread and coherence time $T_C \propto 1/B_{\text{Dop}}$ to check that the above relation is just an inverted version of the frequency domain condition for fast (or time selective) fading in Eq (8.33).

One final question that remains now is the following. Designing a Rx for a static channel during a transmission does not seem to be an excessively difficult task. However, when the channel is changing with time, how can a Rx handle this channel that is acting differently on different data symbols? In other words, how should we incorporate channel gains $\gamma_i(t)$ and channel delays $\tau_i(t)$ that are changing with time. We will answer this question at the end of Section 8.1.7.

[†]Channel coding is an integral part of most wireless systems that introduces time diversity in the signal by design.

8.1.7 From Channel Paths to Channel Taps

We have learned before that the wireless channel acts on the Tx signal in continuous time. However, the Rx samples the arriving signal at a rate of $F_S = L/T_M$ where L is the oversampling factor, or number of samples/symbol. We need to understand how this transformation from continuous-time to discrete-time impacts the Rx signal and what constitutes an equivalent ‘digital channel’.

We start with a simple Quadrature Amplitude Modulated (QAM) waveform as a passband signal consisting of two Pulse Amplitude Modulated (PAM) waveforms $v_I(nT_S)$ and $v_Q(nT_S)$ in I and Q rails.

$$\begin{aligned} I &\rightarrow v_I(nT_S) = \sum_m a_I[m] p(nT_S - mT_M) \\ Q &\uparrow v_Q(nT_S) = \sum_m a_Q[m] p(nT_S - mT_M) \end{aligned}$$

In terms of complex signals, $a[m]$ are complex data symbols comprising of inphase and quadrature data symbols $a_I[m]$ and $a_Q[m]$, respectively. Furthermore, $v_I(nT_S)$ and $v_Q(nT_S)$ combine to form the complex signal $v(nT_S)$ and $p(nT_S)$ is the impulse response of a square-root Nyquist pulse that has a zero Q component.

$$\begin{aligned} v(nT_S) &= a[0]p(nT_S - 0T_M) + a[1]p(nT_S - 1T_M) + \\ &\quad a[2]p(nT_S - 2T_M) + a[3]p(nT_S - 3T_M) + \dots \\ &= \sum_m a[m]p(nT_S - mT_M) \end{aligned}$$

Recall that such a waveform is generated after upsampling the complex data symbols $a[m]$ by L and convolving them with the pulse shape generated at a rate of L samples/symbol. Let us denote the *upsampled data symbols by $\tilde{a}(nT_S)$* . Then, we can write the above expression as

$$v(nT_S) = \tilde{a}(nT_S) * p(nT_S) \quad (8.35)$$

Its continuous-time version $v(t)$ can be expressed as

$$v(t) = \sum_m a[m]p(t - mT_M)$$

The upconversion process produces the Tx signal $s(t)$ as

$$\begin{aligned} s(t) &= \sum_m a_I[m]p(t - mT_M) \sqrt{2} \cos 2\pi F_C t - \\ &\quad \sum_m a_Q[m]p(t - mT_M) \sqrt{2} \sin 2\pi F_C t \\ &= v_I(t) \sqrt{2} \cos 2\pi F_C t - v_Q(t) \sqrt{2} \sin 2\pi F_C t \end{aligned}$$

Now recall Eq (8.6) reproduced below which relates the downconverted complex Rx signal $x(t)$ to $v(t)$.

$$x(t) = \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot v(t - \tau_i) = v(t) * c_B(t) \quad (8.36)$$

Let us now visualize the process that brings us the channel taps from channel paths by concentrating on a single pulse shaped symbol $a[m]$ as it travels through the wireless channel. Owing to the linearity of the channel, whatever happens to this pulse shaped symbol will be repeated to all others symbols as a superposition.

For simplicity of presentation and less complicated figures, we focus only on the I arm as well as a real channel impulse response. *This implies that the waveform in the figure consists of PAM symbols attenuated by different amplitudes of the respective paths, but not phases.* The data symbols are convolved through a square-root Nyquist pulse and then the channel impulse response before the sampling at the Rx. A visualization of the channel convolution part is drawn in Figure 8.33 where the carrier wave is absent due to the downconversion process already accomplished before sampling.

Shown at the top of Figure 8.33 is a channel impulse response with five paths. Below that, notice the arrival of multipath copies of the Tx signal with different amplitudes and delays τ_i according to the Rx time reference. Path 0 arrives a little after the Rx time reference 0 while the paths 1 and 2 are very close to each other. Finally, paths 3 and 4 are reasonably spaced apart.

Here, the Rx is sampling the signal at a rate of $R_M = 1/T_M$ Hz, or $L = 1$ sample/symbol because this is what eventually matters for symbol detection. Due to skipping the time interval in between every two samples, the Rx can only distinguish between paths that are sufficiently spaced apart. Let us find out what sufficiently means in this context.

This process depicts a transformation of actual channel paths (a total of five in continuous-time) to what Rx sees as channel taps (a total of four in discrete-time). Here is the main point.

"A channel tap at instant mT_M is a discrete-time sample that arises from the superposition of all continuous-time channel paths at that instant."

One might think from the channel impulse response that the first 3 paths make up tap 0 of the discrete-time channel while paths 3 and 4 make up taps 1 and 2, respectively, but this is not true. Referring to Figure 8.33, we proceed as follows.

Tap 0: Observe the instant 0 according to the Rx time reference. The main contribution in this tap is from the direct path signal that has not reached its peak yet while there is some reasonable contribution from paths 1 and 2 as well. Consequently, their superposition generates the tap 0 of the discrete-time channel impulse response $h[0]$.

Tap 1: At the next sample time, i.e., T_M , the main contribution is from path 2 and a significant part from path 1 while paths 0 and 3 participate marginally to produce the next channel tap $h[1]$.

Tap 2: Continuing in a similar manner, channel tap $h[2]$ is produced mainly from path 4 with a minor part from path 3, 2 and 1.

Tap 3: Finally, there is no main participation from any path in making up the tap 3. Only if the sum total of the energy is above the noise floor or the threshold set for channel estimation, it would be counted as a 4 tap channel.

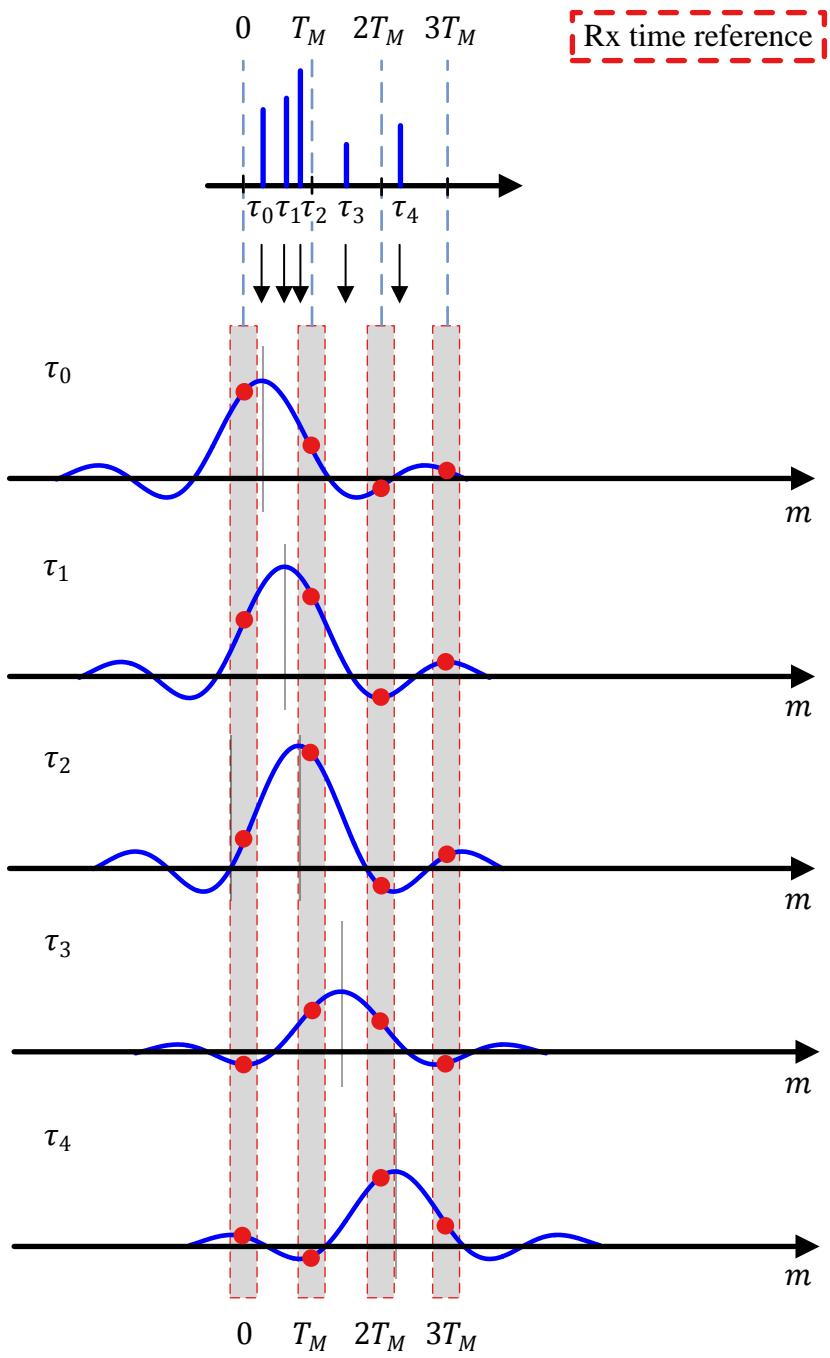


Figure 8.33: From actual continuous-time channel paths (5) to what Rx sees as discrete-time channel taps (4) at $T_S = T_M$ for a real Tx signal and a real channel

For gaining further insight, recall that the pulse bandwidth is directly proportional to the symbol rate $1/T_M$.

$$B_{\text{QAM}} = \frac{1 + \alpha}{T_M}$$

The higher the symbol rate $1/T_M$, the narrower the pulse shape in time domain. Two things happen as a result.

1. Due to the narrower pulse shape, the number of multiple channel paths that superimpose at each other to form each tap reduces.
2. A higher value of $1/T_M$ requires a higher sample rate at the Rx (even at $L = 1$ sample/symbol). This high sample rate implies that the Rx can ‘see’ more of the paths *on an individual level* as opposed to their superimposed sum. For example, paths 1 and 2 in Figure 8.33 are arriving very close to each other and a wide pulse in our example could not resolve them as separate paths. At now higher sample rate at the Rx, it can resolve paths 1 and 2 if $\tau_2 - \tau_1$ is greater than the symbol time T_M . This is one of the reasons the Ultra-Wideband (UWB) systems are popular for ranging applications because they can separate the first arriving path relatively clearly from the later paths. It is the first path arrival time that translates into an accurate distance estimation.

It must be remembered though that this is different than sampling the Rx signal at an arbitrarily high sample rate with the same bandwidth. As Figure 8.33 clearly depicts, sampling the Rx signal at, say $L = 2$ samples/symbol, just gives us more samples of the channel convolved signal without any improvement in multipath resolution. Nevertheless, sampling at higher than the symbol rate $1/T_M$ does safeguard us from experiencing a spectral null due to the worst case timing offset, a situation we encountered in Figure 7.7 during a discussion on timing synchronization.

Many authors treat the channel as first bandlimited with a perfectly rectangular filter according to the signal bandwidth. As a result, its time domain impulse response is assumed to be convolved with a sinc signal and then the composite channel is taken as a summation of scaled and time shifted sinc signals. This can be seen through assuming the excess bandwidth $\alpha = 0$ and viewing the channel response as such a sum in Figure 8.33.

Moving from a single symbol example, we want to have a look at the kind of the signal that arrives at the Rx for a pulse shaped symbol stream. This is drawn in Figure 8.34 where a sequence of 10 Square-Root Raised Cosine pulse shaped symbols with excess bandwidth $\alpha = 0.5$ are sampled at the symbol rate $1/T_M$. It is evident that each sample at the Rx is no longer similar to an ideal case, even if there was a zero timing offset and no noise present. Instead, a series of scaled and delayed pulses add up to form each sample. Although it is not always the case, we filter the resultant sampled signal through a matched filter $p^*(-nT_S)$ to produce the signal $z(nT_S)$ for further processing.

Next, we pick up this story back from Eq (8.36) after which we went on a short

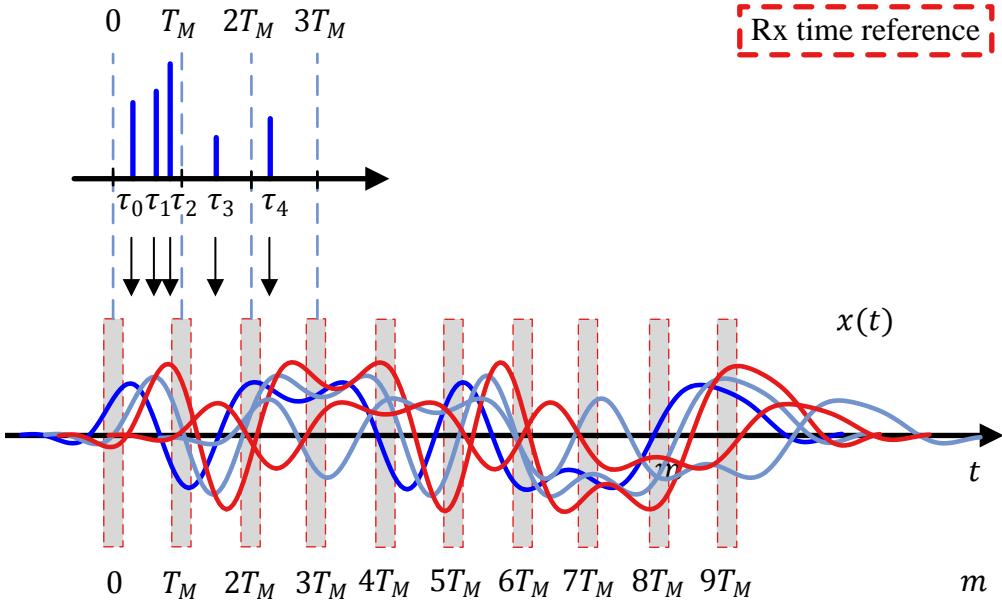


Figure 8.34: A Square-Root Raised Cosine pulse shaped symbol stream with excess bandwidth $\alpha = 0.5$ arriving at the Rx and sampled at the symbol rate ($F_S = 1/T_M$). Only I arm shown and zero phase for each channel path for simplicity

detour. Sampling $x(t)$ at $F_S = 1/T_M$ [†] produces the discrete-time signal $x(nT_S)$ as

$$\begin{aligned} x(nT_S) &= v(nT_S) * c_B(nT_S) \\ &= \sum_{l=0}^{N_{\text{Tap}}} v(lT_S) c_B(nT_S - lT_S) \end{aligned}$$

where the channel length is considered as $N_{\text{Tap}} + 1$. Here, the extra 1 refers to the main tap (which can be the tap with the highest power) while the rest contribute ISI in the signal.

In essence, we are modeling the baseband channel $c_B(nT_S)$ acting on the Tx signal as an FIR filter with $N_{\text{Tap}} + 1$ taps, which is why it is known as a tapped delay-line model. An example with $N_{\text{Tap}} = 3$ is illustrated in Figure 8.35 that generates the input waveform $x(nT_S)$ at the Rx. Although it is not explicitly shown, complex operations are implied in the figure due to the signals $v(nT_S)$ and $c_B(nT_S)$ both being complex. Remember that a complex convolution consists of four real convolutions and produces $I * I - Q * Q$ at the inphase rail and $Q * I + I * Q$ at the quadrature rail.

This signal is filtered at the Rx by the flipped and conjugated version of the pulse shape $p^*(-nT_S)$ to improve the SNR. Note that we have neither used the terminology ‘maximize’ the SNR, nor called it the matched filter here. This is because after the inclusion of the channel, ***the pulse $p^*(-nT_S)$ is not matched to the incoming signal anymore***. Nevertheless, with slight abuse of notation, we continue calling it the

[†]We are using the same notation F_S for Tx and Rx sample rate for convenience; however, the sample rate can, and often is, different at various stages of the Tx and the Rx systems. In fact, both the Tx and Rx are multirate systems that immensely helps in the transition between the digital and analog worlds.

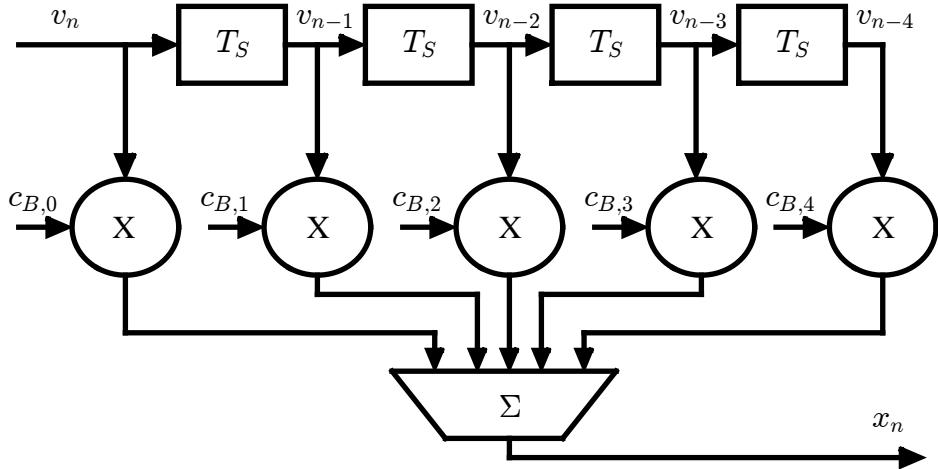


Figure 8.35: Baseband channel impulse response $c_B(nT_S)$ modeled as an FIR filter with $N_{\text{Tap}} + 1$ taps acting on the pulse shaped symbol stream $v(nT_S)$ that generates the input waveform $x(nT_S)$ at the Rx. Complex operations are implied

matched filter to be consistent with the terminology employed in the previous chapters. The matched filter output is then

$$z(nT_S) = x(nT_S) * p^*(-nT_S) = v(nT_S) * c_B(nT_S) * p^*(-nT_S)$$

Combine the above equation with Eq (8.35) where $\tilde{a}(nT_S)$ was the upsampled data symbol stream.

$$z(nT_S) = \tilde{a}(nT_S) * \underbrace{p(nT_S) * c_B(nT_S) * p^*(-nT_S)}_{\text{Composite channel } h(nT_S)}$$

This all comes down to the following pleasant simplification: based on the above relation, we can assume that the signal supplied to the Rx for digital processing is the upsampled data symbols $\tilde{a}(nT_S)$ convolved with a composite channel $h(nT_S)$ that is a linear filter with an impulse response

$$\begin{aligned} h(nT_S) &= p(nT_S) * c_B(nT_S) * p^*(-nT_S) \\ &= r_p(nT_S) * c_B(nT_S) \end{aligned} \tag{8.37}$$

The final matched filter output can now be written as below. Note that the only change from our usual shaped symbol stream is that $p(nT_S)$ has been replaced by $h(nT_S)$.

$$z(nT_S) = \sum_m a[m]h(nT_S - mT_M) \tag{8.38}$$

Each sample of the composite impulse response $h(nT_S)$ is considered a *channel tap* first visualized in Figure 8.33. A block diagram of such a structure is drawn in Figure 8.36.

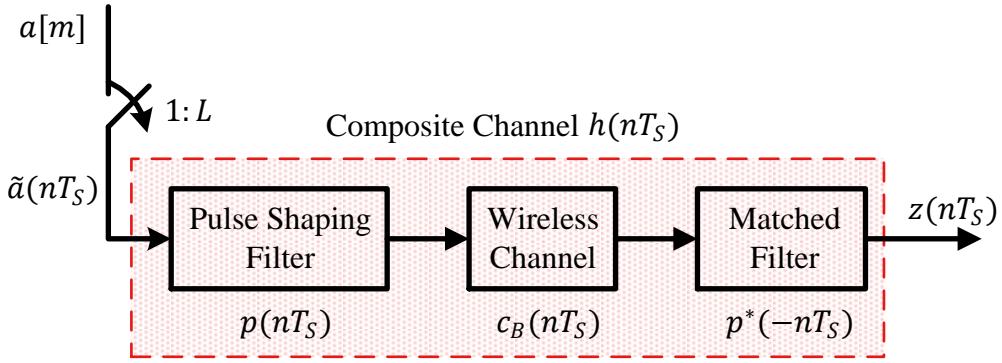


Figure 8.36: A composite channel model consisting of a pulse shaping filter $p(nT_S)$, baseband channel $c_B(nT_S)$ treated as a linear filter and a matched filter $p^*(-nT_S)$

Finally, the model for a T_M -spaced signal at $L = 1$ sample/symbol can be obtained by downsampling the above relation.

$$z[m] = a[m] * h[m] \quad (8.39)$$

As a remark, *a channel tap is different than a channel path due to the finite bandwidth of the Tx signal*. The sample rate F_S determines the bandwidth that can be processed by the Rx. Finite bandwidth systems have a finite rise and fall time of the corresponding time domain signal, as we saw in Figure 8.33 and during the pulse shape design procedure.

Finally, note that we have not included the effect of timing offset in the above model, just like we have not included the effect of carrier phase and frequency distortion as well. A channel is a general filter in which each such distortion can be absorbed as a special case. For example, a simple timing offset ε_Δ implies a channel with only a direct path with delay equal to that timing offset and no amplitude distortion.

Note 8.10 Noise Whitening Filter

When the channel impulse response is known at the Rx, the actual matched filter is formed through the convolution of the pulse shape $p(t)$ with $c_B(t)$ and then flipping and conjugating the resultant response. Then, a symbol rate sampling at $R_M = 1/T_M$ is enough to optimally recover the Tx symbols.

However, the noise samples at the output of such a filter are not uncorrelated anymore. In communications theory, it is important to compare the performance of various compensation techniques through a common evaluation strategy, which becomes non-tractable (no closed-form expressions) due to the correlation among noise samples. Therefore, a noise whitening filter is included after the actual matched filter at the output of which the noise samples taken at rate $1/T_M$ are again uncorrelated.

Since performance evaluation is out of the scope of this book, we proceed with the simple model of Eq (8.38) above.

8.1.8 Dealing with a Time-Varying Channel

In Section 8.1.6, we posed a question about dealing with a mobile channel, i.e., how can a Rx equalize a channel that is varying with time and hence acting differently on different data symbols? From Eq (8.7), we now write

$$c_B(t) = \sum_{i=0}^{N_{MP}-1} \gamma_i(t) \cdot \delta(t - \tau_i(t))$$

i.e., channel gains $\gamma_i(t)$ and channel delays $\tau_i(t)$ are varying with time. With the movement in the channel, the taps in a frequency selective channel are changing according to the rotation rates of path phases, as shown in Figure 8.37. Here, we can see different arriving paths at the Rx which form the cumulative tap gain at a particular sampling instant. These rotating vectors are a beautiful manifestation of the complexity a wireless channel exhibits. When I see world clocks at the check-in counter of a hotel, it reminds me of a frequency selective time-varying channel. When

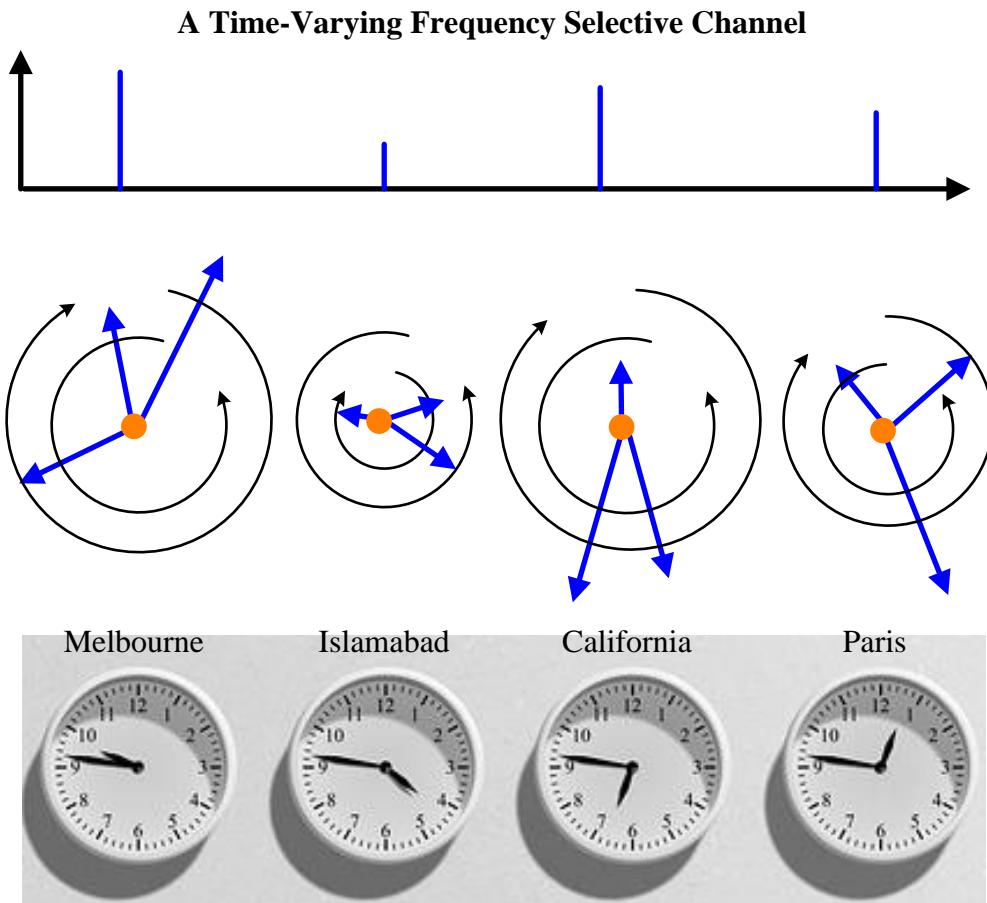


Figure 8.37: The phases in channel taps are continuously rotating giving rise to a time-varying channel

you notice a correlation between clock times of this figure, keep in mind that some correlation also exists among the channel taps.

With channel taps in hand, we present three routes to solve this issue.

Quasi-static assumption: For a block processing scheme where the data is sent over a series of blocks and the channel changes slowly over each block length, it makes sense to assume that the channel remains constant throughout the duration of each block. Then, a new channel is estimated at the start of the next block which is assumed to hold true for the block duration again. This process can be continued till the end of the whole frame. This quasi-static assumption implies that while a different channel emerges for each block, the same channel acts on the data symbols within each block. An example of this scheme is drawn in Figure 8.38.

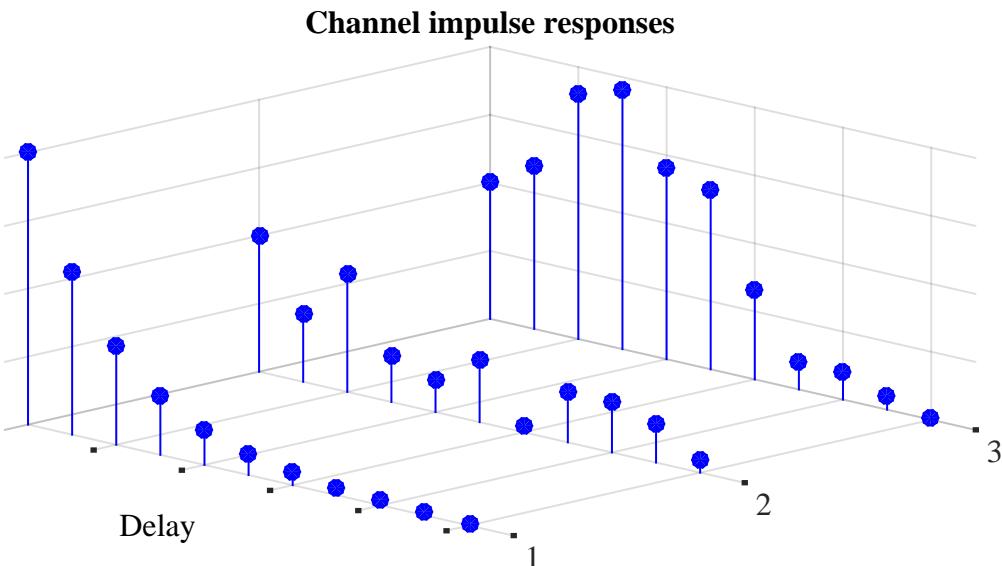


Figure 8.38: A quasi-static assumption implies that the channel stays the same for each block but varies from one block to the next

Channel tracking: Another widely used solution is to insert symbols known at the Rx called ‘pilots’ at uniform intervals within the Tx waveform. Just like an analog signal is sampled at uniform intervals by an ADC, these known pilots ‘sample’ the channel at these locations and the complete channel response can be constructed through interpolating between the pilots. Consequently, the sampling theorem applies in this scenario that links the number of pilots and the rate of change of the channel. Although it costs some spectral efficiency, the advantage is that the channel can be continuously tracked in time.

Adaptive equalization: In Section 8.3, we will study the role of equalizers that mitigate the impact of the channel. For a mostly static channel, these equalizers are preset or fixed at the start of the Rx signal processing and stay so for the duration of the transmission. For a time-varying channel, many equalizers are

designed to adapt themselves according to the channel conditions. Here, a special adaptive algorithm makes the equalizer coefficients follow the variations in the channel impulse response, as we will see later in this chapter.

Now we study the effect of the channel on the actual data symbols.

8.1.9 Effect of Channel on Rx Constellation

Until now, we talked about the physical processes that govern the impairments caused by the wireless channel. Recall that an eye diagram, a transition diagram and a scatter plot are the stethoscopes of a communication system and hence it is imperative to bring in that perspective for a signal convolved with the channel filter.

As an example, we take a really simple channel with just two multipath $N_{MP} = 2$ with an amplitude of 1 and 0.3 and the second path arrives exactly at the start of the next symbol after the first, i.e.,

$$\begin{aligned}\rho_0 &= 1, \quad \tau_0 = 0 \\ \rho_1 &= 0.3, \quad \tau_1 = T_M\end{aligned}$$

Now we draw the eye diagram of such a simple channel in Figure 8.39 for a 4-QAM signal and zero noise. Observe that due to the two paths adding with one another, the eye diagram exhibits 4 distinct levels for each I and Q waveform at the ideal sampling instants and hence the distortion significantly alters the eye pattern. Moreover, owing to the benignity of the channel, the eye is not close and the modulated signal can be recovered through any simple equalization technique.

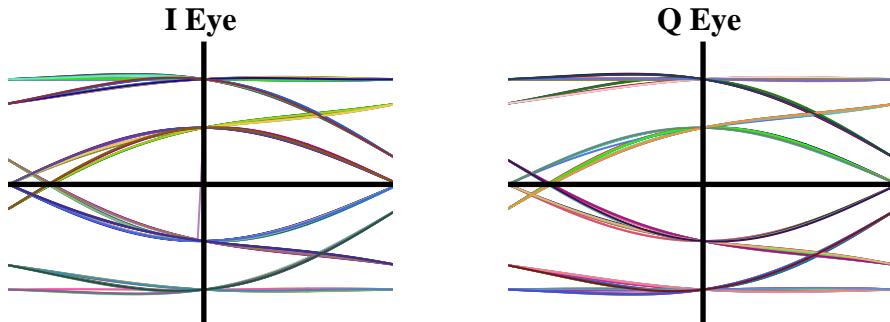


Figure 8.39: Eye diagram for a 4-QAM modulated signal and a simple channel impulse response $h[m] = [1 \ 0.3]$

The transition diagram or phase trajectory for the continuous-time signal arriving at the Rx after passing through the same channel is plotted in Figure 8.40a where the channel characteristics are not clearly visible. On the other hand, downsampling the signal to $L = 32$ samples/symbol and $L = 16$ samples/symbol and drawing the scatter plots in Figure 8.40b and Figure 8.40c, respectively, reveals a certain kind of structure in the Rx signal. This structure arises due to the linearity of the channel resulting in an addition of the channel paths and is inherently different from what we observe for random noise.

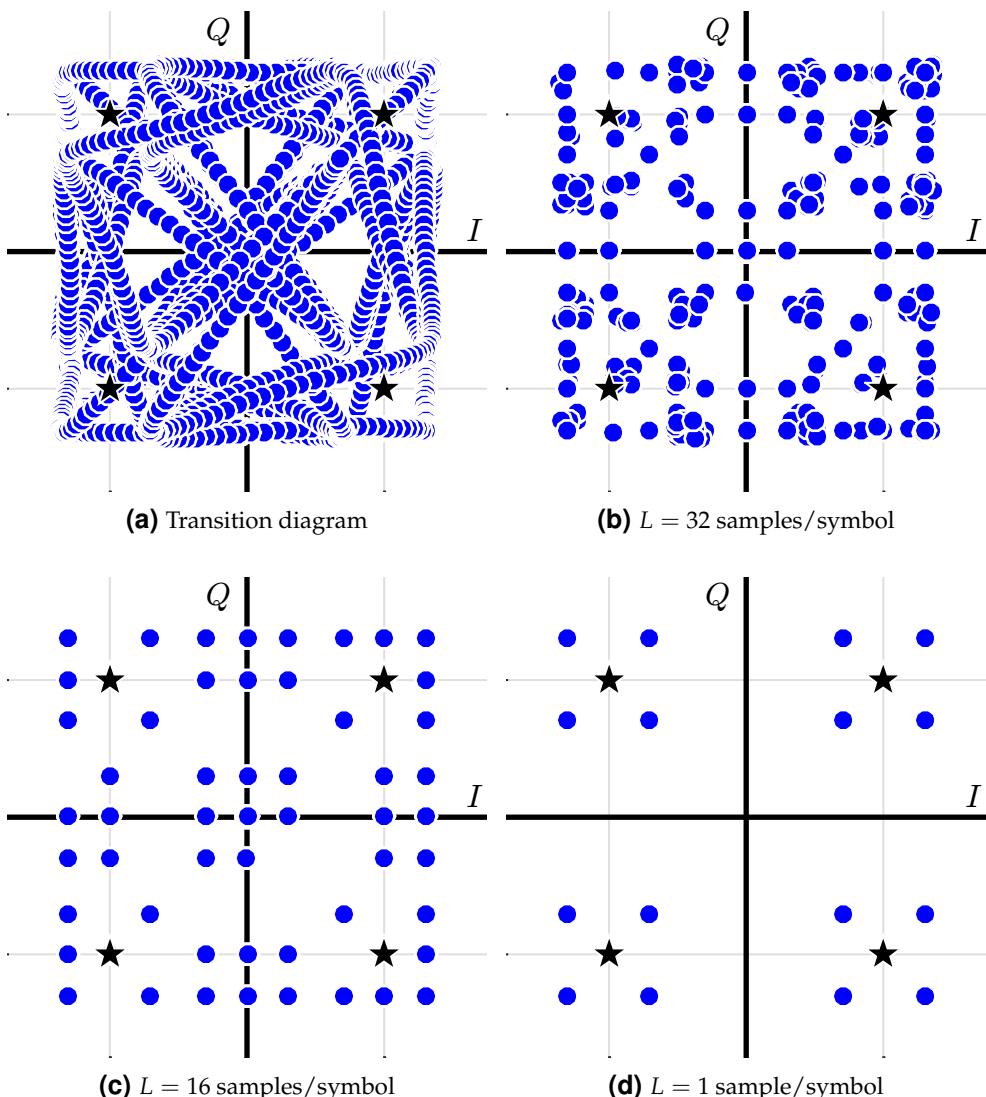


Figure 8.40: Transition diagram and scatter plots for a 4-QAM modulated signal and a simple channel impulse response $h[m] = [1 \ 0.3]$

Finally, a symbol rate scatter plot after downsampling the signal to $L = 1$ sample/symbol is illustrated in Figure 8.40d which is analogous to the ideal sampling instants of the eye diagram in Figure 8.39. Just like the eye diagram manifests 4 distinct levels for each I and Q rail due to the two path channel, the scatter plot also shows the sample locations at equal distances from the ideal constellation points. This is the kind of the signal make-up that enables many equalizers to remove the channel effects.

Keep in mind that a simple example was chosen here to uncover the hidden structure within the channel that is clearly distinguished from other sources of signal impairment.

ment. As the channel gets more complicated with several paths, delays and phases, the eye closes and much randomness appears to spread over the signal. The eye diagrams and scatter plots for such channels are significantly more convoluted than the simple case depicted here. Nevertheless, the channel is still a discrete-time filter and can be mitigated through suitable equalization strategies.

Next, we discuss how the channel taps can be estimated at the Rx. This is because some of the equalization strategies described later in this chapter rely on channel impulse response being known at the Rx.

8.2 Channel Estimation

Channel estimation is a special case of the system identification problem that has a long history in the field of signal processing. The most common method to estimate a channel at the Rx is based on a training sequence, i.e., it is a data-aided scenario and the discussion here is based on Ref. [34].

Correlation

The first idea to estimate the channel comes from the definition of an impulse response: the response to an impulse. Ideally, the channel can be learned by sending a single impulse and observe the Rx signal. An impulse, however, has an infinite bandwidth resulting in high sample rates and power consumption even when approximated. Furthermore, a channel estimate based on a single impulse contains significant noise. Therefore, we turn towards some more practical approaches.

For a symbol spaced channel, one of the most straightforward ways to estimate the channel coefficients is to correlate the Rx signal with a known training sequence $a[m]$ of length N_P , commonly known as a Pseudo-Noise (PN) sequence. The expression pseudo-noise refers to the fact that the sequence is completely deterministic but possesses noise like autocorrelation properties. The quality of the channel estimate from here depends on the careful design of the training sequence. Ideally, we want to employ a sequence such that its autocorrelation closely resembles an impulse, i.e., it is zero for every shift $i_0 \neq 0$ and a sharp peak for $i_0 = 0$.

$$a[m] * a^*[-m] \approx \delta[m]$$

When this sequence is employed as a Tx signal for the correlation operation with a stored copy at the Rx, the output is zero for each shift i_0 for a noiseless case. However, when the shift i_0 attains a value such that the training sequence embedded in the Tx signal gets aligned with the stored training sequence at the Rx, we observe a sharp peak in the correlation output. From here, the channel impulse response can be obtained as follows.

From Eq (8.39), the signal $z[m]$ is the training sequence $a[m]$ convolved with the channel impulse response $h[m]$ which is assumed to be a frequency selective channel with length $N_{\text{Tap}} + 1$ here.

$$z[m] = h[m] * a[m]$$

Employing the definition of correlation for complex signals which is the same as convolution with a flipped signal, we can write the correlation output as

$$z[m] * a^*[-m] = \sum_{i=i_0}^{i_0+N_P-1} z[i] a^*[i-m]$$

where the index i_0 indicates the first sample of the Rx signal used in the correlation. At the instant when the embedded training sequence in the Rx signal and the stored training sequence at the Rx align, we have

$$\hat{h}[m] = h[m] * \underbrace{a[m] * a^*[-m]}_{\approx \delta[m]} \approx h[m]$$

i.e., the correlation output sequence is an estimate of the channel. The estimation error is directly proportional to the number of significant taps.

Next, we turn towards a core statistical technique, namely the least squares, to estimate the channel.

Least Squares

Another method to estimate the channel coefficients is the well known Least Squares (LS) technique. This is probably the oldest statistical signal processing technique employed in all kinds of scientific fields, dating back to its discovery by Legendre and Gauss in early 1800s. To keep the discussion simple, we will work with the following assumptions.

- The channel is *flat fading*, i.e., it is a single tap channel $h[m]$.

$$z[m] = h[m] \cdot a[m]$$

For a length $N_{\text{Tap}} + 1$ channel, i.e., a frequency selective channel, the derivation remains parallel to the one below but matrix operations from linear algebra need to be carried out.

- The channel tap $h[m]$ and the modulated symbols $a[m]$ are real to avoid complex differentiations.

In this setup, the error between the Rx signal $z[m]$ and its actual value $h[m] \cdot a[m]$ is given by

$$e[m] = z[m] - h[m] \cdot a[m]$$

The error can be both positive or negative, so we want to minimize the magnitude squared value of $e[m]$.

$$|e[m]|^2 = |z[m] - h[m] \cdot a[m]|^2$$

The derivative of $|e[m]|^2$ with respect to the channel tap $h[m]$ is

$$\frac{d}{dh[m]} |e[m]|^2 = 2e[m] \cdot \dot{e}[m] \quad (8.40)$$

Utilizing the definition of $e[m]$, its derivative $\dot{e}[m]$ can be calculated as

$$\dot{e}[m] = \frac{d}{dh[m]} (z[m] - h[m] \cdot a[m]) = -a[m]$$

Plugging it back in Eq (8.40) and equating it to zero yields

$$2e[m] \cdot \dot{e}[m] = -2(z[m] - h[m] \cdot a[m])a[m] = 0$$

From here, the estimate $\hat{h}[m]$ of the channel tap in this flat fading scenario is given by

$$\hat{h}[m] = \frac{a[m]z[m]}{|a[m]|^2}$$

For a complex setup, the estimate is given as

$$\hat{h}[m] = \frac{a^*[m]z[m]}{|a[m]|^2} \quad (8.41)$$

Multiple pilots symbols can be embedded in the signal to reduce the estimation error through averaging. The above expressions makes sense due to the following two operations being carried out.

- Owing to the underlying correlation technique, multiplication with $a^*[m]$ cancels the phase originating from the modulated symbol within $z[m]$ and leaves only the phase from the channel tap $h[m]$. Utilizing the value of $z[m]$,

$$a^*[m]z[m] = a^*[m]a[m]h[m] = |a[m]|^2h[m]$$

- Division by $|a[m]|^2$ normalizes the modulation induced magnitude above.

$$\frac{a^*[m]z[m]}{|a[m]|^2} = \frac{|a[m]|^2h[m]}{|a[m]|^2} = h[m]$$

As mentioned above, the same technique can be applied to a frequency selective channel in the context of matrix operations.

For a time-varying channel, we begin with the observation that the index m signifies a changing channel for each symbol in a slow fading scenario, i.e., there is little variation from one symbol to the next. To estimate the channel, pilot symbols can be allocated at uniform intervals on which the fading channel tap can be obtained from the above expression. The values for the intermediate data symbols can then be extracted by applying suitable interpolation techniques.

8.3 Equalization

As discussed before, the wireless channel is a source of severe distortion in the Rx signal and our main task is to remove the resulting ISI from the Rx samples and recover the correct symbols. Equalization refers to any signal processing technique in general and filtering in particular that is designed to eliminate or reduce this ISI before symbol detection. In essence, the output of an equalizer should be a Nyquist pulse for a single symbol case. Linking to Figure 8.36 depicting a composite channel, a conceptual block diagram of the equalization process is shown in Figure 8.41.

It is almost impossible to discuss a broad topic like equalization in a few sections. However, we will manage to understand the main concepts and the most prevalent implementations. For an in-depth exploration, one of the most comprehensive resources on equalization are the wonderful chapters in Ref. [3] by Proakis. We begin with the big picture of equalizers in wireless receivers.

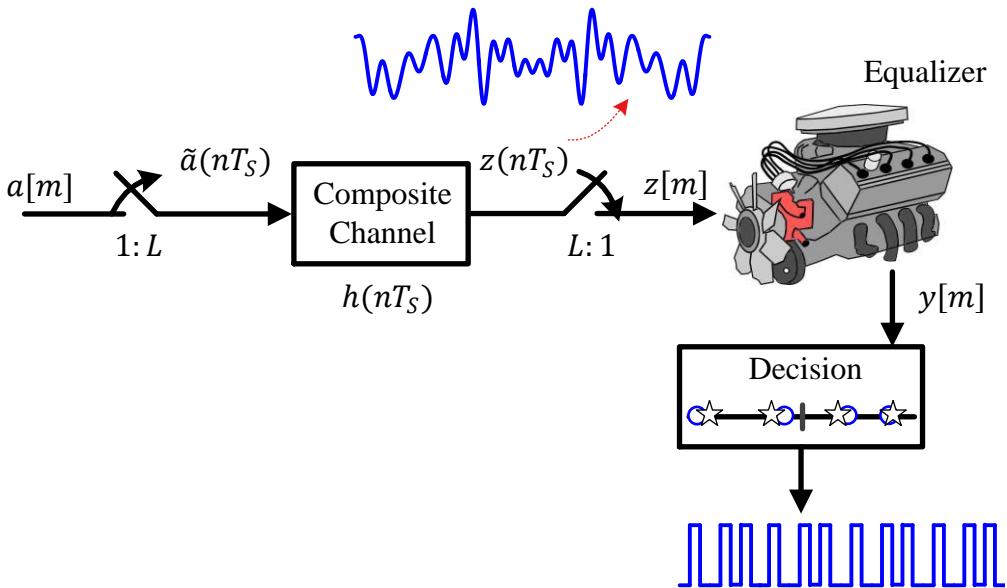


Figure 8.41: An equalizer compensates for the channel distortion at the Rx. A symbol-spaced equalizer is shown here

8.3.1 The Big Picture

The equalizer performs the bulk of the signal processing operations required at the Rx for proper demodulation. For this reason, the categories in which equalization techniques can be divided are far more than encountered in any kind of synchronization. We discuss some of them below.

Symbol-Spaced versus Fractionally-Spaced

Depending on the sample rate, an equalizer is required to operate at $L = 1$ sample/symbol (symbol-spaced equalizers) or $L > 1$ samples/symbol (fractionally-spaced equalizers).

- Symbol-spaced equalizers are also commonly denoted as T_M -spaced equalizers. A tap spacing equal to the symbol time T_M is optimum if the equalizer is preceded by an actual matched filter, i.e., a filter matched to the composite channel $h(nT_S)$ and not just the pulse shape. However, this is not often the case. The sample rate of $1/T_M$ can create a spectral null for an incorrect timing offset, a situation we encountered in Figure 7.7 during a discussion on timing synchronization. Notice the red lines depicting the baseband region in Figure 8.42 for a T_M -spaced equalizer and an overlap from the neighbouring aliases. In this text, we will mainly focus on T_M -spaced equalizers.
- In many situations, the channel response is unknown at the Rx, so the Rx filter is matched to the known pulse shape and a symbol rate sampling is not enough. Due to the adverse effects of wrong timing, it has been shown that a fractionally-spaced equalizer delivers better performance than a symbol-spaced equalizer at

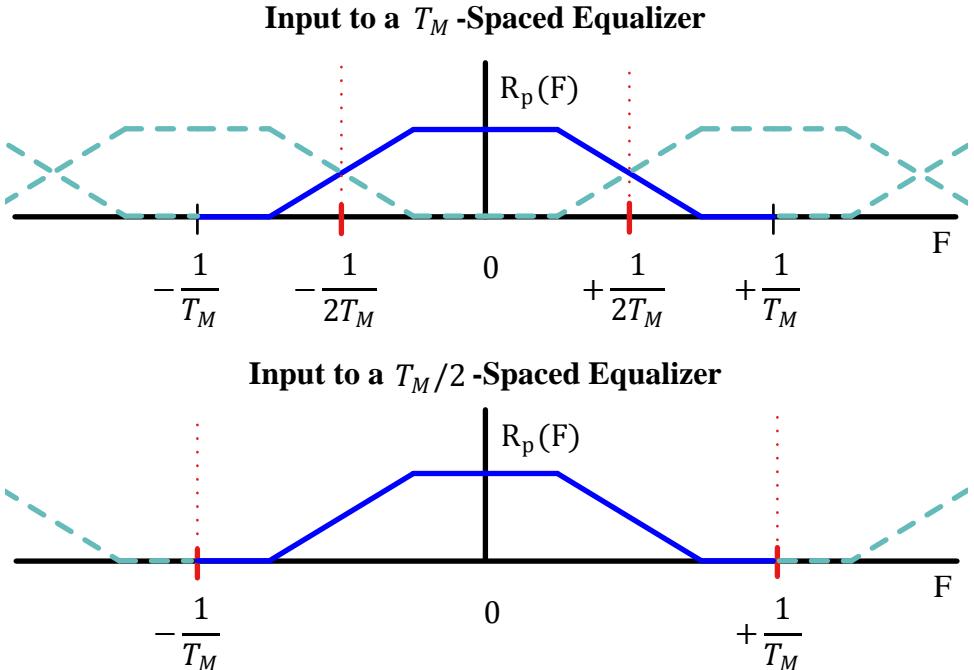


Figure 8.42: A comparison of the input to a T_M versus $T_M/2$ -spaced equalizer. Caused by a timing offset, a spectral null in the T_M -spaced equalizer at $\pm 1/2T_M$ can easily make the signal impossible to rectify

the cost of increased complexity. When employing a fractionally-spaced equalizer, it is a common practice to use $L = 2$ samples/symbol and denote it as a $T_M/2$ -spaced equalizer. Notice the red lines depicting the baseband region in Figure 8.42 in which there is no overlap from the neighbouring aliases.

While we make these deductions about SNR reduction in isolation, keep in mind that the powerful error correcting codes can recover a great deal of information buried in noise and hence a T_M -spaced equalizer suffices in many scenarios. In the discussion that follows, we drop the usual notation $z(mT_M)$ and $y(mT_M)$ in favour of $z[m]$ and $y[m]$, respectively, because it is understood that we are working with a symbol-spaced equalizer, i.e., samples taken at a rate of $F_S = 1/T_M$ or $L = 1$ sample/symbol.

Preset versus Adaptive

We said at the start of this chapter that a carrier phase, carrier frequency or a symbol timing error can be considered as a special case of a wireless channel. On an analogous note, we observe the following.

- A regular equalizer can be considered as general case of a phase, frequency or timing estimator in a feedforward scenario.
- An adaptive equalizer can be viewed as a general case of a phase, frequency or timing locked loop in a feedback setting. We can think of it as a loop with several

arms all driven by a common mechanism, as shown in Figure 8.43. A relevant analogy is an ordinary serpent compared to the *Hydra*, a multi-headed serpent in Greek mythology that is killed by Heracles as one of his 12 labours.

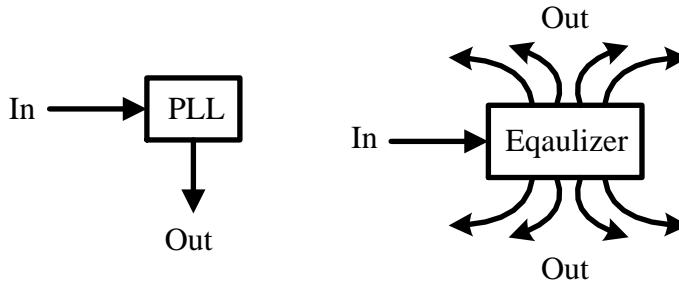


Figure 8.43: An equalizer is analogous to a multi-armed PLL

Hence, an equalizer can compensate for a carrier phase offset, a small frequency offset or a symbol timing offset (in a fractionally-spaced scenario). Nevertheless, the update process will be seriously complicated. Furthermore, the dynamics of the carrier and timing loops are much faster than the equalizer since the loops behave like a single-armed equalizer. Moreover, decoupling the individual distortions and conquering them separately in an independent manner is beneficial in terms of system complexity and performance. Even with the asymmetric pulses arising from multipath, the timing locked loop plays a crucial role of tracking the clock frequency to prevent timing drift, as long as the TED output is stable enough to not jump around when the paths come and go due to the fading. The important point is that timing recovery still brings out the symbol rate periodicity which largely stays intact even after the channel convolution.

For these reasons, an equalizer usually follows the timing synchronization process so that both the equalizer and synchronizer are not competing with each other to steer the timing. Then, the equalizer can compensate for the remaining timing error. A carrier phase loop, on the other hand, can operate at the equalizer output.

Linear versus Non-Linear

Later in Section 8.3.7, we will find that an equalizer is the most demanding and resource intensive component of a conventional wireless Rx. While the exact numbers depend on the specifications and subsequent implementation of the system, it can consume even 75% of the processing resources of a DSP machine! In high speed communication, it is impractical to remain busy processing a symbol within a certain window of time while many future symbols are arriving at the Rx. That will result in filling the buffer faster than emptying it. Therefore, it is imperative to explore simple solutions for mitigating the effect of the channel.

For this reason, an equalizer is commonly divided into two categories, linear and non-linear equalizers.

Linear equalizers: Linearity implies that the equalizers produce symbol estimates $y[m]$ based on simple linear operations on the matched filter output $z[m]$ and hence are inherently fast. The tradeoff is a higher bit error rate as well as an inability to effectively cope with harsh channels.

Non-linear equalizers: In this category, the equalizers generate the symbol outputs through more complicated mathematical operations than simple linear operations. On the up side, they perform significantly better in dealing with a frequency selective channel than linear equalizers in terms of the output bit error rate.

Time versus Frequency Domain

We have learned that equalization complexity is higher than any other signal processing task in a high rate wireless system. It turns out that beyond a certain rate, it is better to transform the signal into frequency domain through an FFT-iFFT pair and equalize the channel in frequency domain. With an ever growing demand for higher data rates, equalization in frequency domain has been the force behind most modern wireless standards.

Data-Aided/Decision-Directed versus Blind

Like synchronizers, many equalizers start their operation by processing a known training sequence embedded within the Rx signal. It is expected that at the end of training, the equalizer has reduced the channel impact to a level where it can switch to a decision-directed mode. Then, it can continue the process either until the end of the packet for burst transmissions or indefinitely for continuous transmissions. On the other hand, blind equalizers work without any knowledge of the data or the channel and switch into a decision-directed mode only after crossing a certain threshold for convergence.

All these categories are summarized in Figure 8.44. Finally, some equalizers require the knowledge of the channel impulse response at the Rx while many others do not. This requirement will become clear from the context of the algorithm.

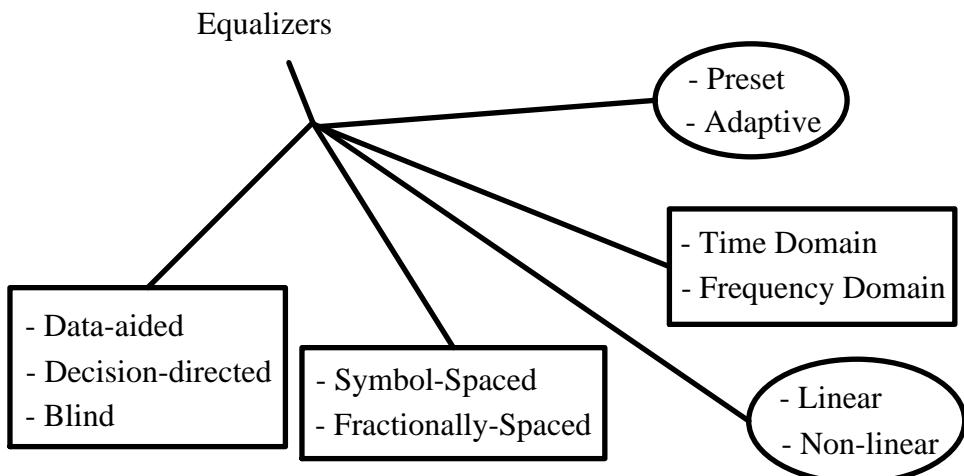


Figure 8.44: Classification of equalization algorithms

We start with an introduction to the principle of correlation applied to the equalization problem.

8.3.2 Brute Force Equalization: ML Sequence Estimation

Having known the channel, we can invite our old friend, *the correlation principle*, to solve the equalization problem. However, we will use a slightly more general concept as follows.

The data symbols $a[m]$ after upsampling at the Tx pass through a composite channel $h[m]$ comprising of the Tx filter, the baseband channel impulse response and the Rx filter as mentioned previously in Eq 8.38. After downsampling to $L = 1$ sample/symbol[†],

$$z[m] = a[m] * h[m] \quad (8.42)$$

Now although we do not have any knowledge of the Tx symbols $a[m]$ at the Rx, we do know that they come from a finite set of size M depending on the modulation type. For example, M is equal to 2 in BPSK and 4 in QPSK modulation. For a binary modulation scheme ($M = 2$) and three data symbols ($N_d = 3$) as an example, we get $M^{N_d} = 2^3 = 8$ options as shown in Figure 8.45.

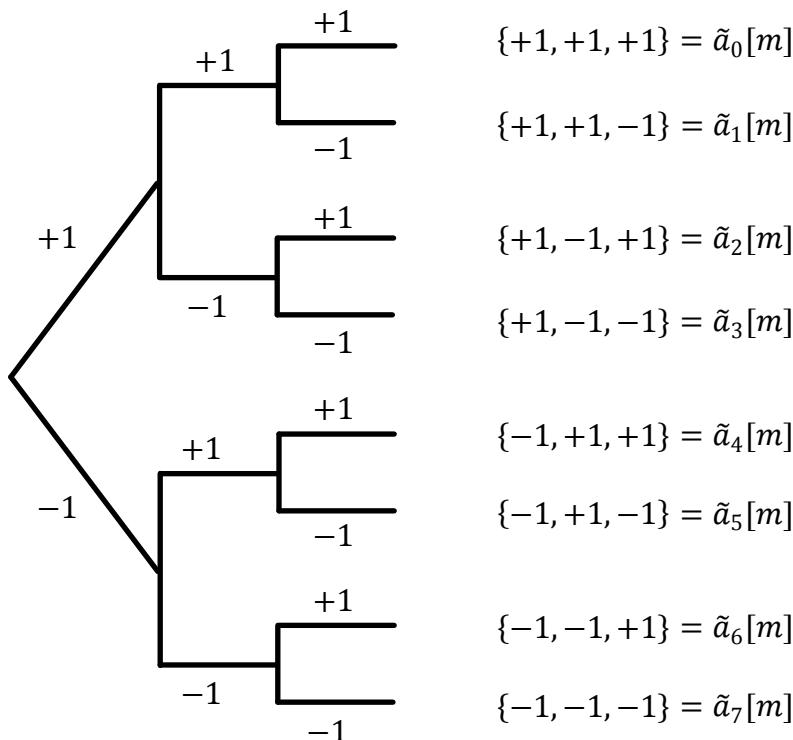


Figure 8.45: Possible data sequences for a binary modulation scheme with length $N_d = 3$ symbols

[†]There is an implicit assumption that the input signal has been filtered through an exact matched filter as well as noise whitening filter for optimal performance as described in Note 8.10. Without any knowledge of the channel, we just utilize a regular matched filter $p(-nT_S)$ in this setup. Furthermore, since measuring the performance is out of the scope of this text, we ignore the noise whitening filter in this discussion.

Also, the channel impulse response can be known through a channel estimation strategy such as in Section 8.2. Armed with this knowledge, we can convolve the channel impulse response estimate $\hat{h}[m]$ with all possible options for the symbol sequence $\tilde{a}[m]$ *at the Rx*. As a result, we get $M^{N_d} = 2^3 = 8$ possible Rx sequences.

$$\tilde{z}_0[m] = \tilde{a}_0[m] * \hat{h}[m]$$

$$\tilde{z}_1[m] = \tilde{a}_1[m] * \hat{h}[m]$$

$$\tilde{z}_2[m] = \tilde{a}_2[m] * \hat{h}[m]$$

$$\tilde{z}_3[m] = \tilde{a}_3[m] * \hat{h}[m]$$

$$\tilde{z}_4[m] = \tilde{a}_4[m] * \hat{h}[m]$$

$$\tilde{z}_5[m] = \tilde{a}_5[m] * \hat{h}[m]$$

$$\tilde{z}_6[m] = \tilde{a}_6[m] * \hat{h}[m]$$

$$\tilde{z}_7[m] = \tilde{a}_7[m] * \hat{h}[m]$$

Let us call each such option $\tilde{z}_i[m]$ a candidate sequence. Then, a strategy is to use a slightly more general concept than correlation, i.e., we can find the error magnitudes between the Rx sequence $z[m]$ and the candidate sequences $\tilde{z}_i[m]$.

$$e_i = \sum_m |z[m] - \tilde{z}_i[m]|^2 \quad (8.43)$$

The mostly likely Tx sequence is the one with the smallest such error. Note that the cross term $z^*[m]\tilde{z}_i[m]$ represents the correlation when we open the magnitude squared expression. The term $|z[m]|^2$ is common to all the error terms and hence not needed. However, the term $|\tilde{z}_i[m]|^2$ comes in due to different energies involved and subsequently required in addition to the correlation expression.

It turns out that this strategy is optimal in the sense of maximizing the likelihood of detected symbols in the presence of AWGN (as such a correlation process should do) and hence known as *Maximum Likelihood Sequence Estimation (MLSE)*. The first known commercial application of MLSE was in a Codex 9.6 kbps voiceband modem of 1969 [31]. The significant disadvantage of this approach is its computational complexity. For a sequence length N_d and modulation size M , there are M^{N_d} possible sequences which is usually a very very large number. Even for a binary modulation with $M = 2$ and a data length of just 10 symbols, this is equal to $2^{N_d} = 1024$ candidate sequences to compare!

The Viterbi Algorithm

To overcome this problem, the most widely known algorithm in digital communications, *the Viterbi algorithm*, can be employed. Viterbi algorithm breaks this process down in stages and works by only retaining the most likely candidates at each step. We discuss the Viterbi algorithm with the help of a simple setup of a binary modulation and a 2 tap channel as described in Ref. [34]. From Eq (8.42), we can write

$$z[m] = a[m]h[0] + a[m-1]h[1]$$

Let one term from the error sum in Eq (8.43) be the metric

$$M\{a[m], a[m-1]\} = |z[m] - a[m]h[0] - a[m-1]h[1]|^2$$

Now consider a diagram known as a trellis in Figure 8.46.

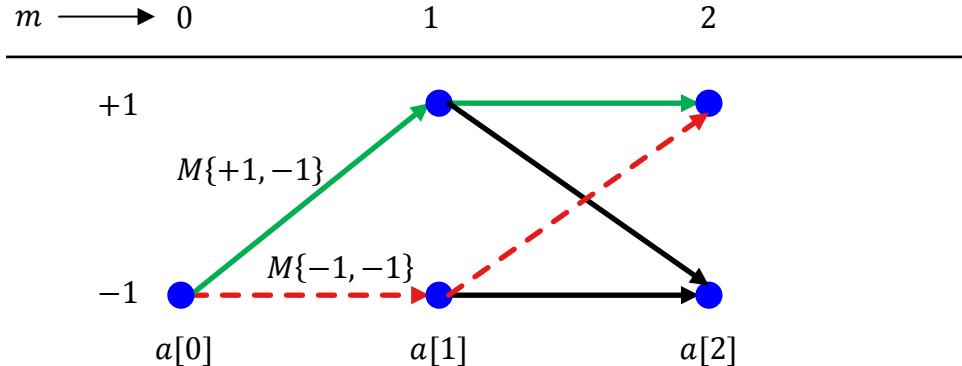


Figure 8.46: A trellis for implementing Viterbi algorithm

- At time $m = 0$, let $a[0] = -1$ be a single known training symbol and we need to detect the remaining symbols for $m > 0$.
- At time $m = 1$, there are two possible values for the metric $M\{a[1], a[0]\}$ because $a[1]$ can attain a value of either $+1$ or -1 . These two metrics correspond to two different branches drawn in Figure 8.46 between $m = 0$ and $m = 1$. We say that the two branches end in two ‘states’ that depend on the possible values for $a[1]$.
- At $m = 2$, there are now four possible branches, two from each state, corresponding to the possible values for the symbol pair $(a[1], a[2])$. Observe from the figure that two branches end at the state corresponding to $a[2] = -1$ while two end at the state corresponding to $a[2] = +1$. The Viterbi algorithm dictates that at each state, only the sequence with minimum error is kept and the other is pruned. This minimum error is determined by the candidate *path metrics* which are an accumulation of $M\{a[m], a[m-1]\}$. And the process continues.

As an example, at state corresponding to $a[2] = +1$, there are only two possibilities for $a[1]$ since $a[2]$ is already determined.

$$a[2] = +1 \quad \rightarrow \quad a[1] = +1 \text{ or } a[1] = -1$$

Since $a[0] = -1$, the corresponding candidate metrics are as follows.

$$a[2], a[1] \qquad \qquad a[1], a[0]$$

$$\begin{array}{rcl} M\{+1, +1\} & + & M\{+1, -1\} \\ M\{+1, -1\} & + & M\{-1, -1\} \end{array}$$

where the former is shown as a solid green line while the latter as a dashed red line from $a[0]$ to $a[2]$ in Figure 8.46. The largest such metric determines which path is kept and its metric accumulated while the other is discarded. This results in each stage possessing a ‘path history’ that determines the sequence for all the previous symbols. In the end, only the most likely sequence survives.

The benefit of the Viterbi algorithm over brute force sequence estimation is that we do not need to process all possible candidate sequences from the start to finish since the candidates are regularly discarded at each step. For a binary modulation scheme, length $N_{\text{Tap}} + 1$ channel and data sequence length N_d , it reduces the brute force estimation of 2^{N_d} comparisons to N_d stages of $2^{N_{\text{Tap}}}$ comparisons through discarding the trellis paths (keep in mind that $N_{\text{Tap}} + 1$ is much smaller than N_d). Nonetheless, it is also prohibitive to implement in most wireless channels for realistic existing packet lengths, channel lengths and modulation sizes.

Example 8.2

The Viterbi algorithm did find its application in GSM, the popular 2G cellular standard (R.I.P.). With the specs such as

- binary modulation $\{+1, -1\}$,
- low symbol rate $1/T_M = 271 \text{ kHz}$, or symbol time $T_M = 3.69 \mu\text{s}$, and
- typical maximum delay spread T_{Del} of a few microseconds in urban environments,

the resultant channel has 4 – 6 taps rendering the maximum likelihood sequence estimation possible. It is important to know that not all ISI in the Rx signal in a GSM system comes from the wireless channel. A portion of it arises due to Gaussian pulse shaping at the Tx which is not a Nyquist filter. This is done to trade-off some higher levels of ISI with increased spectral efficiency.

We feel a little annoyance when we forget an idea or something we are about to say. When that happens, starting with the last thought I remember, I list the next possible thoughts and compare how likely each of them can be. Keeping the one with highest likelihood, I repeat this process and often traverse the original route to capture what had escaped my mind. This could become Viterbi algorithm if I had some mechanism to incorporate an accumulation of path metrics.

A common use of maximum likelihood sequence estimation is during the design stage of the Rx, against which the potential performance of any equalization technique is benchmarked. We now shift towards finding some computationally efficient equalizers.

8.3.3 Linear Equalizers

The most common suboptimum strategy is to employ an FIR digital filter as an equalizer at the Rx with $2L_q + 1$ taps (or coefficients) $q[m]$. As with the channel, the extra 1 tap refers to the main tap usually placed at the center of the equalizer response. The input to the equalizer is the matched filter output $z[m]$ while its output is the best (in terms of a chosen criterion) estimate of the Tx symbol $y[m]$ that can be used for symbol decision $\hat{a}[m]$. These taps are adjusted in a manner such that the filter output

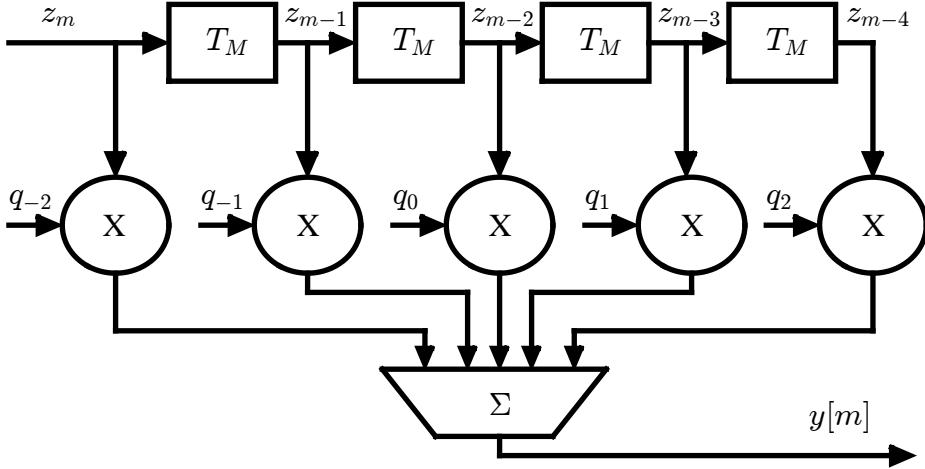


Figure 8.47: Matched filter output $z[m]$ input to an equalizer with taps (or coefficients) $q[m]$ that generates an estimate of the symbol $y[m]$ that can be used for symbol decision $\hat{a}[m]$. Complex operations are implied

is an ISI-eliminated (or at least ISI-reduced) version of the Tx symbols.

$$\begin{aligned} y[m] &= z[m] * q[m] \\ &= \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \end{aligned} \quad (8.44)$$

A block diagram of this approach is illustrated in Figure 8.47. A tapped delay line or a transversal filter are also common names for an FIR filter that is implemented in a manner to directly compute the sum in a convolution operation. Here, each of the two inputs to the adder is coupled with a fixed coefficient multiplier to form a Multiply-Accumulate (MAC) although we just bunched all of them together into one summer. For reference, the other option is a transposed structure of the same equation in which the registers operate as partial sum accumulators and is more popular with the hardware engineers. Since filtering is a linear operation, the resulting structure is called a linear equalizer. Linearity implies that the output is a scaled and summed version of the input.

Next, we discuss an algorithm known as the *Zero Forcing Equalizer (ZFE)* and briefly touch on another known as a *Minimum Mean Square Error Equalizer*. As compared to the maximum likelihood sequence estimation, the performance of these linear equalizers is quite ordinary. However, when implemented within a decision feedback structure to be discussed later, their performance at good SNR becomes reasonably accurate.

Zero Forcing Equalizer (ZFE) in Time Domain

From the previous discussions and particularly the ISI demonstration in Figure 8.33, our main problem is a distorted pulse shape arriving at the Rx that does not pass through zeros at integer multiples of symbol time T_M while being unity at the desired

symbol instant m . In other words, the cascade of the Tx pulse shaping filter, the channel filter and the Rx matched filter does not satisfy the Nyquist no-ISI criterion discussed in Section 3.6 anymore. Using $h[m]$ instead of the pulse auto-correlation $r_p[m]$ (due to the inclusion of the channel response), we can write

$$h[m] \neq \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases}$$

Now expand the equalizer output as

$$\begin{aligned} y[m] &= z[m] * q[m] = \left\{ a[m] * h[m] \right\} * q[m] \\ &= a[m] * \left\{ h[m] * q[m] \right\} = a[m] * \underbrace{\left\{ \sum_{l=-L_q}^{L_q} q[l] \cdot h[m-l] \right\}}_{\text{ZFE tries to force this as } \delta[m]} \end{aligned}$$

We realize that at each symbol instant m , we get the actual undistorted symbol $a[m]$ back if the convolution of our linear equalizer impulse response $q[m]$ with the channel impulse response $h[m]$ yields an overall Nyquist response.

$$q[m] * h[m] = \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases} \quad (8.45)$$

Since the equalizer taps are in our own control as opposed to the channel, the simplest strategy to achieve such a no-ISI condition is to force it ourselves. So as the name says, a *Zero Forcing Equalizer tries to force the equalizer output to pass through zeros* at all multiples of the symbol time T_M , while the output at the main symbol instant m is chosen to be unity. In that sense, a zero forcing equalizer tries to mimic the cascade of the channel filter and equalizer filter as having an output that is unity at a particular symbol time and zero for all other multiples of T_M , thus fulfilling the Nyquist no-ISI criterion.

$$\sum_{l=-L_q}^{L_q} q[l] \cdot h[m-l] = \delta[m] \quad (8.46)$$

The ideal output based on the zero forcing principle is drawn in Figure 8.48. We will shortly see that achieving this ideal response is not possible and there is always a residual ISI left in the Rx symbols as a result.

It is important to note that the first tap might not yield the optimal results due to the scattered channel energy within the channel taps. Therefore, it is a usual practice to introduce an equalization delay m_0 at the output. The value of m_0 can be found through the tap with the largest energy. Consequently, the above equation can be recast as

$$\sum_{l=-L_q}^{L_q} q[l] \cdot h[m-l] = \delta[m - m_0] \quad (8.47)$$

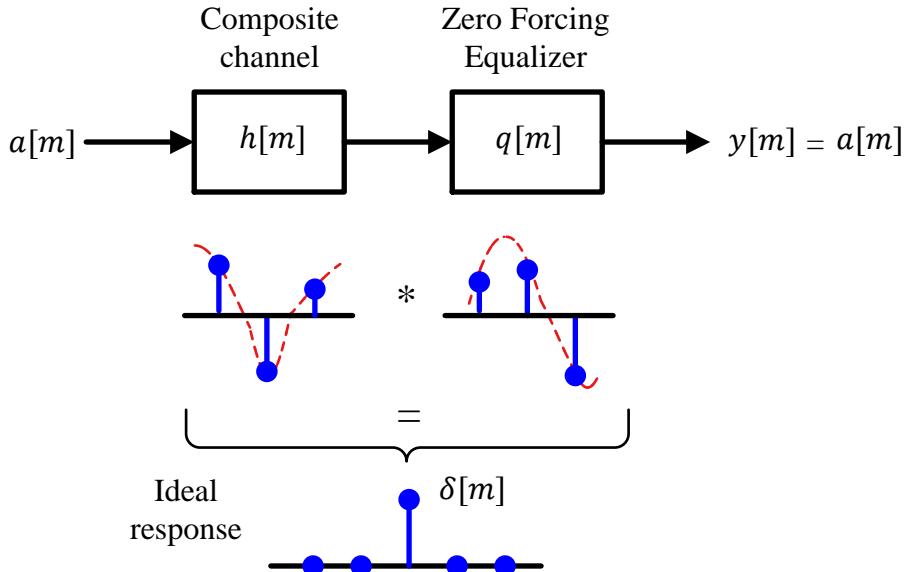


Figure 8.48: Ideal output based on the zero forcing principle which is not possible to achieve

Recalling that the channel length is given by $N_{\text{Tap}} + 1$, the length of the convolution output is given by

$$\begin{aligned} \text{Length}\{q[m]\} + \text{Length}\{h[m]\} - 1 &= 2L_q + 1 + N_{\text{Tap}} + 1 - 1 \\ &= 2L_q + N_{\text{Tap}} + 1 \end{aligned}$$

This equation tells us that there should be $2L_q + N_{\text{Tap}} + 1$ values in our constructed sequence $\delta[m - m_0]$, i.e., $2L_q + N_{\text{Tap}}$ values of m that are zero and only one value that is 1 at $m = m_0$. However, the equalizer has only $2L_q + 1$ adjustable parameters (equalizer coefficients). For this reason, unique equalizer coefficients cannot be chosen to force the remaining

$$(2L_q + N_{\text{Tap}} + 1) - (2L_q + 1) = N_{\text{Tap}}$$

values to zero. Consequently, *N_{Tap} values outside the equalization range are in general nonzero* and contribute towards the residual ISI at its output. In general, therefore, it is impossible to completely eliminate the ISI at the output of the zero forcing equalizer.

Computing the ZFE Taps

The zero forcing equalizer coefficients can be computed through solving the set of equations in Eq (8.47) above. For this purpose, we break Eq (8.47) into subsequent

equations for each m . For example[†],

$$\begin{aligned} & \vdots \quad \vdots \\ & \sum_{l=-L_q}^{L_q} q[l] \cdot h[(m_0 - 1) - l] = \delta[\overline{m_0 - 1} - m_0] = \delta[-1] = 0 \\ & \sum_{l=-L_q}^{L_q} q[l] \cdot h[m_0 - l] = \delta[m_0 - m_0] = \delta[0] = 1 \quad (8.48) \\ & \sum_{l=-L_q}^{L_q} q[l] \cdot h[(m_0 + 1) - l] = \delta[\overline{m_0 + 1} - m_0] = \delta[+1] = 0 \\ & \vdots \quad \vdots \end{aligned}$$

It is evident that $2L_q + 1$ such equations can be constructed from which the taps of a zero forcing equalizer can be found. Let us consider an example for a 5 tap channel and a 3 tap ZFE.

Example 8.3

Assume that the impulse response estimate $\hat{h}[m]$ of the channel at the output of the channel estimator is perfect, i.e., it is equal to $h[m]$ and consists of the following 5 taps.

$$h[m] = [0.1 \ 0.1 \ 0.67 \ 0.19 \ -0.02]$$

with tap 2 being the largest. Since we choose the equalizer length as 3, the output of the equalizer has a length $5 + 3 - 1 = 7$. A ZFE with $2L_q + 1 = 3$ taps implies that $L_q = 1$. Plugging in these values for $m_0 = 2$ in Eq (8.48), we have

$$\begin{aligned} q[-1]h[2] + q[0]h[1] + q[1]h[0] &= 0 \\ q[-1]h[3] + q[0]h[2] + q[1]h[1] &= 1 \\ q[-1]h[4] + q[0]h[3] + q[1]h[2] &= 0 \end{aligned}$$

Solving this set of simultaneous equations through substitution or matrix formation in linear algebra or using a programming language, we get the equalizer taps $q[m]$ as

$$q[m] = [-0.1714 \ 1.6101 \ -0.4617]$$

The composite channel estimate $h[m]$, the equalizer taps $q[m]$ and the output of their convolution are drawn in Figure 8.49. Observe that the middle portion of the output is marked by the arrows showing ISI free region $[0 \ 1 \ 0]$, a condition forced by us. However, the ZFE cannot control the ISI from $7 - 3 = 4$ samples (or $N_{\text{Tap}} = 4$ samples), two on each side. This can also be seen from the response at the equalizer output, rounded to

[†]There is a zero forcing Least Squares (LS) solution in which $h[m]$ are stacked in a matrix \mathbf{H} and the unit impulse signal in a vector \mathbf{u} . Then, you would see an expression as

$$\mathbf{q}_{\text{LS}} = (\mathbf{H}^* \mathbf{H})^{-1} \mathbf{H}^* \mathbf{u}$$

two decimal places.

$$z[m] = q[m] * h[m] = \left[\begin{array}{cccccc} -0.02 & 0.14 & \underbrace{0 \quad 1 \quad 0}_{\text{forced to zero}} & -0.12 & 0.01 \end{array} \right]$$

It also shows that when a sequence of data symbols is transmitted through the same channel and compensated by the ZFE in this example, the interference from each symbol will not affect the two neighbouring symbols but the second and third samples before and after each symbol. The actual amount of introduced ISI depends on the channel impulse response at that instant which was quite mild for this example.

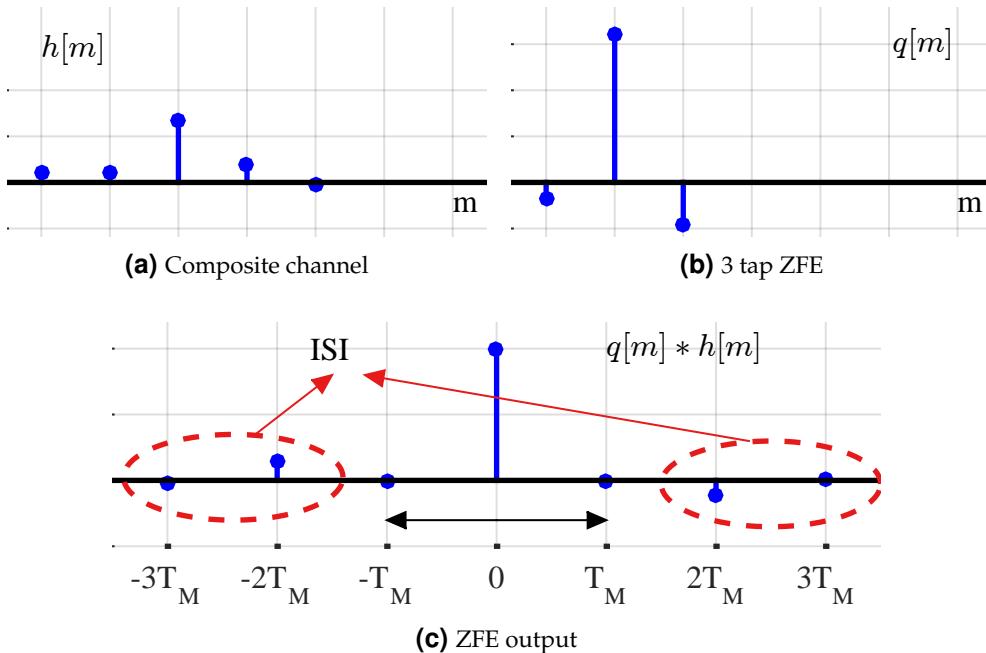


Figure 8.49: Principle of zero forcing equalizer. The output is a unit impulse only for 3 middle samples marked by the arrows, while the remaining samples contribute ISI in the Rx symbols

What happens when the equalizer length is increased? For the same channel length and a longer equalizer, the ISI free region would have been long but it can still not force the outer N_{Tap} samples – one less than the channel length – to zero. This can be seen by increasing the equalizer length to $2L_q + 1 = 5$ taps. Expanding the set of Eq (8.48), using

$$h[-2] = h[-1] = h[5] = h[6] = 0,$$

and assigning the channel indices from 0 to 6 (instead of 1 to 7), we get

$$\begin{aligned} q[-2]h[2] + q[-1]h[1] + q[0]h[0] &= 0 \\ q[-2]h[3] + q[-1]h[2] + q[0]h[1] + q[1]h[0] &= 0 \\ q[-2]h[4] + q[-1]h[3] + q[0]h[2] + q[1]h[1] + q[2]h[0] &= 1 \\ q[-1]h[4] + q[0]h[3] + q[1]h[2] + q[2]h[1] &= 0 \\ q[0]h[4] + q[1]h[3] + q[2]h[2] &= 0 \end{aligned}$$

Solving this set of simultaneous equations again through substitution or matrix formation in linear algebra or using a programming language, we get the equalizer taps $q[m]$ as

$$q[m] = [-0.2176 \ -0.1004 \ 1.5580 \ -0.4717 \ 0.1803]$$

The output of the convolution between the composite channel $h[m]$ and $q[m]$ is drawn in Figure 8.50. Again observe that the middle portion of the output is marked by the arrows showing a longer ISI free region which was forced to become $[0 \ 0 \ 1 \ 0 \ 0]$ by us. However, the ZFE cannot control the ISI from $9 - 5 = 4$ samples, two on each side, i.e., N_{Tap} samples in total. But this time the equalizer output response, rounded to two decimal places, is

$$z[m] = q[m] * h[m] = [-0.02 \ -0.03 \ \underbrace{0 \ 0 \ 1 \ 0 \ 0}_{\text{forced to zero}} \ 0.04 \ 0]$$

which manifests a considerably reduced amount of ISI as compared to a 3 tap equalizer in Figure 8.49c.

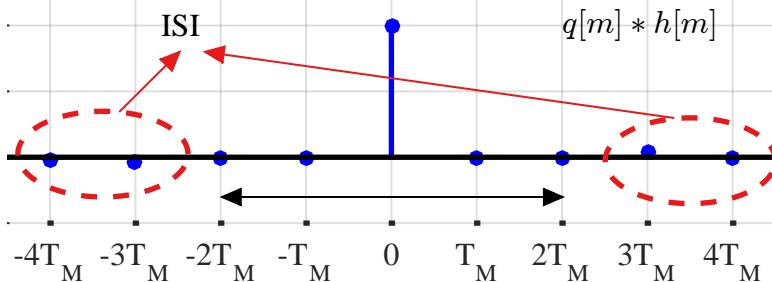


Figure 8.50: Output for a 5 tap ZFE. Observe that the ISI is still present in N_{Tap} samples but reduced in amplitude

The above example deals with a mild channel. For many channel realizations, the equalizer length should be three to five times longer than the length of the channel to properly deconvolve its effect [28].

Zero Forcing Equalizer (ZFE) in Frequency Domain

To view this process in frequency domain, suppose that the equalizer length is increased towards infinity such that the ISI in the outer samples becomes negligible.

$$L_q \rightarrow \infty$$

Then, Eq (8.45) reproduced below becomes an exact unit impulse and the ISI can be completely eliminated.

$$q[m] * h[m] = \begin{cases} 1, & m = 0 \\ 0, & m \neq 0 \end{cases}$$

We can take the Fourier Transform of such a unit impulse from Section 1.10.3 which is a constant equal to 1 at all frequencies.

Now taking the Fourier Transform of both sides of the above equation,

$$Q(F) \cdot H(F) = 1 \quad (8.49)$$

where $Q(F)$ and $H(F)$ are the frequency responses of the equalizer taps $q[m]$ and the composite channel $h[m]$, respectively. From here, the frequency domain expression for the zero forcing equalizer can be obtained as

$$Q(F) = \frac{1}{H(F)} \quad (8.50)$$

This expression tells us that *a zero forcing equalizer simply inverts the channel frequency response*. It scales the magnitude response by the same value while cancels the phase through negating it.

In decibels, the magnitude response in Eq (8.49) can be written as

$$20 \log_{10} |Q(F)| + 20 \log_{10} |H(F)| = 0 \quad (8.51)$$

for an infinite length zero forcing equalizer. A diagram to illustrate this concept is drawn in Figure 8.51 where the magnitude responses in dBs of both the channel and the equalizer are shown. The equalizer can be seen attempting to restore the unit amplitude by inverting the channel. Due to the logarithms involved, unit amplitude is 0 in dBs and inverting a quantity is the same as negating it in dBs.

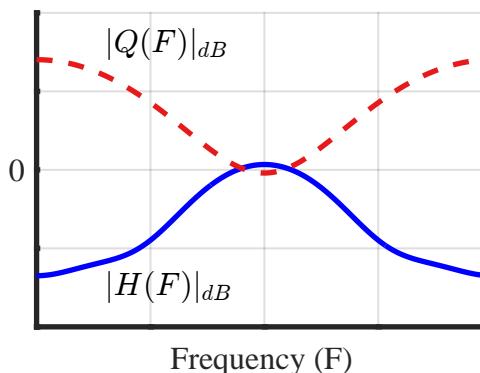


Figure 8.51: A ZFE compensates for the channel by inverting its frequency response (negating in dB)

As stated before, perfect inversion is possible only for an infinite length equalizer. Now we can see why a ZFE cannot completely eliminate the channel distortion. This is because it is a Finite Impulse Response (FIR) filter that is trying to act as an inverse

filter to another FIR filter: the channel frequency response $H(F)$. If we had learned the concept of FIR filters through a z -Transform in digital signal processing, we could immediately see that the inverse of an FIR filter is an Infinite Impulse Response (IIR) filter.

In addition to the residual ISI, now we have a look at another more significant drawback of a zero forcing equalizer, namely noise enhancement.

Noise Enhancement in a ZFE

Until now, we did not consider the additive noise $w[m]$ in our setup. First, we rewrite the matched filter output from Eq (8.42) below.

$$z[m] = a[m] * h[m]$$

After passing through the equalizer, the output signal is

$$y[m] = z[m] * q[m] = a[m] * h[m] * q[m]$$

However, when the AWGN is included, the equalizer output is modified as

$$\begin{aligned} y[m] &= (a[m] * h[m] + w[m]) * q[m] \\ &= \underbrace{a[m] * h[m] * q[m]}_{\text{Signal out}} + \underbrace{w[m] * q[m]}_{\text{Noise out}} \end{aligned}$$

i.e., the ZFE – being a linear operator – acts both on the signal and the noise separately.

A serious problem can be visualized by taking the above equation in frequency domain[†].

$$Y(F) = A(F) \cdot H(F) \cdot Q(F) + W(F) \cdot Q(F)$$

Since $Q(F) = 1/H(F)$, the ZFE response $Q(F)$ has a sharp peak wherever the channel possesses a spectral null. A visualization of this phenomenon is drawn in Figure 8.52.

Next, such a response is multiplied with noise that has a flat spectral density at all frequencies. As a result of this operation, the noise at the output of the equalizer exhibits an infinite noise power spectral density. In practical scenarios, the noise is amplified at frequencies where the channel has a high attenuation, commonly known as noise enhancement. Notice in Figure 8.52 how the peak in ZFE response due to the spectral null in the channel amplifies the noise at those frequencies.

To overcome the noise enhancement problem, a different type of a linear equalizer known as minimum mean square error equalizer is studied next.

Minimum Mean Square Error (MMSE) Equalizer

The fundamental principle behind a Minimum Mean Square Error (MMSE) Equalizer is to focus on the error between the current symbol $a[m]$ at symbol time m and its estimate at the output of the detector $\hat{a}[m]$.

$$e[m] = a[m] - \hat{a}[m]$$

[†]We loosely use the expressions $A(F)$ and $W(F)$ for the Fourier Transforms of $a[m]$ and $w[m]$, respectively, although these are random sequences and should be described in terms of their spectral densities.

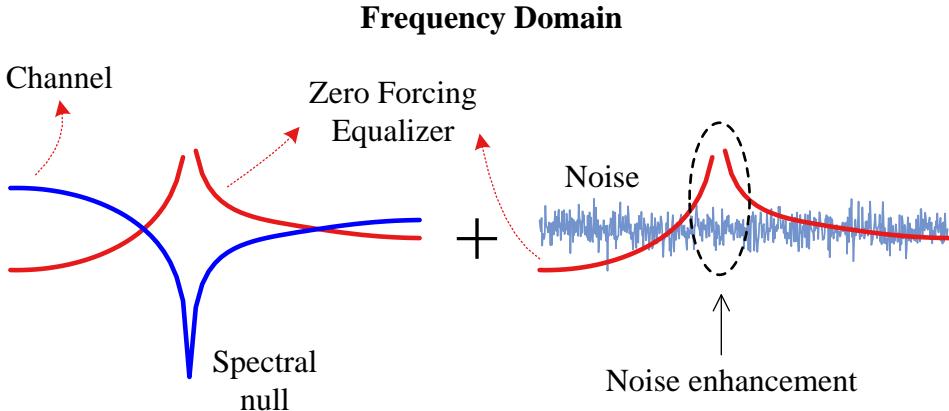


Figure 8.52: Noise enhancement by a ZFE at the frequencies where the channel has a high attenuation

However, $\hat{a}[m]$ is a quantized version of the equalizer output $y[m]$, e.g., $\hat{a}[m]$ is the sign of $y[m]$ for a binary modulation scheme.

$$\hat{a}[m] = \text{sign} \{y[m]\}$$

Two points need to be considered here.

- It is obvious that a reasonable amount of information is lost as a result of such quantization. Therefore, it is more prudent to utilize $y[m]$ directly. With $z[m]$ as the matched filter output and the equalizer input,

$$\begin{aligned} e[m] &= a[m] - y[m] = a[m] - (z[m] * q[m]) \\ &= a[m] - \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \end{aligned} \quad (8.52)$$

- The data symbols $a[m]$ can be complex valued. Furthermore, the error $e[m]$ can be both positive or negative. For these reasons, the equalizer taps should be adjusted to *minimize the magnitude squared value of the error*.

$$|e[m]|^2 = \left| a[m] - \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \right|^2 \quad (8.53)$$

To minimize this magnitude squared error, take the derivative of the above expression with respect to each equalizer tap $q[l]$. To keep the discussion simple, we will consider a binary modulation scheme here as well as a real channel and hence *zero quadrature components*.

$$\frac{d}{dq[l]} |e[m]|^2 = 2e[m] \cdot \dot{e}[m] \quad (8.54)$$

for each $l = -L_q, \dots, L_q$

Utilizing the definition of $e[m]$ from Eq (8.52), $\dot{e}[m]$ can be calculated as

$$\dot{e}[m] = \frac{d}{dq[l]} \left(a[m] - \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \right) = -z[m-l]$$

This is because the derivative with respect to $q[l]$ is nonzero only for the term involving one tap: $q[l]$. Let us plug this back in Eq (8.54) to get

$$\frac{d}{dq[l]} |e[m]|^2 = -2e[m] \cdot z[m-l]$$

for each $l = -L_q, \dots, L_q$

(8.55)

There are two ways to proceed from here.

1. Use the above expression in its current form, which leads to the structure for the most well known adaptive equalization scheme called the Least Mean Square (LMS) equalization. We will discuss this route in Section 8.3.4.
2. Take the statistical expectation of the square error above, which leads to the Minimum Mean Square Error (MMSE) equalization. To avoid getting into a probabilistic framework, I will briefly touch on this route.

For the expression for an MMSE equalizer, open Eq (8.55) by inserting the value of $e[m]$ with a changed index in the summation as follows.

$$\frac{d}{dq[l]} |e[m]|^2 = -2 \left(a[m] - \sum_{i=-L_q}^{L_q} q[i]z[m-i] \right) z[m-l]$$

The reason behind this is that for each l , the whole convolution sum is computed. This process is then repeated for each l from $-L_q$ to L_q . It can now be modified as

$$\frac{d}{dq[l]} |e[m]|^2 = -2a[m]z[m-l] + 2 \sum_{i=-L_q}^{L_q} q[i]z[m-i] \cdot z[m-l]$$

Take mean on both sides of the equation above to turn the squared error into mean squared error. Moreover, since this is a derivative, we can equate it to zero to find its maximum.

$$\begin{aligned} \frac{d}{dq[l]} \text{Mean}|e[m]|^2 &= -2 \text{Mean}\{a[m] \cdot z[m-l]\} + \\ &2 \sum_{i=-L_q}^{L_q} q[i] \cdot \text{Mean}\{z[m-i] \cdot z[m-l]\} = 0 \end{aligned}$$

Next, we can write

$$\text{Mean}\{a[m] \cdot z[m-l]\} = \sum_{i=-L_q}^{L_q} q[i] \text{Mean}\{z[m-i] \cdot z[m-l]\} \quad (8.56)$$

Now recall from Section 1.6 that correlation between two arbitrary signals $f[m]$ and $g[m]$ is defined as

$$\text{corr}_{fg}[n] = \sum_{m=-\infty}^{\infty} f[m]g[m-n]$$

So we can define the following two correlations.

$$\sum_{m=-\infty}^{\infty} a[m] \cdot z[m-l] = \text{corr}_{az}[l] \quad (8.57)$$

and

$$\begin{aligned} \sum_{m=-\infty}^{\infty} z[m-i] \cdot z[m-l] &= \sum_{m=-\infty}^{\infty} z[m-i] \cdot z[(m-i) - (l-i)] \\ &= \text{corr}_z[l-i] \end{aligned} \quad (8.58)$$

Plugging these definitions back into Eq (8.56) produces the final set of equations for an MMSE equalizer. Here, we are replacing the statistical expectation with an averaging operation which is not exactly correct but can be adopted, for example, for real-time computations of the coefficients.

$$\sum_{i=-L_q}^{L_q} q[i] \cdot \text{corr}_z[l-i] = \text{corr}_{az}[l] \quad (8.59)$$

for each $l = -L_q, \dots, L_q$

While we will not pursue MMSE equalizer any further[†], we should know that the MMSE equalizer achieves a balance between inverting the channel and enhancing the noise. Therefore, it exhibits a superior performance as compared to a zero forcing equalizer.

Note 8.11 Wiener filter

The minimum mean square error solution presented above was invented by Wiener in 1940s and hence also known as the *Wiener filter*. Back then, filters were usually implemented through electrical elements lumped together in a circuit design to output a desired frequency response. It was mostly trial and error and there was no way to know the optimal performance against which their design could be gauged. Wiener basically opened the door to optimal filtering by presenting this problem in the minimum mean square error sense.

However, transforming these simple mathematical equations into circuit elements proved to be a challenging task for the engineers at that time. Consequently, it was often impractical to implement the discrete-time version of the Wiener filter until Kalman, Widrow and Hoff appeared on the scene in early 1960s [35]. Kalman came up with the *Kalman filter* for optimal linear filtering problem and Widrow and Hoff proposed the most practical adaptive algorithm to date known as the *Least Mean Square (LMS)* algorithm. We will discuss the equalizer based on LMS algorithm in Section 8.3.4.

Now we turn our attention towards the main adaptive equalization algorithm widely utilized in digital communication systems.

[†]The next step is to employ linear algebra and arrange these terms in a matrix formulation.

$$\mathbf{R} \cdot \mathbf{q} = \mathbf{p}$$

from which the MMSE equalizer weights are computed as

$$\mathbf{q}_{\text{MMSE}} = \mathbf{R}^{-1} \mathbf{p}$$

8.3.4 Least Mean Square (LMS) Equalizer

We start with a motivation to develop an automatic equalizer with self-adjusting taps and then discuss the Least Mean Square (LMS) equalizer.

Channel state information: In developing the brute force equalizer (maximum likelihood sequence estimation) and linear equalizers, we assumed that perfect channel state information is available at the Rx. While this information, commonly known as Channel State Information (CSI), can be gained from a training sequence embedded in the Rx signal, the channel characteristics are unknown in many other situations.

Nevertheless, the quality of the channel estimation is only as good as the channel itself. As the wireless channel deteriorates, so does the reliability on its estimate.

Time variation: Even when the wireless channel is known to a reasonably accurate level, it eventually changes after some time. In Section 8.1.5, we studied in detail how time variations in the channel unfold on the scale of the coherence time T_C and how this impacts the Rx signal. For that reason, an equalizer needs to automatically adjust its coefficients in response to the channel variations. Nature favours those who adapt.

Utilization of available information: In situations where the channel is estimated from a training sequence and a fixed equalizer is employed, it is difficult to incorporate further information obtained from the data symbols. For an adaptive equalizer, the taps can be adjusted first from the training sequence and then easily driven through the data symbols out of the detector in a decision-directed manner in real time.

The LMS algorithm was first proposed by Widrow (a professor at Stanford University) and his PhD student Hoff in the 1960s. Due to its simplicity and robustness, it has been the most widely used adaptive filtering algorithm in real applications. An LMS equalizer in communication system design is just one of those beautiful examples and its other applications include noise and echo cancellation, beamforming, neural networks and so on. We first develop an intuitive understanding of its operation.

An Intuitive Understanding

Before we discuss the LMS algorithm, let us understand this concept through an analogy that appeals to intuition. For what follows, *a gradient is just a mere generalization of the derivative* (slope of the tangent to a curve) for a multi-variable function. We need a ‘multi-variable derivative’ (i.e., a gradient) in our case because the equalizer has multiple taps, all of which need to be optimized.

Assume that you are on a holiday with your family and spending the day in a nice theme park. You are going upwards on a roller coaster and hence the gradient is in the upward direction as shown in Figure 8.53. Also assume that you are sitting in the front seat, have access to a (hypothetical) set of brakes installed and there is no anti-rollback mechanism which prevents the coasters from sliding down the hill.

Suddenly the electricity in the park goes out. Leaving the roller coaster to slide all the way down the hill would be catastrophic, so your strategy is to

- first apply the brakes preventing the sudden and rapid drop,
- leave the brakes to slightly descend *in the direction opposite to the gradient*,
- apply the brakes again, and
- repeat this process.

Repeating the above steps in an iterative manner, you will safely reach the equilibrium point. With this intuition in place, we can discuss the LMS algorithm next.

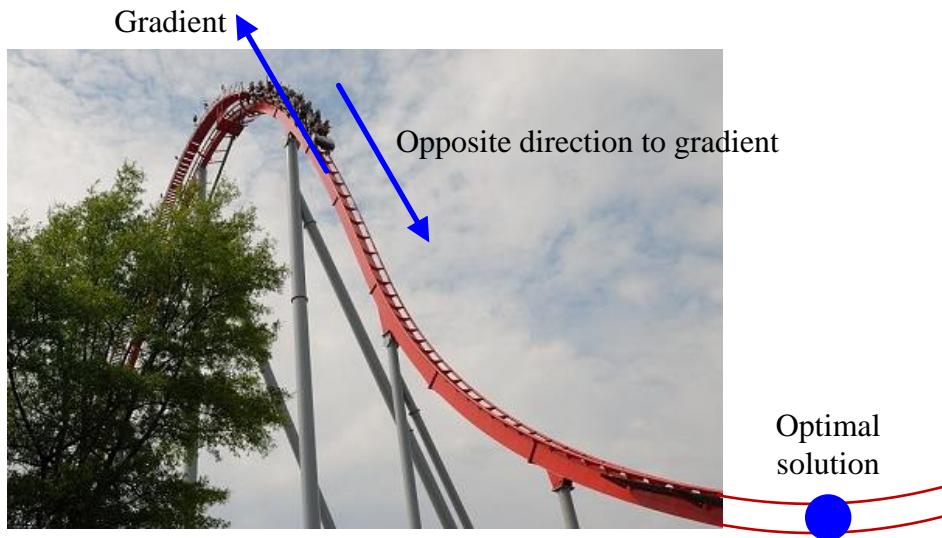


Figure 8.53: Intuitive behind LMS algorithm

The LMS Algorithm

We start with a performance function that has a similar valley type shape, e.g., the squared error we saw in Eq (8.53) reproduced below.

$$\begin{aligned} \text{Mean}|e[m]|^2 &= \text{Mean} |a[m] - y[m]|^2 = \text{Mean} |a[m] - z[m] * q[m]|^2 \\ &= \text{Mean} \left| a[m] - \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \right|^2 \end{aligned}$$

Figure 8.54a draws this quadratic curve as a function of one equalizer tap $q_0[m]$ which is similar to the roller coaster analogy we saw before. On the same note, Figure 8.54b draws the same relationship as a quadratic surface of two equalizer taps, $q_0[m]$ and $q_1[m]$ where a unique minimum point can be identified. Extending the same concept further, Mean $|e[m]|^2$ can be dealt with as a function of $2L_q + 1$ equalizer taps $q_l[m]$ and a unique minimum point can be reached. It is unfortunate that we cannot graphically draw the same relationship for a higher number of taps.

We can start at any point on this Mean $|e[m]|^2$ curve and take a small step in a direction where the decrease in the squared error is the fastest, thus proceeding as follows.

- Previously, the equalizer taps $q[m]$ were constant. We were using the symbol time index m for the equalizer tap $q[m]$ as well because we were treating it as a discrete-time sequence. Now our equalizer taps are changing with each symbol time, so *we need to bring in two indices, m for time and l for the equalizer tap number* which we assign as a subscript. Each tap for $l = -L_q, \dots, L_q$ is updated at symbol time $m + 1$ according to

$$q_l[m + 1] = q_l[m] + \text{a small step}$$

Here, $q_l[m]$ means the l^{th} equalizer tap at symbol time m .

- The fastest reduction in error happens when our direction of update is opposite to the gradient of Mean $|e[m]|^2$ with respect to the equalizer tap weights.

$$q_l[m + 1] = q_l[m] + \text{a small step opposite to the gradient of Mean } |e[m]|^2$$

- The gradient is a mere generalization of the derivative and we bring in a minus sign for moving in its opposite direction.

$$q_l[m + 1] = q_l[m] - \text{Mean} \frac{d}{dq_l[m]} |e[m]|^2 \quad (8.60)$$

- Now we have already found the derivative *for each l* in Eq (8.55) reproduced below.

$$\frac{d}{dq_l[m]} |e[m]|^2 = -2e[m] \cdot z[m - l]$$

for each $l = -L_q, \dots, L_q$

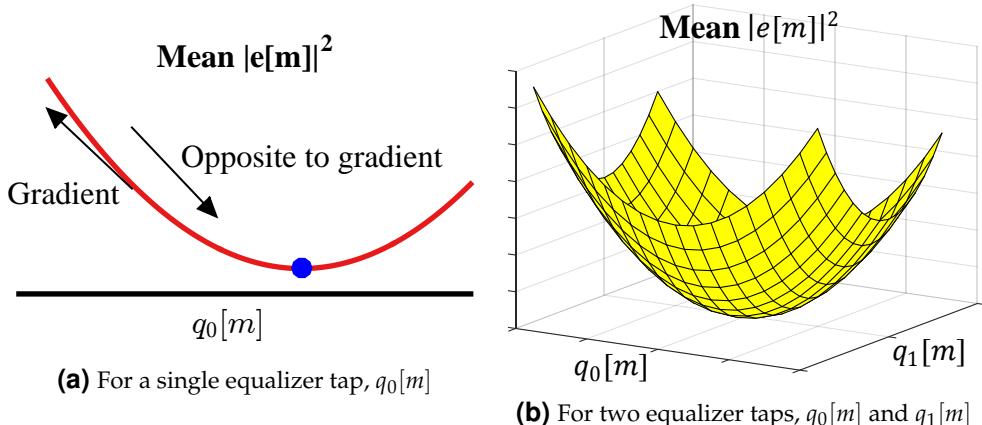


Figure 8.54: Mean $|e[m]|^2$ is a quadratic function of equalizer taps $q_l[m], l = -L_q, \dots, L_q$

Substituting this value back in Eq (8.60), we can update the set of equalizer taps at each step as

$$q_l[m+1] = q_l[m] + 2 \operatorname{Mean} \left\{ e[m] \cdot z[m-l] \right\}$$

- Now *let us remove the mean altogether*, a justification for which we will shortly see, and get

$$q_l[m+1] = q_l[m] + 2e[m] \cdot z[m-l]?$$

- The question mark above is there to indicate that we are probably forgetting something. Remember the brakes in the roller coaster analogy? We need to include an effect similar to the brake here, otherwise the effect of the gradient on its own will result in large swings on the updated taps. Let us call this parameter representing the brakes a *step size* and denote it as μ . We will see the effect of varying μ later.

$$q_l[m+1] = q_l[m] + 2\mu \cdot e[m] \cdot z[m-l]$$

for each $l = -L_q, \dots, L_q$

(8.61)

Note 8.12 A brilliant trick

In an actual derivation of an optimal filter, it is the statistical expectation of squared error that is minimized, i.e.,

$$\operatorname{Mean} |e[m]|^2$$

and hence the term *mean squared error*. This is what we did in the section on MMSE equalizers. Now if Widrow and Hoff wanted to derive the adaptive algorithm that minimizes the mean squared error, they needed to obtain the statistical correlations between the Rx matched filtered samples themselves as well as their correlations with the actual data symbols. This is a very difficult task in most practical applications.

So they proposed a *completely naive solution* for such a specialized problem by removing the statistical expectation altogether, i.e., just employ the squared error $|e[m]|^2$ instead of mean squared error, $\operatorname{Mean} |e[m]|^2$. This is what we saw in the algorithm derived in Eq (8.61). To come up with such a successful workhorse for filter taps adaptation, when everyone else was running after optimal solutions, was a brilliant feat in itself.

It turns out that as long as the step size μ is chosen sufficiently small, i.e., the brakes are tight enough in our analogy, the LMS algorithm is very stable – even though $|e[m]|^2$ at each single shot is a very coarse estimate of its mean.

Figure 8.55 illustrates a block diagram for implementing an LMS equalizer. The matched filter output $z[m]$ is input to a linear equalizer with coefficients $q_l[m]$ at symbol time m . These taps are updated for symbol time $m+1$ by the LMS algorithm that computes the new taps through Eq (8.61). The curved arrow indicates the updated taps being delivered to the equalizer at each new symbol time m .

The input to the LMS algorithm is the matched filter output $z[m]$ and error signal $e[m]$. In general, there are two stages of the equalizer operation.

Training stage: In the training stage, the error signal $e[m]$ is computed through subtracting the equalizer output $y[m]$ from the known training sequence $a[m]$.

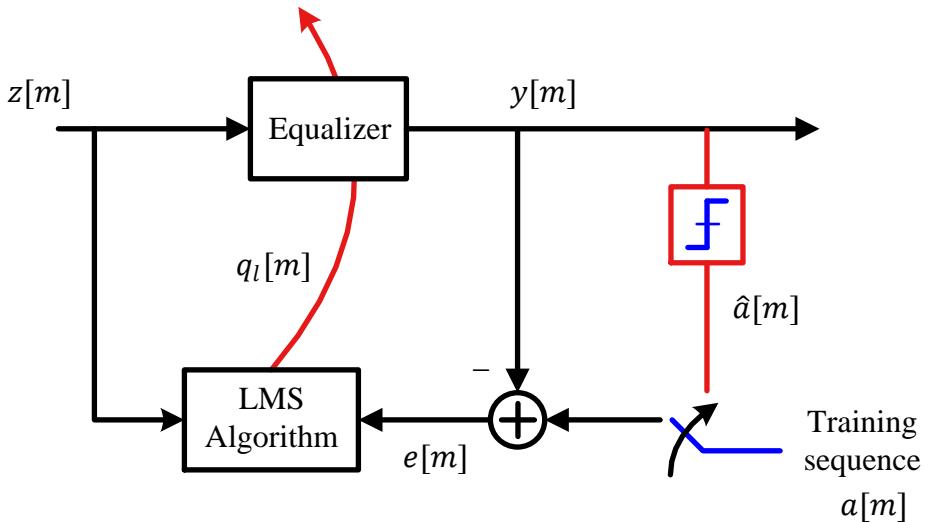


Figure 8.55: A block diagram for implementation of an LMS equalizer

Decision-directed stage: When the training sequence ends, the LMS algorithm uses the symbol decisions $\hat{a}[m]$ in place of known symbols and continues henceforth.

The three steps performed by an LMS equalizer are summarized in Table 8.1. It is important to know that since the update process continues with the input, the equalizer taps – after converging at the optimal solution given by the MMSE solution – do not stay there. Instead, they just keep hovering around the optimal values (unlike the roller coaster analogy which comes to rest at the end) and add some extra error to the possible optimal solution.

Using $y[m] = z[m] * q_l[m]$ and applying the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$, we can see that these are 4 real convolutions for a complex modulation scheme. While the inphase coefficients $q_{l,I}[m]$ combat the inter-symbol interference in the inphase and quadrature channels, the quadrature coefficients $q_{l,Q}[m]$ battle the cross-talk between the two channels. This cross-talk is usually caused by an asymmetric channel frequency response around the carrier frequency which can be seen from the complex conjugate property in Table 2.4.

The Step Size μ

Next, we consider an example applying the LMS algorithm for channel equalization.

Example 8.4

To observe the effect of the step size μ on the performance of the LMS equalizer, we incorporate a 4-QAM modulated symbols shaped by a Square-Root Raised Cosine pulse with excess bandwidth $\alpha = 0.25$. The impulse response of the wireless channel is the same as in Example 8.3 and the resulting curves are averaged over 100 simulation runs for a T_M -spaced equalizer. We consider the following three different values of μ .

$$\mu = 0.01, \quad \mu = 0.04, \quad \mu = 0.1$$

Table 8.1: Least Mean Square (LMS) equalizer

Inputs	Matched filter output $z[m]$ Training symbols $a[m]$ or decisions $\hat{a}[m]$ Equalizer taps $q_l[m]$
Outputs	Equalizer output $y[m]$ Updated equalizer taps $q_l[m + 1]$
Step 1: Filtering	$y[m] = z[m] * q[m]$
Step 2: Error generation	$e[m] = a[m] - y[m]$
Step 3: Taps adaptation	$q_l[m + 1] = q_l[m] + 2\mu \cdot e[m] \cdot z[m - l]$ for each l from $-L_q$ to L_q

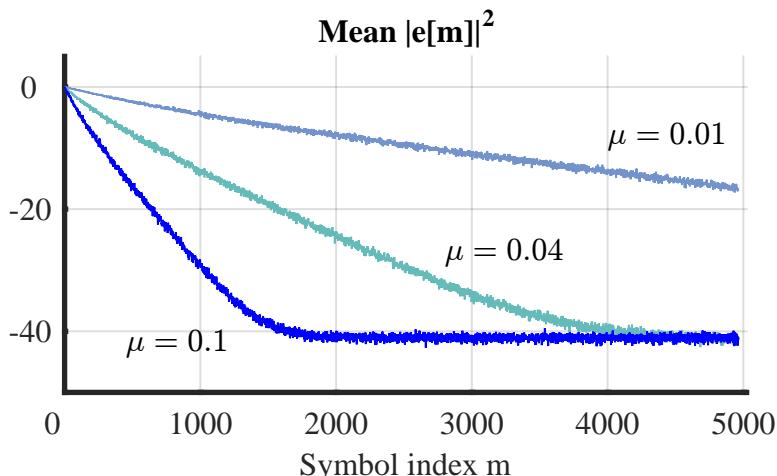


Figure 8.56: Mean $|e[m]|^2$ for a T_M -spaced equalizer for three different values of the step size μ

Figure 8.56 draws the corresponding curves for all three values of μ , regarding which a few comments are in order.

- A large value of μ generates an equalizer response that converges faster than that for a smaller value of μ . This makes sense from the form of the equalizer tap update where the new tap at each symbol time m is generated through the addition of $2\mu e[m]z[m - l]$ in the previous tap value. The larger the μ , the larger the update value and hence faster the convergence.

- So why not choose μ as large as possible? From the roller coaster analogy, it can flip over in any direction if it is thrown towards the equilibrium point too quickly by not properly applying the brakes. Recall that the LMS algorithm, after converging to the minimum error, does not stay there and fluctuates around that value. The difference between this minimum error and the actual error is the excess mean square error. While it is not clear from Figure 8.56, a larger μ results in a greater excess error and hence there is a tradeoff between faster convergence and a lower error.
- From the update expression $2\mu e[m]z[m-l]$ which dictates how each tap is generated through the previous tap value, we also infer that for a given μ , the convergence behaviour (and stability) of the LMS algorithm also depends on the signal power at the equalizer input. In a variant known as *normalized LMS equalizer*, this signal power is estimated to normalize μ at each symbol time m .
- For $\mu = 0.1$, the corresponding equalizer taps are shown converging to their final values in Figure 8.57. The equalizer takes many hundreds of symbols before approaching the tap values with acceptable Mean $|e[m]|^2$. This is typical of this kind of processing in an iterative fashion.

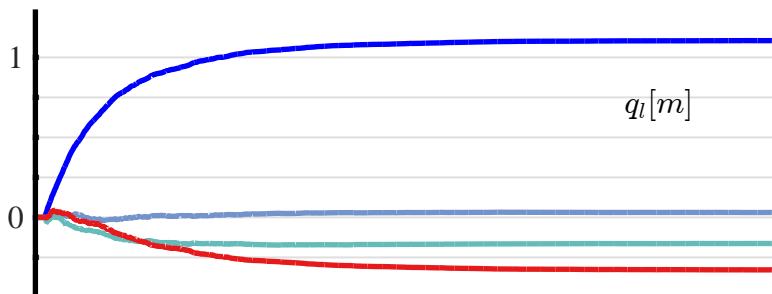


Figure 8.57: Some of the equalizer taps $q_l[m]$ for $\mu = 0.1$

- The convergence time of the LMS equalizer also depends on the actual channel frequency response. If there is not enough power in a particular spectral region, it becomes difficult for the equalizer to prepare a compensation response. As a result, a spectral null reduces the convergence speed and hence requires a significantly large number of symbols. This had been one of the bottlenecks in the high rate wireless communication systems. We will have more to say about this in the discussion on frequency domain equalization in Section 8.3.8.
- An LMS equalizer is also extensively used in high speed serial links in conjunction with Decision Feedback Equalization (DFE) which we study in Section 8.3.5.
- Finally, as we saw in Example 5.2 during the phase synchronization, the process of equalizer taps convergence is more interesting to watch from a constellation point of view as it unfolds in a 3D space for equalizer output $y[m]$, which I called a *dynamic scatter plot*. If this page was a 3D box and we could go towards one side of the scatter plot, we would have viewed it as in Figure 8.58. Since the equalizer taps are initialized as all zeros, the equalizer output $y[m]$ starts from zero. Then, the four 4-QAM constellation points trace a similar trajectory

taken by $\text{Mean}|e[m]|^2$ and $q_I[m]$ before. The figure has been plotted for $\mu = 0.1$. Note that the varying amount of gap between some constellation points arises from the randomness of the data symbols consisting of one out of four possible symbols during each T_M .

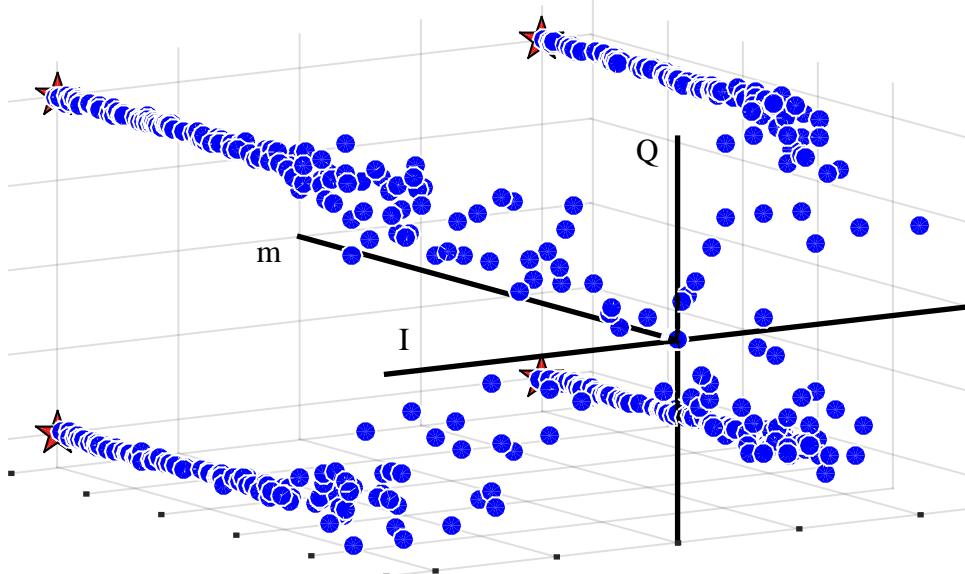


Figure 8.58: A dynamic scatter plot of equalizer output $y[m]$ for $\mu = 0.1$. Observe the convergence towards the actual constellation locations of a 4-QAM scheme

Accelerating the Convergence

An LMS equalizer has been the workhorse for wireless communication systems throughout the previous decades. However, we saw in the above example that it takes many hundreds or thousands of symbols before the LMS equalizer converges to the optimum tap values. As a general convergence property, remember that the shortest settling time is obtained when the power spectrum of the T_M -spaced equalizer input (that suffers from aliasing due to $L = 1$) is flat and the step size μ is chosen to be the *inverse* of the product of the Rx signal power with the number of equalizer coefficients [36]. A smaller step size μ should be chosen when the variation in this folded spectrum is large that leads to slower convergence.

There has been a significant interest in accelerating its convergence rate. Two of the most widely employed methods are explained below.

Variable step size μ : It is evident that the convergence rate is controlled by the step size μ . Just like we saw in Section 4.7 where the Phase Locked Loop constants are reconfigured on the fly, it makes sense to start the LMS equalizer with a large value of μ to ensure faster convergence at the expense of significant fluctuation during this process. After converging closer to the optimal solution, it can be reduced in steps such that μ during the final tracking stage is a small enough value to satisfy the targeted excess mean square error.

Cyclic equalization: Instead of modifying μ , this method focuses on the training sequence that is sent at the start of the transmission to help the Rx determine the synchronization parameters as well as the equalizer taps. It was discovered that if this training sequence is made periodic with the same period as the equalizer length, the taps can be computed almost instantly. This is done through exploiting the periodicity in the stored sequence $a[m]$ and the incoming signal $z[m]$. The periodicity implies that a Discrete Fourier Transform (DFT) of these sequences can be taken and multiplied point-by-point for each DFT index k . After normalizing the result for each k by the Rx power in that spectral bin, an inverse Discrete Fourier Transform (iDFT) is taken to determine the equalizer taps $q[m]^{\dagger}$.

Although cyclic equalization is a very interesting technique, it is largely abandoned in favour of a better alternative for high speed wireless communications, namely the frequency domain equalization. We discuss frequency domain equalization in Section 8.3.8.

LMS Variants

In summary, the LMS equalizer has been incorporated into many commercial high speed modems due to its simplicity and coefficients adaptation of a time-varying channel. For further simplification, some of its variations employ only the sign of the error signal or that of the input samples. Three such possible variations for tap adaptation are as follows.

$$q_l[m+1] = q_l[m] + 2\mu \cdot \text{sign}(e[m]) \cdot z[m-l]$$

for each $l = -L_q, \dots, L_q$

$$q_l[m+1] = q_l[m] + 2\mu \cdot e[m] \cdot \text{sign}(z[m-l])$$

for each $l = -L_q, \dots, L_q$

$$q_l[m+1] = q_l[m] + 2\mu \cdot \text{sign}(e[m]) \cdot \text{sign}(z[m-l])$$

for each $l = -L_q, \dots, L_q$

It is evident that the last variant is the simplest of all, consisting of just the signs of both quantities. On the downside, it also exhibits the slowest rate of convergence.

Exercise 8.1

GNU Radio currently supports only the Decision-Directed version of the LMS equalizer (DD-LMS). Therefore, the Rx can switch to this equalization technique only after reliable decisions are available, i.e., after all other loops have converged to their steady state solutions. The variables used in the block ‘LMS DD Equalizer’ are as follows.

Gain: This is the update rate μ of the equalizer described above.

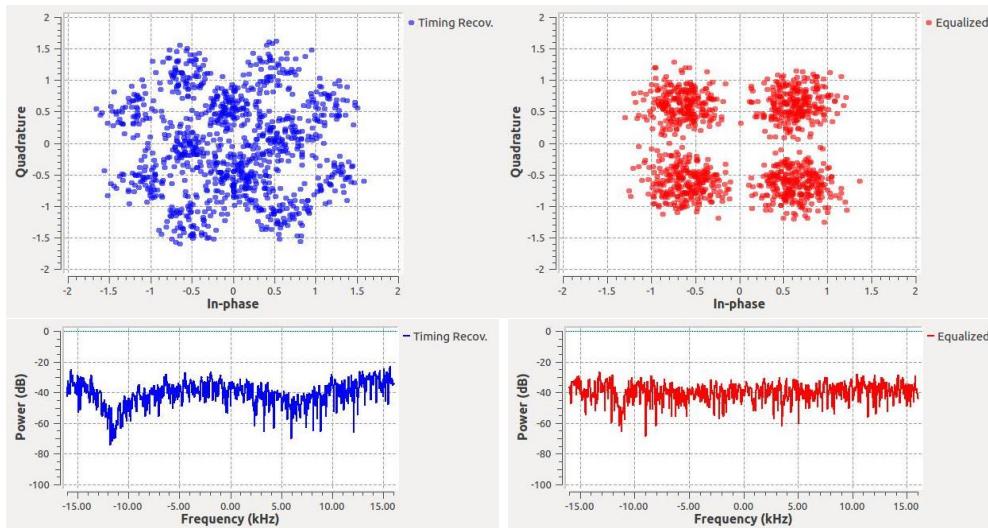
Num. Taps: The number of taps in the equalizer should be significantly longer than the expected channel length.

Samples/Symbol: The number of samples/symbol L determines whether the equalizer is symbol-spaced or fractionally-spaced. A value of $L = 2$ is usually used for fractionally-spaced equalizers that do not suffer from the problem of spectral null, see Figure 7.7.

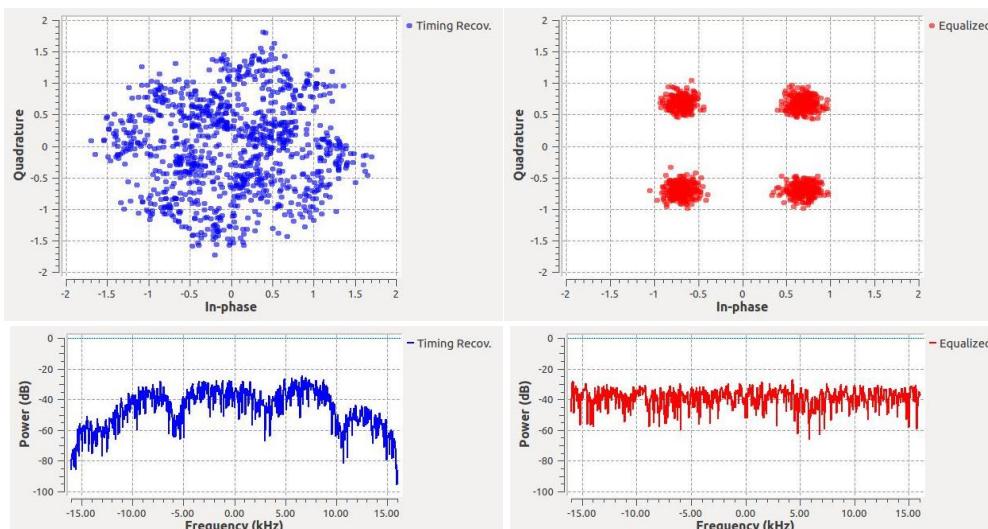
[†]This same method can also be used to estimate the channel impulse response $h[m]$ instead of the equalizer taps and then use another equalizer, such as the maximum likelihood sequence estimation or MMSE equalizer, to remove the channel distortion.

Constellation Object: This is the constellation where the decisions are mapped in the decision-directed mode.

For a gain of $\mu = 0.01$, 11 taps and a QPSK modulation, Figure 8.59 draws the scatter plots and spectra in GNU Radio for a symbol-spaced equalizer ($L = 1$) and a fractionally-spaced equalizer ($L = 2$) for a harsh channel. The purpose of these plots



(a) T_M -spaced equalizer



(b) $T_M/2$ -spaced equalizer

Figure 8.59: Scatter plots and spectra for a symbol-spaced and a fractionally-spaced equalizer

are to show that

- a different sampling rate implies that both equalizers ‘experience’ different channels even though the underlying physical paths are the same,
- a fractionally-spaced equalizer performs better than a symbol-spaced equalizer under almost any condition, and
- after the compensation for channel frequency response, the spectrum is ideally flat at the output of an equalizer.

You can experiment with both T_M and $T_M/2$ -spaced equalizers by changing the timing and observing the equalizer response for both cases.

8.3.5 Decision Feedback Equalization

The most attractive feature of the linear equalizers is their simplicity: each output sample $y[m]$ is a simple linear combination (determined by the equalizer taps $q[m]$) of the input samples $z[m]$. The cost of this simple implementation is their performance. While the minimum mean square error equalizer does not enhance the noise as much as the zero forcing equalizer, its performance is still not acceptable in most practical scenarios.

Is it possible to design an equalizer that carries both features of simplicity and accuracy? While the answer is usually no in most situations, it is possible here to a certain extent through a Decision Feedback Equalizer (DFE).

Fundamental Idea

To understand the concept of decision feedback, we first have to return to the original channel impulse response $h[m]$ in the form of channel taps. Remember that there is usually a main tap with the largest energy that represents the main symbol location. Let us call this tap the main *cursor*.

The remaining channel taps contribute towards the Inter-Symbol Interference (ISI). This ISI manifests itself at two locations:

- before the cursor known as *precursor ISI*, and
- after the cursor known as *postcursor ISI*.

Since these terms can be a little confusing, let us build this concept from the fundamentals. Recall from the previous sections that for a symbol-spaced equalizer, the matched filter output $z[m]$ is generated by the convolution of the data symbols $a[m]$ with the composite channel impulse response $h[m]$.

$$z[m] = a[m] * h[m] = \sum_{l=0}^{N_{\text{Tap}}} a[l] \cdot h[m-l]$$

Assume that the first four data symbols $a[m]$ are

$$a[m] = [+1 \ -1 \ -1 \ +1]$$

and the channel consists of the following 4 taps.

$$h[m] = \begin{bmatrix} -0.5 & 0.9 & -0.3 & 0.2 \end{bmatrix}$$

Also suppose that the Rx has complete knowledge of the channel through a channel estimation procedure such as the one described in Section 8.2. The two signals $a[m]$ and $h[m]$ are shown at the top of Figure 8.60.

Next, we invoke the intuitive explanation of convolution from Section 2.4 and draw the convolution steps in Figure 8.60 from the viewpoint of symbol $a[2]$. I highly recommend you to read that intuitive view on convolution before continuing what follows, even if you already know about convolution.

Symbol $a[0]$: The symbol $a[0] = 1$ kicks out the first copy of the impulse response $h[m]$. The type of the channel implies that the cursor lies at the second tap at position $m = 1$.

Symbol $a[1]$: Now the symbol $a[1] = -1$ induces the second copy of $h[m]$ whose cursor lies at $m = 2$.

Symbol $a[2]$: This is the focus of our current discussion. The third symbol $a[2] = -1$ produces the third copy of $h[m]$ that starts at $m = 2$. However, the cursor lies at time $m = 3$. Observe the dashed red rectangle drawn at time $m = 3$ encompassing the whole figure that contains the ISI injected by each symbol at this time. It is evident that symbols $a[0]$ and $a[1]$ cause the postcursor ISI through 4th and 3rd channel taps, respectively. Furthermore, symbol $a[3]$, a future symbol, induces the precursor ISI through the first channel tap.

But doesn't the Rx already know the two symbols $a[0]$ and $a[1]$ as well as the complete channel information through which it can cancel their contributions? Yes, it does and that is the whole point of canceling the postcursor ISI as explained next.

If we open the convolution equation between the data symbols $a[m]$ and channel $h[m]$ at time $m = 3$ (because the cursor for symbol $a[2]$ lies at time $m = 3$), we get an interesting expression.

$$\begin{aligned} z[3] &= \sum_{l=0}^{N_{\text{Tap}}} a[l] \cdot h[3-l] \\ &= \underbrace{a[0] \cdot h[3] + a[1] \cdot h[2]}_{\text{Postcursor ISI}} + \underbrace{a[2] \cdot h[1]}_{\text{Cursor}} + \underbrace{a[3] \cdot h[0]}_{\text{Precursor ISI}} \end{aligned}$$

Compare these terms with Figure 8.60 where each subfigure in the red dashed rectangle corresponds to a term above. We can rewrite the above expression as

$$z[3] - \left\{ \underbrace{a[0] \cdot h[3] + a[1] \cdot h[2]}_{\text{Postcursor ISI}} \right\} = \underbrace{a[2] \cdot h[1]}_{\text{Cursor}} + \underbrace{a[3] \cdot h[0]}_{\text{Precursor ISI}}$$

After the training sequence ends and the data transmission starts, the decisions $\hat{a}[m]$ taken at each symbol instant m can be employed to cancel the postcursor ISI. This

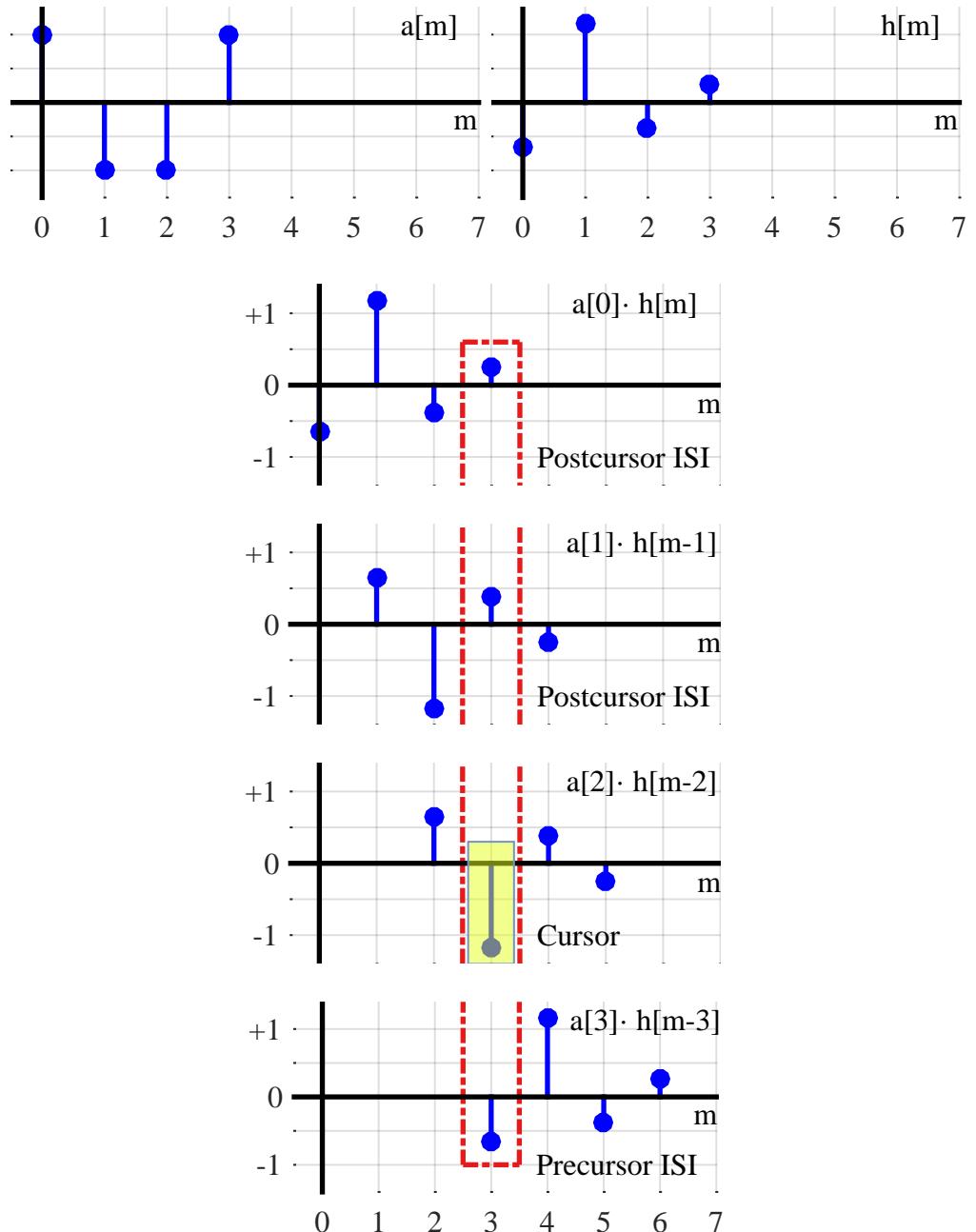


Figure 8.60: Convolution steps between data symbols $a[m]$ and channel $h[m]$ highlighting the cursor, postcursor ISI and precursor ISI

is the idea of the decision feedback equalization. For example, standing at instant

$m = 2$,

$$z[3] - \left\{ \underbrace{\hat{a}[0] \cdot h[3] + \hat{a}[1] \cdot h[2]}_{\text{Decision feedback}} \right\} = a[2] \cdot h[1] + a[3] \cdot h[0] \quad (8.62)$$

A Summary

To summarize what we have learned so far, understand that each symbol generates both precursor ISI and postcursor ISI. Not much can be done regarding the symbols responsible for precursor ISI because it requires looking into the future symbols. This is only possible in an often undesirable delayed output scenario. On the other hand, once a symbol is detected, its effect from the future symbols, i.e., postcursor ISI, can be removed through a proper equalizer design. Let us find out how this is accomplished through a decision feedback equalizer.

Keep in mind that a decision feedback equalizer can be implemented with fixed or preset taps at the start of the transmission that remain so until the end, or its taps can be adapted according to the channel conditions through an adaptive algorithm like LMS.

Design of a Preset Decision Feedback Equalizer

The most straightforward idea to implement a decision feedback equalizer is to imitate what we did in decision-directed synchronization. We can scale the previously detected symbol $\hat{a}[m]$ through the corresponding channel tap $h[m]$ and subtract them from the matched filter output $z[m]$ that enters the threshold detector. This is the option presented by the left side of Eq (8.62).

To illustrate the problem with this approach, consider Figure 8.61 showing a channel with 8 taps. Only the postcursor ISI can be removed by the above mentioned strategy. There is a significant precursor ISI – a total of 3 taps to be exact – in the demonstrated channel that is going to cause decision errors. These decision errors will inject even more ISI in the system due to the wrong decisions fed back. Clearly, a ‘decision-directed equalizer’ is not a good solution for most channels. Let us discuss an alternative approach.

A regular equalizer consists of $2L_q + 1$ taps which act on the incoming signal $z[m]$ (the matched filter output) all at once to produce the output $y[m]$.

$$y[m] = \sum_{l=-L_q}^{L_q} q[l] \cdot z[m-l] \quad (8.63)$$

The strategy adopted by a DFE is to split the equalizer into two separate portions:

- *a feedback filter* that removes the postcursor ISI with the help of symbol decisions, and
- *a feedforward filter* that minimizes the ISI like a regular equalizer. It can be designed to handle either just the precursor portion of the ISI or all of it.

The structure of a decision feedback equalizer with two constituent filters is shown in Figure 8.62.

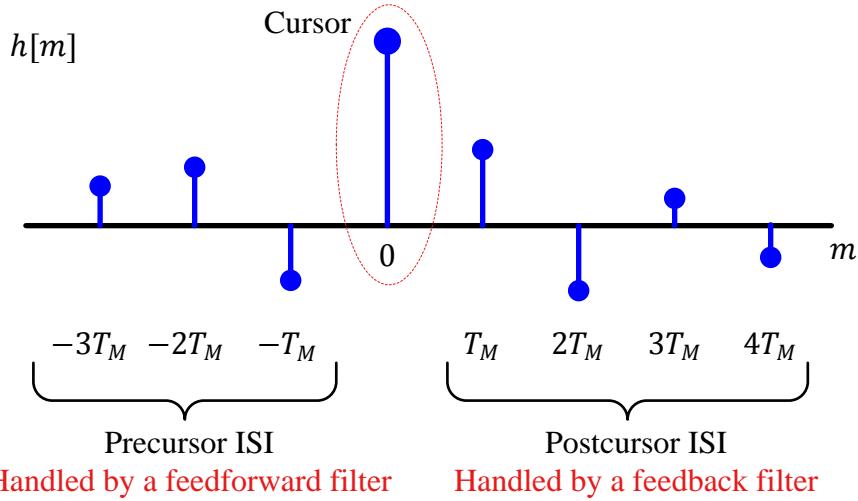


Figure 8.61: A channel with 8 taps demonstrating the main cursor, precursor ISI and postcursor ISI

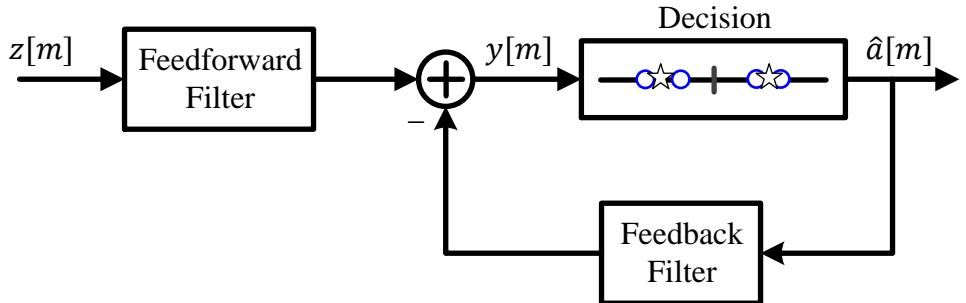


Figure 8.62: A block diagram for the implementation of a decision feedback equalizer

Out of the total number of taps, $L_{q,FF} + 1$ taps are assigned to the feedforward filter that are considered T_M -spaced in the current discussion. The remaining $L_{q,FB}$ taps are then assigned to the feedback filter.

$$2L_q + 1 = L_{q,FF} + 1 + L_{q,FB}$$

The exact division of the taps between these filters depends on the channel response. For example, a channel with a 'long tail' or a large postcursor ISI and less precursor ISI can have $L_{q,FB}$ greater than $L_{q,FF} + 1$. Next, we find out how to implement such a technique.

With the matched filter output $z[m]$ as the equalizer input, we have the output $y[m]$

after splitting the equalizer in Eq (8.63).

$$y[m] = \underbrace{\sum_{l=-L_{q,FF}}^0 q_{FF}[l] \cdot z[m-l]}_{\text{Feedforward filter}} + \underbrace{\sum_{l=1}^{L_{q,FB}} q_{FB}[l] \cdot \hat{a}[m-l]}_{\text{Feedback filter}} \quad (8.64)$$

where $q_{FF}[m]$ and $q_{FB}[m]$ are the taps for feedforward and feedback filter, respectively. The next question is how to choose these taps.

As far as the feedforward filter $q_{FF}[m]$ is concerned, any criteria for handling the ISI can be chosen, e.g., zero forcing or minimum mean square error. In other words, the feedforward filter is a simple linear equalizer discussed before in Section 8.3.3. However, it does not need to approximate the inverse of the channel characteristics and its coefficients can be designed accordingly. For example, for a channel with a large precursor and small postcursor ISI, all the feedforward filter taps can be devoted to removing the precursor ISI (as long as the symbol decisions are correct, the postcursor ISI is removed by the feedback filter). Otherwise, the feedforward filter can be designed to reduce ISI from all the taps whether precursor or postcursor, just like a single equalizer would do.

On the other hand, it seems from Eq (8.62) that the coefficients $q_{FB}[m]$ of the feedback filter should simply be the channel taps $h[m]$ because the input to the feedback section is the data symbol decisions $\hat{a}[m]$, see Figure 8.62. However, while deriving that equation, we did not have any feedforward filter to remove the precursor ISI and hence its input was directly the data symbols $a[m]$ convolved with the channel response $h[m]$. In this case, however, the input to the decision device (without a feedback equalizer) is expressed as

$$y[m] = a[m] * h[m] * q_{FF}[m]$$

The term $h[m] * q_{FF}[m]$ is the modified channel that appears as the distortion to the feedback equalizer. In the light of Eq (8.62) with the modified channel whose effect needs to be cancelled out, the taps of the feedback filter can be written as the convolution of the channel response $h[m]$ with the feedforward filter taps $q_{FF}[m]$.

$$q_{FB}[m] = h[m] * q_{FF}[m]$$

The final expression for the feedback taps is

$$q_{FB}[m] = \sum_{l=-L_{q,FF}}^0 q_{FF}[l] \cdot h[m-l] \quad (8.65)$$

for each $m = 1, 2, \dots, L_{q,FB}$

Removing the impact of $L_{q,FB}$ past symbols from the current symbol results in complete elimination of postcursor ISI as long as the fed back decisions are correct. Once that happens, the performance of a DFE is vastly superior to that of any linear equalizer. However, if the decisions are not correct, the errors are fed back generating more errors in a phenomenon known as *error propagation*. As a consequence, a DFE is a suitable design for a system operating at high SNR. Finally, remember that a DFE is non-linear due to the inclusion of the detected symbols $\hat{a}[m]$ in the feedback filter.

Design of an Adaptive Decision Feedback Equalizer

As we saw in Section 8.3.4 on LMS equalizer, the taps of both feedforward and feedback filters can also be adjusted in an iterative manner instead of fixing them to some determined values. Following the steps that lead to Eq (8.61), we have the following coefficients for the feedforward filter.

$$q_l[m+1] = q_l[m] + 2\mu \cdot e[m] \cdot z[m-l] \quad (8.66)$$

for each $l = -L_{q,FF}, \dots, 0$

where the subscript l now denotes the l^{th} tap, as opposed to the symbol time m . On the other hand, the coefficients of the feedback filter can be adjusted as

$$q_l[m+1] = q_l[m] + 2\mu \cdot e[m] \cdot \hat{a}[m-l] \quad (8.67)$$

for each $l = 1, 2, \dots, L_{q,FB}$

Observe the appearance of the symbol decision $\hat{a}[m]$ instead of the matched filter output $z[m]$ in the above expression because the input to the feedback equalizer is the symbol decisions $\hat{a}[m]$. Rest of the details of the LMS algorithm for the DFE are the same as discussed before.

The application of decision feedback equalization is not restricted to individual symbols basis only. In recent times, the advent of powerful iterative error correcting codes which operate in units of large blocks have motivated decision feedback equalization in a block-wise fashion.

8.3.6 Blind Equalization: Constant Modulus Algorithm (CMA)

In the discussion on brute force (maximum likelihood sequence estimation) and linear equalization, the channel response $h[m]$ was available at the Rx through any channel estimation procedure that requires a training sequence. For adaptive and decision feedback equalization too, first the training sequence symbols and then symbol decisions were employed to tune the equalizer taps. There are many applications, however, where the Rx needs to acquire the equalizer coefficients without any help from the Tx in the form of known symbols. This need primarily arises from mobile communication systems where the Rx can lose the signal due to channel fading or blockage and a blind acquisition is required after the signal returns. Some of the other applications are a general purpose radio to demodulate most wireless communication signals, military radios, signal monitoring and point to multi-point networks.

Blind equalization, also known as self-recovering equalization, is the process of choosing the equalizer taps without any help from a training sequence or symbol decisions. In this chapter, we will discuss the most prevalent blind equalization technique known as *Constant Modulus Algorithm (CMA)* devised by Godard in 1980.

The CMA is very similar to the LMS algorithm discussed in Section 8.3.4 with the only difference being in the performance function. In case of the LMS algorithm, the performance function was the squared error reproduced below.

$$\text{Mean}|e[m]|^2 = \text{Mean}|a[m] - y[m]|^2 = \text{Mean}|a[m] - z[m] * q_l[m]|^2$$

where $a[m]$ are the data symbols, $z[m]$ is the equalizer input, $y[m]$ is the equalizer output, and $q_l[m]$ is the l^{th} equalizer tap at time m . The question is what to do in the absence of any data symbols $a[m]$ or their decisions $\hat{a}[m]$.

For this purpose, we first consider the scatter plot of QPSK symbols received through a multipath channel. When multiple taps are convolved with the Tx symbols, one example for a simple channel was shown in Figure 8.40d after downsampling the signal to $L = 1$ sample/symbol. This figure is redrawn in Figure 8.63a for a 2 tap channel and in Figure 8.63b for a 4 tap channel. Regardless of how convoluted (pun intended) the signal looks like as the multipath scatters the Tx symbols all over the 2D plane, we can still observe a structure in this figure. From our knowledge of wireless channels, we have seen that the channel acts on the Rx signal in a systematic manner *unlike noise*. This gives us a hope of recovering the signal even without any data or channel knowledge.

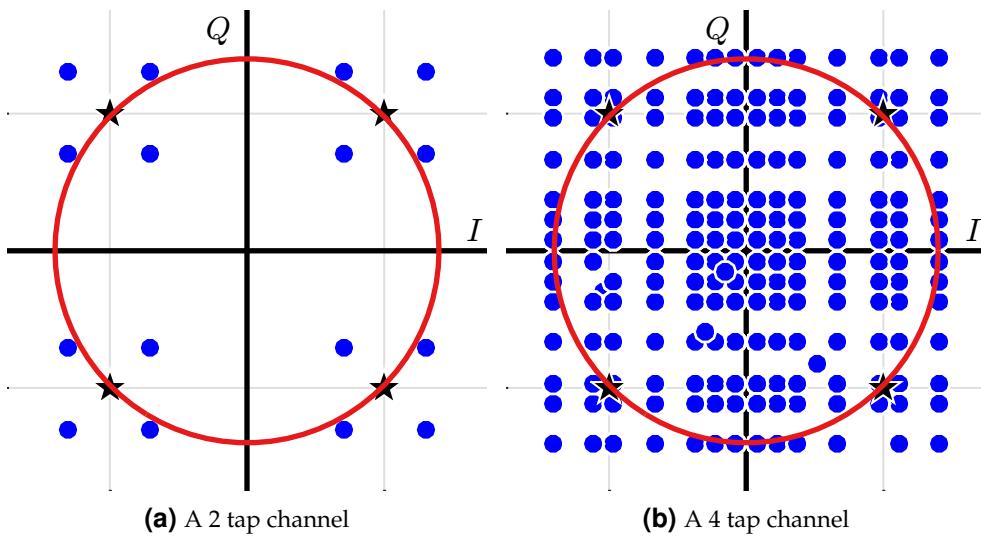


Figure 8.63: Scatter plot for a QPSK signal after filtering through a channel. Notice the scattering of the symbols around the unit circle

Now suppose that symbol timing recovery, even imperfect, is in place and the matched filter output $z[m]$ at $L = 1$ sample/symbol is available. More importantly, we are not interested in carrier phase recovery at this stage. Then, with all the scattered symbols in this 2D plane, one strategy is to form a ring in this 2D plane that passes through the desired symbol locations as drawn in Figure 8.63. For a normalized QPSK constellation,

$$\{a_I[m], a_Q[m]\} = \begin{cases} +\frac{1}{\sqrt{2}}, +\frac{1}{\sqrt{2}} \\ +\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}}, +\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \end{cases}$$

The ring that passes through these symbols has a radius equal to 1. From Figure 8.63, if the distance between the location of actual Rx samples and this ring is minimized, the channel is implicitly equalized regardless of the complexity of the channel. In

this respect, this is a very interesting strategy that is different than other equalization techniques discussed before in the context of available training or decisions.

In light of the above, the performance criterion used by the CMA is

$$\text{Mean} \left(|y[m]|^2 - 1 \right)^2$$

where $|y[m]|^2$ is the energy of the T_M -spaced sample scattered around the constellation and 1 is their desired energy concentrated in a constellation circle. The next steps of the derivation are very similar to the LMS algorithm. $\text{Mean} (|y[m]|^2 - 1)^2$ can be dealt with as a function of $2L_q + 1$ equalizer taps $q_l[m]$ and a unique minimum point on the error surface can be reached by removing the mean, then starting at any point and taking a small step opposite to the gradient with respect to the equalizer tap weights.

$$q_l[m+1] = q_l[m] - \frac{d}{dq_l[m]} \left(|y[m]|^2 - 1 \right)^2 \quad (8.68)$$

As before, $q_l[m]$ means the l^{th} equalizer tap at symbol time m . The derivative in the above expression is written for each $l = -L_q, \dots, L_q$, as

$$\frac{d}{dq_l[m]} \left(|y[m]|^2 - 1 \right)^2 = 2 \left(|y[m]|^2 - 1 \right) \cdot \frac{d}{dq_l[m]} |y[m]|^2 \quad (8.69)$$

Using $y[m] = z[m] * q_l[m]$, we can open $|y[m]|^2$ by considering the case for a real signal only that can be later generalized to complex signals. Then, for each $l = -L_q, \dots, L_q$, we can write

$$\begin{aligned} \frac{d}{dq_l[m]} |y[m]|^2 &= 2y[m] \cdot \frac{d}{dq_l[m]} \sum_{l=-L_q}^{L_q} q_l[m] z[m-l] \\ &= 2y[m] \cdot z[m-l] \end{aligned}$$

Plugging back in Eq (8.69) and ignoring the irrelevant constants,

$$\frac{d}{dq_l[m]} \left(|y[m]|^2 - 1 \right)^2 = y[m] \cdot z[m-l] \cdot \left(|y[m]|^2 - 1 \right)$$

Finally, substitute this value back in the tap update Eq (8.68) and introduce the step size μ for a smooth convergence to obtain the final expression for a CMA equalizer.

$$q_l[m+1] = q_l[m] - \mu \cdot y^*[m] \cdot z[m-l] \cdot \left(|y[m]|^2 - 1 \right) \quad (8.70)$$

for each $l = -L_q, \dots, L_q$

where the conjugate appears as a result of the same derivation for complex signals. A few comments about the constant modulus algorithm are now in order.

Generalization: The actual CMA equalizer is a generalized version of the algorithm derived above. There, instead of taking the magnitude squared of the equalizer

output $y[m]$, any other value of p can be employed. So the performance criterion becomes

$$\text{Mean} \left(\left| y[m] \right|^p - 1 \right)^2$$

Following a similar derivation, the generalized CMA update equation becomes

$$q_l[m+1] = q_l[m] - \mu \cdot y^*[m] \cdot \left| y[m] \right|^{p-2} \cdot z[m-l] \cdot \left(\left| y[m] \right|^p - 1 \right)$$

for each $l = -L_q, \dots, L_q$

PLL: Observe from the expression of the CMA equalizer that it exploits the magnitude of the Rx samples to compensate for the channel. Consequently, the convergence can happen at any phase and a Phase Locked Loop (PLL) can be run at the output with the CMA equalizer to correct for the rotated symbols in a non-data-aided fashion. Carrier phase recovery was discussed in Chapter 5 in which non-data-aided algorithms were studied in Section 5.6.

Mode switch: As happens with the algorithms working without any knowledge of a training sequence or symbol decisions, the convergence time of the CMA equalizer is significantly longer than the LMS algorithm. This is the cost of having no known symbols within the Tx sequence. Furthermore, the symbol decisions are not available until the convergence of the equalizer. When that eventually happens, the tap update mechanism can be switched from CMA to LMS algorithm in a decision-directed fashion.

Multi-Modulus Algorithm (MMA): From the intuition behind CMA algorithm shown in Figure 8.63a, it is evidently more suited to a PSK constellation. To better cope with QAM signals, a Multi-Modulus Algorithm (MMA) separately penalizes the dispersion of the inphase and quadrature parts of the equalizer output. This is equivalent to fitting the constellation on to a square instead of a circle. A natural advantage of MMA is that it converges towards the steady state equalizer taps with a correct phase offset and hence a carrier recovery loop is not required at the final stage.

From 1960s to 1990s, time domain equalization through the methods described above was the major paradigm for combating ISI. Next, we discuss the equalization in frequency domain that solves both problems of a long ISI and slow convergence of the equalizer taps.

8.3.7 From Time to Frequency Domain Equalization

In the study of time domain equalizers until now, we have encountered two significant problems described below.

Convergence time: An equalizer can be considered as a multi-armed phase or timing locked loop, all of which are driven by the same mechanism. As a consequence, the convergence time of an equalizer is significantly longer than a PLL. In a high data rate communication system, more and more symbols are arriving at the Rx while the equalizer has not converged, filling the buffer faster than it

is emptied through the Rx DSP. Subsequently, the buffer quickly overflows and any real-time processing at the Rx becomes impossible. Therefore, time domain equalization – and not synchronization – has historically proved to be a bottleneck for most high rate applications.

Computational Complexity: Each operation at the Rx is the inverse of what happened to the Tx signal. For example, a Carrier Frequency Offset (CFO) is corrected by a continuous rotation by its estimate in the opposite direction, a Symbol Timing Offset (STO) is compensated by interpolating the signal by its estimate in the opposite direction, and so on.

Now in time domain, the Tx signal undergoes a convolution with the wireless channel before arriving at the Rx. The inverse operation at the Rx should be a deconvolution but actually another convolution, with the equalizer taps in a linear case, takes place.

- The convolution of the Tx signal with the wireless channel is performed by nature and is inherently very fast.
- The convolution of the Rx signal with the equalizer is performed by our own DSP machine and is relatively much slower. As we saw in Figure 8.4b for high data rate scenario, each symbol interferes with many dozens of symbols in the future through the late arriving paths, generating an enormous amount of ISI. Moreover, the length of the time domain equalizer should be much longer than the length of the channel for a satisfactory error rate for many channels, at least 3 to 5 times the length of the channel [28].

Taking these factors into account, the computational complexity of a time domain equalizer on the Rx side quickly explodes.

In light of the above, we can say that the time domain equalization is not suitable for high data rate applications as illustrated in Figure 8.64.

The question about how to tackle this problem brings me to what fred harris said in Ref. [1].

“When faced with a problem we can’t solve, we invoke a trick that *Star Trek* enthusiasts will recognize as the *Kobayashi Maru Scenario (Wrath of Khan)*. When faced with an unsolvable problem, change it into one you can solve, and solve that one instead.”

This change is carried out in frequency domain. The solution in frequency domain is faster because convolution in one domain is multiplication in the other. And to combat this channel multiplication, frequency domain division is a relatively much simpler operation as compared to the time domain convolution. The cost of going into frequency domain through an FFT and then returning through an iFFT quickly crosses the breakeven point for a long frequency selective fading channel. This is illustrated in Figure 8.64 with block processing as its integral element centering around a process and dump philosophy, as we find out in single-carrier frequency domain equalization next and OFDM later.

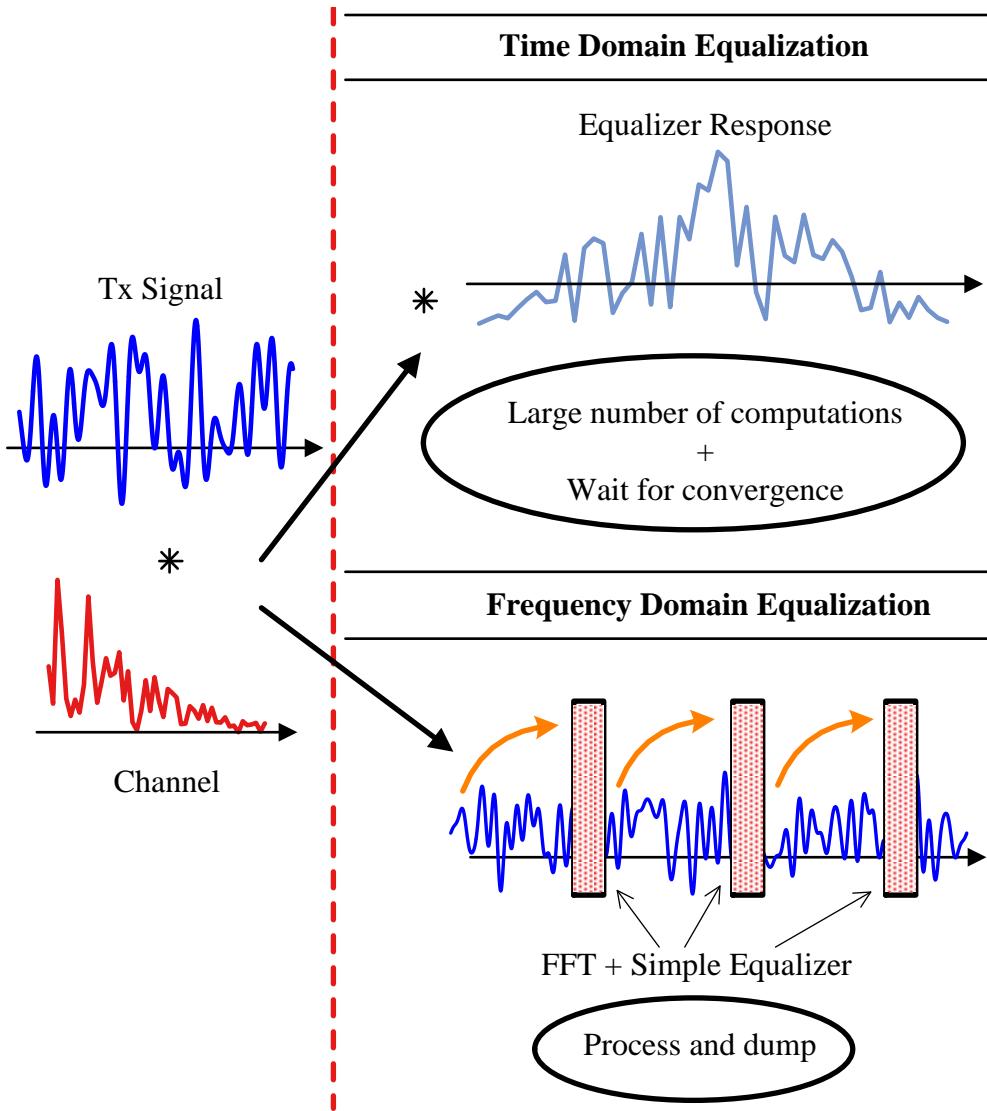


Figure 8.64: An illustration of how time and frequency domain equalization act on the Rx signal

8.3.8 Single-Carrier Frequency Domain Equalization

Having introduced to the frequency domain for the equalization purpose, we discuss the formation of the Tx sequence suitable for frequency domain processing first and then explain the block processing mechanism to mitigate the ISI. The term 'single-carrier' in SC-FDE is made clear when we discuss the other class – multicarrier systems – in Chapter 9.

The Cyclic Prefix (CP)

In the discussion on wireless channel and particularly the conversion from channel paths to channel taps in Section 8.1.7, we saw that the Rx signal is the convolution between the Tx signal and the channel impulse response. At $L = 1$ sample/symbol, the matched filter output is

$$z[m] = a[m] * h[m] \quad (8.71)$$

where $a[m]$ are the data symbols and $h[m]$ is the composite channel impulse response that includes Tx filtering, baseband channel and the Rx filtering.

We also know that convolution in time domain is equivalent to multiplication in frequency domain. This implies that the Rx signal in frequency domain is a product between the Tx spectrum and the channel frequency response[†].

$$Z(F) = A(F) \cdot H(F)$$

However, it is impossible to compute $H(F)$ on a digital machine. To view this process in frequency domain, the only practical tool available to us is the Discrete Fourier Transform (DFT). If we take the N_d -point DFT of $z[m]$ in Eq (8.71) where N_d is the number of data symbols, *is the resulting DFT $Z[k]$ a product between DFT $A[k]$ and DFT $H[k]$?*

$$Z[k] = ? \quad A[k] \cdot H[k] \quad \rightarrow \quad \text{No}$$

The answer is No. We found in Section 2.4.3 that while we can take the DFT of any $z[m]$, it is only equal to $A[k] \cdot H[k]$ when a circular convolution happens between $a[m]$ and $h[m]$. Eq (8.71) tells us that instead of a circular convolution, a regular convolution occurs in the real world.

To recap a little on circular convolution, remember that the way both time and frequency domain sequences are defined within a range $0 \leq n, k \leq N - 1$, DFT works with a circular shift which appears due to the inherent periodicity arising from sampling in frequency domain. Circular convolution between two signals, denoted by \circledast , is very similar to regular convolution except that circular – and not regular – flipping and time shifts are performed. For two signals $s[n]$ and $h[n]$, it is defined as

$$s[n] \circledast h[n] = \sum_{l=0}^{N-1} s[l]h[(n-l) \bmod N]$$

Since a circular convolution between the Tx signal and the wireless channel does not take place in practice, the main question is the following.

"What should be the form of the Tx signal in time domain such that the matched filtered signal $z[m]$ at the Rx is a result of circular convolution between $a[m]$ and $h[m]$?" In other words, how to make the Rx signal 'look like' having come from a circular convolution procedure when in reality it has come from a regular convolution process?

[†]Again, we loosely use the expressions $Z(F)$ and $A(F)$ for the Fourier Transforms of $z[m]$ and $a[m]$, respectively, although these are random sequences and should be described in terms of their spectral densities.

To answer this question, let us implement a convolution of a sequence $b[m]$ with another sequence $h[m]$ given by

$$b[m] = [2 \ -1 \ -2 \ 2 \ 1]$$

$$h[m] = [2 \ -1 \ 3]$$

Here, $b[m]$ has a length of N_d and $h[m]$ has a response length of $N_{\text{Tap}} + 1 = 2 + 1 = 3$. We trace the following sequence of steps.

Regular convolution: The matched filter output born as a result of regular convolution is plotted at the top of Figure 8.65 which has a total length of

$$N_d + (N_{\text{Tap}} + 1) - 1 = 5 + 3 - 1 = 7$$

Of these 7 samples, the first $3 - 1 = 2$ and the last $3 - 1 = 2$ samples constitute the transient response or group delay. The middle 3 samples, however, are the same as those from the circular convolution, as we see next.

Circular convolution: The circular convolution of $b[m]$ with $h[m]$ is also plotted in Figure 8.65. If we had this kind of result $z[m]$ at the matched filter output, only then its DFT $Z[k]$ would have been a product of the signal DFT $A[k]$ and the channel DFT $H[k]$. However, the results of the regular convolution and the circular convolution are equal only for the samples after the channel memory minus one. The problem is how to deal with the problem region, i.e., our task is now to transform the structure of $b[m]$ such that the first N_d samples are the same as the circular convolution.

Single repetition for periodicity: Our first idea is to recall Note 1.11 explained in Section 1.8 that just like sampling in time domain creates replicas of its spectrum in frequency domain, sampling the frequency axis of $B(F)$ – i.e., taking $B[k]$ at N discrete points from $-N/2$ to $N/2 - 1$ – should have created replicas of $b[m]$ in time domain. This directly points to the fact that making the Tx signal $b[m]$ periodic can help.

Let us find this out by performing the regular convolution again, but after repeating one cycle of $b[m]$, i.e.,

$$b'[m] = \left[\underbrace{2 \ -1 \ -2 \ 2 \ 1}_{b[m]} \mid \underbrace{2 \ -1 \ -2 \ 2 \ 1}_{b[m]} \right]$$

$$h[m] = [2 \ -1 \ 3]$$

During convolution, $h[m]$ is flipped and slid over $b[m]$ sample-by-sample. Counting from 0, the symbol instant 5 is particularly important because this is where the second period of $b[m]$ starts. So at symbol instant 5, these two sequences are drawn in the third subplot of Figure 8.65 with $h[m]$ flipped. The sum of these sample-by-sample products generates the regular convolution result at this instant.

The result of the regular convolution with a repeated cycle is drawn at the bottom of Figure 8.65. Notice how the shaded box contains exactly the outcome

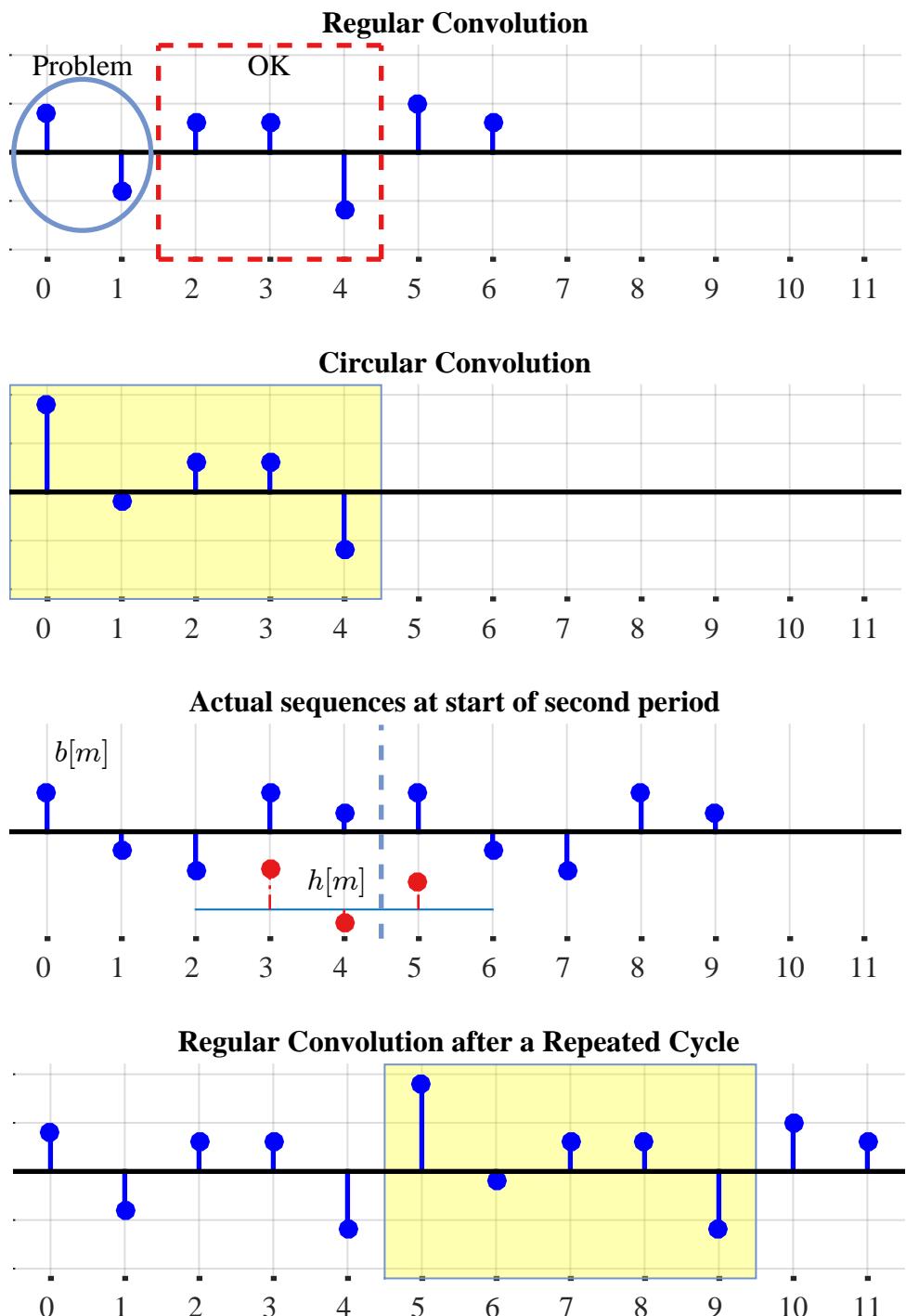


Figure 8.65: A technique to obtain circular convolution from regular convolution

of the circular convolution. Also observe that the first sample of the circular convolution starts at sample 5, exactly where the second period of $b[m]$ begins.

$$b'[m] = \left[\begin{array}{ccccc|ccccc} 2 & -1 & -2 & 2 & 1 & & 2 & -1 & -2 & 2 & 1 \\ & & & & & \uparrow & & & & & \end{array} \right]$$

convolution with flipped $h[m] \rightarrow [3 \quad -1 \quad 2]$

Clearly, making the Tx sequence periodic helps but repeating it all is a huge price to pay! Can we find a compromise?

Cyclic Prefix (CP): Observe that the channel $h[m]$ – at any single output sample of convolution – cannot affect the incoming signal beyond its own length. For example, in the third subplot of Figure 8.65, the channel length is 3, so only $3 - 1 = 2$ previous samples of the Tx sequence contribute in the convolution result at sample 5. These two previous samples are obviously the second last and the last samples of the original Tx sequence $b[m]$.

Therefore, the best compromise is to only repeat a portion of the Tx sequence *from the end* with length equal to channel memory minus one. This last portion of the Tx sequence, prefixed at the start of the modified Tx sequence, is known as a **Cyclic Prefix (CP)**. Since the channel length is $N_{\text{Tap}} + 1$ in our context, the length of the CP is N_{Tap} .

Figure 8.66a shows how to prepend (prefix + append) the last portion of the Tx sequence at its start while Figure 8.66b illustrates the result of regular convolution of the channel $h[m]$ with a cyclic prefixed Tx sequence $b''[m]$.

$$b''[m] = \left[\begin{array}{c|ccccc} \underbrace{2 \quad 1}_{\text{CP}} & 2 & -1 & -2 & 2 & 1 \end{array} \right]$$

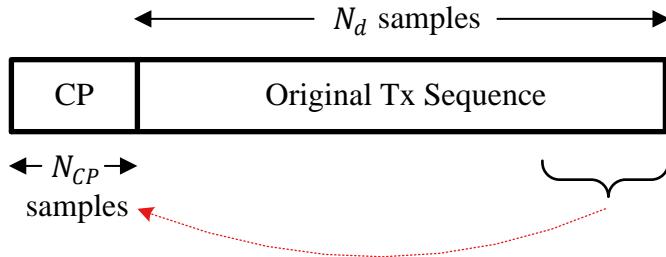
$h[m] = [2 \quad -1 \quad 3]$

It is evident that our desired circular convolution is present within the regular convolution of a cyclic prefixed Tx sequence. Such imitation of the circular convolution starts at sample N_{CP} where N_{CP} denotes the CP length.

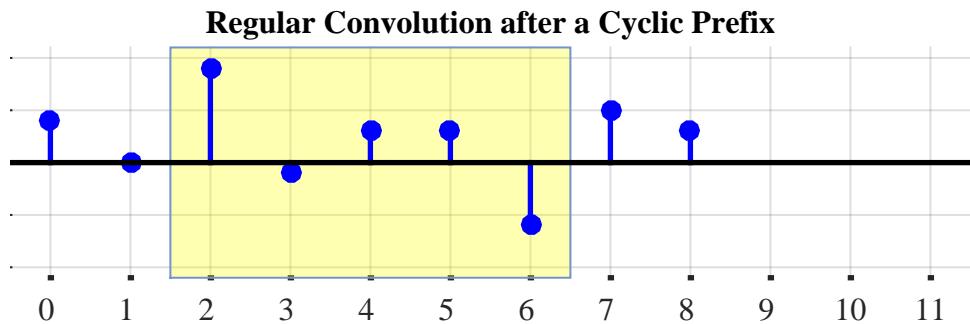
In general, we say that the length of a cyclic prefix should be equal to the maximum expected channel length. We do not know the channel beforehand but important parameters like maximum delay spread T_{Del} , coherence bandwidth B_C , Doppler spread B_{Dop} and coherence time T_C are found through extensive channel measurement campaigns. Then, the maximum delay spread is converted into maximum expected channel length by dividing it through the symbol time T_M and the result is chosen as the CP length N_{CP} .

$$N_{\text{CP}} = \frac{T_{\text{Del}}}{T_M}$$

Since the Rx sequence starting from sample N_{CP} is a circular convolution between the Tx sequence and the channel impulse response, we can now easily take it into frequency domain by taking the N_d -point DFT $Z[k]$ of this sequence *after discarding the CP*. This DFT is now a legitimate product of the DFTs of the Tx sequence and the



(a) A Cyclic Prefix (CP) consists of the last N_{CP} samples of the Tx sequence



(b) Regular convolution of the channel with a cyclic prefixed Tx sequence produces the result of circular convolution starting at sample N_{CP} chosen as N_{Tap} in this example

Figure 8.66: Prepending a Cyclic Prefix (CP) to the Tx sequence before regular convolution with the channel imitates the circular convolution

channel response.

$$Z[k] = A[k] \cdot H[k] \quad \text{for each } k = -N_d/2, \dots, N_d/2 - 1 \quad (8.72)$$

The CP is now useless as it just contains the transient response of the channel. Later, we will see that it also gets polluted with ISI from the previous symbol when used with block processing. Therefore, it is discarded right away at the Rx side. The CP is the price we pay for launching a high data rate signal (geared up for simpler frequency domain equalization) into the environment. This is similar to a rocket launching a shuttle into space and then it either falls off in the ocean, burns up or becomes space junk.

Now we have a deeper look into the frequency domain processing.

A View into Frequency Domain

With a CP in place and hence a circular convolution, the DFT is a perfect representation of what happens in frequency domain. To understand this process, we have to see what the signal in frequency domain is made of. For this purpose, consider the

DFT definition.

$$\begin{array}{ll} I \rightarrow & Z_I[k] = \sum_{m=0}^{N_d-1} \left[z_I[m] \cos 2\pi \frac{k}{N_d} m + z_Q[m] \sin 2\pi \frac{k}{N_d} m \right] \\ Q \uparrow & Z_Q[k] = \sum_{m=0}^{N_d-1} \left[z_Q[m] \cos 2\pi \frac{k}{N_d} m - z_I[m] \sin 2\pi \frac{k}{N_d} m \right] \end{array}$$

for each $k = -\frac{N_d}{2}, -\frac{N_d}{2} + 1, \dots, -1, 0, 1, \dots, \frac{N_d}{2} - 1$.

Recall from Section 1.8 that each $Z[k]$ output is just a sum of term-by-term products between $z[m]$ and complex sinusoids whose frequencies are k complete cycles in an interval of N_d samples (or k/N_d cycles per sample). In other words, the DFT process eventually finds the contribution of every such cosine/sine wave in construction of $z[m]$ through *correlating* the input signal with the reference sinusoids.

Now the input signal $z[m]$ is a convolution between the data symbols $a[m]$ and the channel impulse response $h[m]$. Consequently, each frequency bin k contains the contribution from all data symbols $a[m]$ multiplied with the complex channel coefficient $H[k]$ in frequency domain (which is equal to the continuous channel frequency response in the middle of the bin k). This is known as a *subchannel k* and is drawn in Figure 8.67 for subchannels 0 to 3.

Once the Rx signal is in frequency domain through computing its FFT, the equalization process is straightforward. Assuming a known channel estimate $\hat{h}[m]$ and hence $\hat{H}[k]$ as well as negligible noise, the Tx signal can be easily equalized from Eq (8.72) according to either zero forcing or minimum mean square error criteria discussed before in Section 8.3.3. According to zero forcing criterion for instance, the equalized signal in frequency domain is

$$\hat{A}[k] = \frac{Z[k]}{\hat{H}[k]} \quad (8.73)$$

for each $k = -N_d/2, \dots, N_d/2 - 1$

From the above equation, the data symbols $a[m]$ can be computed by taking the iDFT of the above sequence.

$$\hat{a}[m] = \text{iDFT } \{\hat{A}[k]\} \quad (8.74)$$

for each $m = 0, 1, \dots, N_d - 1$

We now discuss block processing that is the key to overcoming ISI in high rate systems and how a CP fits into that picture.

Block Processing Mitigates ISI

We have previously seen our fundamental problem that the time domain equalizer takes a long time to converge before we can actually detect valid Tx symbols. We have also studied the role of CP in converting the Rx sequence into frequency domain through a DFT. However, notice that there is not much use in frequency domain processing for a long Tx sequence if we need to

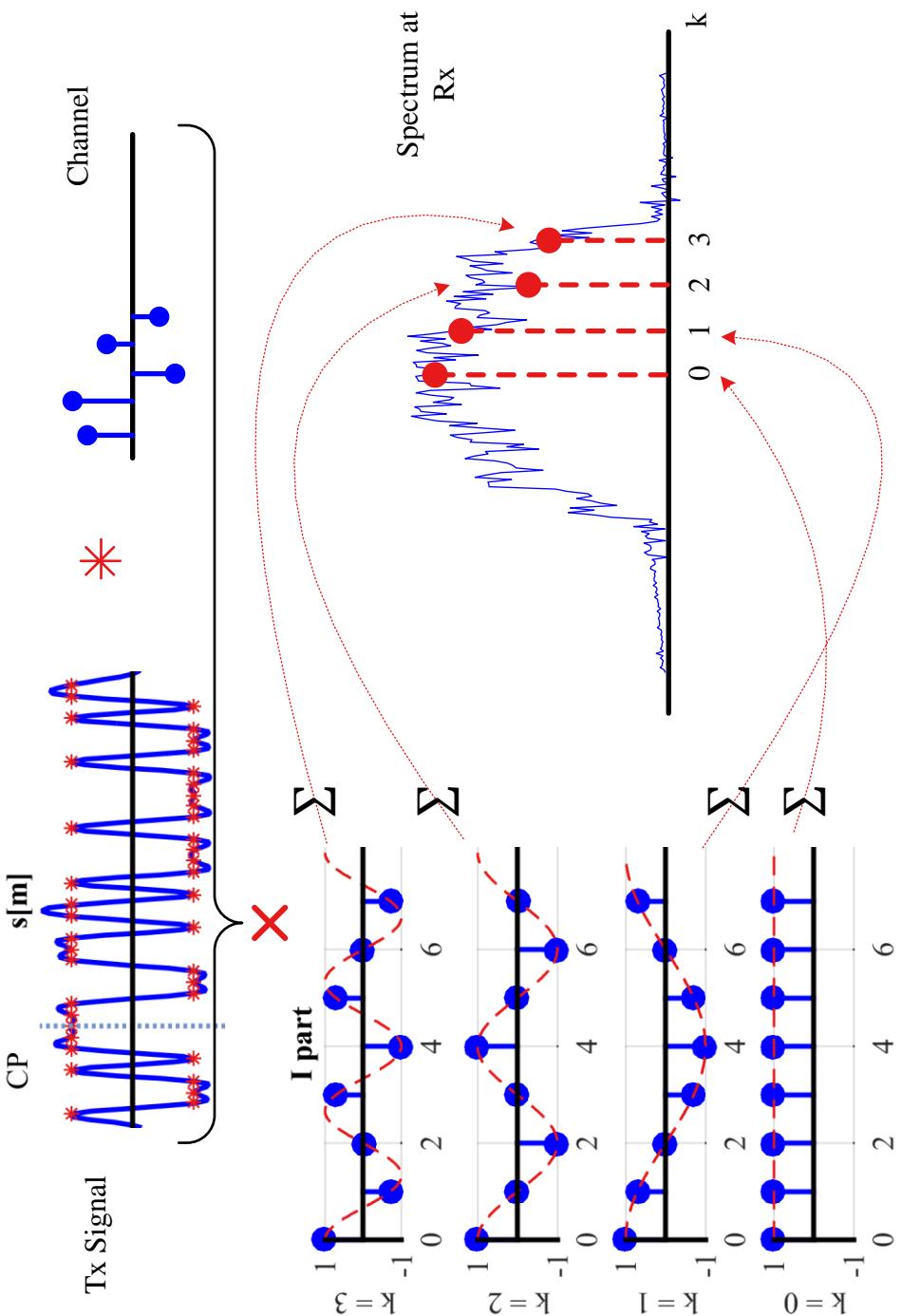


Figure 8.67: Subchannels created at the Rx as a result of DFT processing

- wait at the Tx side for the end of the sequence to prepend the CP at the start, and
- wait for the Rx sequence to finish before taking its DFT.

The solution is to break the Tx sequence down into fixed size segments or blocks for ease of processing. Segmentation has been an effective solution throughout the history in tackling complex problems, just like carriages in a train or drawers in a cupboard.

In block processing, the data symbols are collected into groups of size N through a serial-to-parallel converter as drawn in Figure 8.68. The last N_{CP} symbols from the end of each such group are copied to the start of the sequence. Hence, the output sequence for each block is $N + N_{CP}$ symbols in length, which is then converted from parallel to a serial stream. The rest of the operations are similar to a regular

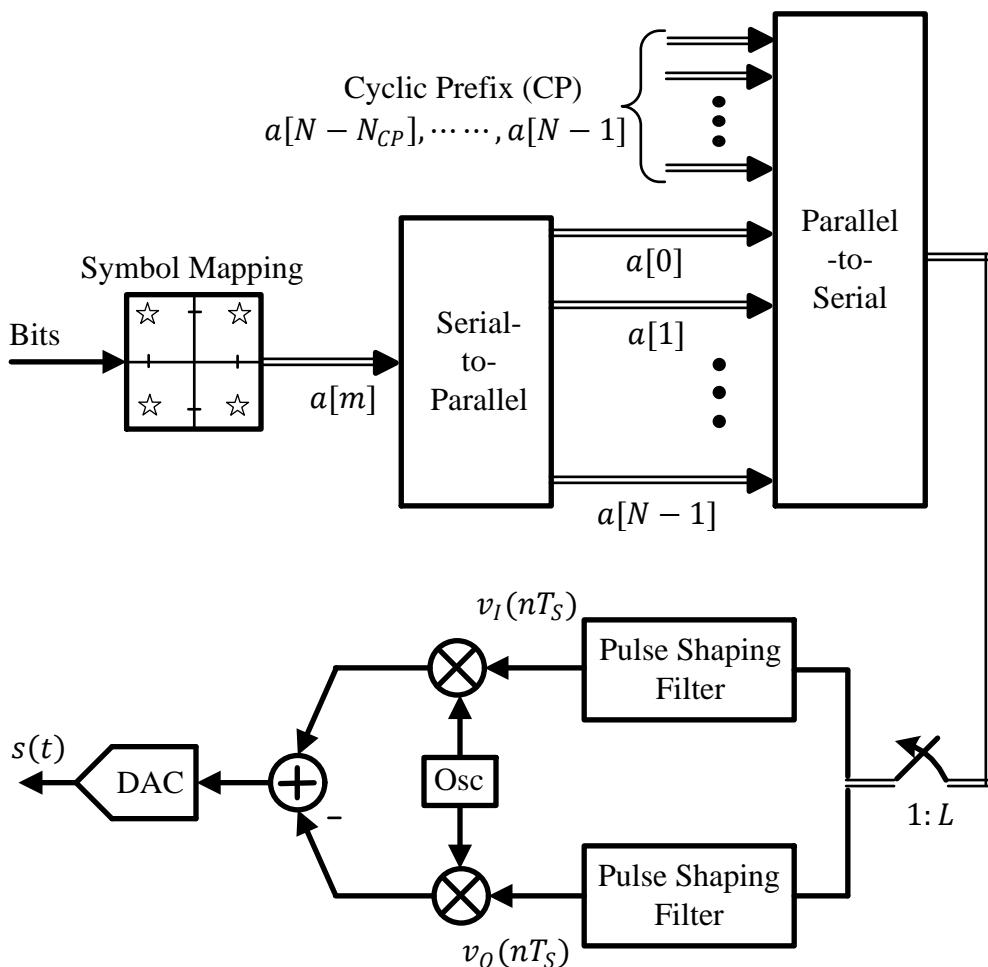


Figure 8.68: A block diagram for the Tx that facilitates block processing for a frequency domain equalizer

communication system.

The main point about the signal processing in Figure 8.68 is that it is a continuing procedure in which the Tx chops the incoming set of symbols into fixed length blocks before the cyclic prefix insertion. As we now see, converting regular convolution with the channel into circular convolution is not the only reason for which a cyclic prefix is employed. We have a second, even more compelling, reason for using the CP as a guard interval that mitigates the ISI from one block corrupting the next. Recall that we covered a conventional and an intuitive method to study convolution in Section 2.4. We apply both methods here to see this utility.

Conventional method: During the convolution process as observed in a conventional manner, the channel impulse response $h[m]$ slides over the cyclic prefixed Tx sequence. When the convolution with one block reaches the last sample of that block as shown in Figure 8.69a, *the channel influence has not reached beyond the CP of the next block*. Since the DFT window starts at sample N_{CP} , no interference from the previous block touches the currently received block and Inter-Symbol Interference (ISI) is avoided. In this context, we should rather say that Inter-Block Interference (IBI) is avoided but the underlying concept stays the same.

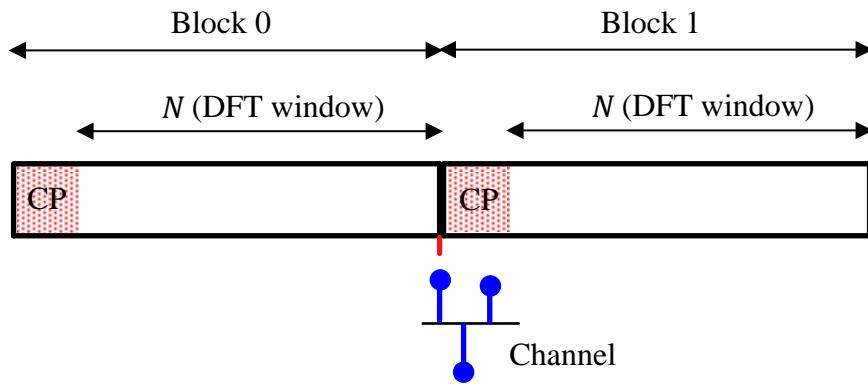
Intuitive method: The convolution through intuitive method between the Tx sequence and the wireless channel is drawn in Figure 8.69b. Block 0 cannot have any ISI since there is no previous block. However, Block 1 is contaminated by the reflections of Block 0 due to the multipath channel. Nevertheless, as long as the length of the channel remains shorter than the cyclic prefix, the last arriving path cannot add any part of Block 0 to the start of Block 1 (where its CP ends). The DFT window of size N taken for Block 1 clearly depicts an ISI-free scenario. The multipath reflections do add up at the Rx antenna but the summation happens between the portions of the same block. Therefore, an equalizer is still needed but the channel is successfully fooled into becoming a flat fading channel (for one block's interference into another) despite the high data rates. We will say more about this in multicarrier communications in Chapter 9.

Continuing the same reasoning, the tail of Block 1 then injects interference into the CP portion of Block 2 (not shown) and so on.

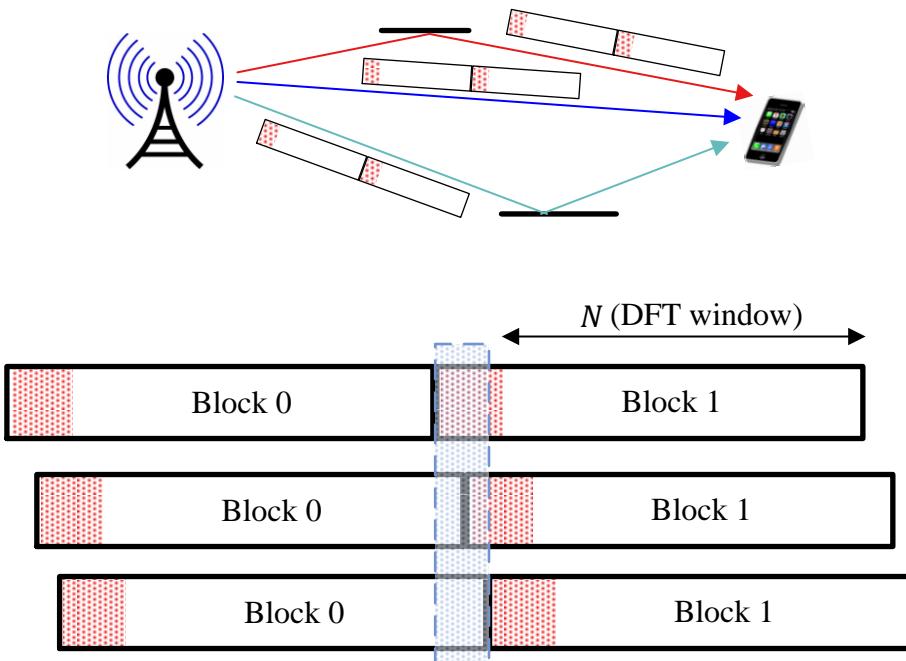
In a scenario where the channel impulse response is longer than the cyclic prefix, the ISI from the previous symbol distorts the current symbol and the performance of the simple frequency domain equalizer deteriorates. It is therefore imperative to run extensive channel measurements so as to figure out the most suitable numbers for the block and CP lengths that work well in real transmissions.

The Rx Structure

At the Rx side, the conventional tasks of matched filtering, frame synchronization, timing synchronization and channel estimation are performed. Then, a serial-to-parallel converter prepares the input for DFT computation. However, the first N_{CP} samples are first discarded since they are contaminated from the ISI of the previous block. Next, an N -point DFT is taken by using its efficient version, namely FFT, and multiplied with the channel inverse $1/\hat{H}[k]$ to equalize in frequency domain. A subsequent iDFT in



(a) Conventional method



(b) Intuitive method

Figure 8.69: Conventional and intuitive methods for convolution between the Tx sequence and wireless channel. Notice no Inter-Block Interference for $N_{CP} > N_{\text{Tap}}$

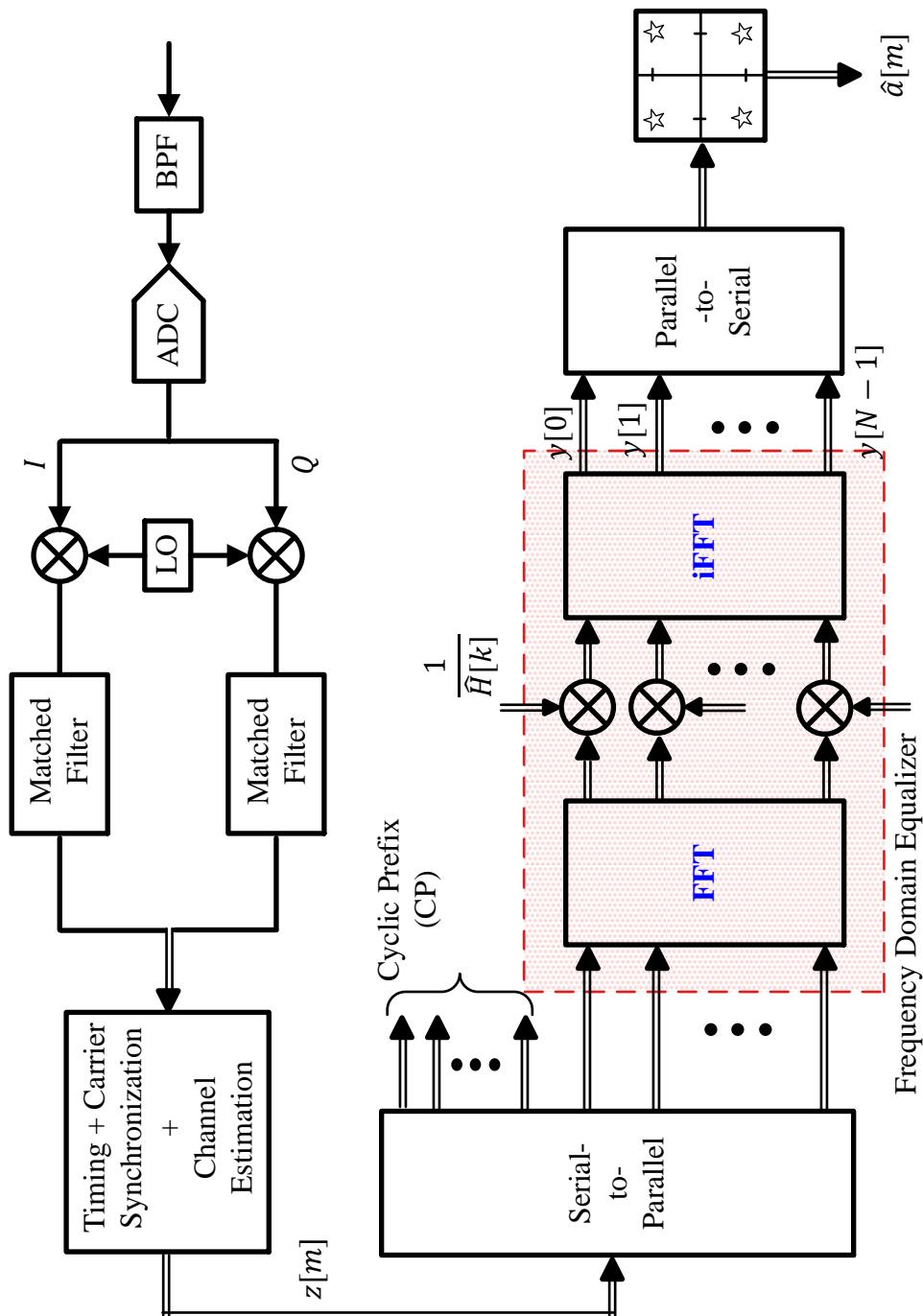


Figure 8.70: A block diagram for the Rx that implements a frequency domain equalizer

the form of iFFT transforms the equalized signal back in time domain which is parallel-to-serial converted and provides the input to the decision device. A block diagram of such a Rx is drawn in Figure 8.70.

We now consider an example to illustrate the process of frequency domain equalization.

Example 8.5

Consider a sequence of $N_d = 32$ Pulse Amplitude Modulated (PAM) data symbols shaped by a Raised Cosine pulse with excess bandwidth $\alpha = 0.5$ and $L = 4$ samples/symbol. A Raised Cosine pulse is used instead of a Square-Root Raised Cosine just for demonstration purpose because with zero noise in the linear system, the order of Tx, channel and Rx filtering can be interchanged. In the form of a cyclic prefix with length 8, the last eight symbols are prepended to the sequence.

Such a pulse shaped symbol stream is drawn in Figure 8.71a along with its spectrum in Figure 8.71b. Next, a realization of length 7 random multipath channel (shorter than the cyclic prefix) is generated whose magnitude response is plotted in Figure 8.71c. Observe channel nulls within the portion that overlaps with the Tx signal spectrum.

The equalizer input signal spectrum is a product in frequency domain between the Tx spectrum and channel frequency response and plotted in Figure 8.71d for $L = 1$ sample/symbol. It exhibits a distorted spectrum compared to that of the Tx signal spectrum. As a consequence, the Rx waveform is distorted in time domain as well. Since the channel is in general complex, the Rx signal consists of both I and Q time domain components. This is shown in Figure 8.71e and Figure 8.71f where both I and Q parts of the Rx signal can be seen to be severely distorted as compared to the original Tx signal. Assuming perfect frame and timing synchronization, the CP has already been discarded at the Rx side.

Finally, a frequency domain zero forcing equalizer from Eq (8.73) is applied to the Rx waveform and the equalized spectrum is shown in Figure 8.71g for $L = 1$ sample/symbol. Due to the perfect channel estimate available at the Rx and infinite SNR, perfect signal recovery is possible. Therefore, the time domain waveform and the data symbol decisions in Figure 8.71h match those at the Tx (starting after the CP in Figure 8.71a).

The equalizer job seems straightforward in the example above and the Tx signal is almost exactly recovered at the Rx. This does not happen in practice because

- the channel estimates $\hat{h}[m]$ available at the Rx are not perfect,
- the presence of noise spreads the Rx constellation in the form of clouds around their ideal locations and causes decision errors, and
- there is noise enhancement in the particular structure of the zero forcing equalizer.

Cyclic Prefix vs Zero Padding

Figure 8.69a and Figure 8.69b also reveal that as an alternative to the cyclic prefix, a zero padding approach can also be employed in which no signal is transmitted instead of N_{CP} samples. This is because the real purpose of block processing is to mitigate ISI from one block into the next *at the Rx* and any guard interval – even all-zero samples – suffices in this regard.

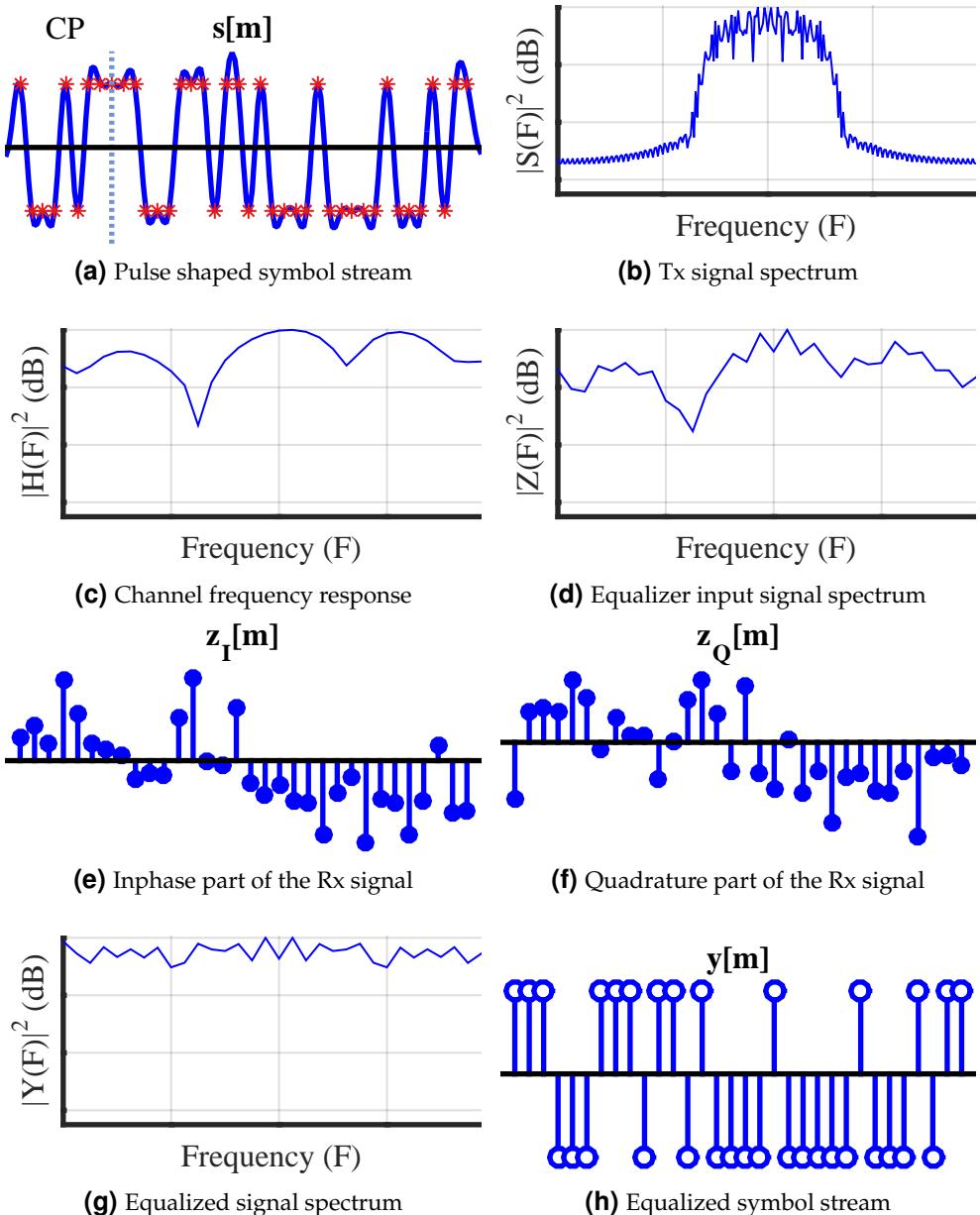


Figure 8.71: Equalization in frequency domain

In such an implementation, a regular convolution can be converted into a circular convolution through adding the samples from the next empty guard interval back into the start of the block. This is evident in Figure 8.69a because if the CP was replaced with all-zero samples, the guard interval will contain the convolution result of the end samples with the channel, which when added cyclically back to the start of the previous block produces an effectively circular convolution.

The main advantage of a zero padding approach is slightly reduced average power

because no energy is allocated to transmit CP samples. However, there are two disadvantages of such an approach.

1. Adding the samples from the next guard interval back to the start of the block results in double addition of noise, one originally present in the samples at the start of the block and the other coming from the added samples from the guard interval. This is somewhat similar to noise enhancement through equalization at the Rx; if the channel was perfectly known at the Tx (and the dynamic range allowed it), doing equalization at the Tx would not enhance any noise.
2. A zero padded block, i.e., an empty guard interval, increases the peak to average power ratio which puts constraints on the Tx analog frontend. Specifically, for the same average power, power amplifiers that can handle higher peak power prove more costly for the system.

Nevertheless, zero padding has been used in some Ultra-Wideband (UWB) and millimeter-wave SC-FDE systems.

A Comparison of Time and Frequency Domain Equalizers

Single-Carrier Frequency Domain Equalization (SC-FDE) is more suitable for high data rate wireless communication systems as compared to time domain equalizers due to the following reasons.

Computational complexity and convergence: As explained in Section 8.3.7, a time domain equalizer for higher data rates becomes the main bottleneck in maintaining continuous transmission. In some cases, it is not surprising for the designer to allocate 70% or higher resources at the Rx to the equalizer operation alone [28]. In addition to this computational complexity, the equalizer needs a reasonably long time to converge. Imagine a high rate system waiting for the equalizer to converge or loops to settle[†]!

On the other hand, as long as the CP is longer than the channel length $N_{\text{Tap}} + 1$, the complexity of the frequency domain equalizer is low and determined by the DFT window size N . Since Fast Fourier Transform (FFT) is the same as DFT with the redundant operations squeezed out, an FFT and iFFT is used at the Rx side for equalization purpose. The complexity of the FFT operation is proportional to $N \log_2 N$.

As a consequence, FDE is a preferred method of operation unless the channel length $N_{\text{Tap}} + 1$ is very small. Then, the cost of going into the frequency domain through an FFT and returning to time domain through an iFFT exceeds that of the direct time domain equalization. As a side remark, frequency domain equalization can be implemented in conventional single-carrier systems without a CP by using fast convolution methods such as overlap-add or overlap-save. Nevertheless, their performance is degraded with respect to a CP based block structure.

Channel inverse: As we saw earlier, a zero forcing equalizer attempts to operate on the Rx signal in time domain such that the channel in frequency domain is

[†]Switching from time domain to frequency domain for equalization purpose has been the single most significant factor in wireless systems breaking through the promise of delivering data rates in hundreds of Mbps.

effectively inverted. However, we also observed that perfect channel inversion is only possible for an infinite length equalizer because the result of the convolution between the Rx sequence and the equalizer is always longer than that of the equalizer impulse response.

In SC-FDE, the implicit circular convolution brings at the Rx a signal that can be taken into frequency domain and perfectly inverted. This is because the equalizer works on the exact DFT based complex sinusoids instead of an implicit approximate inversion in time domain.

Time variations: A single-carrier system with an adaptive time domain equalizer does not need to worry about a slowly time-varying channel because it can track such variations similar to a Phase Locked Loop (PLL) that can accommodate a slowly varying phase. The price paid for such luxury is the initial convergence time and the corresponding discarded symbols that is paid upfront once and for all.

For each block of an SC-FDE system, we assume that the channel is static throughout the duration of $N + N_{CP}$ samples so that the system is LTI and the Rx signal can be treated as a convolution output. This is known as a *quasi-static* assumption and is widely utilized in the design of most wireless systems that deal with ISI through block processing. In other words, the coherence time T_C of the channel is greater than the block length.

$$T_C > N + N_{CP}$$

For high rate systems, it makes sense to restrict N to a value such that a quasi-static assumption holds true and a time-varying channel does not need to be taken into account. Moreover, the FFT, iFFT and the processing of other algorithms at the Rx can be managed reasonably well with a restricted N .

On the other hand, we can also see that with each block, there is an overhead equal to the guard length N_{CP} . In terms of percentage,

$$\text{Transmission overhead} = \frac{N_{CP}}{N + N_{CP}} \times 100\%$$

Clearly, the environment where the system is deployed determines the CP length and hence the reduction in spectral efficiency. This transmission overhead can be controlled through a larger N . A good design eventually balances both of the above conflicting requirements for choosing the block length N .

Note 8.13 Antifragile

According to Merriam-Webster dictionary, the meaning of fragile is “easily broken or damaged, very delicate, not strong”. The opposite of fragile is resilient or robust. According to Nassim Taleb,

“Some things benefit from shocks; they thrive and grow when exposed to volatility, randomness, disorder, and stressors and love adventure, risk, and uncertainty. Yet, in spite of the ubiquity of the phenomenon, there is no word for the exact opposite of fragile. Let us call it antifragile. Antifragility is beyond resilience or robustness. The resilient resists shocks and stays the same; the antifragile gets better.”

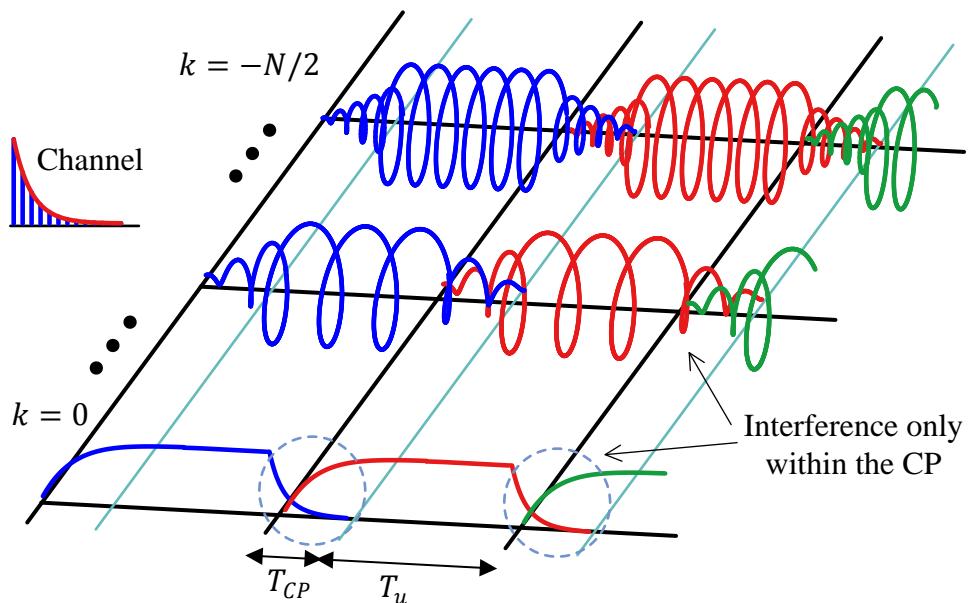
Nassim Nicholas Taleb - Antifragile: Things That Gain From Disorder

From the discussion on equalizers, we have come to know that an equalizer makes a wireless communication system robust against multipath fading by exploiting the sum of multiple copies and hence provides frequency diversity. The presence of multipath, however, does not make such a system better.

On the other hand, Multiple Input Multiple Output (MIMO) systems can exploit a rich multipath fading environment to create independent spatial streams between a Tx and a Rx. We can say that such a framework makes the wireless systems antifragile by gaining from the multipath disorder.

Chapter 9

Orthogonal Frequency Division Multiplexing (OFDM)



“If you want to find the secrets of the universe, think in terms of energy, frequency and vibration.”

Nikola Tesla

Since around the start of the 21st century, high data rate wireless communications is often being realized through an Orthogonal Frequency Division Multiplexing (OFDM) system. OFDM is a special case of Multi-Carrier (MC) systems as opposed to a conventional Single-Carrier (SC) system we have discussed so far in the previous chapters.

OFDM is the technology behind many high speed systems such as WiFi (IEEE 802.11a, g, n, ac), WiMAX (IEEE 802.16), 4G Long Term Evolution (LTE) and 5G mobile communication systems. A close cousin, Discrete Multi-tone (DMT), is used in ADSL and powerline communication systems. As a consequence, there has been a tremendous research activity in this field since 1990s.

The concepts on which OFDM is based are so simple that reading from many resources is not required to understand the fundamentals. Furthermore, as opposed to single-carrier systems where the prevalent material everywhere can be too daunting for a non-mathematician radio enthusiast, a large number of excellent tutorials and books are available on OFDM. Nonetheless, I bring my own perspective of viewing the OFDM signals in this chapter.

Single-Carrier vs Multicarrier Systems

A few points regarding single and multicarrier systems are in order.

5 vs 1 chapters: OFDM is a multicarrier technique as opposed to a single-carrier system discussed so far. We have devoted five chapters to understanding the implementation of a single-carrier system, so why just a single chapter on OFDM? There are several reasons for this choice.

First, the material on single-carrier systems enabled us to deeply visualize how the signals behave and subsequently develop neat mental models for understanding communication signals. This is a valuable tool and is useful in any context, even when you go on to read about OFDM or Multiple Input Multiple Output (MIMO) systems not discussed in this text. So it is not really necessary to allocate one chapter to each problem arising in an OFDM system (although it can certainly be done). The tools devised before can help us get to the crux of the problem and solve it right away. For example, many of the insights from symbol timing synchronization in a single-carrier system can be brought into looking at the effect of carrier frequency offset in an OFDM system.

Second, a single-carrier system can be implemented in a relatively diverse manner. For equalization and each type of synchronization block, feedforward and feedback, data-aided/decision-directed and blind solutions are available that can be mixed and matched together. As we will later see, OFDM is implemented through a series of fast processing routines that operate in a process and dump fashion one after another. This ease of processing comes from the particular signal structure that leaves little room for a diverse mix of algorithms. Although the literature available on OFDM far exceeds that on any other technology, it

still mostly consists of modifications around that fixed signal structure and hence less versatile than single-carrier systems.

The role of single-carrier systems: As mentioned above, a number of modern high-speed wireless communication systems are based on OFDM or multicarrier technique in general. It is easy to then focus on understanding OFDM or multicarrier systems only. However, single-carrier systems are just as important and in fact, a multicarrier system can be considered as another kind of a single-carrier system in terms of an elongated symbol described later. So an understanding of single-carrier systems helps solve the issues in multicarrier context as well.

Before we start, I touch upon a bigger picture along with a brief history of OFDM.

9.1 The Big Picture

The basic multicarrier principle of segmenting one bit stream into several parallel streams and using them to simultaneously modulate independent carriers was first used in the Collins Kineplex System. It largely remained in the shadows for almost three more decades with some sporadic developments of significance. This is because an array of parallel sinusoidal generators was quite complex to build through analog modules.

Then in 1971, Weinstein and Ebert [37] showed that a Discrete Fourier Transform (DFT) – a fast version of which is FFT (Fast Fourier Transform) – can be employed to efficiently implement such a system. Later in 1980s, OFDM was proposed as the solution to the challenging problem of digital TV broadcasting through terrestrial networks in Europe in the context of a single frequency network.

Moore's law for the integrated circuits[†] in the meantime was catching up in the radio domain and by the 1980s, real-time digital signal processing advanced to a level that many of the operations performed by the analog circuits could be replaced by a mere play with numbers, thus giving birth to the concept of a Software Defined Radio (SDR). It is good to note that the SDR concept was not born out of thin air but instead was originally pushed forward in 1970s mainly by the efforts of the usual suspect – the US Department of Defense.

Interestingly, an SDR is *the natural partner* to digital communication theory in general. We know that digital communication is mostly about conveying discrete amplitudes and phases, i.e., concrete numbers, from point A to point B. It was a great mismatch then that a major portion of such number manipulation was being performed through analog circuitry and the SDR concept just brought the digital information processing back to where it belonged.

In light of the above, we can say that OFDM is the most prominent flag-bearer of the general SDR concept. This is no surprise then that the two landmark papers announcing the supremacy of OFDM and SDR, respectively, were published very close in time to each other:

[†]Gordon Moore, the co-founder of Fairchild Semiconductor and Intel, predicted in 1965 that the number of transistors in a dense integrated circuit doubles about every 18 months. This scaling down of the transistor size has spawned revolutionary growth in the world of computing, electronics and society in general in the last 50 to 60 years. In fact, if we subtract the effect of Moore's law from the past half a century, humankind has relatively little to show for its progress.

1. “Multicarrier Modulation for Data Transmission: An Idea whose Time Has Come” by J. Bingham [38], and
2. “The Software Radio” by J. Mitola [39].

While SDR was the future of both digital and analog communication in any case, OFDM played a significant role in its traction.

Now even more interestingly, there was a noteworthy development around the same time in computer peripheral communication history. While a serial interface transmits one bit at a time, a parallel port sends multiple bits of data at once through multiple parallel wires in its cable and port connector. The most prominent parallel port, commonly known as a printer port shown in Figure 9.1, was widely used in personal computers from the 1970s to 2000s. At that time, designers found it easier to add extra hardware for high rates such that the ideal aggregate data rate was the individual rate multiplied with the number of parallel links. Again, the integrated technology has progressed to a point that serial links (e.g., a USB) with fewer pins and subsequent device size reduction have become the standard and parallel port has gradually sunk into oblivion.



Figure 9.1: A parallel port (image from Wikipedia) has gradually been replaced by a serial port (USB)

This development has exactly been opposite to the wireless systems where the trend has moved from serial to parallel communication. We will learn in this chapter that OFDM is nothing but a set of several parallel waves in the air instead of one (and hence the term multicarrier). This parallel trend has infiltrated the wireless industry not only through a multicarrier system, but also from multiple antennas at the Tx and/or the Rx. Such a system is known as a Multiple Input Multiple Output (MIMO) system that has added a delightful extension to the Rx algorithm design which unfortunately is outside the scope of this text.

The main highlight of this parallel processing for fast data rates is the emergence of *process and dump* philosophy of the algorithm design, see Figure 8.64 as a reference. Whether it is the timing or frequency synchronization, phase synchronization or (most importantly) equalization, each Rx block takes as an input a set of samples and processes them in a feedforward manner. The natural casualty of such a design are recursive loops which are slowly reducing in scope from the packet based systems. Who wants a time domain equalizer to converge (or to a lesser extent, an x-locked loop to settle) while watching an HD video on the train!

Another aspect of this history appears when we consider that the rise of OFDM almost coincides with the rise of capacity approaching error correcting codes. Also known as a Forward Error Control (FEC), it is a process of detecting and correcting bit errors in digital communication systems by adding some redundancy to the Tx

bit stream in an intelligent manner. Turbo Codes took the communications world by storm (and enough suspicion) when their discovery was first published in 1993. Later on, Low Density Parity Check (LDPC) codes were rediscovered in 1996 that exhibit slightly superior performance in some respects.

Like a trick played by nature, the main feature that propels these codes closer to the channel capacity is their *feedback loop* during the decoding process: several iterations need to be performed within the decoder to *converge* towards a desirable

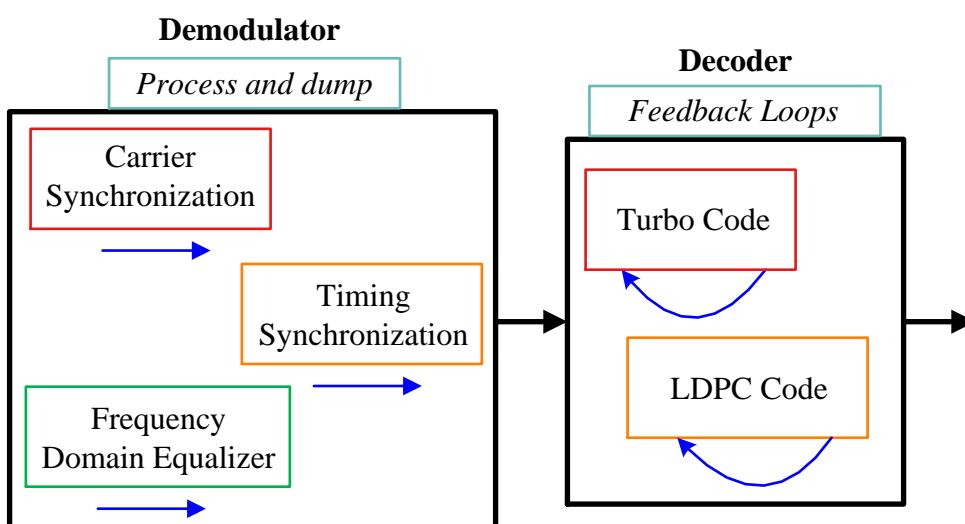
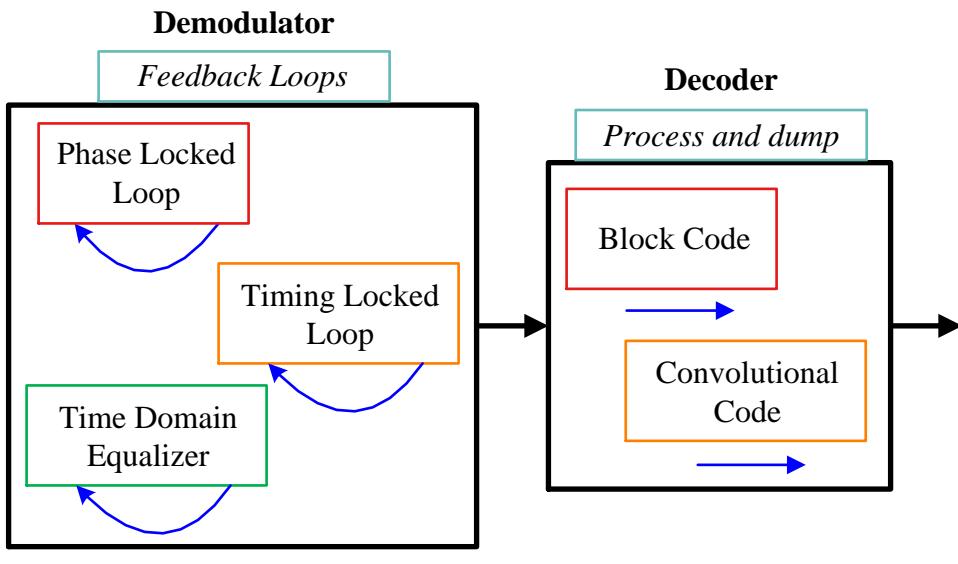


Figure 9.2: During the radio evolution, feedback loops are now less preferred in the demodulator but necessarily implemented in the iterative decoders

point on the BER/SNR curve. As illustrated in Figure 9.2, the feedback loops that were gradually being dismissed from the signal demodulation process have reappeared in the signal decoding procedures! It seems that the radio is evolving like an organism.

This evolution is also manifested in the form of a very large number of antennas in massive MIMO systems. A direct consequence of this transformation is that beam-forming – a luxury for 20th century radios – has become the defining feature of the next generation of wireless systems. This is hardly surprising since a luxury for one generation almost always becomes a necessity for the next.

With this background in place, it seems imperative to have a signal level understanding of how OFDM works.

9.2 How OFDM Works

We start this topic with an analogy that can help us better visualize the transmission and interference processes. I highly recommend reading about the wireless channel in Section 8.1.1 to fully comprehend what comes next.

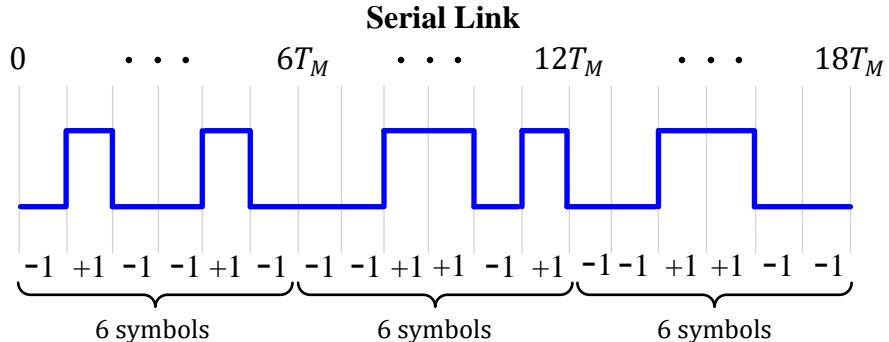
9.2.1 From the Printer Port

First, recall from Section 8.1.1 that the major problem with high speed wireless communication is that the multipath components arriving at the Rx cause different attenuations plus phase shifts and consequently constructive and destructive interference throughout the signal span. This multipath depends on the physical environment around the Tx and Rx. For low data rates, the multipath arrive very close in time to each other with respect to the symbol rate $1/T_M$. However, as we increase the data rate, each symbol starts interfering with dozens of symbols in the future, a phenomenon known as Inter-Symbol Interference (ISI).

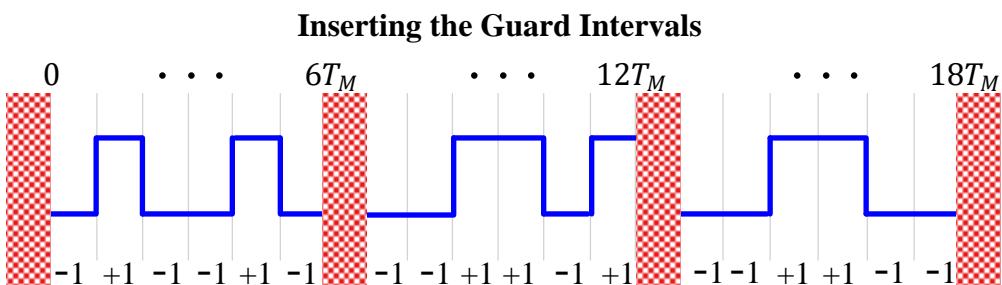
Since this physical environment cannot be changed, a safe method to avoid this long reaching effect of ISI is to reduce our symbol rates. The reduction of symbol rate is then insufficient for meeting our growing needs for faster data rates. What to do in this situation?

Assume that like a printer (parallel) port shown in Figure 9.1 before, the system designer has access to an array of parallel wires in the air that do not interfere with each other during their flight. We will see later how these parallel wires can be elegantly created. It is evident that if N such parallel wires are included in the design, the symbol rate $1/T_M$ can be simply reduced by a factor of N , i.e., the new symbol rate (actually, the block rate) can be $1/NT_M$ without compromising on the overall throughput of the system. This is shown in Figure 9.3 where the block duration $NT_M = 6T_M$ is indicated at the bottom.

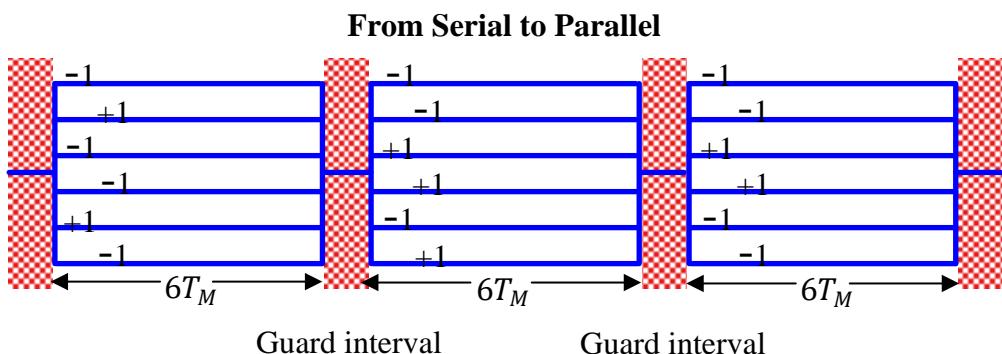
In Figure 9.3a, a serial data stream with symbol duration T_M is shown. Next, it is segmented into blocks of $N = 6$ symbols and a guard interval is inserted within each such block as plotted in Figure 9.3b. If you have read Section 8.3.8 on frequency domain equalization, then this guard interval is actually the Cyclic Prefix (CP). Otherwise, just take this guard interval as a shield for preventing the next block from the multipath interference of the previous block. Next, each symbol in a block of $N = 6$ symbols is sent on a set of $N = 6$ parallel wires shown in Figure 9.3c where each wire is $N = 6$ times longer than the symbol duration T_M . Due to parallelism, there is



(a) A serial symbol stream with symbol duration T_M



(b) A guard interval of duration T_M inserted into the above signal



(c) $N = 6$ symbols collected into one block and sent on N parallel wires

Figure 9.3: A serial data stream is segmented into blocks of $N = 6$ parallel wires where the length of each wire is $N = 6$ times that of the original symbol T_M

no interference within the data symbols on one set. Moreover, the guard interval prevents the multipath of one block smearing into the next as long as the channel length is lesser than the guard interval.

The main theme to focus in Figure 9.3 is how the serial symbol stream of +1s and

-1 s are mapped on parallel symbol streams and sent on separate wires in the air. For example, the first six symbols $-1, +1, -1, -1, +1, -1$ are all sent to the first block starting from top to bottom.

A simple example demonstrates what we gain from such a serial to parallel conversion of the data symbols.

Note 9.1 Fooling the Channel

Assume that the wireless channel in which such a communication system is deployed has a maximum delay spread T_{Del} of about a symbol time $1T_M$, i.e., it interferes with one next symbol. Then, the multipath copies of the serial link will arrive T_M seconds later. Temporarily neglecting the impact of carrier wave, each symbol copy will overlap with the next symbol, thus completely randomizing the Rx sequence (since the source generates an independent stream of $+1$ s and -1 s). According to the definition, this is a frequency selective channel and the resultant signal is drawn in the upper part of Figure 9.4.

For the parallel mode, a copy of the elongated symbol of duration $6T_M$ will also be delivered after a delay of $1T_M$, since the multipath channel is the same. However, as opposed to a complete overlap of one symbol into the next, we will have a relatively short overlap of just $1T_M$ which is $1/6$ or 16.7% of the parallel symbol duration $6T_M$. Consequently, this is the case of a frequency flat channel if it is defined as having a delay spread of approximately 15% of the symbol time.

“By introducing parallelism, we have fooled a frequency selective channel into behaving as a frequency flat channel!”

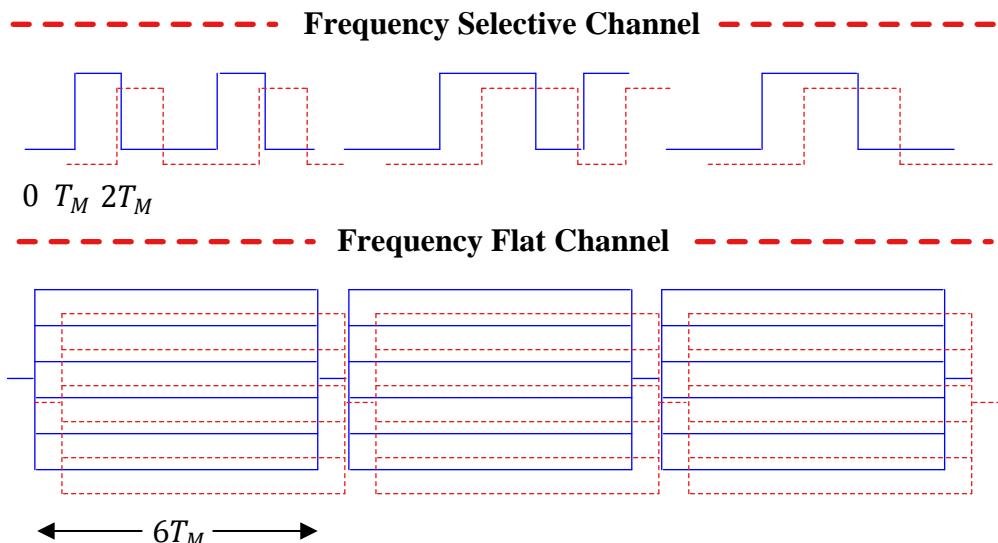


Figure 9.4: By rearranging the symbols from serial to parallel, a frequency selective channel (for symbols of duration T_M) has been converted into a frequency flat channel (for elongated symbols of duration $6T_M$)

This is shown in the lower part of Figure 9.4 where by rearranging the symbols from serial to parallel, a frequency selective channel for symbols of duration T_M has been converted into a frequency flat channel for elongated symbols of duration $6T_M$.

Also observe that due to the guard interval, the delay spread of the first elongated symbol will not bump into the next elongated symbol. By increasing N significantly larger than 6, a severe (much longer) frequency selective channel can be turned into a frequency flat channel as well.

Parallel Wires in the Air

To avoid using the confusing terms of a symbol and an elongated symbol, we will call this *an OFDM symbol* from now onwards. By definition, the duration of an OFDM symbol is NT_M , i.e., N times longer than a conventional symbol duration T_M (later, we will see that its duration increases after inserting the guard interval). Now, the question is how to create these parallel wires in the air.

To create those parallel wires of length NT_M in the air, let us go back to Section 1.7 on the set of discrete frequencies. For N symbols in a serial link, we collect a total of N samples at a rate of $F_S = 1/T_M$, or $L = 1$ sample/symbol, thus spanning a total time duration of NT_M seconds. This description is just to enforce the concept of 1 sample per data symbol which we now change to NT_S seconds below.

In single-carrier systems, we have used the notation $F_S = 1/T_S$ as the sample rate and $R_M = 1/T_M$ as the symbol rate. The relation between them is determined by the number of samples/symbol L .

$$F_S = L/T_M, \quad T_M = LT_S$$

A signal oversampled by L is required in single-carrier systems for Tx and Rx filtering, an example of which is pulse shaping at the Tx and matched filtering at the Rx. Such an oversampled signal is also employed (but not necessary) to perform various tasks such as symbol timing synchronization and fractionally-spaced equalization. In the above description, we saw that the target is to convert a high rate serial symbol stream into a bank of N low rate parallel symbol streams. Since this represents $L = 1$ sample per symbol and most of the blocks (such as synchronization and equalization) in an OFDM Rx design do not require oversampling, we switch to the conventional $F_S = 1/T_S$ notation to denote the sample rate and sample time, respectively.

Now operating on N samples, consider that the lowest frequency that can be represented by these N samples is the one by a sinusoid that completes one full cycle – and no more – during this interval of NT_S seconds. Being an inverse of time period, such a frequency is given by $1/(NT_S)$ Hz and is known as the fundamental frequency F_1 .

$$\begin{aligned} I &\rightarrow V_I(t) = \cos 2\pi \frac{1}{NT_S} t = \cos 2\pi F_1 t \\ Q &\uparrow V_Q(t) = \sin 2\pi \frac{1}{NT_S} t = \sin 2\pi F_1 t \end{aligned}$$

Let us call the expression NT_S that represents one block of N samples as T_u where u stands for the useful part of the symbol. We will shortly see what useful means in this

context.

$$T_u = NT_S \quad (9.1)$$

which implies that $F_1 = 1/T_u$. Now we consider N integer multiples F_k of this fundamental frequency F_1 .

$$F_k = kF_1 = \frac{k}{NT_S} = \frac{k}{T_u} \quad (9.2)$$

which are all orthogonal to each other in T_u seconds. To see why, let us sample these complex sinusoids at a rate of $F_S = 1/T_S$.

$$\begin{aligned} I &\rightarrow V_I[n] = \cos 2\pi \frac{1}{NT_S} t \Big|_{t=nT_S} = \cos 2\pi \frac{1}{NT_S} nT_S = \cos 2\pi \frac{1}{N} n \\ Q &\uparrow V_Q[n] = \sin 2\pi \frac{1}{NT_S} t \Big|_{t=nT_S} = \sin 2\pi \frac{1}{NT_S} nT_S = \sin 2\pi \frac{1}{N} n \end{aligned}$$

If the integer multiples k/N of the fundamental frequency $1/N$ above are constructed for

$$k = -\frac{N}{2}, -\frac{N}{2} + 1, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1$$

then this is the same set we encountered before in Section 1.7. There, we defined orthogonality in Eq (1.47) reproduced below.

$$\begin{aligned} I &\rightarrow \sum_{n=0}^{N-1} \cos 2\pi \frac{k-k'}{N} n = \begin{cases} N & k = k' \\ 0 & k \neq k' \end{cases} \\ Q &\uparrow \sum_{n=0}^{N-1} \sin 2\pi \frac{k-k'}{N} n = 0 \end{aligned} \quad (9.3)$$

and this concept was illustrated in Figure 1.46. In words, all complex sinusoids having frequencies k/N are orthogonal to each other in a duration of $T_u = NT_S$ seconds. By orthogonality, we imply that over a duration of NT_S seconds (one period of the sinusoid with fundamental frequency), a sum of sample-by-sample products of any two of these sinusoids is zero which is the same as their correlation at lag 0.

There we have a set of N parallel wires or waves that do not interfere with each other when correlated over a duration of N samples. *This set of complex sinusoids acts as our invisible printer port in the air!* Next, we move towards a more visual approach of these parallel waves.

Enter the Correlation (Real Sinusoids)

Although we can directly go to the case of complex sinusoids due to their coverage in the previous chapters, we first consider the inphase components only. This is because it allows us to draw the modulated OFDM signal in the form of these sinusoids in Figure 9.6 later in a simple manner.

Some real sinusoids for $N = 6$ are illustrated in Figure 9.5 where $k = 0, 1, \dots, N-1$. To verify their orthogonality, take for example $k = 0$ and $k = 1$. The sinusoid

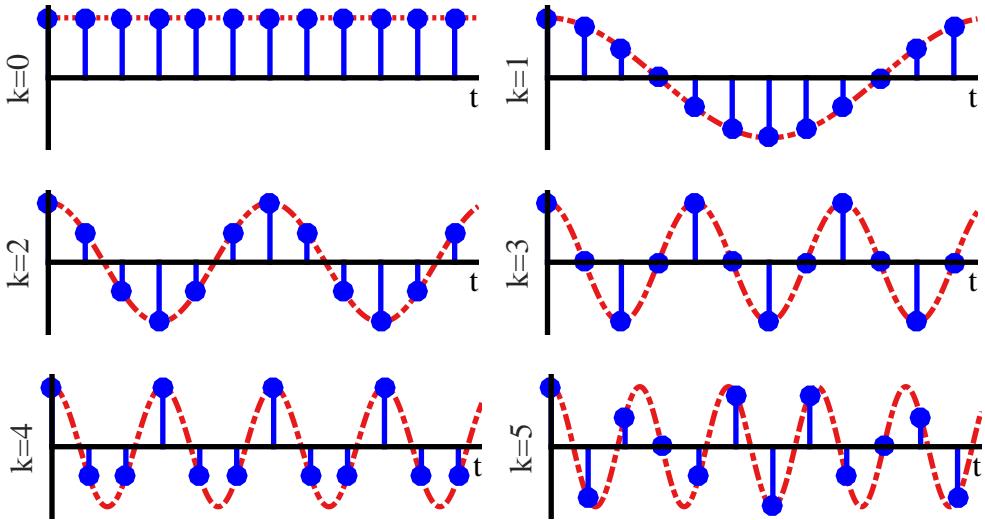


Figure 9.5: Real sinusoids act as a set of parallel waves in the air

with $k = 0$ are all ones, so its sample-by-sample product with $k = 1$ sinusoid yields the same sinusoid back. So for $k = 1$, notice that the sum of all samples are zero.

You can repeat this computation for $k \neq 0$ by simulating in software, the sum of sample-by-sample product between any two sinusoids is zero as long as their frequencies are a multiple of a fundamental frequency, $k = 1$ in this example.

Consequently, as the data symbols $a[k]^+$ are produced, we modulate each such sinusoid with one data symbol. For a real sinusoid with frequency k/N ,

$$a[k] \cdot \cos 2\pi \frac{k}{N} n, \quad k = 0, 1, \dots, N-1$$

Finally, all such waveforms are added together to form the cumulative signal as drawn in Figure 9.6. Notice that the symbol stream is the same as the first set of $N = 6$ symbols from Figure 9.3a.

Owing to orthogonality, at the Rx side, a correlation of the above signal is performed with each such sinusoid of frequency k'/N . We know that

$$\sum_{n=0}^{N-1} \left\{ a[k] \cos 2\pi \frac{k}{N} n \right\} \cdot \cos 2\pi \frac{k'}{N} n = \begin{cases} a[k] & k = k' \\ 0 & k \neq k' \end{cases}$$

where a constant scaling factor of 2 has been ignored. This can be proved as follows.

- For $k \neq k'$, the orthogonality condition in Eq (9.3) simply implies the result to be zero. This was proved for a real sinusoid in the discussion after Eq (1.46).
- For $k = k'$, use the identity $\cos^2 \theta = 0.5(1 + \cos 2\theta)$. The first term is a constant and produces $a[k]$ at the output, while the second term is a cosine whose summation over a whole period (or a set of periods) for $n = 0$ to $N - 1$ is zero.

[†]We will shortly see why the data symbols are denoted as $a[k]$ here instead of $a[m]$ as before in the text. This is due to the presence of the iDFT next, the primary task of which is considered as taking a signal from frequency domain into time domain. For this reason, the symbols are assumed to be produced in frequency domain.

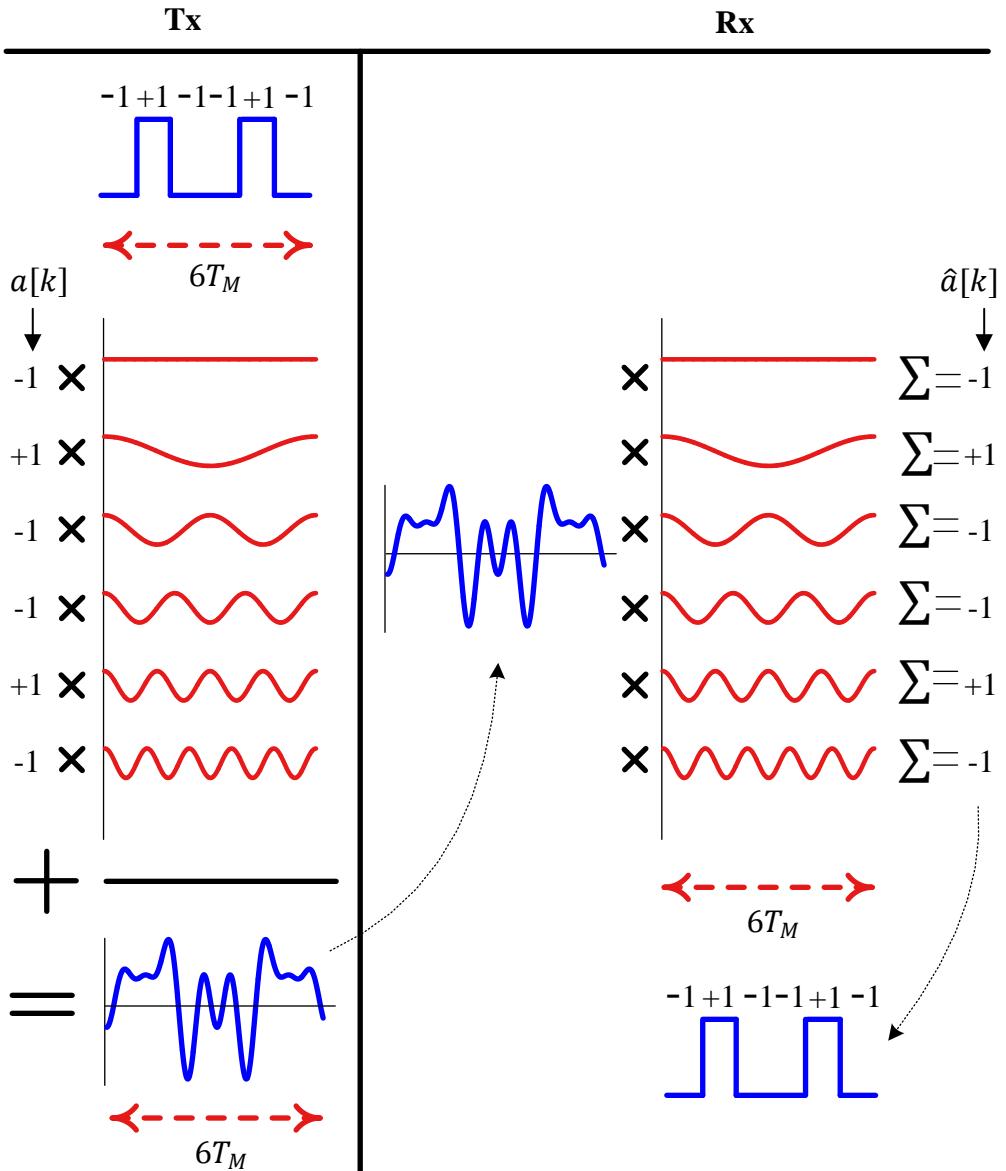


Figure 9.6: Data symbols modulated onto orthogonal sinusoids and added together to form the Tx signal. At the Rx, it is correlated with orthogonal sinusoids to generate symbol decisions

Consequently, if each such sinusoid of frequency k/N was modulated with a different data symbol $a[k]$ and added to all the others, each data symbol would have come out at the Rx side undistorted from the presence of other $N - 1$ symbols in parallel. This is illustrated in Figure 9.6 on the Rx side and can be verified as follows.

Let us form a cumulative signal by adding all N symbols modulated with their

respective sinusoids.

$$\begin{aligned} v[n] &= a[0] \cos 2\pi \frac{0}{N} n + a[1] \cos 2\pi \frac{1}{N} n + \cdots + a[N-1] \cos 2\pi \frac{N-1}{N} n \\ &= \sum_{k=0}^{N-1} a[k] \cos 2\pi \frac{k}{N} n \end{aligned}$$

A correlation with $\cos 2\pi(k'/N)n$ at the Rx would yield

$$\sum_{n=0}^{N-1} \left\{ \sum_{k=0}^{N-1} a[k] \cos 2\pi \frac{k}{N} n \right\} \cdot \cos 2\pi \frac{k'}{N} n = a[k]$$

where a factor of 2 has been ignored again. This can be seen by taking an example of $k' = 1$.

$$\begin{aligned} &\sum_{n=0}^{N-1} \left\{ a[0] \cos 2\pi \frac{0}{N} n \right\} \cdot \cos 2\pi \frac{1}{N} n + \sum_{n=0}^{N-1} \left\{ a[1] \cos 2\pi \frac{1}{N} n \right\} \cdot \cos 2\pi \frac{1}{N} n + \\ &\quad \cdots + \sum_{n=0}^{N-1} \left\{ a[N-1] \cos 2\pi \frac{N-1}{N} n \right\} \cdot \cos 2\pi \frac{1}{N} n \\ &= 0 + a[1] + \cdots + 0 = a[1] \end{aligned}$$

This process is drawn in Figure 9.6 on the right hand side, where the cumulative waveform is multiplied with each orthogonal sinusoid generating the estimate for each data symbol $\hat{a}[k]$. Since these sinusoids carry the data symbols $a[k]$ in an independent fashion, these are known as **subcarriers**. A subcarrier is the word you would hear a lot during any discussion on OFDM.

Enter the Correlation (Complex Sinusoids/Subcarriers)

Having clarified the simple case of real sinusoids, we now focus on the realistic scenario of complex sinusoids known as subcarriers in the context of OFDM. A set of subcarriers for $k = -N/2, \dots, N/2 - 1$ is drawn in a 3D plane in Figure 9.7. Since some people are more comfortable with 2D figures, a set of subcarriers in terms of I and Q components is also illustrated in Figure 9.8 for $N = 4$.

For a subcarrier with frequency k/N and a modulation scheme with no Q component (such as BPSK), we can write

$$\begin{array}{ll} I & \rightarrow a[k] \cos 2\pi \frac{k}{N} n \\ Q & \uparrow a[k] \sin 2\pi \frac{k}{N} n \end{array}$$

More generally, for a QAM scheme with $a_I[k]$ and $a_Q[k]$ as I and Q parts, respectively, we can multiply them with a subcarrier with frequency k/N as follows.

$$\begin{array}{ll} I & \rightarrow a_I[k] \cos 2\pi \frac{k}{N} n - a_Q[k] \sin 2\pi \frac{k}{N} n \\ Q & \uparrow a_Q[k] \cos 2\pi \frac{k}{N} n + a_I[k] \sin 2\pi \frac{k}{N} n \end{array}$$

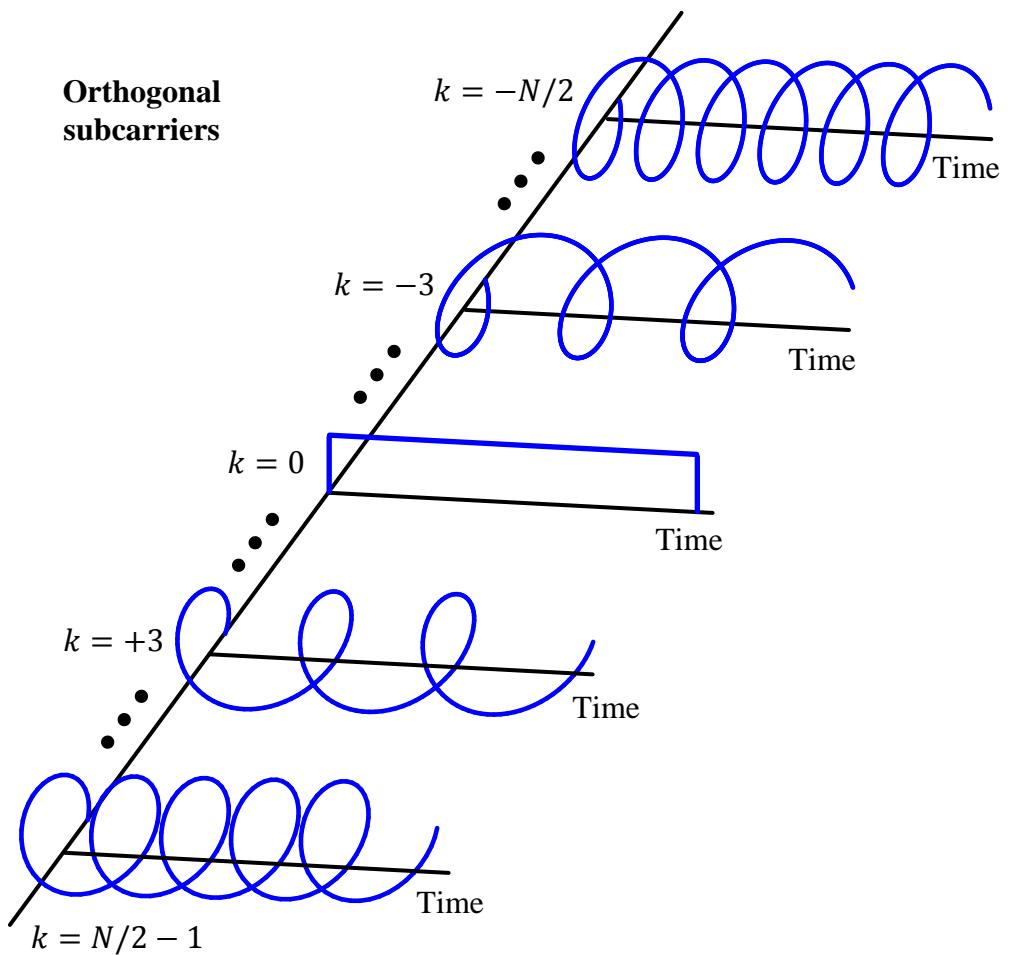


Figure 9.7: Complex subcarriers (which are nothing but complex sinusoids) act as a set of parallel waves in the air. Notice that these are the same sinusoids used in the computation of the DFT

With the concept of subcarriers in place, we can clearly see how this modulation process can be very efficiently implemented: comparing the above expression with the definition of an iDFT in Eq (1.54), *the subcarriers themselves are the DFT sinusoids and the data symbol modulation process is exactly taking the iDFT of the symbol*

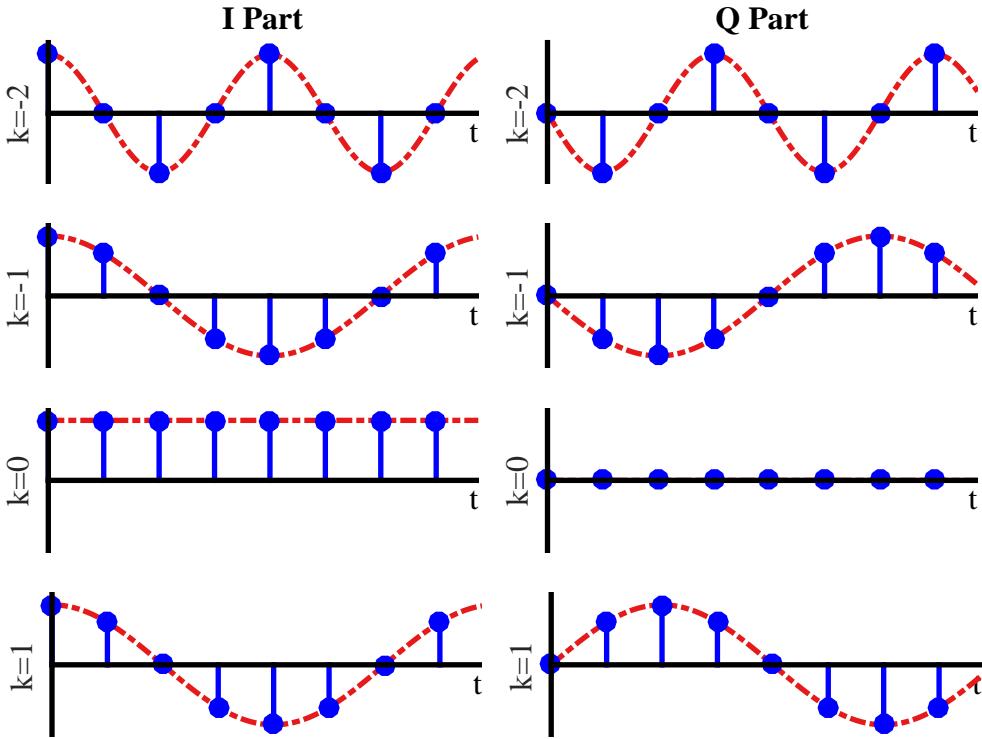


Figure 9.8: *I* and *Q* components of the subcarriers (which are nothing but complex sinusoids) that act as a set of parallel waves in the air. Notice that these are the same sinusoids used in the computation of the DFT

sequence $a[k]$. Consequently, OFDM can be implemented through iDFT as[†]

$$\begin{aligned}
 I \rightarrow & v_I[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[a_I[k] \cos 2\pi \frac{k}{N} n - a_Q[k] \sin 2\pi \frac{k}{N} n \right] \\
 Q \uparrow & v_Q[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[a_Q[k] \cos 2\pi \frac{k}{N} n + a_I[k] \sin 2\pi \frac{k}{N} n \right]
 \end{aligned} \tag{9.4}$$

Comparing it with the definition of iDFT which is used to transform a signal from frequency domain to time domain, we can see why the terminology ‘frequency domain symbols’ is used for data symbols denoted by $a[k]$. This has nothing to do with some frequency domain property of data symbols but simply because the Tx signal is naturally considered to be in time domain and any signal before the iDFT block is then considered to be present in frequency domain.

[†]In other resources, you will often see the equivalent expression below.

$$v[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} a[k] \cdot \exp \left(j2\pi \frac{k}{N} n \right)$$

Logically, the reverse operation, i.e., a DFT, is performed at the Rx side to transform the Rx signal into frequency domain for both equalization and detection purpose. The data symbol estimates are given as

$$\begin{aligned} I &\rightarrow \hat{a}_I[k] = \sum_{n=0}^{N-1} \left[v_I[n] \cos 2\pi \frac{k}{N} n + v_Q[n] \sin 2\pi \frac{k}{N} n \right] \\ Q &\uparrow \hat{a}_Q[k] = \sum_{n=0}^{N-1} \left[v_Q[n] \cos 2\pi \frac{k}{N} n - v_I[n] \sin 2\pi \frac{k}{N} n \right] \end{aligned} \quad (9.5)$$

Although a mathematical proof can be constructed by utilizing the subcarriers orthogonality, suffice it to say that the iDFT operation and DFT operation are inverse of each other and hence the data symbols appearing in Eq (9.4) can be perfectly recovered by applying Eq (9.5).

Needless to say, both the iDFT and the DFT are implemented through the iFFT (Inverse Fast Fourier Transform) and FFT (Fast Fourier Transform) procedures, respectively, which is not a separate transform but a very efficient algorithm to compute the iDFT and DFT, respectively. From here onwards, we will refer to going to and coming back from frequency domain through FFT operations interchangeably for the DFT.

Before performing such an FFT operation, however, we need to ensure that the convolution between the Tx signal and the wireless channel happens in a circular fashion instead of a linear one. This is what we explain next.

Inserting the Cyclic Prefix (CP)

An actual OFDM symbol is completed after inserting the Cyclic Prefix (CP) and consists of

$$N_{\text{OFDM}} = N + N_{\text{CP}}$$

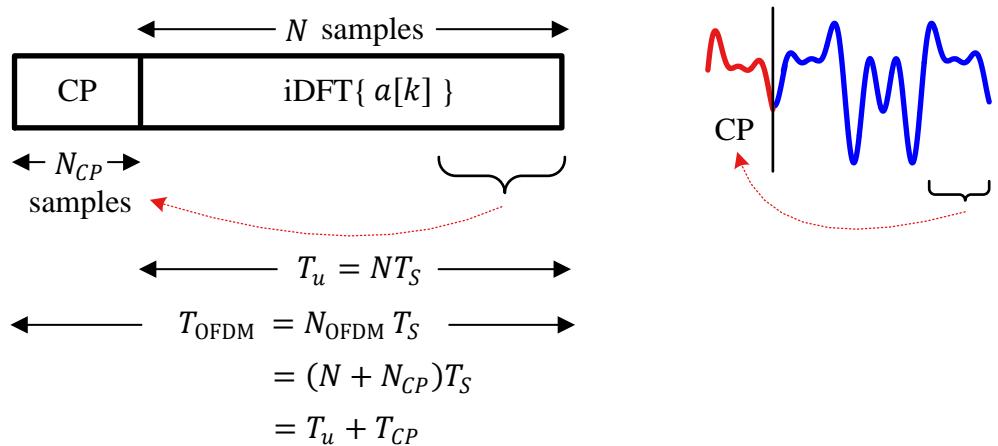


Figure 9.9: A Cyclic Prefix (CP) in the context of an OFDM symbol

samples instead of N samples. We will now see how useful part $T_u = NT_S$ implies the actual iFFT of $a[k]$ while a Cyclic Prefix (CP) is just an added redundancy.

In the discussion on Single-Carrier Frequency Domain Equalization (SC-FDE) in Section 8.3.8, we found that a Cyclic Prefix (CP) consists of the last N_{CP} samples of the Tx sequence (where N_{CP} is determined by the maximum expected channel length). For an OFDM symbol of length N , the process to insert the CP is shown in Figure 9.9. The time duration of the complete OFDM symbol is therefore defined as

$$T_{\text{OFDM}} = N_{\text{OFDM}} T_S = (N + N_{CP}) T_S = T_u + T_{CP}$$

where

$$T_{CP} = N_{CP} T_S$$

It is important to note that as opposed to the last N_{CP} symbols prefixed back in an SC-FDE system, the last $N_{CP} = T_{CP}/T_S$ samples of $\text{iDFT}\{a[k]\}$ are prepended at the start of the OFDM symbol. The job of the CP is threefold in an OFDM system.

Inter-Symbol Interference (ISI): To provide a guard interval for the channel such that no interference from the previous OFDM symbol enters the next OFDM symbol. This happens when the length of the CP is chosen as the maximum expected channel length. This was explained in detail in the context of single-carrier frequency domain equalization in Section 8.3.8 and I recommend revisiting that section to understand this point.

Converting linear convolution into circular convolution: A CP converts linear convolution between the Tx signal and the wireless channel into circular convolution.

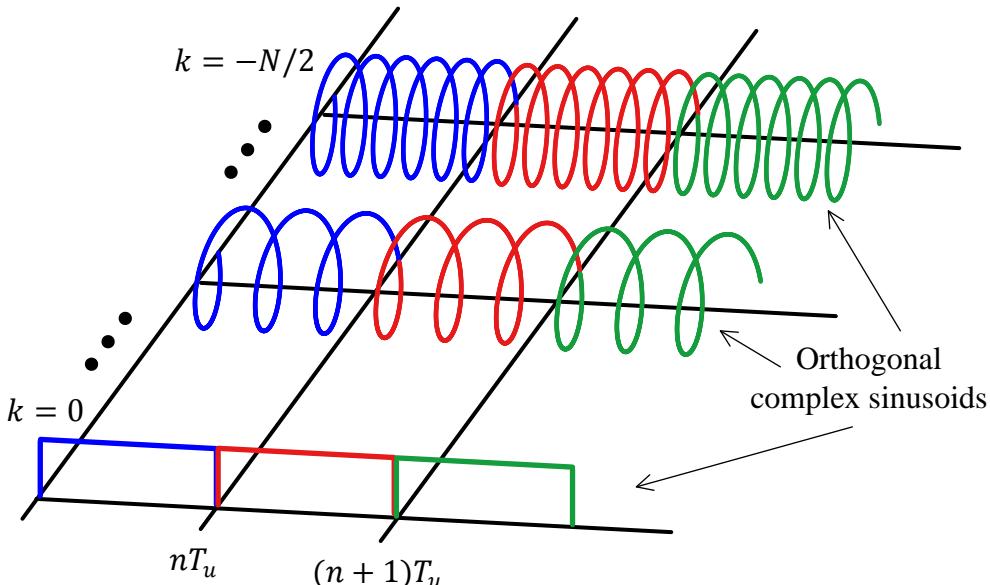


Figure 9.10: Orthogonal subcarriers (complex sinusoids) in an OFDM symbol without a Cyclic Prefix (CP) or guard interval

This circular convolution enables the discrete frequency domain product between their respective transforms. Again, this was explained in significant detail in the context of single-carrier frequency domain equalization in Section 8.3.8.

Inter-Carrier Interference (ICI): In my opinion, a detailed mathematical reasoning on the prevention of ICI through a CP is not necessary and probably too cumbersome for a non-mathematician. For the understanding of the basic mechanism, I just demonstrate this concept with the help of a few figures.

Figure 9.10 depicts components of an OFDM symbol without any CP or guard interval, where *only the negative subcarriers are drawn* to keep the figure size small. The negative sign in the frequency of the last sinusoid, $-N/2$, appears due to its clockwise direction of rotation. These components are the subcarriers that are orthogonal to each other (earlier drawn in Figure 9.8 with I and Q parts separated) and scaled by data symbol $a[k]$ (all assumed +1 here).

The actual OFDM signal is formed through summing all the corresponding samples in these subcarriers. Notice that since the subcarriers are completing their full number of cycles within each OFDM symbol, they are all orthogonal to each other within that OFDM symbol and neither ISI nor ICI occurs.

Next, Figure 9.11 draws the same OFDM signal without any CP but after it has passed through a wireless channel. In this figure, only $k = 0$ subcarrier is shown for all three intervals while the rest of the subcarriers are drawn for OFDM symbol 1 only for clarity. It is straightforward to identify the ISI due to the intruding of one OFDM symbol into the next. The issue of ICI is not as clear here.

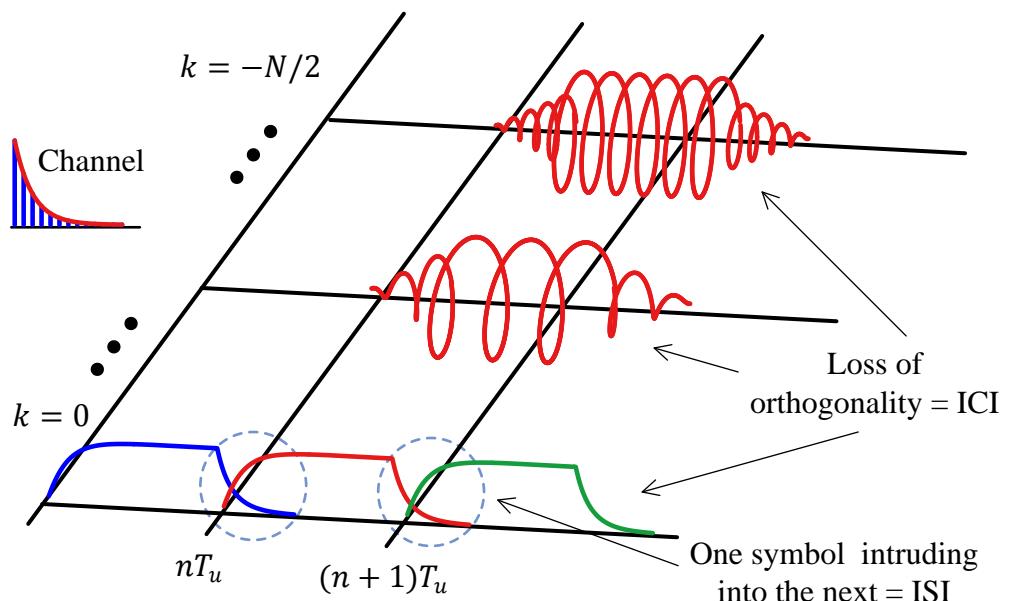


Figure 9.11: After passing through the wireless channel, Inter-Carrier Interference (ICI) arises due to the loss of orthogonality among subcarriers and Inter-Symbol Interference (ISI) arises due to one OFDM symbol intruding into the next

For this purpose, we recall Section 1.10.1 here in which Figure 1.64 and Figure 1.65 hold particular importance. In these figures, we discovered that signals with frequencies other than the set k/N are not periodic in the observation window and are suddenly terminated without completing the final cycle in full. This discontinuity violates the orthogonality relation yielding a non-zero result and becomes responsible for spectral contributions over the entire set of frequencies k/N . Due to the convolution with the wireless channel, a loss of exact complete periods within an OFDM symbol is evident in our current Figure 9.11 which gives rise to ICI among subcarriers. *The data symbols $a[k]$ are not travelling on independent parallel waves anymore!*

On a side note, this ICI was not an issue in single-carrier frequency domain equalization systems because there are no independent subcarriers related to the data symbols $a[k]$. There are subchannels though, which exhibit the impact of each data symbol inherently spread over all of them.

When a CP is inserted with a length greater than the maximum expected channel length, the effect of ISI remains limited to the CP duration. This is drawn in Figure 9.12a. The CP also acts as a buffer against introducing ICI in the system because the loss of orthogonality only happens within the CP duration. When it is removed, it not only helps in getting rid of the ISI in the system but the remaining parts of the subcarriers exhibit a completion of rotation within the useful period T_u , thus restoring the orthogonality among the subcarriers. The OFDM symbol is ICI-free and the data symbols $a[k]$ can then be independently detected. This is illustrated in Figure 9.12b.

Later, we will study that ICI is also caused by a Carrier Frequency Offset (CFO) and the methods to prevent it in that context.

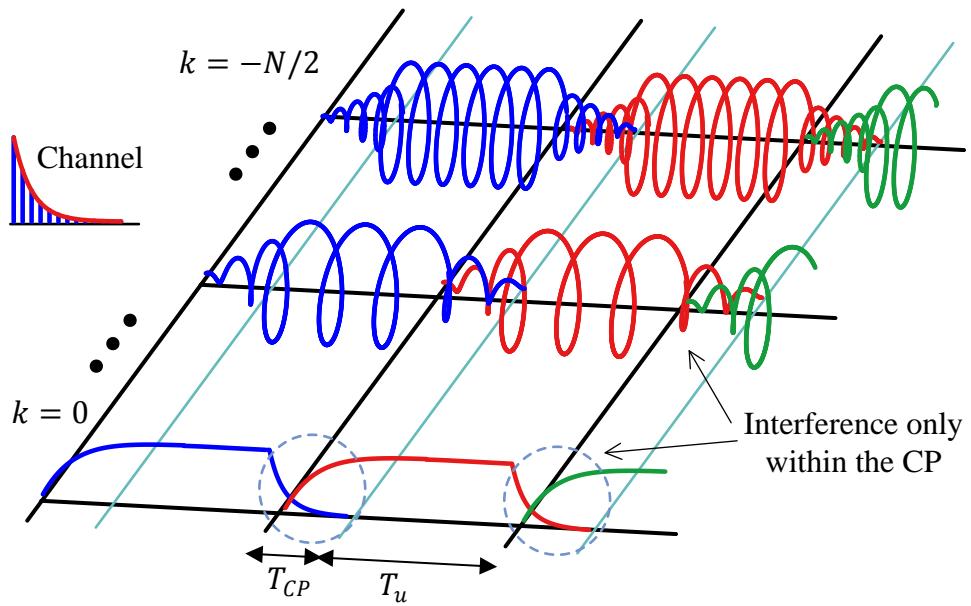
Now we discuss why the equalization becomes simpler in an OFDM system as compared to a single-carrier system.

Why Equalization Becomes Simpler

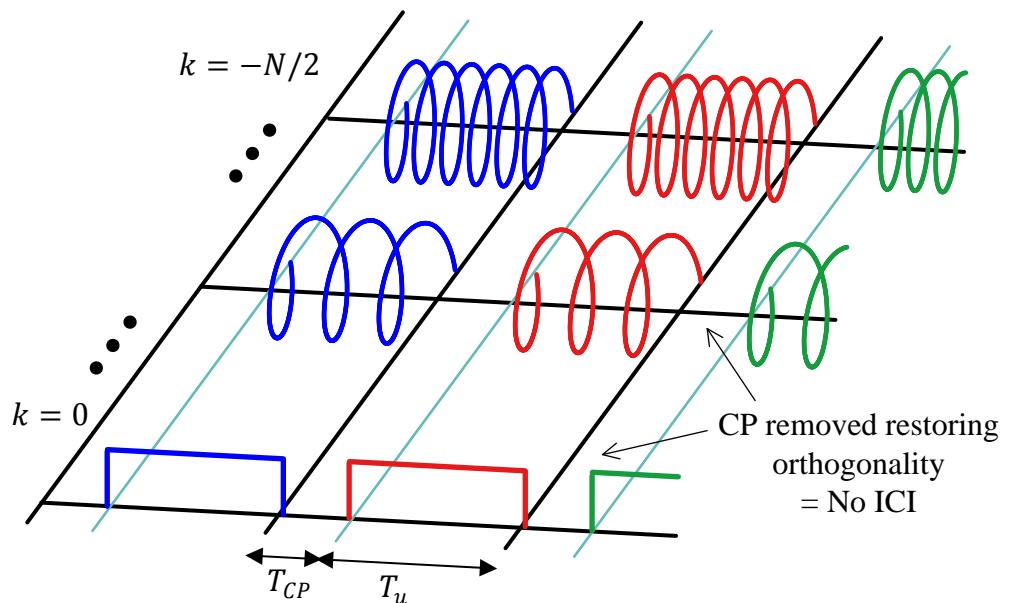
Complexity of equalization has been the bottleneck in successful implementation of high rate single-carrier systems. The introduction of subcarriers in OFDM simplifies this process to a considerable level and has been the major factor in its adoption in high speed wireless systems. Although this process of simple equalization is better explained in frequency domain later, we still have a look at the channel impact in time domain to get an idea.

For this purpose, we continue with $N = 6$ subcarriers shown earlier in Figure 9.6 as an example and see what happens in the presence of a multipath channel. This is drawn in Figure 9.13 where two observations are important.

1. We have explained before that as long as the delay spread of the channel remains less than 10 or so percent of the symbol duration, the channel is frequency flat fading and hence multiplies the signal with a single coefficient, instead of a complete convolution as in the case of a frequency selective fading channel. Now the channel example shown here is a frequency selective channel for the serial symbol stream because the multipath duration is equal or greater than a symbol time T_M . However, due to the elongated symbol length, i.e., NT_M here, it has been converted into a frequency flat fading channel *for each subcarrier*.



(a) ISI is limited to the CP interval



(b) With CP removed, the subcarriers are orthogonal and hence no ICI occurs

Figure 9.12: Effect of CP on ISI and ICI

- With the addition of a CP, the orthogonality of each subcarrier is ensured and the channel impacts each subcarrier in an individual manner. Notice in Figure 9.13 how each subcarrier will be deteriorated by the channel but discarding the CP

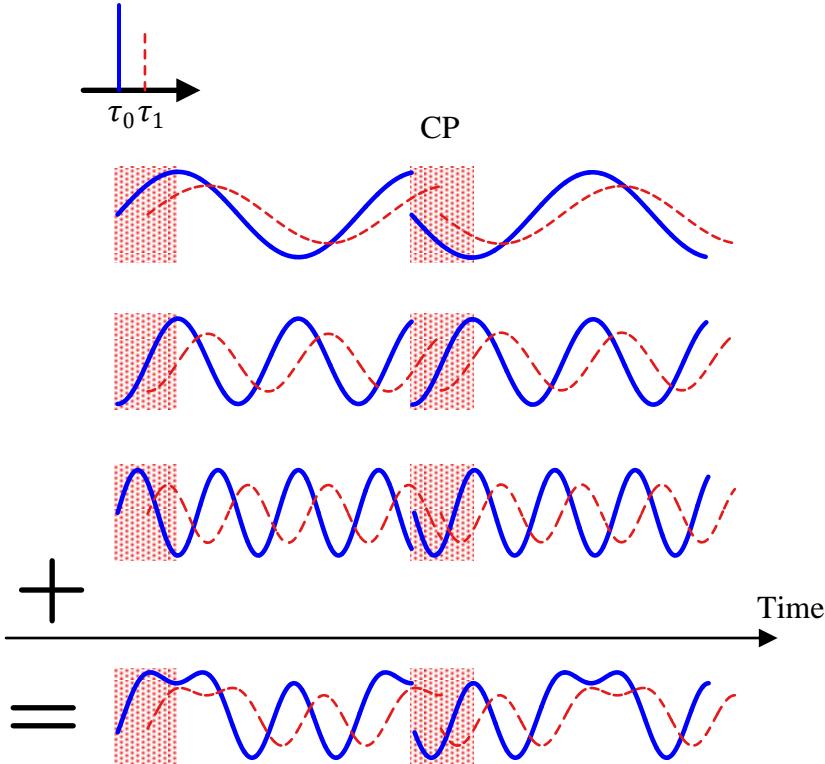


Figure 9.13: In the presence of a CP, multipath in time domain impact each subcarrier individually due to their orthogonality. Moreover, an elongated symbol implies flat fading for each individual subcarrier

gets rid of both ISI from one OFDM symbol to the next as well as ICI among difference subcarriers. Then, for each subcarrier, the channel is just a summation of complex sinusoids *of the same frequency with different amplitudes and delays* which all add up to generate a resultant amplitude and phase without changing the frequency. Let us derive this resultant channel for a subcarrier k .

Referring to Figure 9.13, assume that the delay of the direct path $\tau_0 = 0$ and its amplitude $\rho_0 = 1$. With a single multipath with delay τ_1 (which is $n_1 = \tau_1/T_S$ in terms of samples) and amplitude ρ_1 , the I output can be expressed as below. Recalling that the subcarriers are complex sinusoids, *the actual amplitudes γ_i are complex* and given as a function of carrier frequency F_C in Eq (8.5). Nevertheless, for removing clutter to simplify the derivation, here we take them as real and equal to ρ_i .

$$\begin{aligned}
 & \cos 2\pi \frac{k}{N} n + \rho_1 \cos 2\pi \frac{k}{N} (n - n_1) = \cos 2\pi \frac{k}{N} n + \\
 I \rightarrow & \quad \rho_1 \cos 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n_1 + \rho_1 \sin 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n_1 \\
 & = \left(1 + \rho_1 \cos 2\pi \frac{k}{N} n_1 \right) \cos 2\pi \frac{k}{N} n + \left(\rho_1 \sin 2\pi \frac{k}{N} n_1 \right) \sin 2\pi \frac{k}{N} n
 \end{aligned}$$

where we have used the identity $\cos(A - B) = \cos A \cos B + \sin A \sin B$. On a similar note,

$$\begin{aligned} & \sin 2\pi \frac{k}{N} n + \rho_1 \sin 2\pi \frac{k}{N} (n - n_1) = \sin 2\pi \frac{k}{N} n + \\ Q \uparrow & \quad \rho_1 \sin 2\pi \frac{k}{N} n \cdot \cos 2\pi \frac{k}{N} n_1 - \rho_1 \cos 2\pi \frac{k}{N} n \cdot \sin 2\pi \frac{k}{N} n_1 \\ & = -\left(\rho_1 \sin 2\pi \frac{k}{N} n_1\right) \cos 2\pi \frac{k}{N} n + \left(1 + \rho_1 \cos 2\pi \frac{k}{N} n_1\right) \sin 2\pi \frac{k}{N} n \end{aligned}$$

where $\sin(A - B) = \sin A \cos B - \cos A \sin B$. From the multiplication rule of complex signals $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$, we can see that the subcarrier k is being multiplied with a *single channel coefficient* given by

$$\begin{aligned} I \rightarrow & H_I[k] = 1 + \rho_1 \cos 2\pi \frac{k}{N} n_1 \\ Q \uparrow & H_Q[k] = -\rho_1 \sin 2\pi \frac{k}{N} n_1 \end{aligned} \tag{9.6}$$

For each subcarrier, even when the number of multipath copies increases, the number of terms in the above expression increases but the frequency of their sum signal remains the same (equal to k/N). Subsequently, the important point to remember is that each such sum, for each subcarrier, remains orthogonal to all the other subcarriers $k' \neq k$.

Finally, we need to change the following in the above equation.

- We can write using the useful symbol duration $T_u = NT_S$ from Eq (9.1),

$$2\pi \frac{k}{N} n_1 = 2\pi \frac{k}{N} \frac{\tau_1}{T_S} = 2\pi k \frac{\tau_1}{T_u}$$

- When the actual *complex gain* γ_1 is taken into account, the above expression includes complex multiplication of γ_1 with the complex sinusoid of frequency $-2\pi k \tau_1 / T_u$, see Eq (9.6).
- The actual channel response consists of N_{MP} paths that are summed at the Rx antenna.

Taking all this into account, the channel coefficient becomes

$$\begin{aligned} I \rightarrow & H_I[k] = \sum_{i=0}^{N_{MP}-1} \gamma_{i,I} \cos 2\pi k \frac{\tau_i}{T_u} + \gamma_{i,Q} \sin 2\pi k \frac{\tau_i}{T_u} \\ Q \uparrow & H_Q[k] = \sum_{i=0}^{N_{MP}-1} \gamma_{i,Q} \cos 2\pi k \frac{\tau_i}{T_u} - \gamma_{i,I} \sin 2\pi k \frac{\tau_i}{T_u} \end{aligned} \tag{9.7}$$

which is the very well known single tap channel coefficient for an OFDM system exhibiting flat fading for each subcarrier[†].

[†]In complex notation, these channel gains are given by

$$H[k] = \sum_{i=0}^{N_{MP}-1} \gamma_i \cdot \exp(-j2\pi k \tau_i / T_u)$$

While we will have a better look at this process in frequency domain, the orthogonal subcarriers each experiencing a flat fading channel is the main reason behind simpler equalization required for an OFDM system.

In summary, OFDM increases the symbol duration from T_M to $T_u = NT_M$ such that the information is transmitted in *many low rate parallel streams*. The duration of each low rate stream T_u is significantly longer than the delay spread T_{Del} of the channel and hence no ISI occurs, thanks to the guard interval.

$$T_u \gg T_{\text{CP}} > T_{\text{Del}} \quad (9.8)$$

This eases the equalization part of the Rx processing when we use our Master Algorithm, i.e., correlation, at the Rx to separate the subcarriers.

Having viewed the concept of OFDM in time domain, we look at its frequency domain interpretation.

9.2.2 To a Sliced Bread

We start with the problem high rate single-carrier systems face in frequency domain and then explain how OFDM addresses that issue.

Increasing the Symbol Rate

The wireless channel from Chapter 8 has an impulse response derived from the contribution of each multipath component. Its frequency response was also drawn in several figures throughout that chapter, two of which are redrawn here in Figure 9.14a for a frequency flat fading channel and in Figure 9.14b for a frequency selective fading channel. Here, $C(F)$ and $S(F)$ are the frequency domain representations of the channel and Tx signal, respectively.

With respect to frequency domain, the signal at the Rx is a product of the spectra of the Tx signal and the wireless channel. The low data rate signal in Figure 9.14a needs less manipulation by the Rx to get the original data back. Essentially for this kind of signal, the channel acts just as a single multiplier that can be equalized through estimating that channel coefficient and dividing the Rx signal by the estimated value.

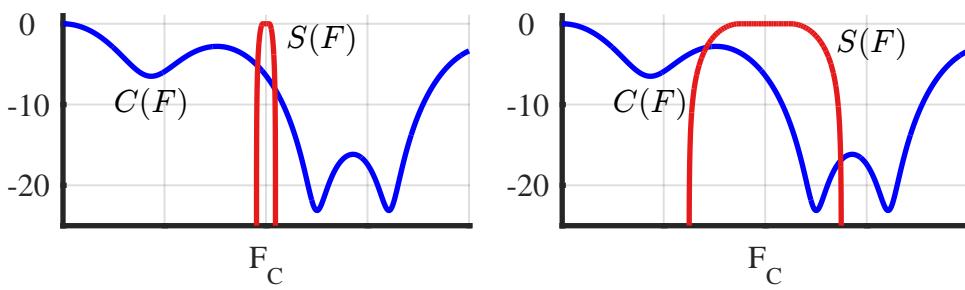


Figure 9.14: Our symbol rate determines the signal bandwidth and consequently whether the channel behaves as a frequency flat or selective fading channel

Consequently, the equalization process reduces to a single division operation[†].

Coming to Figure 9.14b, it is evident that due to the high data rate and hence large signal bandwidth, the whole bandwidth extends beyond the coherence bandwidth of the channel and hence the Rx magnitude is distributed across a wide spectral region. As the term selectivity implies, when some part of the signal spectrum is in a deep fade, the rest of the portion is not. This discriminatory treatment of different portions of the signal spectrum by the channel is a form of frequency diversity and hence a good thing if properly equalized.

However, this high data rate signal needs a lot of Rx processing to equalize the channel due to its large bandwidth. Such a computationally complex equalizer is incompatible with the high rate symbol stream as discussed in detail in Section 8.3.7 in the context of the drawbacks of a time domain equalizer.

Subcarriers/Complex Sinusoids in Frequency Domain

To view this process in frequency domain, we need to know how the Fourier Transform of a complex sinusoid looks like.

- If we had infinitely long subcarriers, then their Fourier Transform would just have been an impulse at that frequency. This is shown at the top of Figure 9.15. Recalling the definition of frequency as cycles/second, the frequency of this impulse is $3/T_u$ because there are three complete cycles of the complex sinusoid in our measurement time T_u .
- However, the subcarriers are only T_u wide in time which is the same as multiplying them with a rectangular signal of length T_u . A rectangular signal has a sinc signal as its Fourier Transform shown in the middle of Figure 9.15.
- Finally, a multiplication in time induces a convolution in frequency. For this reason, the sinc signal in frequency gets shifted to the frequency of the subcarrier, $3/T_u$ in our example.

Now we want to inspect all the subcarriers together in frequency domain. For this purpose, the following two questions need to be answered.

Spacing between any two subcarriers: First, remember from Eq (9.1) that the duration of each subcarrier, or complex sinusoid, is

$$T_u = NT_S \quad (9.9)$$

and there are N such subcarriers forming the overall signal. The indices of these subcarriers range from $k = -N/2$ to $k = N/2 - 1$. From this information, we deduced the actual frequency F_k of each subcarrier using $k/N = F/F_S$ as

$$F_k = F_S \cdot \frac{k}{N} = \frac{k}{NT_S} = \frac{k}{T_u}$$

[†]There can be a question of how to recover when this signal encounters a deep channel fade. In that case, nothing but diversity (a signal replica in some form whether in time, frequency, space, etc.) can recover the signal.

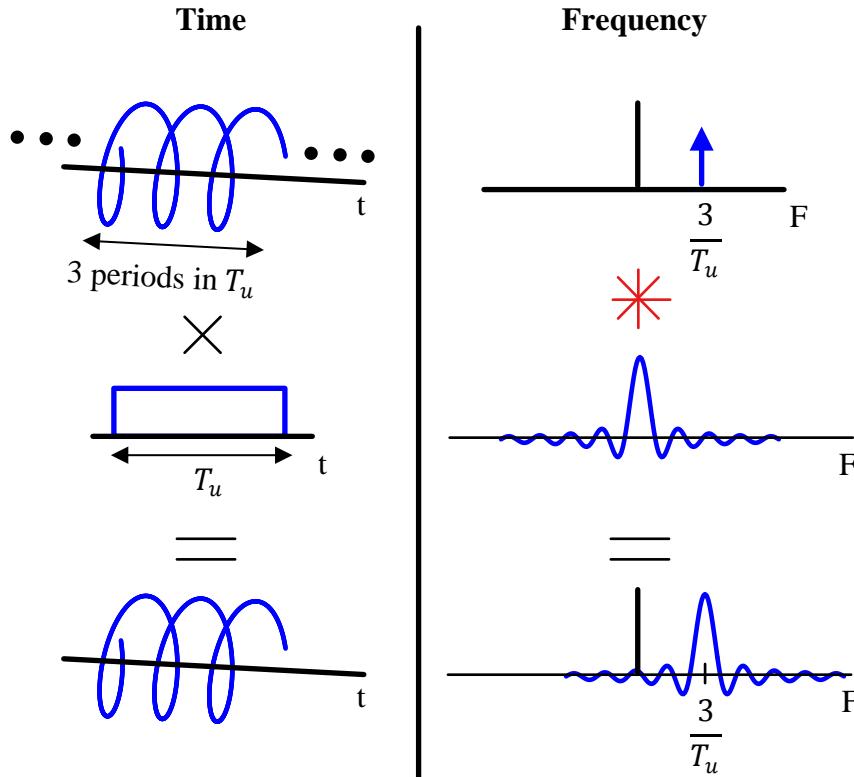


Figure 9.15: Limiting the complex sinusoid or subcarrier in time domain creates its spectrum as a sinc signal in frequency domain

So we have the maximum subcarrier frequency as $-N/(2T_u)$. From $k = 0$ and $k = 1$, we can find the spacing Δ_F between any two subcarriers.

$$\Delta_F = \frac{1}{T_u} - \frac{0}{T_u} = \frac{1}{T_u} \quad (9.10)$$

From Eq (9.9) and Eq (9.10), we have the total bandwidth $B = 1/T_S$ as

$$\frac{1}{\Delta_F} = T_u = NT_S = \frac{N}{B} \rightarrow \text{or } B = N\Delta_F$$

Note 9.2 Bandwidth utilization

By using $T_u = NT_S$, we have derived the subcarrier frequencies as $F_k = k/T_u = k/NT_S$. Here, we have the maximum subcarrier frequency as $-N/(2NT_S) = -1/(2T_S) = -F_S/2$ which is the maximum frequency allowed by the Nyquist's sampling theorem to avoid aliasing. This points to the fact that OFDM efficiently utilizes the whole available bandwidth, although the usage of null subcarriers at

the edges reduces this efficiency. Those null subcarriers are required to give enough transition bandwidth to subsequent simple digital and analog filters thus avoiding costly filters.

Other factors reducing the bandwidth efficiency are the Cyclic Prefix (CP), pilot subcarriers (subcarriers modulated with known data for tracking the channel) and training sequences for initial acquisition.

Location of sinc signal's first zero crossing: To look at the zero crossing of a subcarrier in frequency domain, recall Section 1.10.1 on DFT of a rectangular signal. There, we found in Eq (1.69) that for a full length rectangle (i.e., N samples), the discrete frequency index k_{zc} of the first zero crossing is located at

$$k_{zc} = 1$$

Since the length of the rectangle in our scenario is N samples that cover the duration T_u , we get the first zero crossing frequency F_{zc} as

$$\frac{F_{zc}}{F_S} = \frac{k_{zc}}{N} \rightarrow F_{zc} = \frac{1}{NT_S} = \frac{1}{T_u} \quad (9.11)$$

which interestingly is the same as Eq (9.10).

It is not surprising that the spacing Δ_F between any two subcarriers and the first zero crossing of the sinc signal is the same, equal to $1/T_u$. *This is what orthogonality means in frequency domain!*[†]

With this information at hand, we draw the subcarriers in frequency domain, all at the same time in Figure 9.16a. Notice the zero crossings of the 0th subcarrier passing through

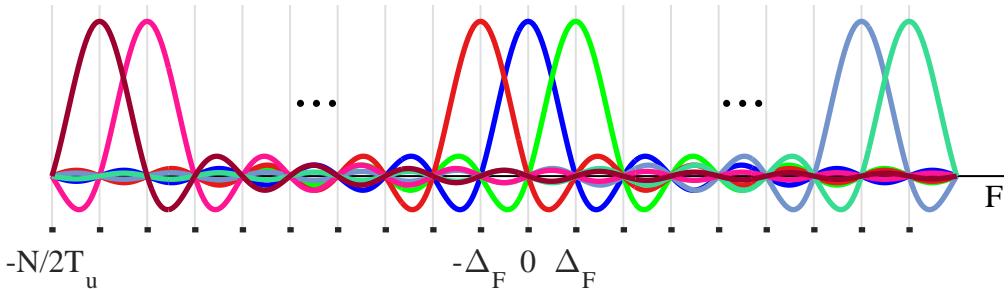
$$\pm\Delta_F = \pm\frac{1}{T_u}, \quad \pm 2\Delta_F = \pm\frac{2}{T_u}, \quad \dots$$

and so on, right at the peaks of the other subcarrier locations $k\Delta_F = k/T_u$.

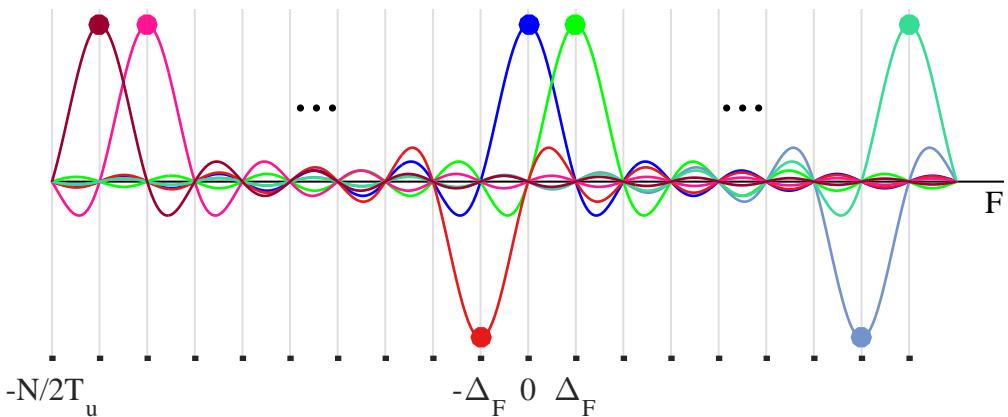
When a linear modulation scheme, such as BPSK, is utilized, the subcarrier amplitudes are modulated as $a[k] = +1$ or $a[k] = -1$ and an iDFT is taken to generate the time domain signal. The DFT at the Rx then needs to be taken at exactly the same set of frequencies k/T_u . In that way, the Rx 'samples' the OFDM waveform in frequency domain at exactly the peaks of subcarriers, exhibiting zero Inter-Carrier Interference (ICI) – maximum contribution comes from the desired subcarrier and zero contribution from all the rest – a phenomenon similar to Inter-Symbol Interference (ISI) and Nyquist's no-ISI criterion in single-carrier systems. This is plotted in Figure 9.16b. Later, we will see how the presence of a CFO causes the 'sampling' in frequency domain at the wrong instants thus introducing ICI in the system.

This is a beautiful dual of the no-ISI phenomenon encountered in time domain in the context of modulated Nyquist pulses and drawn in Figure 3.40 earlier. In time domain of single-carrier systems, the task of the symbol timing synchronization unit is to sample the waveform composed of Nyquist pulses at the peak of the eye diagram

[†]Another kind of orthogonality in frequency domain can be constructed through choosing completely non-overlapping bands for the transmission of two signals. This is the conventional and well known Frequency Division Multiplexing (FDM) system that is too dull for our purpose.



(a) Unmodulated OFDM subcarriers in frequency domain. Notice the zero crossings of each subcarrier passing through other subcarrier frequencies $k\Delta_F = k/T_u$



(b) Modulated OFDM subcarriers in frequency domain. The Rx utilizes exactly the same frequencies $k\Delta_F = k/T_u$ for DFT that ‘samples’ the signal in frequency domain at ideal ICI-free instants

Figure 9.16: Unmodulated and modulated OFDM subcarriers in frequency domain spaced $\Delta_F = 1/T_u$ apart from each other

(points of no ISI). In frequency domain of multicarrier systems, the task of the carrier frequency synchronization unit is to ‘sample’ the waveform composed of subcarriers at the peak of each subcarrier (points of no ICI). Two minor points are in order here.

- Using inverted commas for ‘sampling’ in frequency domain, I refer to the Rx carrier synchronization output. If it contains a zero Carrier Frequency Offset (CFO), we say that the waveform is ‘sampled’ at optimal locations. If there is a residual CFO present, the waveform is ‘sampled’ at a location other than the peaks of the subcarriers.
- Referring to the time frequency duality, a frequency domain eye diagram can also be drawn for an OFDM signal where the peak implies the point of zero CFO, just like the eye diagram in time domain exhibits the point of zero Symbol Timing Offset (STO).

The Puzzle of Violating the Sampling Theorem

From the discussion so far, one might think that these infinitely long sinc signals in frequency domain cause spectral aliasing due to not being bandlimited. This is what the sampling theorem has always promised us. It turns out that the aliasing does occur but in a harmless manner.

Interestingly, sampling at N samples/OFDM symbol implies that an exact replica of the spectrum seen in Figure 9.16 is also present centered at integer multiples of $F = F_S$ Hz or integer multiples of index $k = N$. Nevertheless, the zero crossings of those replica still pass through frequencies $k\Delta_F = k/T_u$ – no matter how far away from their center frequency – and hence inject zero ICI for this sample rate. This could be seen from Figure 9.16 if there was another subcarrier drawn to the left of the subcarrier at $F = -N/2T_u$. That would have been present at frequency

$$F = -F_S + \left(\frac{N}{2} - 1 \right) \frac{1}{T_u}$$

and its first right zero crossing would have passed through $F = -N/2T_u$.

In hindsight, this makes perfect sense since this is analogous to Nyquist no-ISI criterion for single-carrier systems, see Note 3.5.

Why Equalization Becomes Simpler

In Section 9.2.1, we saw the time domain picture of the multipath components of the orthogonal subcarriers in Figure 9.13 and claimed that the simpler equalization of OFDM systems is better explained in frequency domain. Let us explore this fact here.

We do not need to denote the DFT of the data symbols $a[k]$ since they are already assumed to be in frequency domain. By convention, the signal before the iDFT at the Tx is assumed to be in frequency domain and after the iDFT in time domain. These $a[k]$ modulated the subcarriers in Figure 9.16b. On the other hand, we can denote the DFT of the channel impulse response as $H[k]$.

To find the nature of these channel coefficients $H[k]$ in frequency domain, we first refer to Figure 9.13 where delayed copies of the Tx signal were drawn for each individual subcarrier k . Now we saw in Section 1.9 that a delay in time domain induces a *frequency dependent phase shift* in frequency domain.

$$\text{Time shift } s[(n \pm n_0) \bmod N] \longrightarrow \pm 2\pi \frac{k}{N} n_0 \quad \text{Phase shift}$$

The delay τ_i in seconds becomes τ_i/T_S in samples. Consequently, while each multipath copy of a subcarrier k arrives with the same frequency k/N but an amplitude ρ_i [†], its phase is also changed by an amount $-2\pi(k/N)\tau_i/T_S$. For example, for a single multipath copy arriving with amplitude ρ_1 at τ_1 seconds after the direct path, this phase shift is illustrated in Figure 9.17 along with the direct path which is a graphical manifestation of what we got in Eq (9.6) during time domain description.

The result for several multipath signals can be produced by extending the same concept, i.e., when many of these vectors are added for the same subcarrier frequency

[†]As described before in time domain, remember that the subcarriers are complex sinusoids, *the actual amplitudes γ_i are complex* and given as a function of carrier frequency F_C in Eq (8.5). Nevertheless, for removing clutter to simplify the derivation, here we take them as real and equal to ρ_i .

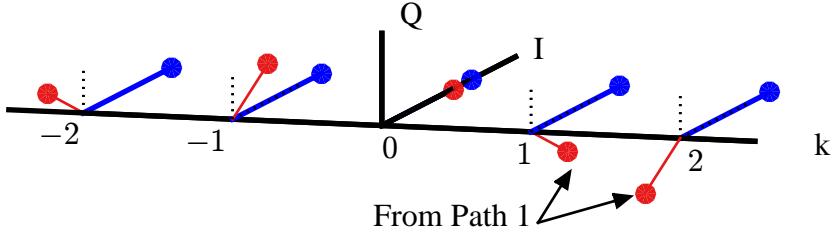


Figure 9.17: Multipath in frequency domain impact each subcarrier individually due to their orthogonality. Moreover, these narrow subcarriers induce flat fading for each k (here, complex channel gains γ_i are assumed real for simplification)

k/N , we get the channel coefficient $H[k]$ for each k as a vector sum of the channel taps, previously expressed in Eq (9.7).

In summary, the output of the channel – a product between the data symbols $a[k]$ and the channel frequency response $H(F)$ – is sampled at N locations in frequency domain by the subcarriers. At the Rx, the time domain signal is sampled at a rate of N samples per OFDM symbol before taking the DFT. The presence of the Cyclic Prefix (CP) converts the linear convolution with the channel into circular convolution. I wanted to write the exact mathematical derivation of this process but then this involves too many indices and instead can become a cause of confusion.

Treading on the visualization path, a signal level diagram was shown in Figure 8.67 in the context of single-carrier frequency domain equalization. There, we also saw how the FFT output $Z[k]$ in Eq (8.72) is a product between the data symbols DFT $A[k]$ and channel frequency response $H[k]$. However, the difference here is that an iDFT has already been taken at the OFDM Tx after defining the data symbols $a[k]$ in frequency domain. Therefore, in this context, the DFT output at the Rx is just the product between the Tx signal DFT $S[k]$ and $H[k]$.

$$Z[k] = S[k] \cdot H[k] \quad (9.12)$$

for each $k = -N/2, \dots, N/2 - 1$

Owing to the orthogonality of the subcarriers which manifests itself in the form of each subcarrier k/T_u passing through zero at other subcarrier frequencies k'/T_u (i.e., zero ICI), $S[k]$ for each k is nothing but the data symbol value $a[k]$. Thus, we can write

$$Z[k] = a[k] \cdot H[k] \quad (9.13)$$

for each $k = -N/2, \dots, N/2 - 1$

Similar to Figure 8.67 in the single-carrier case, Figure 9.18 illustrates this process for our example channel and an all-ones modulated OFDM symbol where the sinc sidelobes in dB are clearly visible. The channel frequency response $H(F)$ is sampled by the DFT at the subcarrier frequencies $k\Delta_F = k/T_u$ to yield $H[k]$. This is shown as the dashed red curve, the Δ_F -spaced samples of which are multiplying each subcarrier data individually. It is evident that such a wideband channel has been converted into many parallel **frequency flat subchannels** through using the subcarriers.

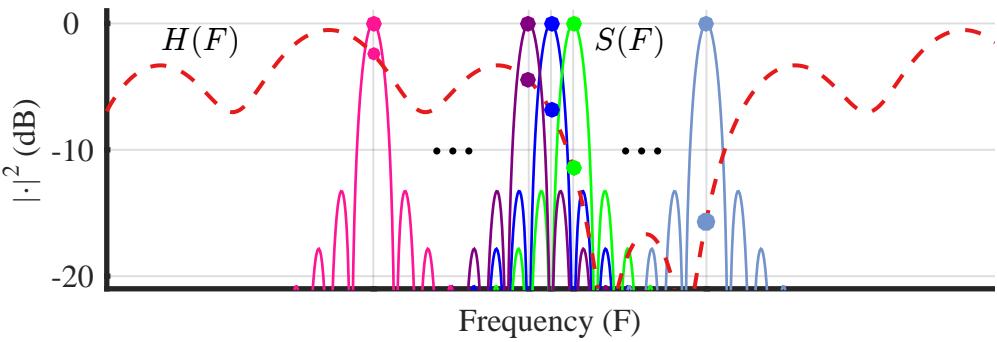


Figure 9.18: The DFT output at the Rx is the spectral product between the data symbol values $a[k]$ and channel frequency response $H[k]$ sampled at subcarrier frequencies $k\Delta_F = k/T_u$

Therefore, to recover the data symbol value when a channel estimate $\hat{H}[k]$ is available, all we have to do is divide each DFT output $Z[k]$ from $k = -N/2$ to $N/2 - 1$ by its corresponding channel coefficient $\hat{H}[k]$.

$$\hat{a}[k] = \frac{Z[k]}{\hat{H}[k]} \quad (9.14)$$

This is the fundamental concept of equalization in an OFDM system. That comes out to be N divisions for N symbols, or **1 division/symbol**, a huge computational saving over a time domain equalizer for a similar frequency selective channel.

Note 9.3 The sliced bread

The OFDM operation is very similar to processing a bread. Long ago, each time a person wanted to eat bread, they had to take a knife and cut a piece of bread for themselves. Then came sliced bread in July 1928 invented by a jeweler Otto Frederick Rohwedder that changed everything. Processing each individual slice got much easier; you could put jam, butter or cheese on different slices, see Figure 9.19.

It was difficult to process the whole bread before that invention. Similarly, it is difficult to process the collective spectrum for communication purposes before OFDM came on the scene. By slicing the spectrum, OFDM not only made it easier to equalize the wireless channel but also made it possible to send different modulation signals on different subcarriers, e.g., subcarriers experiencing a ‘good’ channel can be used to transmit a higher-order modulation signal (e.g., 16 or 64-QAM) that translates into more bits within the same time. It also made possible to assign different subcarriers to different users for transmission or receive different subcarriers from different users by instructing them beforehand. This kind of spectral slicing and manipulation helped in adoption of OFDM over many transmission schemes, even ahead of those with a slightly superior performance in some aspects.

On a lighter note, now we have a formal proof that OFDM is the best thing that happened since sliced bread.

In summary, what OFDM does in frequency domain is fairly simple. It just segments the available bandwidth of a frequency selective channel into **many parallel**



Figure 9.19: Just like a whole bread needs to be sliced for eating convenience, OFDM slices the spectrum for communication convenience. Compare with Figure 9.18

frequency flat channels through utilizing those sinusoidal subcarriers. This is because the segmentation factor N is chosen such that the bandwidth of these frequency flat channels Δ_F is within the coherence bandwidth B_C of the channel. From Eq (9.8) and using $\Delta_F = 1/T_u$,

$$\frac{1}{T_u} < \frac{1}{T_{\text{Del}}} \quad \Rightarrow \quad \Delta_F < B_C \quad (9.15)$$

Hence, equalization for each narrow slice requires just a single division operation, rendering the computational load of the equalizer to a total of N divisions. A relevant analogy is that of a train versus a fleet of trucks. While the train carriages might look quite similar, they are units in one complete set but the fleet of trucks comes with an independence of handling and manipulating each truck in an individual manner.

9.3 An OFDM Transceiver

Having covered the fundamental concepts, we now discuss the Tx and Rx structure for an OFDM system. We will see that it is very different to a Tx and Rx structure for a single-carrier system described in detail in Chapter 3. The most interesting thing to observe is the reversal of roles in time and frequency domains which will allow us to later exploit the time frequency duality for construction of OFDM synchronization algorithms.

Tx and Rx Structure

With all the background information in place, we can easily draw a block diagram for the Tx and Rx structure of an OFDM system shown in Figure 9.20. The information bits are mapped to the symbols in accordance with the chosen modulation scheme, e.g., PSK or QAM. Next, the symbols are grouped in blocks of N by a serial-to-parallel converter and an N -point iFFT is taken to generate the time domain waveform of the complex signal. The last N_{CP} samples of this time domain waveform are prepended back to the start of the sequence as a Cyclic Prefix (CP) for both as a guard interval and for converting the linear convolution with the channel into circular convolution. A parallel-to-serial converter then produces the serial sequence that is upconverted to the carrier frequency with the help of a Local Oscillator (LO). This signal is input to a DAC that outputs the continuous-time OFDM waveform.

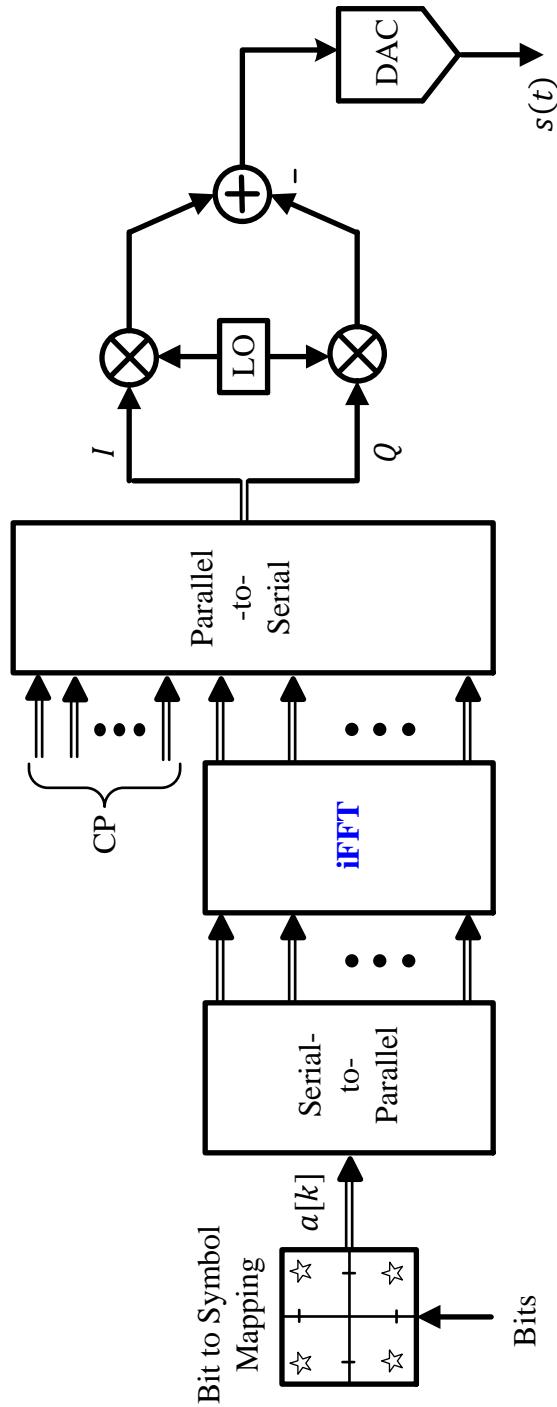


Figure 9.20: Generation of an OFDM signal using an N -point iFFT

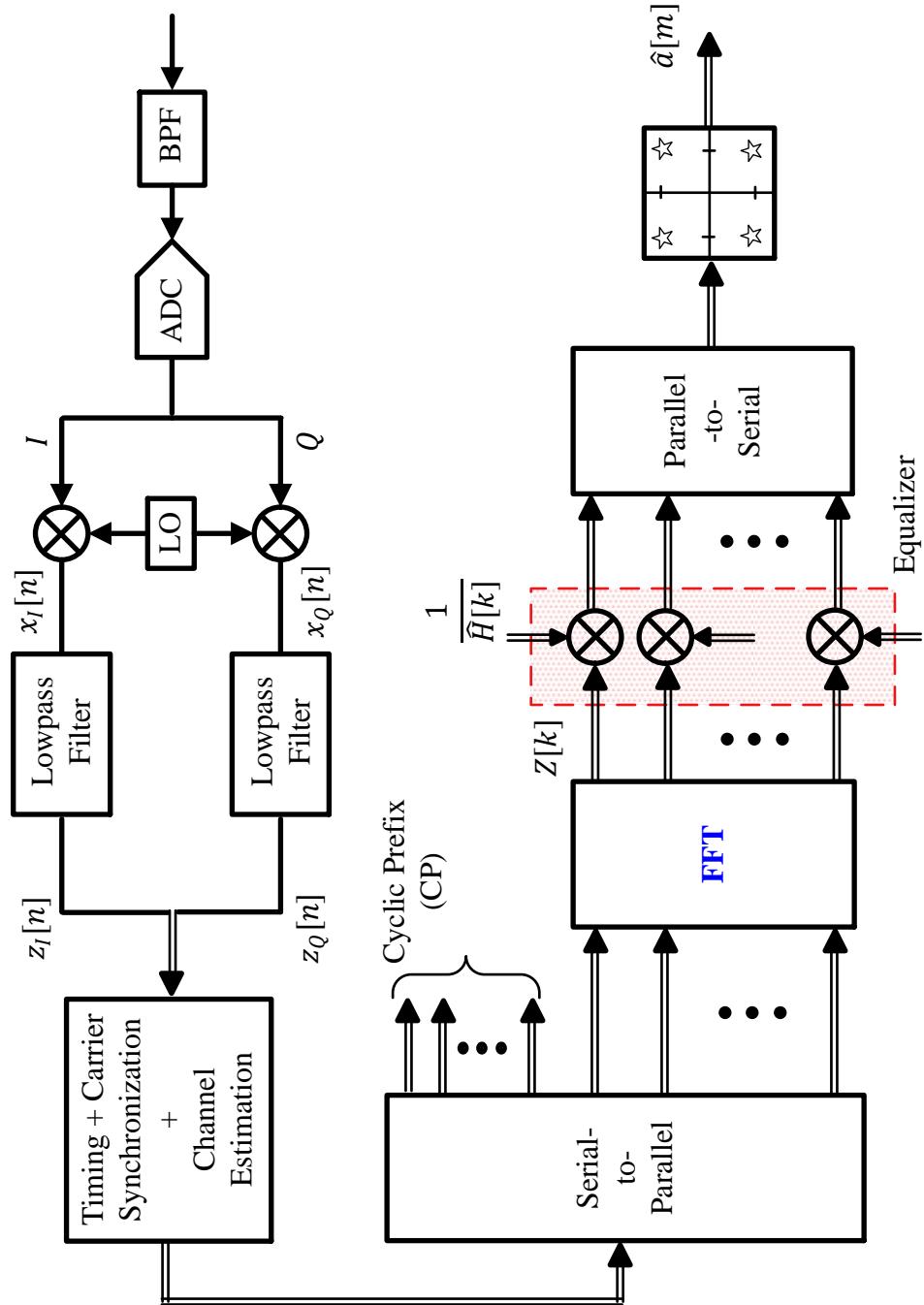


Figure 9.21: Detection of an OFDM signal using an N -point FFT before equalization

A block diagram of such a Rx is drawn in Figure 9.21. The OFDM Rx implements similar operations as the Tx but in reverse order. A continuous-time waveform is first bandpass filtered and downconverted with the help of a Local Oscillator (LO) where the choice of the ADC placement depends on the Rx architecture. This is lowpass filtered to reduce interference and noise as well as for sample rate conversion. Timing synchronization, carrier synchronization and channel estimation are performed at this stage. Then, a serial-to-parallel converter prepares the input for DFT computation. However, the first N_{CP} samples are first discarded since they are contaminated from the ISI of the previous block. Next, an N -point DFT is taken by using its efficient version, namely FFT, and multiplied with the channel inverse $1/\hat{H}[k]$ to equalize in frequency domain. Since the data symbols $a[k]$ were produced in frequency domain (which is the same as saying that an iFFT has already been taken) at the Tx, the equalized signal is ready for symbol detection which is parallel-to-serial converted and input to the decision device.

Turning to an example, we can now understand the relevant parameters of the well known IEEE 802.11a standard.

Example 9.1

In IEEE 802.11a standard, there are $N = 64$ subcarriers in total that can be scaled by the modulation symbols. In fact, only 48 out of 64 subcarriers are used for data transmission.

$$k = \{-26, \dots, -22, -20, \dots, -8, -6, \dots, -1, \\ +1, \dots, +6, +8, \dots, +20, +22, \dots, +26\}$$

Out of the remaining 16 subcarriers, 4 are used as pilot symbols for synchronization and channel tracking purposes (usually transmitted with a boosted power) while nothing is transmitted on 12 subcarriers. Subcarrier 0 is not used due to the DC offset problem described in Chapter 10 whereas 6 subcarriers on the left and 5 subcarriers on the right are used as guard bands to ease spectral and filtering requirements (recall that an OFDM signal consists of sinc signals in frequency domain which have a slow rolloff). In summary, the data modulated subcarriers are as below^a.

The subcarriers $\{-21, -7, +7, +21\}$ are used for inserting pilot subcarriers. After all these assignments, the subcarrier arrangement is shown in Figure 9.22.

After the mapping, an iDFT (or iFFT) is taken to convert the subcarriers to time domain, a cyclic prefix is inserted and finally the resulting waveform is truncated to a single OFDM symbol length by applying time domain windowing. The timing parameters of such an OFDM signal are given in Table 9.1.

^aThese subcarriers for later generations of 802.11 systems kept increasing for higher data rates. For example, an 802.11n OFDM signal contains 28 subcarriers on each side of the spectrum in a 20 MHz channel and 57 subcarriers on each side in a 40 MHz channel. On the other hand, 802.11ac utilizes 121 subcarriers on each side for an 80 MHz channel.

A Comparison with SC-FDE

Recall that in case of Single-Carrier Frequency Domain Equalization (SC-FDE) discussed in Section 8.3.8, both the FFT and iFFT were performed at the Rx. In an OFDM system on the other hand, the iFFT is implemented at the Tx and the FFT at the Rx. The respective block diagrams of the two systems are drawn in Figure 9.23.

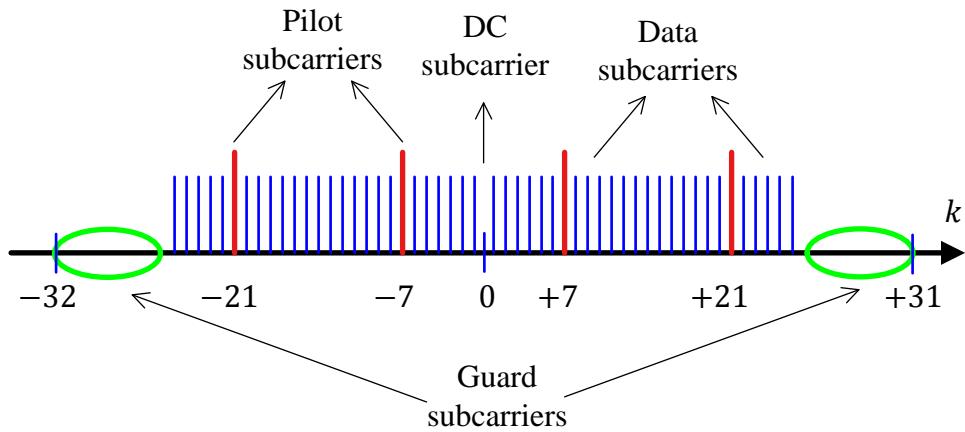


Figure 9.22: Mapping of the DC, pilot, data and guard subcarriers in IEEE 802.11a

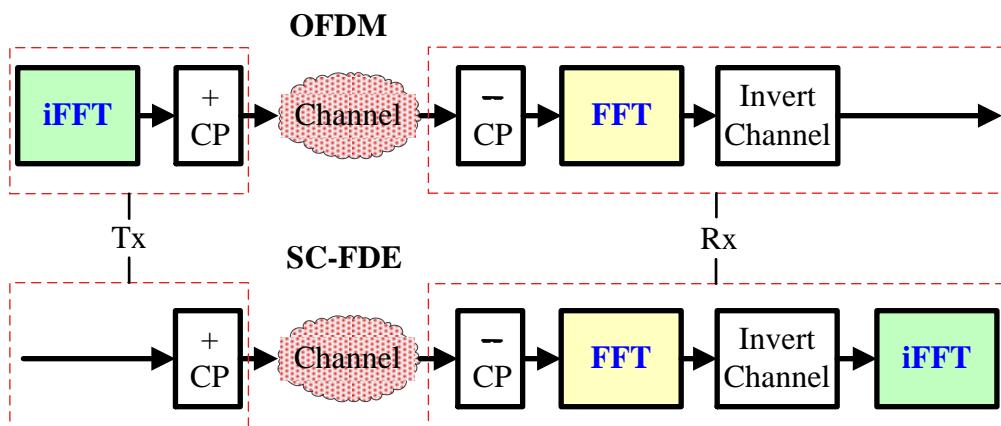


Figure 9.23: A block diagram comparison of an OFDM and an SC-FDE system

Table 9.1: Timing for IEEE 802.11a OFDM System

Parameter	Value
Bandwidth $B = 1/T_S$	20 MHz
Data subcarriers	48
Pilot subcarriers	4
Null subcarriers	12
Total subcarriers	$48 + 4 + 12 = 64$
Subcarrier spacing $\Delta_F = B/N$	$20\text{MHz}/64 = 312.5 \text{ kHz}$
Useful symbol duration (FFT period) $T_u = 1/\Delta_F$	$3.2\mu\text{s}$
Cyclic prefix length T_{CP} (25% of T_u)	$T_u/4 = 0.8\mu\text{s}$
Total symbol duration T_{OFDM}	$3.2 + 0.8 = 4\mu\text{s}$

The overall effect of this change is that OFDM transmits the symbols on orthogonal subcarriers whereas an SC-FDE system spreads the information of each data symbol across all subchannels, thus providing a little extra frequency diversity. Consequently, once a subcarrier in OFDM gets in a deep fade, the information about that data symbol is completely gone and noise enhancement varies with different subcarriers. On the other hand, an SC-FDE Rx still gathers some of that information from other subchannels and each symbol sees the same effective noise variance. Hence, the performance of an SC-FDE system is slightly better than an OFDM system in general.

This remains true in an uncoded system. Channel coding is a process in which extra redundancy is added to the information bits to make them more resilient to errors occurring in the transmission process. It turns out that in a system with channel coding and interleaving included, the performance of both systems is comparable to each other.

In other aspects, OFDM is more sensitive to RF impairments than SC-FDE systems. For example, it is sensitive to power amplifier non-linearities because the ratio between the peak of the OFDM signal and its average value – known as *peak to average power ratio* – is higher. Furthermore, OFDM systems suffer more in the presence of a carrier frequency offset and phase noise as compared to single-carrier systems.

Why is OFDM still the dominant technology for high rate wireless communication so far when an SC-FDE Rx can also be implemented? Besides the fact that OFDM balances the computational load through an iFFT at the Tx, the flexibility of having a sliced spectrum at hand open to manipulation of any kind cannot be overlooked. As stated before, a different modulation and coding scheme can be allocated to each

subcarrier based on their individual channel gains. In addition, different users can be assigned a set of subcarriers to perform multiplexing known as Orthogonal Frequency Division Multiple Access (OFDMA).

It should be noted that although SC-FDE was proposed decades earlier, it was revived in 1995 by an article from Hikmet Sari in [40]. I must add that *the rediscovery of SC-FDE did take the original magic away from the beauty of OFDM transmission.* When the idea of OFDM was treated as a new revolution, Sari's article slightly shook this excitement and told us that we already had something equivalent we ignored during all this time.

This is the time we briefly touch on the concept of pulse shaping in OFDM systems.

Pulse Shaping in OFDM Systems

This discussion heavily relies on the excellent article by Keysight Technologies in Ref. [41]. Originally, OFDM is a block processing scheme and hence there is no extra pulse shaping required after the iDFT at the Tx. Instead, a certain number of subcarriers, both on the positive and negative edges of the spectrum, are left unmodulated to cater for the decay of sinc sidelobes thus alleviating the subsequent digital and analog filtering tasks. In Example 9.1 on IEEE 802.11a standard, 6 subcarriers on the negative edge and 5 subcarriers on the positive edge of the spectrum were assigned as null subcarriers.

Nevertheless, many systems still employ pulse shaping in OFDM systems to smooth the edges at the OFDM symbol boundaries that reduces the out of band power to meet the regulatory requirements. Recall that the pulse shape employed in a single-carrier system is a Raised Cosine in frequency domain. Therefore, from time frequency duality, we already know that the Raised Cosine pulse shape in an OFDM system is in time domain. We have covered the topic of pulse shaping in depth in Section 3.6 and hence briefly describe this operation in the current context.

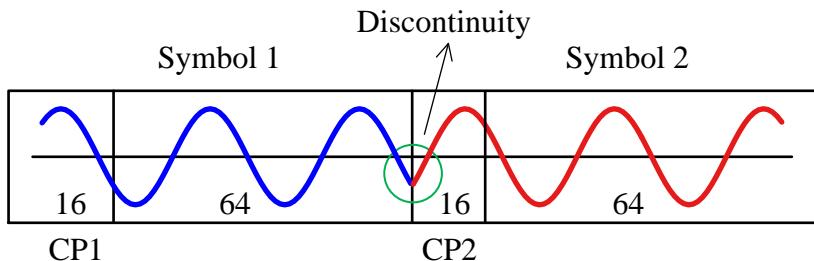
We created the Cyclic Prefix (CP) by appending the last N_{CP} samples from the iFFT output to the beginning of the OFDM symbol. The total symbol duration thus becomes $N + N_{CP}$ samples. For obvious reasons, the next OFDM symbol begins with a different amplitude and phase at the boundary of the current symbol, as shown in Figure 9.24a for $N = 64$ and $N_{CP} = 16$. This discontinuity causes spectral regrowth that violates the spectral mask and creates interference for adjacent channels. Therefore, our task is to assemble consecutive OFDM symbols in a smooth fashion.

To introduce a smooth transition between the current and next OFDM symbol, a cyclic suffix (also known as cyclic postfix) is introduced. Just like a cyclic prefix is the last N_{CP} samples in an N sample symbol, a cyclic suffix is the first N_{CS} samples out of N that are attached at the end.

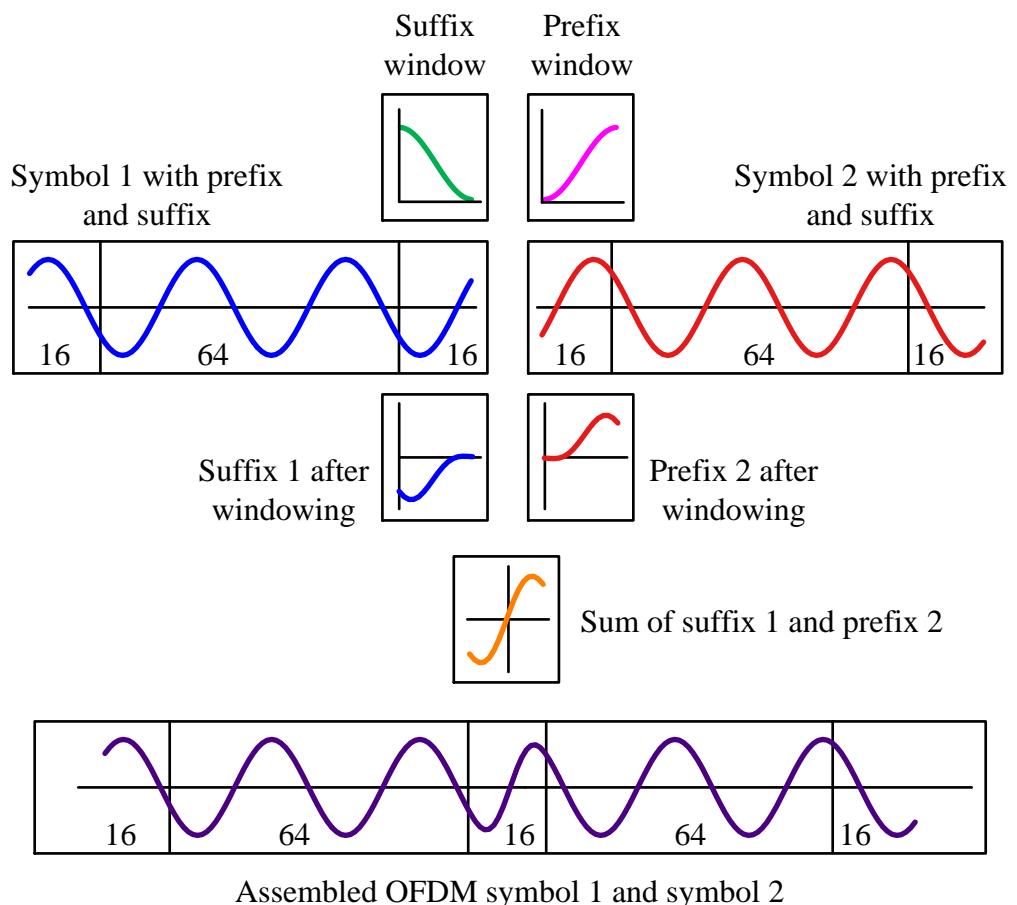
Let us create an example for $N = 64$ and $N_{CP} = 16$. The total number of samples in this OFDM symbol is equal to $N + N_{CP} = N_{OFDM} = 80$ as in Figure 9.24a and it should remain the same after the windowing operation. Here, we assume a full length cyclic suffix of length $N_{CS} = 16$ samples to understand the concept.

When the first 16 samples of OFDM symbol 1 are appended in the end as a cyclic suffix, the symbol becomes 96 samples long which violates the $N_{OFDM} = 80$ sample condition of the original signal. Instead, an alternative scheme is applied as follows for which a graphical representation is illustrated in Figure 9.24b.

- Length 16 cyclic suffix of symbol 1 is windowed with the transition part of a Raised Cosine that rolls off from 1 to 0.



(a) Discontinuity arising without pulse shaping



(b) Pulse shaping produces a smooth waveform

Figure 9.24: At the boundary of two OFDM symbols, pulse shaping smoothes the edges, resulting in the avoidance of spectral regrowth

- Length 16 cyclic prefix of symbol 2 is windowed with the transition part of the Raised Cosine that rolls on from 0 to 1.

- These two segments are overlapped and summed together to generate a length 16 portion that is attached between both symbols.

Notice that the waveform is now continuous going from one symbol to the next while the total OFDM symbol duration is still 80.

Recall that one purpose of the cyclic prefix is to convert linear convolution with the channel into circular convolution. This circular convolution is a result of the periodicity that appears due to the cyclic prefix being the same as the last part of the symbol. However, now a full length cyclic suffix of the first symbol interferes with the cyclic prefix of the next symbol, thus leaving no room for the periodicity to appear.

The solution is to reduce the length of the window, i.e., cyclic suffix, to as less number of samples as possible. For the current example, a length 4 cyclic suffix implies that the multipath immunity has reduced from $N_{CP} = 16$ samples to $N_{CP} - N_{CS} = 12$ samples. Only a maximum channel length of $N_{Tap} = 12$ can now induce a single tap flat fading at each subcarrier of the Rx DFT output. This is the price we pay for the spectral smoothing of the waveform.

A variant of this technique is known as Windowed-OFDM (W-OFDM) in which windowing is applied in a manner that the window overlaps with the complete OFDM symbol as well as the CP and the cyclic suffix. Moreover, the window amplitude is -3 dB at the start and end of the OFDM symbol. For a perfect reconstruction of the OFDM symbol, the cyclic suffix and cyclic prefix are not discarded and instead are added to the start and the end, respectively.

OFDM Time-Frequency Grid

In single-carrier systems, the frequency domain is not defined in an isolated sense as in OFDM systems. Instead, the spectrum of the signal contains contributions from all the modulation symbols. From the discussion so far, we come to know that OFDM is particularly attractive due to its ability to treat the frequency domain in a sliced manner, thanks to the orthogonal subcarriers. This gives rise to the concept of time-frequency grid where the index k defines individual subcarriers on the frequency axis while the index m defines successive OFDM symbols on the time axis. This is plotted in Figure 9.25 where the indices m and k lie on horizontal and vertical axis, respectively.

The concept of time-frequency grid brings much freedom in manipulating the frequency axis. For example, it makes tasks such as synchronization in general and channel estimation in particular quite straightforward. Furthermore, interpolation between those subcarriers produces reliable channel estimates for the whole spectrum. This is what we will discuss in Section 9.7.

Next, we discuss the synchronization techniques for OFDM systems which is mainly influenced by Ref. [42] and [43].

9.4 Timing Synchronization

We have arrived here by covering enough scenarios of time frequency duality that synchronization issues in an OFDM system can be easily understood. The two major impairments discussed above are

- Inter-Symbol Interference (ISI) where one OFDM symbol can leak into the other OFDM symbol, and

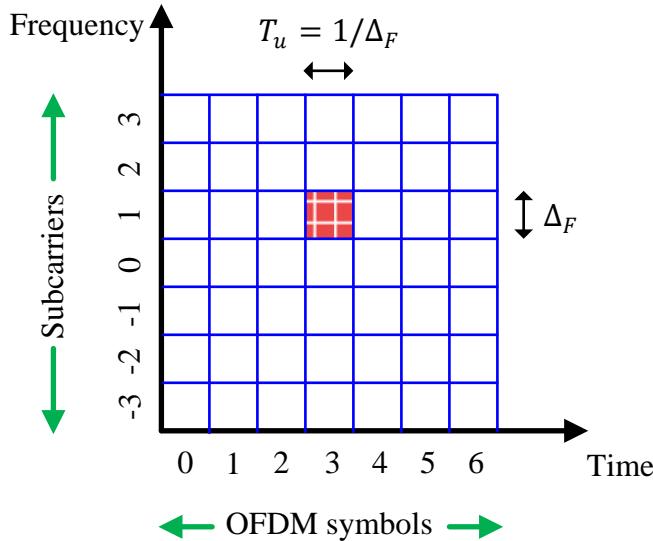


Figure 9.25: OFDM time-frequency grid

- Inter-Carrier Interference (ICI) where the orthogonality among the subcarriers is destroyed.

While we have discussed that ISI can happen when the channel length is longer than the CP, it can also happen if we choose the wrong window of N samples before taking the DFT at the Rx, i.e., misalignment in timing synchronization. Although this same phenomenon also causes ICI, the subcarriers also suffer from ICI due to a misalignment in carrier frequency synchronization (or to a minor extent, sampling clock frequency synchronization).

The choice between a feedforward and a feedback implementation depends on the performance requirements and system specifications. By virtue of block processing in OFDM, feedforward approaches naturally complement the burst mode transmissions. On the other hand, most continuous transmissions like Digital Audio Broadcasting (DAB) and Digital Video Broadcasting (DVB) can implement a closed loop approach. There are also many hybrid techniques where the acquisition is done in a feedforward manner while the pilot-based estimates are used to drive loops that slowly correct the radio impairments.

In this chapter, we focus on feedforward techniques, as feedback methods such as a PLL, FLL and TLL are discussed in enough details in the previous chapters and the reader can easily carry over similar feedback techniques from single-carrier to OFDM systems.

Next, we discuss the effects of a timing impairment without help of many equations.

Effect of Timing Mismatch

To avoid using many indices, we skip the OFDM symbol index m in the following expressions to focus on each OFDM symbol. Moreover, upconversion and downconversion through a carrier at frequency F_C is also bypassed and complex baseband

representations are employed. Finally, no additive white Gaussian noise is present to study the effects of each impairment in isolation.

With these settings in place, a baseband OFDM signal at the Tx is given in Eq (9.4) and reproduced below.

$$\begin{aligned} I \rightarrow \quad v_I[n] &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[a_I[k] \cos 2\pi \frac{k}{N} n - a_Q[k] \sin 2\pi \frac{k}{N} n \right] \\ Q \uparrow \quad v_Q[n] &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[a_Q[k] \cos 2\pi \frac{k}{N} n + a_I[k] \sin 2\pi \frac{k}{N} n \right] \end{aligned} \quad (9.16)$$

Keep in mind that the data symbols $a[k]$ arise from a linear modulation scheme in exactly the same manner as in single-carrier systems but are assumed to be in frequency domain here due to the iFFT operation involved.

After the DAC, what is the form of the continuous-time signal? Recall the expression for subcarriers in continuous domain in Eq (9.2) given by $2\pi F_k t$ where each subcarrier frequency F_k in Hz is given by

$$F_k = \frac{k}{NT_S} = \frac{k}{T_u} = k\Delta_F \quad \text{for each } k = -N/2, \dots, N/2 - 1$$

where we have used $k/N = F/F_S$. The above relation reinforces the concept that as the subcarrier frequencies F_k should be integer multiples of subcarrier spacing Δ_F , see Figure 9.16a. Now using $(k/N)n = (F_k/F_S)n = F_k n T_S$ which translates to $F_k t$ in continuous time domain, the OFDM signal in continuous-time becomes

$$\begin{aligned} I \rightarrow \quad v_I(t) &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} [a_I[k] \cos 2\pi F_k t - a_Q[k] \sin 2\pi F_k t] \\ Q \uparrow \quad v_Q(t) &= \frac{1}{N} \sum_{k=-N/2}^{N/2-1} [a_Q[k] \cos 2\pi F_k t + a_I[k] \sin 2\pi F_k t] \end{aligned}$$

The spectrum of such an OFDM signal is drawn in Figure 9.26. Notice the rectangular structure within the spectrum and sinc like sidelobes at the edges. To constrain the spectrum, a time domain Raised Cosine (RC) pulse shape can also be used which was described earlier.

With zero noise, the same signal is received at the Rx convoluted by the channel and the baseband version, following our conventions from the previous chapters, is denoted by $x(t)$. Before we sample $x(t)$ to go to the discrete domain, we need to know how a Carrier Frequency Offset (CFO) F_Δ and a Symbol Timing Offset (STO) ε_Δ affects the Rx signal. For this purpose, let us focus on the expression $F_k t$. After the synchronization errors, this becomes

$$2\pi F_k t \longrightarrow (F_k + F_\Delta)(t + \varepsilon_\Delta)$$

As opposed to introducing a negative ε_Δ as in $x(t - \varepsilon_\Delta)$ done in single-carrier systems before, there is a reason to choose a positive ε_Δ here that will become clear in the discussion on timing synchronization. Now after sampling at a rate of $F_S = 1/T_S$, we

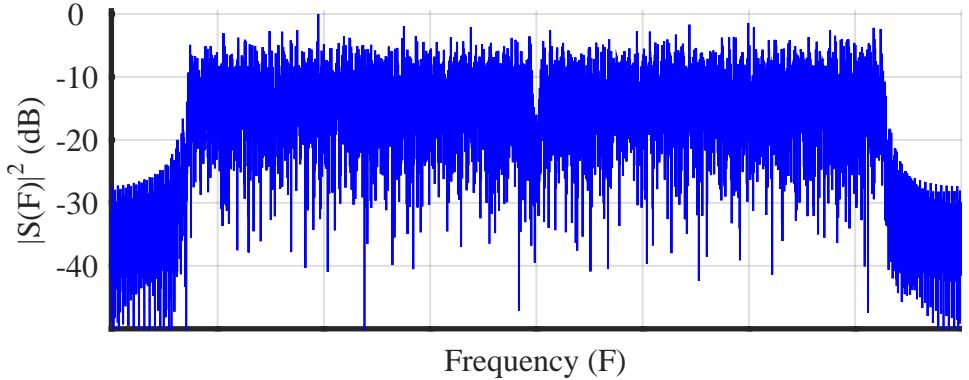


Figure 9.26: Spectrum of a typical OFDM signal

can write

$$\begin{aligned}
 2\pi(F_k + F_\Delta)(t + \varepsilon_\Delta) \Big|_{t=nT_S} &= 2\pi(F_k + F_\Delta)(nT_S + \varepsilon_\Delta) \\
 &= 2\pi(F_k + F_\Delta) \left(n + \frac{\varepsilon_\Delta}{T_S} \right) T_S \\
 &= 2\pi \frac{F_k + F_\Delta}{F_S} \left(n + \frac{\varepsilon_\Delta}{T_S} \right) \\
 &= 2\pi \left(\frac{F_k}{F_S} + \frac{F_\Delta}{N\Delta_F} \right) \left(n + \frac{\varepsilon_\Delta}{T_S} \right) \\
 &= \frac{2\pi}{N} \left(k + \frac{F_\Delta}{\Delta_F} \right) \left(n + \frac{\varepsilon_\Delta}{T_S} \right)
 \end{aligned}$$

where we have used $F_k = k\Delta_F$ and

$$\frac{1}{\Delta_F} = T_u = NT_S = \frac{N}{F_S} \Rightarrow F_S = N \cdot \Delta_F$$

Also, F_Δ/Δ_F [†] and ε_Δ/T_S are normalized CFO and STO, respectively, as encountered before in the context of single-carrier systems. We denote them by

$$\nu_0 = \frac{F_\Delta}{\Delta_F} \quad \varepsilon_0 = \frac{\varepsilon_\Delta}{T_S}$$

With these settings in place, the above expression can be modified as

$$2\pi(F_k + F_\Delta)(t + \varepsilon_\Delta) \Big|_{t=nT_S} = \frac{2\pi}{N} (k + \nu_0) (n + \varepsilon_0) \quad (9.17)$$

Now we can use the above relation to get the sampled baseband version of the Rx signal $x[n]$ which after lowpass filtering as in Figure 9.21 produces the signal $z[n]$. Skipping the mathematical details of this process, all we need to do is include the

[†]As a reminder, F_Δ here is the CFO and Δ_F is the subcarrier spacing.

product of a channel coefficient $H[k]$ with data symbol $a[k]$ for each subcarrier k into Eq (9.16). Defining $Z[k] = a[k]H[k]$ from Eq (9.13),

$$I \rightarrow z_I[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[Z_I[k] \cos \frac{2\pi}{N}(k + \nu_0)(n + \varepsilon_0) - Z_Q[k] \sin \frac{2\pi}{N}(k + \nu_0)(n + \varepsilon_0) \right] \quad (9.18a)$$

$$Q \uparrow z_Q[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \left[Z_Q[k] \cos \frac{2\pi}{N}(k + \nu_0)(n + \varepsilon_0) + Z_I[k] \sin \frac{2\pi}{N}(k + \nu_0)(n + \varepsilon_0) \right] \quad (9.18b)$$

For a zero CFO and STO, the above equation is just an iDFT[†] of $Z[k]$. Hence after the DFT at the Rx, we get our $Z[k]$ back which are the modulation symbols $a[k]$ scaled by the channel coefficients $H[k]$.

We conclude that the timing synchronization problem in OFDM systems is slightly different than single-carrier systems. There, a symbol timing offset is the fractional interval between 0 to symbol time T_M due to the necessity of just 1 sample/symbol for symbol detection. Here, while the sample/symbol is still 1, these symbols are grouped into N samples of an OFDM symbol. Consequently, there is an integer part of the Symbol Timing offset (STO) and a fractional part.

Integer STO: The integer part of the STO problem corresponds to locating the right sample. What exactly is the right sample? A major portion of the processing at the Rx is composed of taking the N -point DFT (actually, FFT) of the incoming signal. For this purpose, the Rx must know the starting sample of the sequence which will form the input to this FFT routine. This samples coincides with the end of the CP and constitutes of the useful portion of the OFDM symbol T_u . As a result, all N samples participate in estimating the location of the right sample. This availability of N samples/OFDM symbol implies that the (OFDM) symbol timing synchronization problem is quite similar to the frame synchronization part in the single-carrier systems. This is shown in Figure 9.27 where the boundaries of the OFDM symbols in an OFDM frame are drawn for clarification purpose. The task of the timing synchronization unit is to locate these boundaries.

Fractional STO: As far as the fractional part of the STO is concerned, it is a bit surprising but we do not really need to estimate it. Recall from Section 1.9 that a delay in time domain induces a *frequency dependent phase shift* in frequency domain.

$$\text{Time shift } s[(n \pm \varepsilon_0) \bmod N] \longrightarrow \pm 2\pi \frac{k}{N} \varepsilon_0 \quad \text{Phase shift} \quad (9.19)$$

Consequently, after the DFT, the fractional timing offset appears as a phase rotation at each individual subcarrier k and *hence becomes part of the phase of*

[†]If you are familiar with complex notations in OFDM, you can recognize this equation as

$$z[n] = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} a[k] H[k] \exp \left(j \frac{2\pi}{N} (k + \nu_0)(n + \varepsilon_0) \right)$$

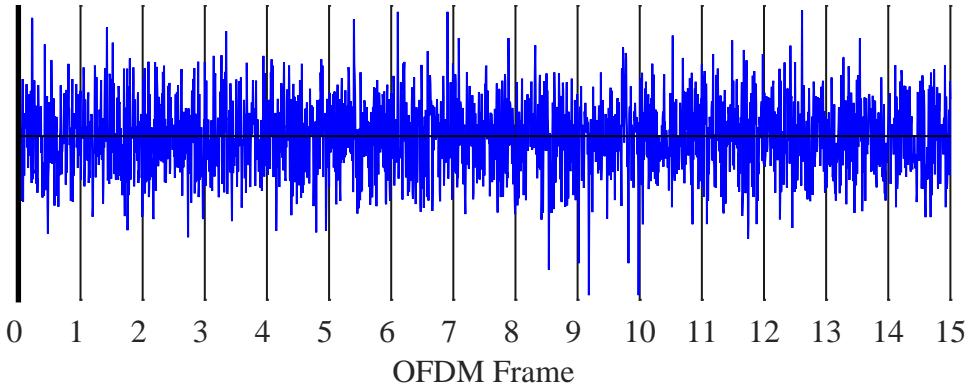


Figure 9.27: The task of the timing synchronization unit is to locate the OFDM symbol boundaries (i.e., their starting samples) in an OFDM frame

the channel gain $H[k]$. Since we were not slicing the spectrum in separate subcarriers in single-carrier systems, we could not apply this result there for rate $1/T_M$ sampling. But the timing phase offset does become part of the channel phase for a fractionally-spaced equalizer, enabling it to compensate for this timing phase. A timing recovery loop, however, is still required to track the waveform periodicity in the form of clock frequency.

The question now is the following: how far away the timing offset can be from the ideal sampling instant such that no ISI or ICI is contributed in the DFT input. For this purpose, refer to Figure 9.28 that shows two OFDM symbols in succession.

- The ideal sampling instant is the sample where the CP ends. However, even if our first sample is behind this location, there is no harm done as long as it lies after the contamination from the previous symbol (i.e., beyond the channel length). Carefully look at the waveform portion of Figure 9.28. In this region of no concern, *only the multipath copies of the same subcarrier* are being summed up for each subcarrier and this can only induce a change in phase. This change in phase merges into the channel phase $\angle H[k]$ which is compensated by the equalizer later.

If the STO ε_0 is within this region, and a DFT is computed at the Rx, we know from our knowledge of time frequency duality that the effect of a timing error on an OFDM symbol should cause a spinning of the symbols defined in frequency domain, just like a CFO in a single-carrier systems rotated the constellation of our symbols defined in time domain, see Figure 6.3. Therefore, after the DFT and owing to Eq (9.19), the Rx symbols will be rotating in circles with an inverse period ε_0 (whether it is an integer or a fraction), as long as ε_0 is within the region of no concern. This is drawn in Figure 9.29a.

To confirm our intuition, we manipulate the argument of the subcarriers at the

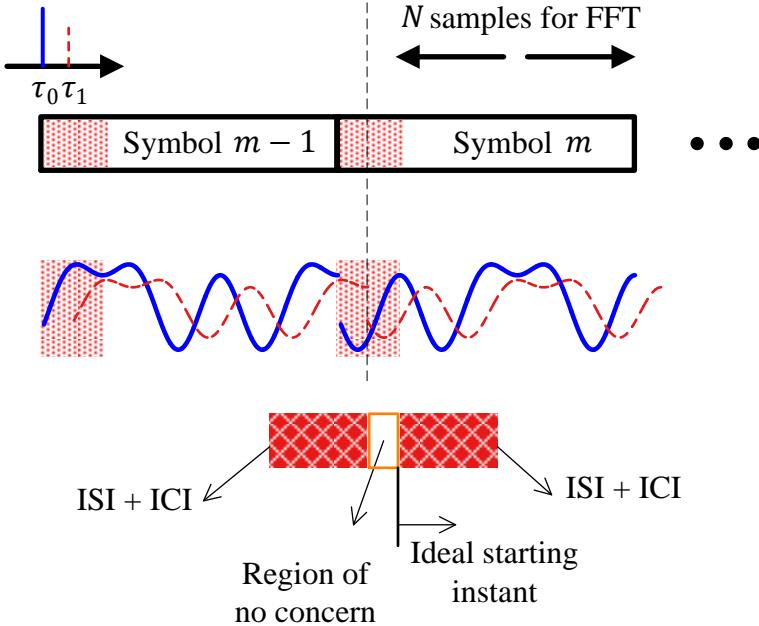


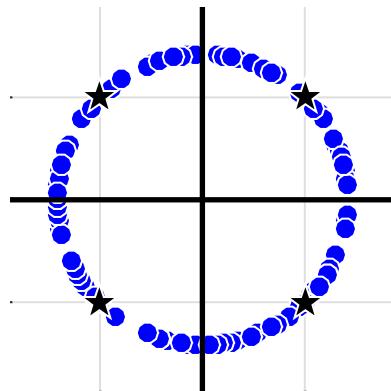
Figure 9.28: The effect of STO ε_0 on an OFDM symbol

Rx FFT input in Eq (9.17), reproduced below, as follows.

$$\begin{aligned} 2\pi(F_k + F_\Delta)(t + \varepsilon_\Delta)|_{t=nT_s} &= \frac{2\pi}{N} (k + \nu_0) (n + \varepsilon_0) \\ &= \frac{2\pi}{N} \left\{ \underbrace{kn}_{\text{Term 1}} + \underbrace{\nu_0 n}_{\text{Term 2}} + \underbrace{k\varepsilon_0}_{\text{Term 3}} + \underbrace{\nu_0 \varepsilon_0}_{\text{Term 4}} \right\} \end{aligned}$$

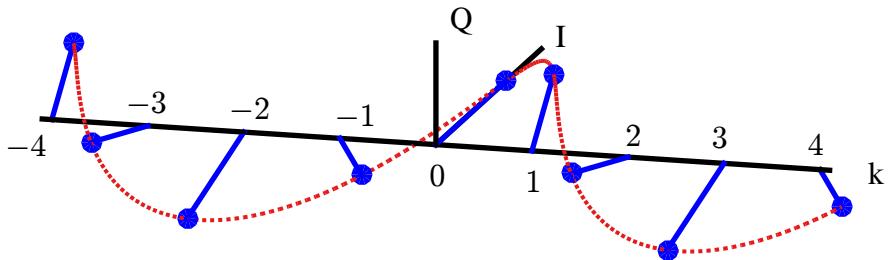
- ◊ From the DFT definition in Eq (1.53), we know that the FFT at the Rx consists of subcarriers with the arguments $-kn$ which simply cancels out Term 1 above.
- ◊ Term 2 appears as a normalized CFO at the FFT output because it is a function of time n .
- ◊ Term 4 is neither a function of n nor a function of k . Hence, it appears as a constant phase that gets merged into the channel phase $\angle H[k]$.
- ◊ This leaves Term 3 as a function of STO and clearly it is being multiplied with the subcarrier index k . As a result, the data symbols at each subcarrier k get rotated by the product of this index k varying from $-N/2$ to $N/2 - 1$ with the STO ε_0 and this is what causes the constellation rotation. In other words, these data symbols are multiplied with a complex sinusoid $2\pi(k/N)\varepsilon_0$ with inverse period ε_0 and we see this complex sinusoid rotating in frequency domain in Figure 9.29b, which is exactly the same as drawn in Figure 9.29a in the form of a scatter plot.

- Referring back to Figure 9.28, when the starting sample of the OFDM symbol is



(a) A small STO in the DFT input spins the DFT constellation output in circles. Compare this scatter plot with that in Figure 6.3a

OFDM Data Symbols with a Timing Offset



(b) Frequency domain representation of the data subcarriers are modulated with a complex sinusoid with inverse period ε_0

Figure 9.29: Effect of a timing offset ε_0 on the data subcarriers: scatter plot and frequency domain subcarriers

before the region of no concern, the subcarriers from the previous OFDM symbol interfere with the current OFDM symbol inducing both ISI and ICI.

- Finally, when the input window is after the ideal sampling instant, then a similar ISI and ICI come from the next OFDM symbol. If you are not clear on why ICI occurs in such a case, refer to Figure 9.11.

Having established the necessity to have synchronization in place, we discuss some of the relevant estimation techniques. In the single-carrier systems, our Master Algorithm, the correlation, was mostly employed in an implicit manner in the sense that the fundamental expression was derived from the correlation and then various approximations were found to estimate the desired parameter. In the case of OFDM, the

correlation is overwhelmingly utilized in an explicit manner for time domain estimation of synchronization parameters. The discussion is mostly based on the synchronization techniques in Ref. [44].

Training Correlation

First, consider the correlation of a pseudo-random sequence of +1s and -1s with itself (i.e., autocorrelation) for several time shifts[†]. The summation in the correlation of such a long stream of opposite signs will mostly cancel out and the resultant value will be large only when they exactly coincide. From Section 2.8, we know that the autocorrelation of white noise is a unit impulse which was drawn in Figure 2.39. When we draw the autocorrelation of a pseudo-random sequence in Figure 9.30 for a range of negative and positive time shifts, we can see that such a sequence approximately behave like noise. The important result is that *the autocorrelation of this sequence has a sharp peak when completely aligned*, much higher than any other time shift.

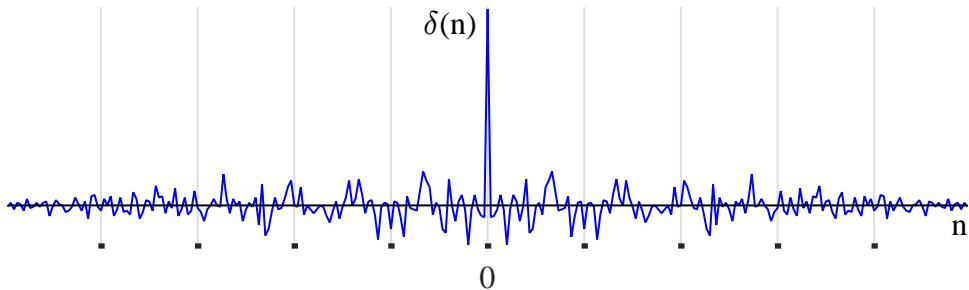


Figure 9.30: Autocorrelation of a pseudo-random sequence is similar to an impulse, just like white noise

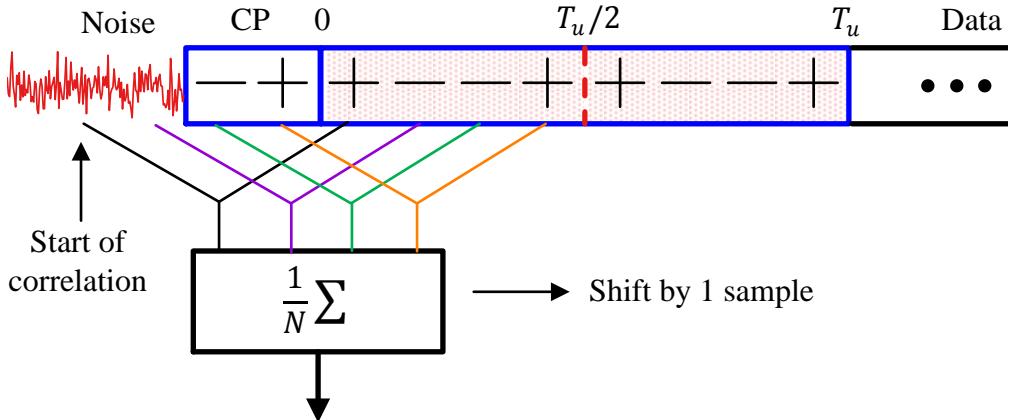
Second, two copies of such a pseudo-random sequence is prepended before the data symbols without any iFFT at the Tx, i.e., the training sequence is in time domain. The Rx is sampling the signal in the expectation of the OFDM frame. Here, the initial samples are composed of just noise, followed by the above training sequence and then the OFDM symbols as illustrated in Figure 9.31a. We choose an initial sample and compute the correlation as follows. If a sample in $z[n]$ is multiplied with the conjugate of another sample spaced $N/2$ apart (i.e., $T_u/2$ seconds), and $N/2$ such multiplications are carried out and summed together, then this will yield a large value only when this starting sample coincides with the start of the training sequence. An example of such a correlation is shown in Figure 9.31 for a length $N = 4$ training sequence composed of $\{+, -, -, +\}$.

- In Figure 9.31a which illustrates a wrong location for the correlation start, assume that the two noise samples have a - and + sign, respectively. Then, the correlation result here (normalized by $1/N$) becomes

$$\frac{1}{4} \sum \left\{ (- \cdot +), (+ \cdot -), (- \cdot -), (+ \cdot +) \right\} = 0$$

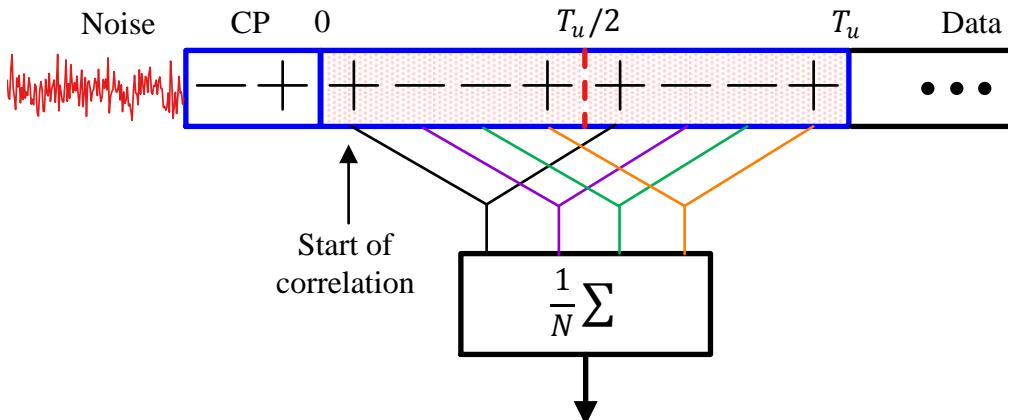
[†]There are many specialized sequences available for this purpose such as Gold codes and Kasami codes but since optimization is not our purpose, we consider a random stream of +1s and -1s here.

Training sequence with identical halves



(a) The correlation start is located before the time domain training

Training sequence with identical halves



(b) The correlation start is located at the initial sample of the time domain training

Figure 9.31: A time domain training sequence with two identical halves containing a pseudo-random sequence. The correlation is a large value only when the correlation start coincides with the initial sample of the training

One could argue that the noise samples could have opposite signs and hence generate a large value. But remember that we are considering an example for a very small $N = 4$. For a large value of N , it is almost improbable that so many noise samples have exactly the same signs in succession as the training sequence. Furthermore, the example is taken near the training sequence where the last two values in the window are already the same due to the presence of a CP. This partial similarity gives rise to a gradually increasing slope of the

correlation result, instead of a sharp peak.

- Next, we shift this correlation window by 1 sample to the right and compute the $N/2$ -spaced correlation among the samples falling within this window. And continue repeating this process.
- When this correlation window reaches the actual frame start, i.e., the initial sample of the training sequence, the following result is produced.

$$\frac{1}{4} \sum \left\{ (+ \cdot +), (- \cdot -), (- \cdot -), (+ \cdot +) \right\} = 1$$

which corresponds to the exact alignment of the length $N/2$ training sequence. Connect this result with Figure 9.30 illustrating $\delta[n]$ as its autocorrelation.

The Coarse Timing Metric

To form such a signal, we construct a single OFDM symbol that consists of the length $N/2$ iFFT of a pseudo-random sequence, i.e., the training sequence is in frequency domain now and consists of complex samples. Then, we repeat it twice in time domain which makes the second half of this OFDM symbol the same as the first half. This forms our training sequence prepended at the start of an OFDM frame consisting of two identical halves with length $N/2$ each.

Note that such a training sequence can also be generated by assigning the length $N/2$ sequence to even subcarriers and zeros to odd subcarriers. This is equivalent to upsampling by 2 the length $N/2$ sequence in frequency domain. Therefore, its length N FFT is composed of two copies in the time domain, just like upsampling by P in time domain consists of P spectral replicas in frequency domain.

After the addition of a CP, the convolution of the channel will affect both halves of the training sequence in a similar manner. The samples received are denoted as $z[n]$ and let us compute a correlation in this sequence in the following manner. Utilizing the definition of correlation, we define the index ε_0 as the sample where the correlation starts and a metric $P(\varepsilon_0)$ based on this index as

$$P(\varepsilon_0) = \sum_{i=0}^{N/2-1} z \left[\left(\varepsilon_0 + \frac{N}{2} \right) + i \right] z^* [\varepsilon_0 + i] \quad (9.20)$$

This correlation involves a window of N samples starting at ε_0 that slides to the right while searching for the training symbol. The conjugate operation is necessary to rotate the length $T_u/2$ subcarriers in the first half in an opposite direction to cancel them out and add the signs of the pseudo-random sequence in a coherent manner. This is drawn in Figure 9.32 where a length N correlation window is also visible for the ideal correlation start. Although this could be seen from plugging the value of $z[n]$ from Eq (9.18), we avoid this route to keep the discussion simple.

There are two factors that can cause a wrong timing offset if only $P(\varepsilon_0)$ is used for this purpose.

Phase/frequency rotation: Recalling that OFDM is a passband and hence complex modulation scheme and the subcarriers are composed of both I and Q components, $P(\varepsilon_0)$ in Eq (9.20) above consists of complex samples. How should we compare this metric for different values of ε_0 , through the I part, Q part or the

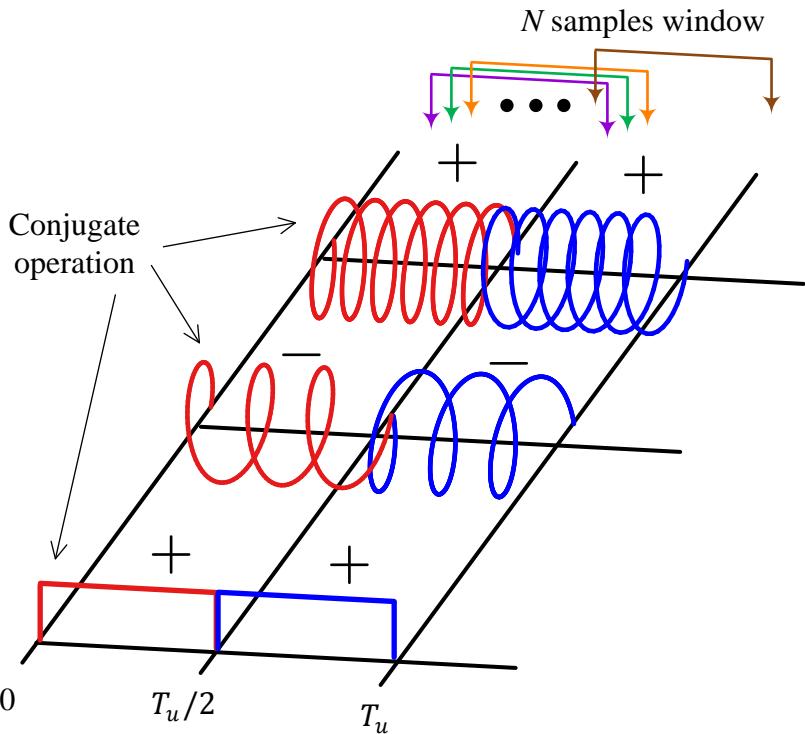


Figure 9.32: Computation of the metric $P(\varepsilon_0)$ involves a correlation sum at a time difference of $T_u/2$ seconds. Notice the opposite rotation of each subcarrier in the first half (after scaling by + or - sign)

magnitude? If there is a phase rotation between the two identical halves of the training symbol, the inphase part will not produce a correct amount of correlation. For a worst case phase offset of $\pi/2$ between them, a zero appears at the I output while all the correlation magnitude appears in the Q output. Similar is the case when a frequency offset is present in the Rx signal that spins the modulation symbols and makes the two identical halves non-identical. Observe that if this was not the case, then the best method to acquire the timing is to correlate the incoming signal (in which the training is embedded) with a locally stored copy of the training at the Rx which is free of noise and other distortions. The presence of a frequency offset ruins the correlation of two otherwise identical sequences. One simple remedy for this problem is to focus on the magnitude squared of the metric $P(\varepsilon_0)$.

Signal energy: Since the channel is not AWGN, the Rx signal energy can vary throughout the duration of the transmission. Consider a case where the energy is high in the first half and low in the second half. Consequently, the correct peak will exhibit a low value and hence can be easily missed even for the correct ε_0 . The solution here is to normalize the metric $P(\varepsilon_0)$ with the energy of the samples within the correlation window. However, each sequence in the correlation consists of $N/2$ samples, so only $N/2$ samples need to be taken into account.

While energy in any of the two halves can be measured, the second half is usually chosen as a precaution against an unexpectedly long channel.

$$E(\varepsilon_0) = \sum_{i=0}^{N/2-1} \left| z \left[\left(\varepsilon_0 + \frac{N}{2} \right) + i \right] \right|^2$$

Taking into account the above two factors, a new timing synchronization metric can be defined as

$$M(\varepsilon_0) = \frac{|P(\varepsilon_0)|^2}{[E(\varepsilon_0)]^2} \quad (9.21)$$

And the optimal timing instant $\hat{\varepsilon}_0$ (recall that timing synchronization in OFDM corresponds to locating the correct sample, i.e., an integer offset) is given as

$$\hat{\varepsilon}_0 = \max_{\varepsilon_0} M(\varepsilon_0) \quad (9.22)$$

Here, the denominator $E(\varepsilon_0)$ can be considered as a virtual AGC to provide a suitable signal gain for accurate timing metric detection. A block diagram of such an implementation is illustrated in Figure 9.33. We will later come back to this scheme to find out how it helps in coarse CFO correction.

Recursive Implementation

The main attraction of such a strategy is the correct result even in the presence of the CFO due to the magnitude squared operations involved. Furthermore, from a hardware point of view, such a timing metric can be implemented in a recursive manner thus simplifying its implementation. For this purpose, write $P(\varepsilon_0 + 1)$ from Eq (9.20) as

$$P(\varepsilon_0 + 1) = \sum_{i=0}^{N/2-1} z \left[\left(\overline{\varepsilon_0 + 1} + \frac{N}{2} \right) + i \right] z^* [\varepsilon_0 + 1 + i]$$

This is the same correlation window as computed for ε_0 but shifted by 1 sample to the right. Therefore, the leftmost sample (corresponding to $i = 0$) can be excluded from $P(\varepsilon_0)$ and an additional sample from the right (corresponding to $i = N/2$) can be added to $P(\varepsilon_0)$ to get $P(\varepsilon_0 + 1)$.

$$P(\varepsilon_0 + 1) = \underbrace{\sum_{i=0}^{N/2-1} z \left[\left(\varepsilon_0 + \frac{N}{2} \right) + i \right] z^* [\varepsilon_0 + i]}_{P(\varepsilon_0)} - z \left[\varepsilon_0 + \frac{N}{2} \right] z^* [\varepsilon_0] + z [\varepsilon_0 + N] z^* \left[\varepsilon_0 + \frac{N}{2} \right]$$

Similarly, the denominator $E(\varepsilon_0)$ can be recursively computed as

$$E(\varepsilon_0 + 1) = E(\varepsilon_0) - \left| z \left[\varepsilon_0 + \frac{N}{2} \right] \right|^2 + |z [\varepsilon_0 + N]|^2 \quad (9.23)$$

For these reasons, such a correlation window is implemented in most of the OFDM receivers for timing synchronization purpose.

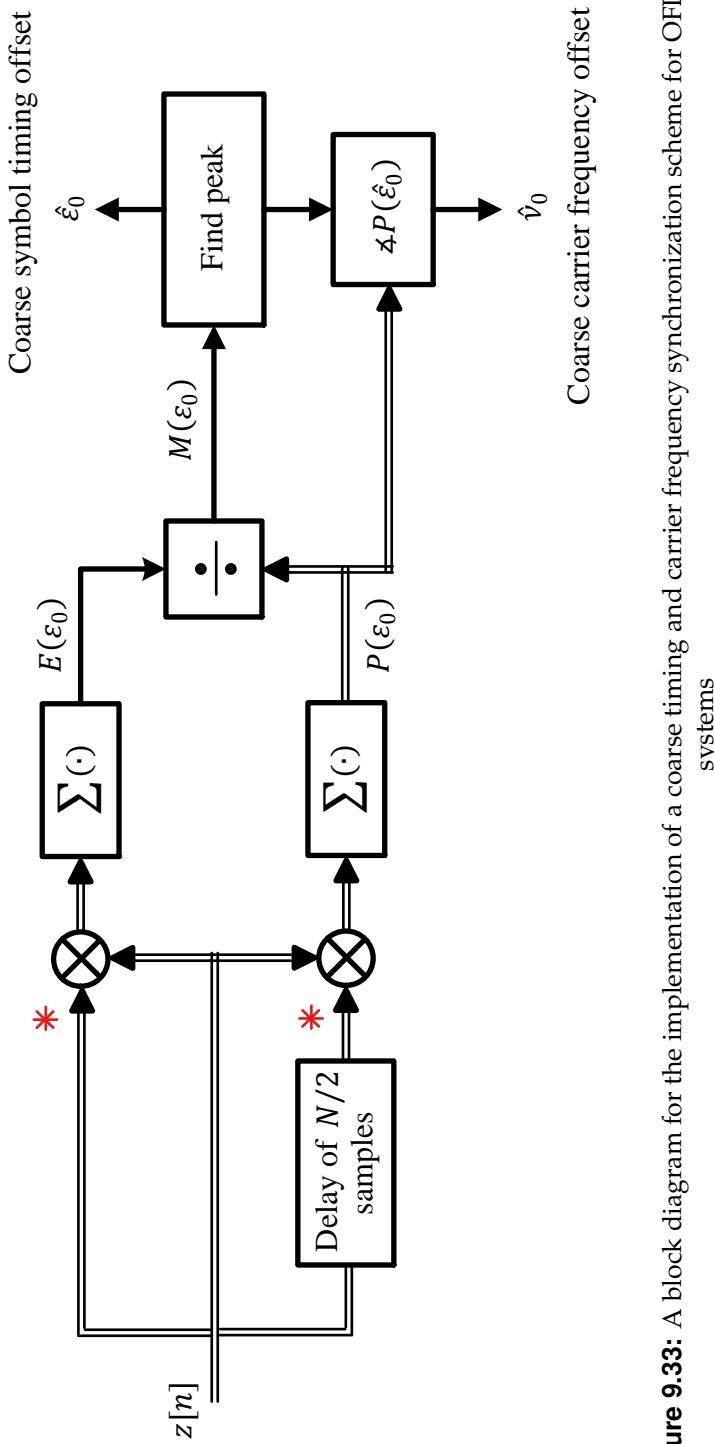


Figure 9.33: A block diagram for the implementation of a coarse timing and carrier frequency synchronization scheme for OFDM systems

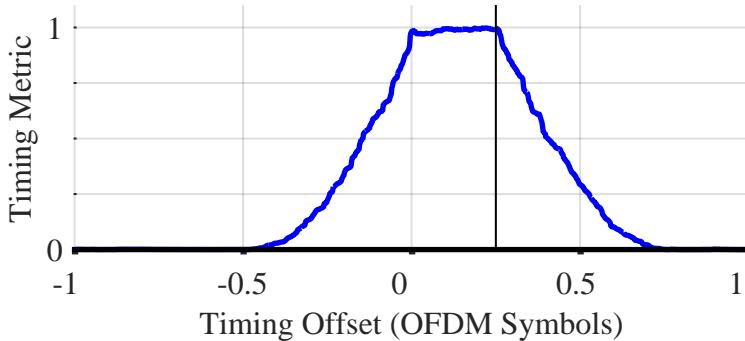


Figure 9.34: The timing metric $M(\varepsilon_0)$ of Eq (9.21) drawn for $N = 512$ subcarriers with a 25% CP, an AWGN channel and a random training sequence

Improvements on the Timing Scheme

We have seen above an implementation of a coarse timing estimation scheme through autocorrelation, i.e., correlation between the samples of the same sequence. When such a timing synchronization scheme is implemented for an OFDM frame in which there is a training sequence prepended at the start, we get the timing metric $M(\varepsilon_0)$ of Eq (9.21) drawn in Figure 9.34 for $N = 512$ subcarriers, a 25% CP, a random training sequence and an AWGN channel. Notice that as the correlation window slides past the OFDM signal, the timing metric gradually rises near the training, reaches a plateau when the window hits the CP, remains constant for the duration of the CP (25%) and then slides down gradually after the training begins.

Since this is an AWGN channel, there is no ISI within this plateau to distort the signal and the first sample can be taken anywhere within this region. This was shown earlier in Figure 9.28 as a region of no concern. In a frequency selective channel, the length of this plateau is actually given as

$$\text{Plateau length} = \text{CP length} - \text{channel length}$$

In an AWGN channel, the plateau is as long as the CP.

The next question is how to utilize the timing metric $M(\varepsilon_0)$ for synchronization purpose. The most straightforward way is to raise a flag of preamble detection whenever $M(\varepsilon_0)$ rises past a certain threshold. However, the metric possesses a flat timing plateau instead of exhibiting a sharp peak. Moreover, due to this flat plateau, the variance of the timing synchronization estimate is quite high. For reliable operations and exact analysis, we like to have a repeatable experience with as less a variance as possible.

Many variations on this timing synchronization scheme have been proposed, some of which we discuss next.

Peak detection through 90% points: While a naive solution is to find the maximum point $\varepsilon_{0,\max}$ of $M(\varepsilon_0)$, this is not a reliable solution as well since it will move around in each realization due to the random noise. A better solution [44] is to detect the peak by finding this maximum point, then locate the points to the left and right of this maximum value that are 90% of this maximum value, and average the time points corresponding to these two 90% values.

Smoothing with a moving average filter: With the knowledge that the plateau has a maximum length equal to the CP in an AWGN channel, the flatness of this region can be turned into a linearly rising curve by convolving it with a moving average filter having a length equal that of the CP. The rationale is that a flat signal resembles a rectangular signal and convolution of a rectangle with another rectangle in the moving average filter is a triangle that exhibits a relatively pronounced peak.

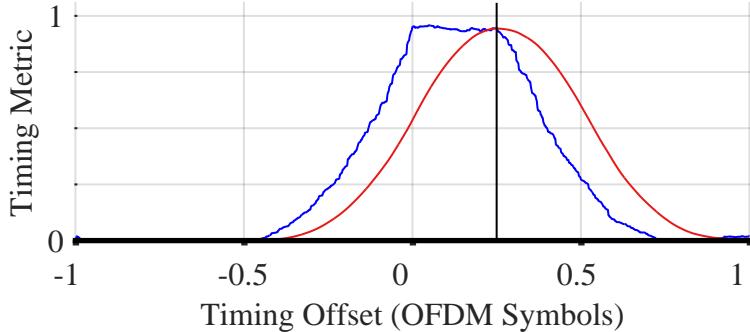


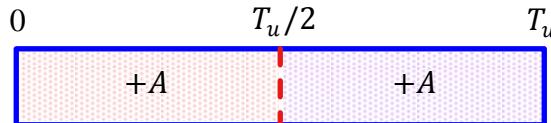
Figure 9.35: The timing metric $M(\epsilon_0)$ of Figure 9.34 convolved with a moving average filter of length equal to the CP length

Such a modified metric is plotted in Figure 9.35 where the single point of the sequence start can be isolated. Note that the convolution result is not a triangle because the plateau is not a rectangular signal in isolation. In any case, with the peak of the timing metric $M(\epsilon_0)$ in hand, many peak finding strategies – similar to the symbol centric and zero crossing techniques in Chapter 7 – can also be applied if a long frame of OFDM symbols is available and convergence time is not much of a concern. This happens for example in continuous transmission systems.

Repetitions with the same or different signs: Ref. [45] proposed an interesting alternative technique to improve the timing metric $M(\epsilon_0)$ in Eq (9.21). Consider the training sequence with 2 identical halves as a special case of a training sequence with L repetitions as well as different signs. For example, denote the length $N/2$ training sequence by A . Then, until now, we have worked on a sequence A repeated twice and with ++ signs, as illustrated in Figure 9.36a. By ++ sign, we do not mean the signs of the individual training symbols before the iFFT but that the whole length $N/2$ sequence A after the iFFT is multiplied with a +1 to generate $\{+A, +A\}$.

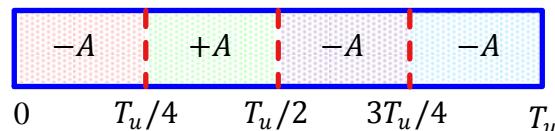
Now consider Figure 9.36b where a training sequence with $L = 4$ identical parts is drawn. While the underlying training before the length $N/4$ iFFT is the same, each iFFT output quarter is multiplied with a different pattern, $\{-A, +A, -A, -A\}$ in this case, and concatenated together. The advantage of such an approach is that even after the insertion of the CP, there is no plateau left due to the presence of *only one unique time shift* for which the whole correlation window exactly matches. The timing metric $M(\epsilon_0)$ is still given by the ratio of $P(\epsilon_0)$ and $E(\epsilon_0)$ but both of these metrics are modified to implement the summation of products between the samples of the repetitive parts. This is shown in

Training sequence with 2 identical parts
with signs ++



(a) 2 identical parts

Training sequence with 4 identical parts
with signs - + - -



(b) 4 identical parts

Figure 9.36: A training sequence with L identical parts multiplied with different signs is a generalization of 2 identical parts with ++ signs

Figure 9.37 where the logic of sign inversion responsible for the steep rolloff of the timing metric is also evident. Later, we will see how such a strategy is useful for CFO estimation as well.

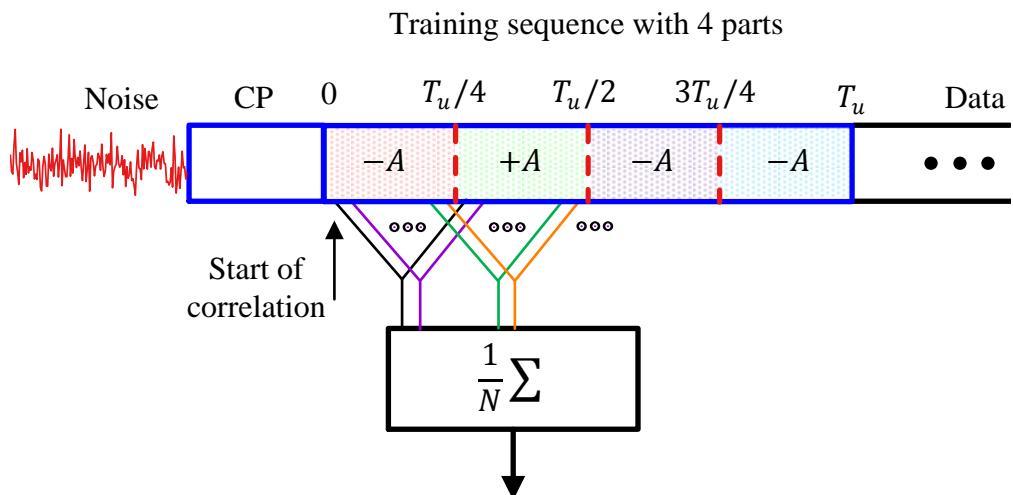


Figure 9.37: The logic behind the correlation window involving L identical parts

It must be kept in mind that during the computation of $P(\varepsilon_0)$, the actual repeat-

itive pattern (e.g., $\{-A, +A, -A, -A\}$) must be multiplied with the corresponding training part for canceling its sign. Figure 9.38 compares the timing metric $M(\varepsilon_0)$ for $L = 4$ repetitions with the pattern $\{-A, +A, -A, -A\}$ and original $L = 2$ repetitions employing the pattern $\{+A, +A\}$. It is evident that the plateau has disappeared and a steep rolloff of the timing metric helps in locating the exact timing sample for synchronization purpose. The concept demonstrated here for 4 repetitions can be easily generalized to L repetitive parts.

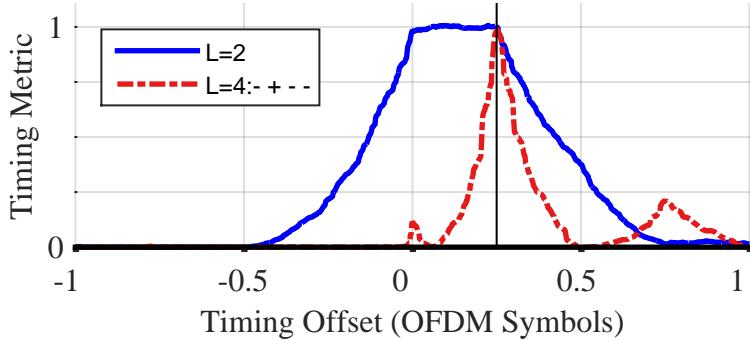


Figure 9.38: The timing metric $M(\varepsilon_0)$ for $L = 4$ repetitions with the pattern $\{-A, +A, -A, -A\}$ and original $L = 2$ repetitions with the pattern $\{+A, +A\}$

Let us now return towards the preamble of IEEE 802.11a standard discussed in Example 9.1.

Example 9.2

In 802.11a WiFi standard, the preamble consists of a *Short Training Field (STF)* with 10 repetitions and a *Long Training Field (LTF)* with 2 repetitions, as drawn in Figure 9.39 where S denotes the STF. Other than coarse timing synchronization, the STF is also utilized for several different purposes (e.g., signal detect, AGC, diversity selection and coarse frequency synchronization). The LTF consists of two identical halves used for fine frequency synchronization as well as channel estimation while a GI2 represents a guard interval of twice the normal length.

The STF in 802.11a is generated by taking an iFFT of the sequence below. Notice that only a few subcarriers from $k = -26$ to $k = +26$ are employed for this purpose which are multiples of 4, the effect of which we shortly see. From $k = -26$ to $+26$,

$$a_{\text{STF}}[k] = \sqrt{\frac{13}{6}} \left\{ 0, 0, \underbrace{+V}_{-24}, 0, 0, 0, \underbrace{-V}_{-20}, 0, 0, 0, \underbrace{+V}_{-16}, 0, 0, 0, \underbrace{-V}_{-12}, 0, 0, 0, \right. \\ \left. \underbrace{-V}_{-8}, 0, 0, 0, \underbrace{+V}_{-4}, 0, 0, 0, \underbrace{0}_{0}, 0, 0, 0, \underbrace{-V}_{+4}, 0, 0, 0, \underbrace{-V}_{+8}, 0, \right. \\ \left. 0, 0, \underbrace{+V}_{+12}, 0, 0, 0, \underbrace{+V}_{+16}, 0, 0, 0, \underbrace{+V}_{+20}, 0, 0, 0, \underbrace{+V}_{+24}, 0, 0 \right\}$$

where V is a complex number given by

$$\begin{array}{ll} I & \rightarrow & V_I = 1 \\ Q & \uparrow & V_Q = 1 \end{array}$$

The scaling factor $\sqrt{13/6}$ comes from normalizing the average power of the OFDM symbol which employs 12 out of 52 available subcarriers. Combined with $\sqrt{1/2}$ that is used to make V above a unit energy number,

$$\sqrt{\frac{1}{2} \cdot \frac{52}{12}} = \sqrt{\frac{13}{6}}$$

What is the implication of only the subcarriers in STF being multiples of 4 utilized for modulation purpose? This can be seen as a case of upsampling a signal consisting of 13 values by 4. From Section 2.7.2, we know that upsampling by inserting L zeros in one domain causes a repetition by L in the other domain. In frequency domain, this repetition was visible in the form of the L spectra becoming a part of the upsampled signal spectrum. Here, the zero insertion is in frequency domain and hence $L = 4$ repetitions appear in time domain. The time duration of the iFFT output is the FFT size N multiplied with the sample time $T_S = 1/20 \times 10^6 = 50$ ns. Thus, the time period of one repetition consisting of $64/4 = 16$ samples is given by

$$\frac{NT_S}{4} = \frac{64 \times 50 \times 10^{-9}}{4} = 0.8\mu\text{s}$$

and the total duration of the STF field is $10 \times 0.8 = 8\mu\text{s}$. As another example, the LTF is generated through the iFFT of the following symbols.

Observe that all 52 subcarriers are utilized. There are numerous other metrics and even more variations on those metrics that have been discovered in the enormous research literature on OFDM systems. Nevertheless, the fundamental concepts remain similar to what has been discussed above. An interested reader should study the intriguing training sequences defined in various standards implementing the OFDM systems such as WiFi (802.11a/g/n/ac), WiMax (802.16), LTE (4G), NR (4.5G) and 5G systems.

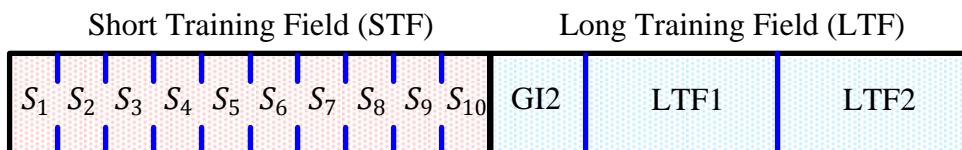


Figure 9.39: Preamble for IEEE 802.11a PHY

Once a coarse timing estimate $\hat{\varepsilon}_0$ is located, it is also customary to advance it by a predetermined number of samples as a safety margin against channel dispersion. Recall from the discussion on the effect of STO that taking the FFT between the ideal sampling instant and the point where the channel ends has no effect on the demodulator performance because any residual phase becomes part of the channel phase $\angle H[k]$ for each k . This safety margin introduces a cushion in synchronization design where an advance of a few samples can be afforded but not a delay of even 1 sample.

As a final remark, we have established the necessity of some kind of periodicity in the Rx signal to mark the timing synchronization instant. For OFDM systems consisting of long packets or continuous transmissions such as DAB or DVB, the timing synchronization can be accomplished through using the cyclic prefix which is essentially a short repetition in an OFDM symbol. Nevertheless, the underlying concepts related to building useful metrics and defining thresholds remain unchanged. For example, the timing metric using a CP can be written as

$$P_{CP}(\varepsilon_0) = \sum_{i=-N_{CP}}^{-1} z[(\varepsilon_0 + N) + i] z^* [\varepsilon_0 + i] \quad (9.24)$$

where the repetition happens at a time difference of N samples and gives rise to a high correlation. The other two metrics, namely $R(\varepsilon_0)$ and $M(\varepsilon_0)$, can also be defined in a similar manner. Keep in mind that for one length N OFDM symbol, the sample indices including the CP are defined as follows.

$$-N_{CP}, \dots, -1, 0, 1, 2, \dots, N-1$$

Fine Timing Synchronization

The coarse timing offset above was computed through using the autocorrelation, i.e., correlation between the samples of the same sequence. Advantages of this approach are its computational complexity due to recursive structure in Eq (9.23) and that it is immune to any impairment caused by the CFO. Nonetheless, the disadvantage of such an approach is that both sets of samples are noisy and hence the performance remains suboptimal.

One remedy to this problem is that we already have a ‘clean’ copy of the training sequence stored at the Rx. Why not correlate the noisy Rx signal with the stored clean sequence? Recall that the noisy version of the training is affected by the CFO and hence even its correlation with the clean sequence will be rotating instead of generating a single sharp peak.

To solve this issue, a coarse frequency offset is also computed after the coarse timing estimation and compensated for which we discuss in the next section. Once the noisy Rx sequence is mostly free of the coarse CFO, we denote it by $\tilde{z}[n]$ and a fine timing estimate is computed by cross-correlating it with the clean training sequence $s[n]$ stored at the Rx. The new metric $\tilde{M}(\varepsilon_0)$ is given by

$$\tilde{M}(\varepsilon_0) = \sum_{n=0}^{N-1} \tilde{z}[n + \varepsilon_0] s^*[n]$$

Such a strategy is also known as matched filtering because the correlation is implemented with the known training sequence $s[n]$. In practice, such a cross-correlation produces highly accurate results. To reduce the acquisition time, this operation can be carried out partially with a portion of $s[n]$ as well. For IEEE 802.11a example, after the coarse timing computation from the Short Training Field (STF), a partial correlation with 32 or 64 samples of the Long Training Field (LTF) can be carried out instead of the complete length 128 sequence for a fine timing estimate.

On the other hand, in the continuous or frame-based OFDM systems, fine timing can be accomplished through averaging over a number of OFDM symbols. This is because the acquisition time requirements are relevantly much more relax as compared

to packet based systems and a simple tradeoff between extending the averaging interval for an increased accuracy is feasible.

Next, we turn our attention towards the problem of CFO estimation in an OFDM system.

9.5 Carrier Frequency Synchronization

In OFDM systems, the Carrier Frequency Offset (CFO) is also corrected in multiple stages, a coarse and a fine stage. Before the synchronization procedures, we need to understand how a CFO impacts the demodulated symbols.

Effect of Carrier Frequency Mismatch

Just like a single-carrier signal, the baseband OFDM signal at the Tx is upconverted by a local oscillator to a carrier frequency F_C and downconverted at the Rx by its local oscillator at a frequency slightly different than F_C . Along with any Doppler shift encountered in the channel path, this imparts a Carrier Frequency Offset (CFO) equal to F_Δ in the Rx signal[†].

As we saw before in the discussion on the timing mismatch, we can divide the CFO F_Δ into an integer and a fractional part.

Integer CFO: What happens in a single-carrier system when a symbol timing offset is an integer multiple of symbol time, say $2T_M$, away from the ideal instant? It would not induce any ISI but our symbol numbers will become wrong. For an STO of $2T_M$, our symbol 0 would appear at symbol instant 2 and a similar shift will appear for the rest of the symbols.

Exactly in a similar way, and by virtue of time frequency duality, an integer CFO will not induce any ICI in the Rx signal and the orthogonality among the subcarriers is maintained. However, the symbols after the DFT will appear at wrong locations and an absence of ICI does not indicate a successful transmission. These wrong symbol indices either cause a signal decoder failure or failed header checks in a higher layer.

Fractional CFO: Similar to a fractional STO in a single-carrier system, a fractional CFO would ‘sample’ the Rx waveform in frequency domain at a shifted location thus inducing ICI. This is plotted in Figure 9.40 for a 25% CFO where the subcarrier demodulation is not being performed at optimal instants. Compare this frequency domain ICI with the time domain ISI in Figure 7.2.

On the same note, a scatter plot isolating the CFO distortion in OFDM resembles the scatter plot isolating the STO distortion in single-carrier systems and drawn in Figure 7.4 where the demodulated symbols form a cloud around the ideal constellation points. A little difference can arise from the presence of a CP that causes a phase rotation of those clouds away from those constellation points.

[†]Here, we ignore another major source of carrier distortion, namely the phase noise.

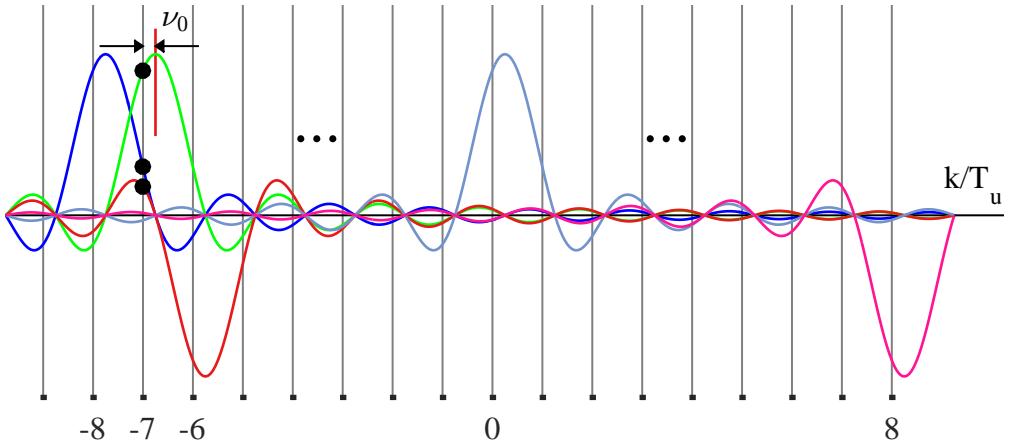


Figure 9.40: A fractional Carrier Frequency Offset (CFO), shown here as 25% of subcarrier spacing $\Delta_F = 1/T_u$ causing ICI in the neighbouring symbols

Coarse Frequency Synchronization

First, recall Section 6.4.1 on a delay and multiply technique to estimate the CFO in a feedforward manner. The mathematical details of this section were elaborated in the context of a single-carrier system. A similar method can be applied to understand coarse frequency synchronization in an OFDM system. The main idea is that the frequency of any complex sinusoid can be estimated by taking a phase difference between any of its two samples at known time instants, say T_2 and T_1 . This is because by definition, the phase difference is given by

$$\Delta\phi = 2\pi F(T_2 - T_1)$$

Since T_2 and T_1 are known and $\Delta\phi$ is estimated through some method, the underlying frequency is

$$F = \frac{\Delta\phi}{2\pi(T_2 - T_1)} \quad (9.25)$$

The good thing about OFDM is that *we have N such complex sinusoids (embedded in noise) already available.*

For this purpose, consider Figure 9.41 which illustrates a single subcarrier (out of N) with index $k = 2$ that is rotating at a frequency of $2/N$ cycles/sample (i.e., it completes 2 cycles in N samples, or T_u seconds). The first cycle is shown in a solid blue line in the first half of the OFDM symbol while the second cycle in a dashed red line in the second half of the OFDM symbol.

Also shown in a dotted line is another complex sinusoid with a slightly different frequency that is the same subcarrier shifted by a normalized CFO ν_0 . The actual subcarrier frequency k/N is known but the Rx signal contains N such subcarriers shifted in frequency by ν_0 and our task is to estimate this unknown CFO ν_0 . If we implement a conjugate product of the actual subcarrier at k/N and this Rx subcarrier at frequency $(k + \nu_0)/N$, then we will be left with a complex sinusoid rotating at ν_0/N .

We already have the relevant samples $z[n]$ in the form of the Rx OFDM signal, so the only question before we can apply Eq (9.25) is that which two time instants T_2 and T_1 should be chosen? We will later see that this choice establishes the acquisition

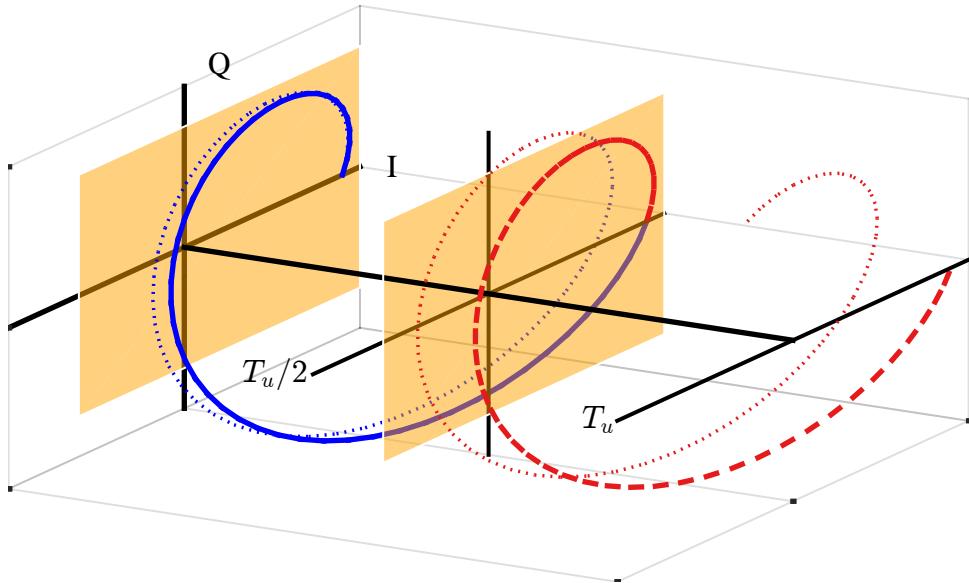


Figure 9.41: Phase difference of a complex sinusoid at two time instants gives its frequency

range of the CFO estimation technique. If we recall the definition of the metric $P(\varepsilon_0)$ from Eq (9.20), we find that the phase difference between two samples has already been computed during the timing synchronization procedure if

$$T_2 = T_u/2, \quad T_1 = 0$$

Not only that, but computation of $P(\varepsilon_0)$ (reproduced below) involves a summation of $N/2$ such phase differences, the next between $T_2 = T_u/2 + 1$ and $T_1 = 1$, and so on until it reaches $T_2 = T_u$ and $T_1 = T_u/2$.

$$P(\varepsilon_0) = \sum_{i=0}^{N/2-1} z \left[\left(\varepsilon_0 + \frac{N}{2} \right) + i \right] z^* [\varepsilon_0 + i]$$

In all these terms, the time difference is equal to $T_u/2$ while the summation is simply a moving average filter. As shown previously in Figure 9.32, the conjugate operation rotates the length $T_u/2$ subcarriers in the first half in an opposite direction to cancel them out as well as the modulation signs of the pseudo-random sequence. As a result of this procedure, it produces the phase differences at each sample of that window equal to

$$\Delta\phi = 2\pi F_\Delta \frac{T_u}{2} = \angle P(\varepsilon_0)$$

Using the definition of normalized frequency offset $\nu_0 = F_\Delta/\Delta_F = F_\Delta T_u$, the above

equation yields

$$\begin{aligned}\hat{\nu}_0 &= \frac{1}{\pi} \angle P(\hat{\varepsilon}_0) \\ &= \frac{1}{\pi} \angle \left\{ \sum_{i=0}^{N/2-1} z \left[\left(\hat{\varepsilon}_0 + \frac{N}{2} \right) + i \right] z^* [\hat{\varepsilon}_0 + i] \right\}\end{aligned}\quad (9.26)$$

where $\hat{\varepsilon}_0$ is the coarse timing offset that marks the beginning of the training sequence. A block diagram of such a scheme to estimate the coarse frequency offset $\hat{\nu}_0$ along with the timing offset $\hat{\varepsilon}_0$ was drawn in Figure 9.33. This is the most widely used coarse frequency synchronization method in OFDM systems due to its simplicity and robustness.

Note that if the CP was used for the coarse timing synchronization procedure, then from Eq (9.24), we get

$$\hat{\nu}_0 = \frac{1}{2\pi} \angle P_{CP}(\hat{\varepsilon}_0) = \frac{1}{2\pi} \angle \left\{ \sum_{i=-N_{CP}}^{-1} z [(\hat{\varepsilon}_0 + N) + i] z^* [\hat{\varepsilon}_0 + i] \right\}$$

The factor 2π instead of π in the denominator arises due to N sample spacing between the two correlation participants, instead of $N/2$ in the case of the training sequence with two identical halves. Furthermore, in OFDM systems with continuous transmissions, the imaginary part of the above expression can be used as a frequency error detector in a frequency locked loop.

Acquisition Range

One drawback of the above coarse frequency synchronization technique is its narrow acquisition range. First, recall the definition of phase $\Delta\phi = 2\pi F(T_2 - T_1)$ and the uniqueness of phase within $\{-\pi, +\pi\}$. Then, for $T_2 - T_1 = T_u/2$, we get the maximum frequency $|\nu_{0,\max}|$ that can be uniquely determined.

$$2\pi |F_{\Delta,\max}| \frac{T_u}{2} < \pi, \quad \text{or} \quad |F_{\Delta,\max}| < \frac{1}{T_u} = \Delta_F$$

$$|\nu_{0,\max}| < 1 \quad (9.27)$$

where $\nu_{0,\max} = F_{\Delta,\max} T_u$. There will be an ambiguity when the CFO F_Δ approaches **one subcarrier spacing** Δ_F either from the positive or the negative side.

Another route to understand this concept is through the sampling theorem. For samples taken at a rate of $F_S = 1/T_S$, the maximum frequency to avoid aliasing is $F_S/2 = 1/2T_S$. Here, the expression

$$z \left[\left(\hat{\varepsilon}_0 + \frac{N}{2} \right) + i \right] z^* [\hat{\varepsilon}_0 + i]$$

in $P(\varepsilon_0)$ is essentially ‘sampling’ the waveform at two time instants $N/2$ samples (or $T_u/2$ seconds) apart, see Figure 9.41 (the summation in $P(\varepsilon_0)$ is just to average several such samples to reduce the additive noise). Thus, according to the sampling

theorem, the maximum frequency before aliasing is $1/2(T_u/2) = 1/T_u = \Delta_F$. Consequently, this is the largest unique CFO F_Δ that can be determined through this technique.

What is the most straightforward method to increase the maximum allowable CFO from the perspective of the sampling theorem? This is simply to increase the sampling rate, i.e., taking the two samples closer in time to each other. In this context, we can increase the desired samples by taking the correlation at closely spaced intervals than $T_u/2$. This can be done by using a training sequence that is identical not only for two halves but for L identical parts. We saw one such scenario in the preamble of IEEE 802.11a standard in Example 9.2 where the short training field consists of 10 identical repetitions, each consisting of 16 samples. Although the FFT size in IEEE 802.11a is $N = 64$, the total length of the short (as well as long) training field is 160 samples, i.e., there are $L = 64/16 = 4$ repetitions within $N = 64$ samples.

For a general training sequence with L identical parts, the time difference between the two correlation participants is reduced from $N/2$ to N/L samples. Then, the correlation expression above can be changed to

$$z \left[\left(\hat{\varepsilon}_0 + \frac{N}{L} \right) + i \right] z^* [\hat{\varepsilon}_0 + i]$$

There are N/L such terms in the correlation window, so we can add them together to average out the noise. Moreover, when the full correlation window of N samples is taken into account, we get

$$\begin{aligned} \hat{\nu}_0 = \frac{L}{2\pi} \angle & \left\{ \sum_{i=0}^{N/L-1} z \left[\left(\hat{\varepsilon}_0 + \frac{N}{L} \right) + i \right] z^* [\hat{\varepsilon}_0 + i] + \right. \\ & \sum_{i=N/L}^{2N/L-1} z \left[\left(\hat{\varepsilon}_0 + \frac{N}{L} \right) + i \right] z^* [\hat{\varepsilon}_0 + i] + \dots \\ & \left. \sum_{i=(L-2)N/L}^{(L-1)N/L-1} z \left[\left(\hat{\varepsilon}_0 + \frac{N}{L} \right) + i \right] z^* [\hat{\varepsilon}_0 + i] \right\} \end{aligned} \quad (9.28)$$

where the factor $L/2\pi$ is a generalization of $2/2\pi = \pi$ in $L = 2$ identical parts from Eq (9.26). From the previous analysis that lead to Eq (9.27), the new acquisition range is easily seen to be

$$\begin{aligned} 2\pi |F_{\Delta,\max}| \frac{T_u}{L} < \pi, \quad \text{or} \quad |F_{\Delta,\max}| < \frac{1}{2T_u/L} = \frac{L}{2} \Delta_F \\ |\nu_{0,\max}| < \frac{L}{2} \end{aligned} \quad (9.29)$$

With this modification, the CFO acquisition range increases as L increases. On the other hand, the number of samples available for computing one correlation term gets reduced in proportion to L . Next, we turn our attention towards improving the frequency estimate through applying finer techniques.

Fine Frequency Synchronization

The preamble in many OFDM systems consists of not one but two training sequences. The first training sequence is used for coarse synchronization strategies while the second preamble is used for fine frequency synchronization and channel estimation.

We discussed in Example 9.2 that the Short Training Field (STF) in IEEE 802.11a standard consists of 10 repetitions of the same length 16 sequence. After the signal detection, AGC convergence and coarse timing synchronization, the last repetitions of the STF can be employed for a coarse frequency estimate. Next, the Long Training Field (LTF) consists of a training sequence with 2 identical halves, each 64 samples in length.

- Since the residual CFO is now small, acquisition range and ICI distortion are not significant issues anymore. Therefore, it can be compensated for by employing the same correlation strategy over these long periods.
- Since this is not a timing synchronization problem anymore, the metric in the denominator to normalize the samples energy is not required.

Assuming that by virtue of successful timing synchronization, $z_{\text{LTF}}[n]$ refers to the portion of the Rx signal corresponding to the LTF. Then, we estimate $\hat{\nu}_0$ as

$$\hat{\nu}_0 = \frac{1}{2\pi} \angle \left\{ \sum_{n=0}^{64-1} z_{\text{LTF}}[64+n] z_{\text{LTF}}^*[n] \right\}$$

There is no other factor with 2π because the normalized CFO is defined with respect to T_u which corresponds to $N = 64$ samples in IEEE 802.11a. This is the same as the length of the LTF and hence $T_2 - T_1$ in Eq (9.25) is T_u as well.

Since the conjugate operation cancels the data modulation, the resulting signal turns out to be a complex sinusoid of frequency F_Δ embedded in noise. Notice that there is only one correlation being computed at a certain lag. We can also follow a strategy similar to the estimators in Section 6.3.4 that compute multiple correlations at all possible lags and subsequently generate a much accurate CFO estimate. Algorithms solving such a problem can then certainly be applied in the context of OFDM as well. Nevertheless, it defeats the purpose of fast synchronization manifested by a block processing configuration in general and OFDM systems in particular.

Finally, as far as the *integer CFO correction* is concerned, it is usually corrected by taking the DFT after completing the timing and frequency synchronization procedures. Then, the demodulated subcarriers in a training sequence or embedded pilots within an OFDM symbol are compared with the expected locations of those subcarriers in the Rx signal. If the integer CFO is zero, the training or pilots in the Rx signal are present at the correct locations. Otherwise, the shift in the pattern can be found by examining the Rx signal. Let us find out how.

Until now, we ignored the OFDM symbol index m in our expressions to avoid index cluttering. Now this index needs to be introduced to compute the integer CFO $[\nu_0]_{\text{int}}$. As described before, when an integer CFO $[\nu_0]_{\text{int}}$ is present, each subcarrier k appears at a location $k + [\nu_0]_{\text{int}}$, including the pilot subcarriers. This spectral shift can be found by exploiting the following facts about the Tx pilots.

- They remain the same for each OFDM symbol $m - 1, m$, and so on, i.e.

$$a_{m-1}[k] = a_m[k] \quad \text{if } k \text{ is a pilot subcarrier}$$

- They are boosted in power by a factor of B^2 .

Let us denote this set of pilot indices by k_p . Recall from Example 9.1 regarding an IEEE 802.11a system that the set $k_p = \{-21, -7, +7, +21\}$ is used for inserting the

pilot subcarriers drawn in Figure 9.22. This implies that correlating the samples of the FFT output for two consecutive OFDM symbols $m - 1$ and m and each subcarrier k_P yields

$$\text{corr}_0[m] = \sum_{k_P} Z_m[k_P] \cdot Z_{m-1}^*[k_P] \quad (9.30)$$

Assuming a slowly varying channel, the channel coefficients remain almost the same for these two OFDM symbols.

$$H_{m-1}[k] \approx H_m[k]$$

Using $Z[k] = a[k]H[k]$ and ignoring other phase distortions due to the oncoming absolute value operation of the correlation metric, the correlation output thus becomes

$$\begin{aligned} |\text{corr}_0[m]| &= \left| \sum_{k_P} a_m[k_P] H_m[k_P] \cdot a_{m-1}^*[k_P] H_{m-1}^*[k_P] \right| \\ &= \begin{cases} B^2 \sum_{k_P} \left\{ |a_m[k_P]|^2 \cdot |H_m[k_P]|^2 \right\} & [\nu_0]_{\text{int}} = 0 \\ \left| \sum_{k_P} \left\{ a_m[k_P] a_{m-1}[k_P] \cdot |H_m[k_P]|^2 \right\} \right| & [\nu_0]_{\text{int}} \neq 0 \end{cases} \end{aligned}$$

Notice that if $[\nu_0]_{\text{int}} = 0$, k_P corresponds to the true pilots and the output magnitude is proportional to $B^2 |a_m[k_P]|^2$ which is a large number. On the other hand, if $[\nu_0]_{\text{int}} \neq 0$, the correlation output is proportional to $\sum_{k_P} a_m[k_P] a_{m-1}[k_P]$ which is a small number owing to the data randomness and different data modulation symbols for each m [†]. This is drawn in Figure 9.42 for $[\nu_0]_{\text{int}} = -2$ where you can observe the correlation sum being performed at shift $I = 0$. Due to the random data, this summation turns out to be a small value.

Next, a similar procedure is carried out for the set of indices $k_P - 1$ to compute $|\text{corr}_{-1}[m]|$, indices $k_P + 1$ to compute $|\text{corr}_{+1}[m]|$, and so on, up to a prescribed limit to which the integer CFO can vary. The magnitude of the largest such correlation $|\text{corr}_I[m]|$ gives the estimate $[\hat{\nu}_0]_{\text{int}}$.

$[\hat{\nu}_0]_{\text{int}} \Rightarrow \max_I |\text{corr}_I[m]| \quad (9.31)$

This procedure is usually known as post-FFT synchronization due to its implementation after the FFT block. In a similar fashion, the previous approaches are known as pre-FFT synchronization. Both time domain (pre-FFT) and frequency domain (post-FFT) synchronization strategies are crucial for correct data symbols demodulation. Target specifications and existing constraints guide the system designers to choose a combination of appropriate approaches.

[†]Recall Eq (7.77) on data randomness, reproduced below.

$$\text{Mean}\{a[m] \cdot a[i]\} = \begin{cases} A^2 & i = m \\ 0 & \text{otherwise} \end{cases}$$

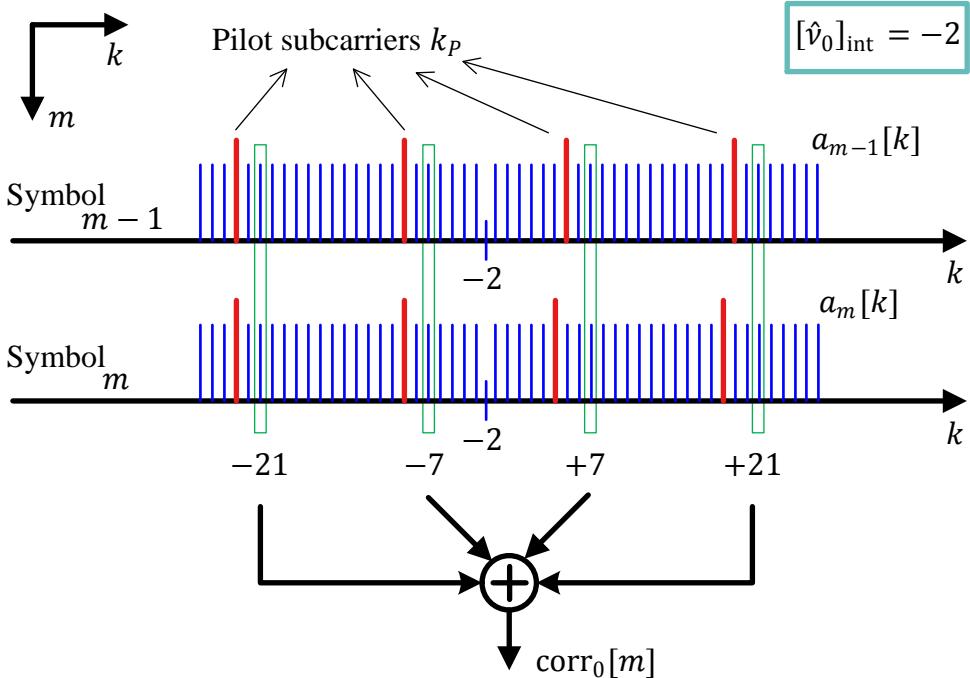


Figure 9.42: Estimation of an integer CFO $[\nu_0]_{\text{int}} = -2$ through pilot subcarriers

Note 9.4 Timing and frequency synchronization

In a study of single-carrier systems, you will find a large number of solutions for timing synchronization, far exceeding those on frequency synchronization. The situation in multicarrier systems is its dual and we find many more algorithms on carrier frequency rather than timing synchronization. This seems a bit puzzling because a single-carrier system will simply not work if even a small uncompensated frequency offset is present in the signal that steadily spins the Rx constellation. On the other hand, a small timing offset degrades the SNR by spreading the Rx constellation but the system will still keep delivering many correct bits.

This is exactly why timing synchronization in single-carrier systems is more important. Once we solve the problems that could completely stop the show, we want to iron out the last bit of inefficiency that remains. So even if the timing is *almost* right, 1 error in 100 bits is too bad and 1 error in 1 million bits is barely good enough. Probably this is the same reason why many of the problems for those living in an affluent part of a city are almost non-existent for those living in an impoverished one.

Now we explain a sampling clock synchronization strategy that simultaneously corrects for any residual CFO as well and hence can be categorized as fine tracking.

9.6 Sampling Clock Synchronization

The nature of a Sampling Clock Offset (SCO), i.e., a difference in sampling clock frequency and also known as a symbol timing frequency offset, has been explained

in Section 7.10 before. It was also visually demonstrated through Figure 7.58 the meaning of a sampling clock offset between the Tx and Rx waveforms. Here, our purpose is to devise a strategy to correct this error in OFDM systems.

We start with the effect of an SCO on the Rx signal.

Effect of Timing/Clock Frequency Mismatch

We derived Eq (9.18) to understand how the normalized CFO and the normalized STO affect the FFT input at the Rx. Now assuming a successful timing synchronization $\varepsilon_0 = 0$, we study the effect of the residual normalized CFO and a normalized Sampling Clock Offset (nSCO). One question that arises here is that why a perfect CFO cannot be assumed in the system model like the STO. Recall from the timing synchronization problem in single-carrier systems that timing needs to be continuously adjusted and can almost never be set and done. Similar is the case with the frequency offset in an OFDM systems and a residual CFO still needs to be included.

For a Rx sampling the waveform at sampling intervals of \tilde{T}_S instead of T_S , an SCO and its normalized versions are defined as

$$\xi = \tilde{T}_S - T_S, \quad \xi_0 = \frac{\tilde{T}_S - T_S}{T_S} = \frac{\xi}{T_S}$$

Next, taking the same route used to derive Eq (9.17) for $\varepsilon_\Delta = \varepsilon_0 = 0$ but sampling at $t = n\tilde{T}_S$, we get

$$\begin{aligned} 2\pi(F_k + F_\Delta)t|_{t=n\tilde{T}_S} &= 2\pi(F_k + F_\Delta)n\tilde{T}_S \\ &= 2\pi(F_k + F_\Delta)(n\tilde{T}_S + nT_S - nT_S) \\ &= 2\pi(F_k + F_\Delta)\left(nT_S + n\frac{\tilde{T}_S - T_S}{T_S}T_S\right) \\ &= 2\pi(F_k + F_\Delta)(n + n\xi_0)T_S \\ &= 2\pi\frac{k\Delta_F + F_\Delta}{F_S}(1 + \xi_0)n \\ &= \frac{2\pi}{N}(k + \nu_0)(1 + \xi_0)n \end{aligned}$$

where we have used $F_k = k\Delta_F$, $F_S = N\Delta_F$ and $F_\Delta/\Delta_F = \nu_0$. In the presence of the CFO and STO, the argument of the subcarriers in the FFT input of Eq (9.18) was derived in Eq (9.17). In the current settings of an SCO and zero STO, this argument becomes

$$\begin{aligned} 2\pi(F_k + F_\Delta)t|_{t=n\tilde{T}_S} &= \frac{2\pi}{N}(k + \nu_0)(1 + \xi_0)n \\ &= \frac{2\pi}{N}\left\{\underbrace{kn}_{\text{Term 1}} + \underbrace{\nu_0(1 + \xi_0)n}_{\text{Term 2}} + \underbrace{(k\xi_0)n}_{\text{Term 3}}\right\} \end{aligned} \quad (9.32)$$

Notice that the index n is common to all the terms above, as opposed to the situation we encountered in timing offset synchronization before. As we saw in that case, these terms are handled as follows.

- From the DFT definition in Eq (1.53), the FFT at the Rx consists of subcarriers with the arguments $-kn$ which simply cancels out Term 1 above.

- Being a function of time n , term 2 appears as a CFO equal to $\nu_0(1 + \xi_0)$ at the FFT output. The part $\nu_0 \cdot \xi_0$ is in addition to the original CFO ν_0 and is very similar to the Doppler shift (actually superimposed over the CFO by a moving or drifting Rx clock).
- This leaves Term 3 as a function of both k and n . The data symbols at each subcarrier k get rotated by the product of this index k varying from $-N/2$ to $N/2 - 1$ with the SCO ξ_0 and this is what causes the constellation rotation. However, in the timing offset case, this rotation was equal to $k\varepsilon_0$ and hence the same for each OFDM symbol. Here, it is different from one OFDM symbol to the next due to the presence of the factor n and depends on the OFDM symbol number in the frame. For example, symbol 3 will experience a different amount of SCO induced rotation as compared to symbol 100. Considering the relation $N_{\text{OFDM}} = N + N_{CP}$, sample number n is equal to $mN_{\text{OFDM}} + N_{CP}$ at the start of m^{th} OFDM symbol. Plugging $n = mN_{\text{OFDM}} + N_{CP}$ in Eq (9.32) after removal of Term 1 through a DFT at the Rx, we get[†]

$$\text{Phase rotation} = \frac{2\pi}{N} \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} (mN_{\text{OFDM}} + N_{CP})$$

We can simplify our view of this effect if we only consider this phase rotation from the previous OFDM symbol to the present and denote it by $\varphi_\Delta[m]$. Assuming the same data symbols $a_{m-1}[k] = a_m[k]$ (which is the case for pilot subcarriers) and slowly varying channel $H_{m-1}[k] \approx H_m[k]$,

$$\begin{aligned} \varphi_\Delta[m] &= \frac{2\pi}{N} \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} \left\{ mN_{\text{OFDM}} + N_{CP} \right\} - \\ &\quad \frac{2\pi}{N} \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} \left\{ (m-1)N_{\text{OFDM}} + N_{CP} \right\} \\ &= \frac{2\pi}{N} \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} N_{\text{OFDM}} \\ &= 2\pi \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} \frac{N + N_{CP}}{N} \end{aligned}$$

Let us denote

$$\lambda = \frac{N + N_{CP}}{N}$$

so that the above equation becomes

$$\varphi_\Delta[m] = 2\pi\lambda \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} \quad (9.33)$$

i.e., $\varphi_\Delta[m] \propto k\xi_0$. While this looks similar to the STO case where the phase rotation was equal to $k\varepsilon_0$, the difference is that $k\xi_0$ is just the rotation between two consecutive OFDM symbols and keeps increasing with time. Ref. [42] draws the data symbols rotation at a subcarrier level corresponding to $\nu_0(1 + \xi_0) + k\xi_0$ in an elegant manner which is the inspiration behind Figure 9.43. Observe that the constant frequency shift is given by $\nu_0(1 + \xi_0)$ while maximum rotation is experienced by the subcarriers at the edges of the spectrum, namely $k = -N/2$ and $k = N/2 - 1$.

[†]In this complete discussion on OFDM synchronization topics, we are ignoring extra noise arising from ICI and/or ISI for the sake of simplicity.

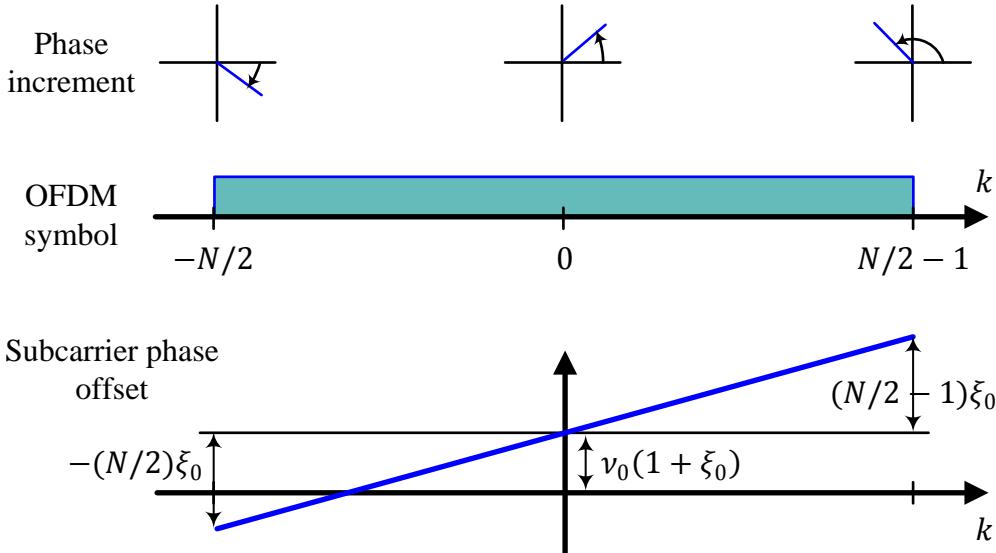


Figure 9.43: Data symbols riding on the subcarriers experience a rotation due to residual CFO and sampling frequency offset

In addition to the above, there is a sliding of the Rx FFT window due to the SCO between the Tx and Rx clocks. The details of such a window drift are similar to what was encountered in the context of single-carrier systems in Section 7.10. A similar timing locked loop can stuff or delete an extra sample while slowly correcting for the SCO. For this purpose, an SCO error detector is needed which is what we discuss below. In the documentation of QIRX software defined radio, Ref. [46] presents an excellent description of OFDM in general and a real world example of sampling clock misalignment in particular.

Joint Sampling Clock and Residual CFO Synchronization Strategy

Now we are in a position to devise an SCO compensation technique that also helps in correcting any residual CFO in the FFT output. Recall from Eq (9.30) (reproduced below) that the *correlation magnitude* between m^{th} OFDM symbol and $(m - 1)^{\text{th}}$ OFDM symbol at the FFT output was exploited to estimate the integer CFO.

$$\text{corr}[m] = \sum_{k_P} Z_m[k_P] \cdot Z_{m-1}^*[k_P]$$

where k_P denotes the set of pilot subcarriers. In the current context, we will use the *correlation phase* but in a slightly different manner.

First, notice that the phase of $\text{corr}[m]$ above for a single k_P is just the phase difference between two consecutive OFDM symbols, m and $m - 1$. Fortunately, we already have the phase for one term of this correlation from Eq (9.33) when we considered the phase rotation from one OFDM symbol to the next and denoted it by $\varphi_\Delta[m]$.

$$\varphi_\Delta[m] = 2\pi\lambda \left\{ \nu_0(1 + \xi_0) + k\xi_0 \right\} = \angle \left\{ Z_m[k_P] \cdot Z_{m-1}^*[k_P] \right\} \quad (9.34)$$

The problem here is that this is only one expression and there are two unknowns, ν_0 and ξ_0 . A single equation with two unknowns cannot have a unique solution.

To overcome this limitation, we divide the available set of pilot subcarriers in two subsets, one in the positive half of the spectrum and the other in the negative half and denote them by k_P^+ and k_P^- , respectively (where the subscript P stands for pilot). For an ease of understanding, we again consider the IEEE 802.11a standard from Example 9.1 in which there are 4 pilot subcarriers at locations $-21, -7, +7$ and $+21$. The positive and negative pilots subcarrier sets are

$$k_P^+ = \{+7, +21\}$$

$$k_P^- = \{-7, -21\}$$

We draw these subcarriers in Figure 9.44 which is an *IQ* version of Figure 9.43 that depicted a 2D plot. Observe that they are symmetrically and uniformly distributed in both halves of the spectrum around the DC subcarrier ($k = 0$).

The reason we have redrawn this figure here is to clarify one point. It seems that a CFO should shift the spectrum to the right or left as this is what usually happens. However, notice that we are not drawing the spectrum of the Rx OFDM signal $z[n]$. Instead, we are drawing the spectrum of the correlation between pilot subcarriers k_P of m^{th} OFDM symbol $Z_m[k_P]$ and $(m-1)^{th}$ OFDM symbol $Z_{m-1}[k_P]$ at the FFT output. In view of Eq (9.34), the CFO then just imparts a constant phase offset proportional to λ . If SCO $\xi_0 = 0$, then this portion of the phase offset contributed by the CFO is shown by red dotted lines in Figure 9.44 which is an equal *phase rotation* (but not a spectral shift) for all subcarriers! It is only after the addition of the SCO induced term (proportional to subcarrier index k) that the subcarriers in the subset k_P^+ rotate in the anticlockwise direction while those in the subset k_P^- rotate in the clockwise direction in proportion to λ .

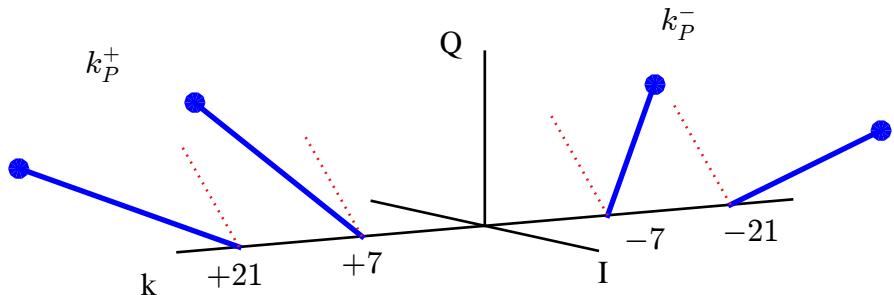


Figure 9.44: Division of pilot subcarriers k_P into two sets, one in the positive half and the other in the negative half of the spectrum. The phase of the correlation between one OFDM symbol to the next is also drawn depicting a residual CFO and a sampling frequency offset

Next, we examine the phase term of *one such subcarrier* in each half.

$$\varphi_\Delta^+[m] = 2\pi\nu_0(1 + \xi_0) + 2\pi\lambda k_P^+ \xi_0$$

$$\varphi_\Delta^-[m] = 2\pi\nu_0(1 + \xi_0) + 2\pi\lambda k_P^- \xi_0$$

Here, each positive subcarrier k_P^+ has a corresponding negative subcarrier k_P^- (for example, +7 and -7). Therefore, we can add the above two equations to cancel their second terms.

$$\varphi_{\Delta}^+[m] + \varphi_{\Delta}^-[m] = 2\pi\lambda 2\nu_0(1 + \xi_0) \quad (9.35)$$

Similarly, we can subtract the above two equations to cancel the first terms containing ν_0 .

$$\varphi_{\Delta}^+[m] - \varphi_{\Delta}^-[m] = 2\pi\lambda(k_P^+ + k_P^-)\xi_0 \quad (9.36)$$

With this understanding in place, we have two choices to average out the noise.

- We can measure the *angles* of all the positive pilot subcarriers and add them together. Similarly, we can measure the *angles* of all the negative pilot subcarriers separately and add them together.
- As an alternative, we can add the positive pilot subcarriers first (instead of their angles) and take the angle of the result later. Similarly, we can add the negative pilot subcarriers first (instead of their angles) and take the angle of the result later.

The latter strategy is clearly superior to the former because measuring the angle is a non-linear operation and hence it is better to sum the subcarriers first to average out the actual noise contribution instead of the noise angles.

So we redefine $\varphi_{\Delta}^+[m]$ and $\varphi_{\Delta}^-[m]$ to construct the following two different correlation phases from one OFDM symbol to the next.

$$\begin{aligned}\varphi_{\Delta}^+[m] &= \angle \left\{ \sum_{k_P^+} Z_m[k_P^+] \cdot Z_{m-1}^*[k_P^+] \right\} \\ \varphi_{\Delta}^-[m] &= \angle \left\{ \sum_{k_P^-} Z_m[k_P^-] \cdot Z_{m-1}^*[k_P^-] \right\}\end{aligned}$$

The final estimates of the ξ_0 and ν_0 are given as follows, see Eq (9.35) and Eq (9.36).

$$\begin{aligned}\hat{\nu}_0 &= \frac{1}{2\pi\lambda} \cdot \frac{1}{2(1 + \hat{\xi}_0)} \cdot \left\{ \varphi_{\Delta}^+[m] + \varphi_{\Delta}^-[m] \right\} \\ \hat{\xi}_0 &= \frac{1}{2\pi\lambda} \cdot \frac{1}{K/2 + 1} \cdot \left\{ \varphi_{\Delta}^+[m] - \varphi_{\Delta}^-[m] \right\}\end{aligned} \quad (9.37)$$

Here, K denotes the total number of used subcarriers which is less than the total number of subcarriers N . For example, as evident from Table 9.1, $K = 48 + 4 = 52$ in IEEE 802.11a WiFi standard for OFDM systems. It appears in the estimate $\hat{\xi}_0$ because of the summation over terms involving the subcarrier index k , either in the form of k_P^+ or k_P^- . When pilots are symmetrically and uniformly distributed around DC, their average on each side yields the desired term.

The final question is that whether a feedback or a feedforward strategy should be adopted for this purpose. With high sample rates of the modern OFDM systems, it is possible to estimate and compensate for the SCO in a feedforward manner. In fact, many receiver systems designed for OFDM estimate and compensate for the SCO on a block by block basis by taking advantage of its frame structure.

Otherwise, a feedback decision-directed based approach can also be employed in a timing locked loop where the above expressions are fed into their respective Proportional plus Integrator loop filters. Remember that until now for the case of pilot subcarriers $a_{m-1}[k] = a_m[k]$ and furthermore, we assumed $H_{m-1}[k] \approx H_m[k]$. In the decision-directed case, the first assumption does not hold true and the data decisions can be used to nullify their impact. Using $Z_m[k] = a_m[k] \cdot H_m[k]$, we get

$$\begin{aligned} Z_m[k] \cdot Z_{m-1}^*[k] \cdot \hat{a}_m^*[k] \cdot \hat{a}_{m-1}[k] &= \\ a_m[k] H_m[k] \cdot a_{m-1}^*[k] H_{m-1}^*[k] \cdot \hat{a}_m^*[k] \cdot \hat{a}_{m-1}[k] &\approx |a_m[k]|^2 \cdot |a_{m-1}[k]|^2 \cdot |H_m[k]|^2 \end{aligned}$$

The only contribution towards the phase of this expression comes from the residual CFO and SCO, as in the scenario discussed before. Notice the use of index k instead of k_p which implies that all subcarriers are employed for tracking purpose thus improving the synchronization performance.

9.7 Channel Estimation

We covered some techniques to estimate a wireless channel in the context of single-carrier systems in Section 8.2. In OFDM systems, each subcarrier acts as an independent channel as long as there is no Inter-Carrier Interference (ICI) left in the synchronized signal. The options of both a training sequence and individual pilots are available for channel estimation and the choice between the two depends on time variation rate of the channel as well as the computational complexity. Many systems acquire the channel through the preamble while employ the pilots for channel tracking. The discussion in this section is mostly based on Ref. [47].

From Eq (9.13), we can write the signal after the FFT at the Rx as a product between the data symbols $a[k]$ and channel frequency response $H[k]$.

$$\begin{aligned} Z[k] &= a[k] \cdot H[k] \\ \text{for each } k &= -N/2, \dots, N/2 - 1 \end{aligned}$$

This is the frequency domain representation of the Rx samples that will now be exploited for the estimation purpose.

Training Based Estimation

In some systems, training OFDM symbols are periodically sent in the Rx signal so that the channel estimate can be updated at the same rate. In such a scenario, all subcarriers of an OFDM symbol are occupied by the training as shown in Figure 9.45 where OFDM symbols for $m = 0$ and $m = 4$ are the training symbols. After estimating the channel for the initial training sequence, it is common to assume a time-invariant channel (and hence valid estimates for subsequent OFDM data symbols) until the next training sequence arrives. Then, the channel estimate is updated and used again for subsequent OFDM symbols and this process is known as piecewise constant interpolation. For example, training based channel estimation is adopted in IEEE 802.11a/b/g and fixed WiMAX systems.

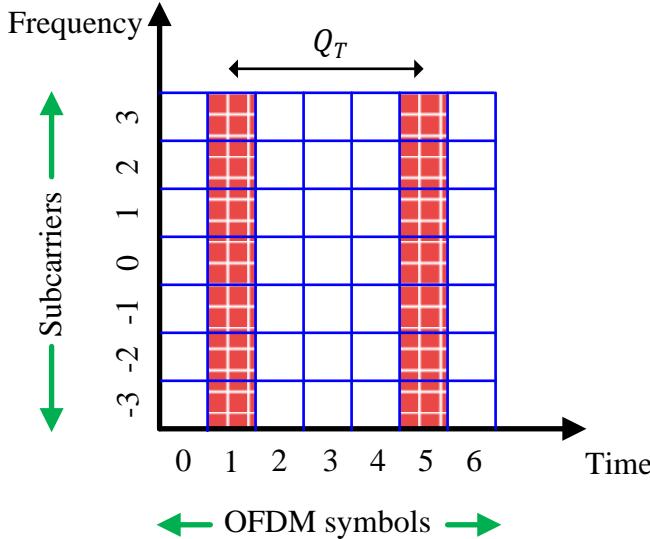


Figure 9.45: Training sequence occupies all the subcarriers in frequency domain for one OFDM symbol

Since the symbols $a[k]$ are known in this context, we can write

$$\hat{H}[k] = \frac{Z[k]}{a[k]} \quad (9.38)$$

for each $k = -N/2, \dots, N/2 - 1$

This is commonly known as Least Squares (LS) channel estimation which in case of OFDM subcarriers is also Maximum Likelihood (ML) solution. Depending on the channel characteristics and the system requirements, this can suffice alone or can be computed as a first estimate before refining it through a more advanced technique.

From Figure 9.45, it is evident that frequency selectivity of the channel is not a problem due to its division into underlying flat fading subcarriers. Even if the channel is drastically different from one subcarrier to the next, it can satisfactorily be estimated by probing on all frequencies.

On the other hand, it is the time domain where the gaps are left and hence susceptible to time selectivity of the channel. Ideally, the training sequences need to be as far apart as possible to minimize the overhead and maximize the data rate. To find out the maximum required spacing between them, we refer to the sampling theorem. Let us denote by Q_T the period between two training sequences which needs to be at least as frequent as the coherence time of the channel T_C which is the inverse of the Doppler spread B_{Dop} (normalized by subcarrier spacing Δ_F), see Eq (8.30). Therefore, we have

$$Q_T \leq \frac{1}{B_{\text{Dop}}/\Delta_F} = \frac{1}{B_{\text{Dop}} T_u}$$

where the scaling factors have been ignored. Since the OFDM symbols between the training are used to deliver the data symbols, a decision-directed approach can also be employed to smooth the channel tracking between the training sequences.

Pilot Based Estimation

The other option is to insert regularly spaced pilots in some subcarriers that are kept occupied for each OFDM symbol. In this way, the pilots are multiplexed with the data and this arrangement is shown in Figure 9.46. For example, the subcarriers $k = -2$ and $k = 1$ are taken by the pilots in Figure 9.46. Then, Eq (9.38) can also be used for pilot-aided channel estimation for values of k corresponding to the pilot symbols.

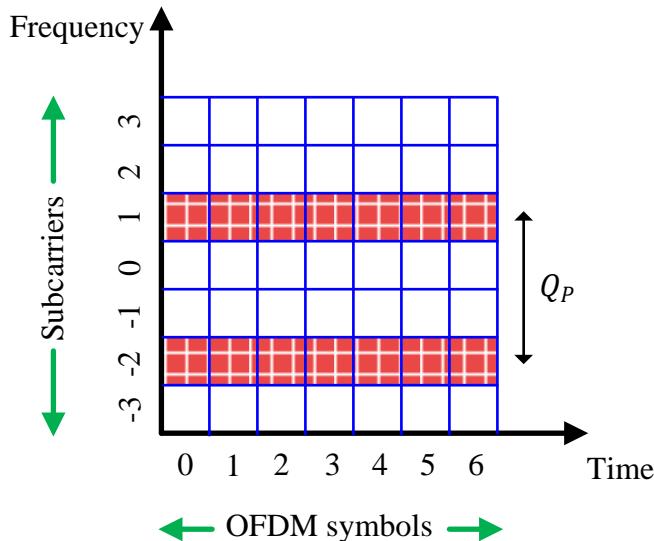


Figure 9.46: Pilot symbols occupy only fixed subcarriers in frequency domain for all OFDM symbols

In this case, time variations of the channel are not a problem since the pilots are present for each OFDM symbol. On the other hand, there is a gap between them in the frequency domain and hence they are susceptible to frequency selectivity of the channel. For this purpose, the maximum gap between two pilots, Q_P is governed by the coherence bandwidth B_C . We know that B_C is inversely proportional to the channel delay spread T_{Del} (which is T_{Del}/T_u samples) leading to the following conclusion.

$$Q_P \leq \frac{1}{T_{\text{Del}}/T_u} = \frac{1}{T_{\text{Del}}\Delta F}$$

As before, the data symbols between the pilots can be exploited in a decision-directed manner to improve on the channel estimates. Some comments are now in order.

Optimal pilot locations: This is determined by the channel fading characteristics in time and frequency domains. Our objective is to insert a minimum number of pilots in the OFDM time-frequency grid. A larger number is a waste of power while a smaller number is a performance loss due to insufficient channel sampling. The tradeoffs involved are accuracy (the more, the better) and spectral efficiency (the lesser, the better). Once a certain number of pilots is determined, it has been reported that the minimum mean square error is obtained when the pilots are equispaced with maximum distance from each other.

Power and modulation: Many systems transmit more power at the pilot subcarriers than the data subcarriers. Also, a lower-order modulation scheme, such as BPSK, is well suited for pilot assignment since a higher data rate is not an objective and maximum power can be allocated to BPSK symbols instead of dividing it into M PSK symbols for normalization, thus implying a longer range.

Interpolation: Interpolation between the pilot subcarriers can be carried out in both time and frequency domains. A piecewise constant interpolation technique that implies a constant channel between two training sequences can be employed. In the case of pilot subcarriers, linear interpolation offers better performance without adding much complexity to the block. Linear interpolation was studied in detail in the context of timing synchronization in Section 7.8. Assume that two pilot measurements are available at subcarriers $k = k_1$ and $k = k_2$ where

$$k_2 - k_1 = Q_P$$

Then, we can find the channel estimate for any other k by linear interpolation.

$$\hat{H}(k) = \mu_k \hat{H}[k_1] + (1 - \mu_k) \hat{H}[k_2], \quad k_1 < k < k_2$$

where μ_k is given by

$$\mu_k = \frac{k}{Q_P}$$

It can be viewed as a 2 tap filter applied to the channel estimate sequence after upsampling the channel estimates at pilot locations by Q_P . This is shown in Figure 9.47. Better interpolation techniques such as Wiener, Gaussian, cubic or spline are also frequently applied for this purpose. Finally, taking an iFFT/FFT pair also accomplishes pilot interpolation if the FFT size is divisible by the pilot spacing.

Another option is to arrange the pilots on adjacent subcarriers in adjacent OFDM symbols such that they cover the time-frequency grid in a diagonal fashion. This helps in sampling the channel at all times and all subcarriers in a periodic manner.

Transform Domain Techniques

There are scenarios in which the number of taps in the channel impulse response is much less than the Cyclic Prefix (CP) length. This extra information helps in refining an initial channel estimate (such as the Least Squares (LS) estimate) by concentrating on the channel taps within the delay spread T_{Del} and nullifying the effect of the noise outside the farthest channel tap.

Consider an OFDM system with the CP length N_{CP} and the number of subcarriers N operating in a channel with length $N_{\text{Tap}} + 1$. Let $\hat{H}[k]$ for each subcarrier k denote the Least Squares estimate for the corresponding channel gain. Next, we take the iFFT of $\hat{H}[k]$ and denote it by $\hat{h}[n]$ that represents an estimate of the channel impulse response $h[n]$. This sequence in time domain is given by

$$\hat{h}[n] = h[n] + \text{noise}, \quad n = 0, 1, \dots, N - 1$$

Since the actual channel length is $N_{\text{Tap}} + 1$, all the taps after N_{Tap} consist of noise only. Here, we form a new channel estimate $\hat{h}_{FFT}[n]$ by removing all the taps from $\hat{h}[n]$

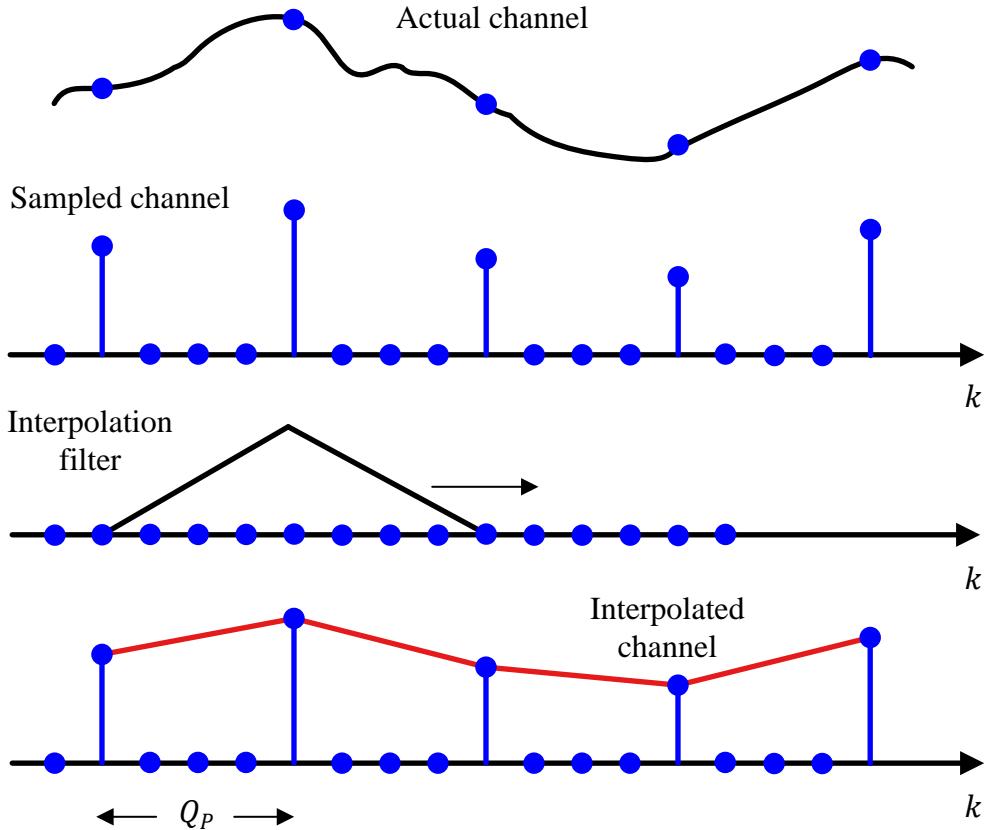


Figure 9.47: Linear interpolation between pilot subcarriers

beyond N_{Tap} .

$$\hat{h}_{\text{FFT}}[n] = \begin{cases} h[n] + \text{noise} & n = 0, 1, \dots, N_{\text{Tap}} \\ 0 & n = N_{\text{Tap}} + 1, \dots, N - 1 \end{cases}$$

Now when we perform an FFT of the above sequence to go back to the frequency domain, the resultant channel estimates are free of noise beyond $n = N_{\text{Tap}}$ and consequently yield better performance.

$$\hat{H}_{\text{FFT}}[k] = \text{FFT}\{\hat{h}_{\text{FFT}}[n]\}$$

A block diagram for the implementation of such an approach is drawn in Figure 9.48. Although a similar concept can be applied to some other transforms, the FFT is particularly attractive due to its efficient implementation.

Decision-Directed Channel Estimation

Due to its widespread use in single-carrier systems, decision-directed channel estimation was one of the earliest methods explored for OFDM systems. It is particularly

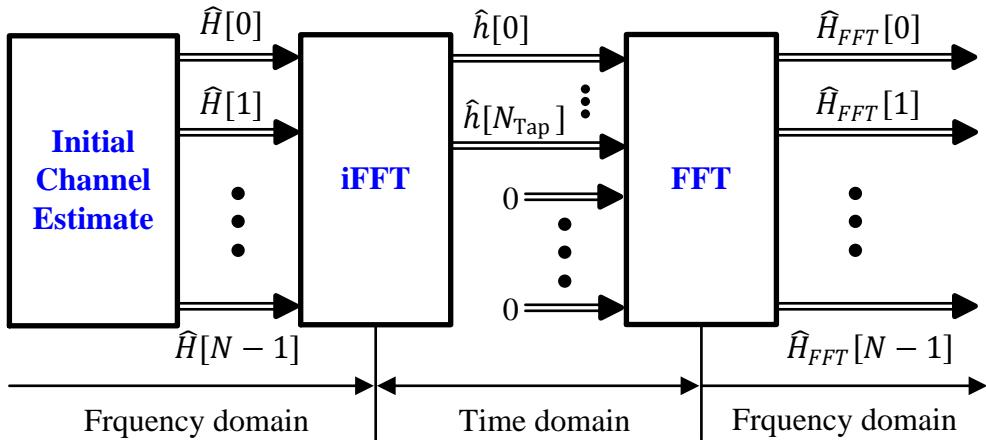


Figure 9.48: Channel estimation through the iFFT-FFT operations

useful because after the initial estimates, the channel coefficients can be updated with the help of decisions from data symbols. The basic idea is to employ the channel estimates from the previous OFDM symbol to detect the data in the current OFDM symbol.

- Assume that the channel is slowly varying from one OFDM symbol to the next. Let $Z_m[k]$ be the m^{th} Rx signal after the FFT and $\hat{H}_{m-1}[k]$ be the channel estimate for $(m-1)^{\text{th}}$ OFDM symbol. Then, it can be employed to detect the current data symbols as

$$\hat{a}_m[k] = \frac{Z_m[k]}{\hat{H}_{m-1}[k]}$$

- Next, the symbol candidate $\hat{a}_m[k]$ can be used for estimating the channel $\hat{H}_m[k]$ at time m as

$$\hat{H}_m[k] = \frac{Z_m[k]}{\hat{a}_m[k]}$$

As another option, the data symbol decision sign $\{\hat{a}_m[k]\}$ can also be used for this purpose to reduce the effect of noise.

$$\hat{H}_m[k] = \frac{Z_m[k]}{\text{sign}\{\hat{a}_m[k]\}}$$

There are two main problems with this kind of decision-directed approach.

Channel estimate expiry: For a quickly varying channel on a time scale of an OFDM symbol, the channel estimates at time $m-1$ are no longer valid at time m . Consequently, the detected symbols $\hat{a}_m[k]$ increasingly suffer from errors.

Error propagation: Wrong data symbols fed back to estimate the current channel gives rise to error propagation which renders the system dysfunctional. This is a problem associated with all decision-directed techniques whether in the context of equalization or synchronization.

To resolve such issues, significant improvement over the decision-directed channel estimation as well as training and pilot based methods described above can be gained through involving a powerful error correcting code for the channel estimation process. Such codes operate in a recursive manner and improve the reliability of symbol decisions in each iteration. Inserting the channel estimation process within such a feedback loop improves the quality of the channel estimates which in turn refines the reliability of the decisions.

Now when we have discussed several synchronization and channel estimation strategies, it is interesting to see the direction where the high rate wireless systems are heading. For this purpose, we compare IEEE 802.11a standard specifications studied in the examples of this chapter to later and modern IEEE 802.11 standards such as n, ac and ax. Such a comparison is given in Table 9.2. Notice that better frequency synchronization enables a closer spacing of the subcarriers, 4 times closer in ax for example. Moreover, the fundamental rate increase comes from either increasing the channel bandwidth (160 MHz), the number of antennas (8) or (to some extent) a higher-order modulation scheme (1024-QAM).

Table 9.2: A comparison of IEEE 802.11 systems

	802.11a	802.11n	802.11ac	802.11ax
Frequency (GHz)	5	2.4, 5	5	2.4, 5
Bandwidth (MHz)	20	40	160	160
Δ_F (kHz)	312.5	312.5	312.5	78.125
T_u (μ s)	3.2	3.2	3.2	0.8, 1.6, 3.2
MIMO	1	4	8	8
Multi-User MIMO	No	No	Downlink	Uplink, Downlink
Modulation	64-QAM	64-QAM	256-QAM	1024-QAM
Data rate (Mbps)	54	600	1000-3000	10000

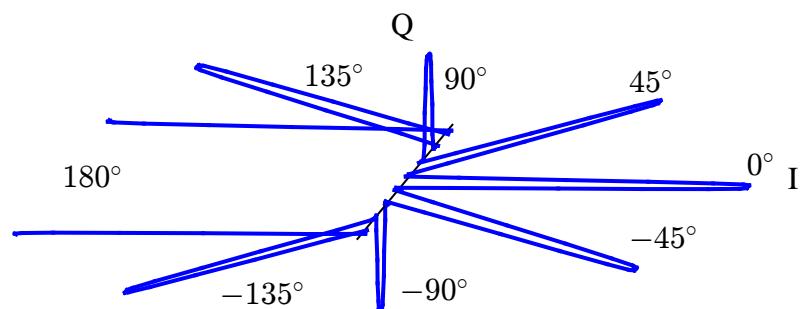
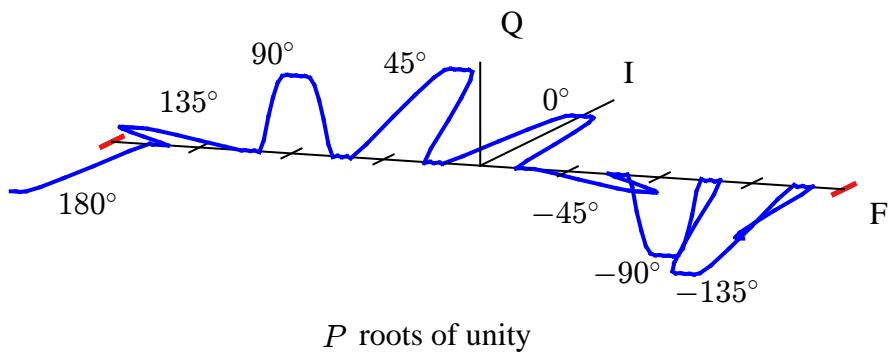
9.8 The Small Picture

In high rate wireless systems, multipath effect becomes the major bottleneck due to resource intensive time domain equalization techniques. This problem is overcome in time domain by transmitting the information in several low rate parallel streams which segments in frequency domain the available bandwidth into many parallel frequency flat channels. Since convolution in time is multiplication in frequency, equalization for each narrow slice requires just a single division operation, significantly reducing the computational load of the equalizer.

Viewed from another angle, OFDM employs a set of complex sinusoids in time domain that *samples the channel in frequency domain* at discrete intervals in frequency. See Note 3.5 for the dual phenomenon in time domain. I find this perspective very fascinating and beautiful.

Chapter 10

The Chapter with No Title



“Software is eating the world.”

Marc Andreessen

In the previous chapters, we have covered the issues of synchronization and equalization in single and multicarrier systems in a reasonable detail. The purpose of this final chapter[†] is to explain the transceiver architectures that enable the modulated signal transmission and reception from a DSP perspective. In early years of digital communications, receiver design mostly consisted of analog signal processing circuits and few operations aside from the detector were performed in the digital domain. The cost-benefit tradeoff between analog and digital signal processing kept pushing the discrete domain towards the antenna, a trend we now know as a Software Defined Radio (SDR). This fundamental shift in the signal playing field by necessity had to give rise to rethinking of many design issues.

Just like we encountered a seesaw perspective over and over again in the discussion on timing synchronization, the common theme in this context is that the hardware is expensive while the software is economical as well as reconfigurable. If a component can be traded off with some lines of code as a result of a design decision, we almost always strive to exercise this option. In addition to better performance, this leads to a reduced Bill of Materials (BOM), enhanced engineering flexibility and reduced design cycle time. With this push of digital technology towards the antenna, good RF engineers need to master both analog and discrete domains before and after the ADC, respectively, to deliver the required system specifications for a truly flexible radio. Their job is unique in this regard because there are very few roles in the world where your own efficient (and certainly elegant) designs slowly shrink your cohort out of the job market.

Note 10.1 The digital revolution

At the start of Chapter 1, we mentioned a hypothetical example for transporting a person or object by converting them into energy beams as an analogy of the shift from the world of physics to the world of bits. In reality as well, the digital revolution has transformed many fields to create new industries while wiping out the old ones and taken many individuals from zero to one and some from one to zero. Adam Grant tells us the following story in his book *Originals* about the famous American inventor and founder of Polaroid, Edwin Land.

“In 1980, Land was approached by Sony founder Akio Morita, who confided that the chemical processing of film might not be the wave of the future, and expressed interest in collaborating on an electronic camera. Land saw the world in terms of chemistry and physics, not zeros and ones. He dismissed the idea, insisting that customers would always want prints, and that the quality of digital photos would never approach that of chemically processed ones.” The rest, as they say, is history.

Before we start, it should be kept in mind that both the radio Tx and Rx have two frontends: an *analog frontend* and a *digital frontend*. As a result, we have $2 \times 2 = 4$ different choices for frontend design in a transceiver as shown in Figure 10.2. Each

[†]A reader might think that the chapter title could have been ‘Transceiver Architectures’ but to me, an equally important focus of this chapter should be downsampling the signal through a polyphase filterbank where the magic of multirate signal processing manifests itself in its full glory. Hence, I chose the current title as a hint that the chapter contains something more than what is apparent. Obviously, anything with no name reminds us of “the man with no name” :)



Figure 10.1: Photography is just one of the countless areas revolutionized by the digital electronics

kind of Rx architecture has a corresponding Tx architecture. The digital frontend at the Rx for example employs powerful signal processing techniques to optimize the downconversion and downsampling tasks of the input signal. For this reason, digital frontends are mostly based on a direct conversion (zero-IF) architecture. The analog frontend, particularly for the Rx, has many more practical issues to handle and hence must choose from a diverse set of available options depending on the system specifications, cost and complexity.

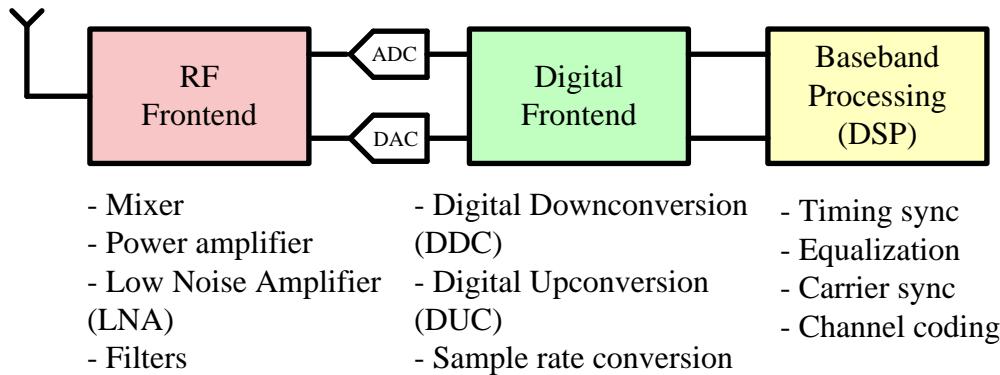


Figure 10.2: A software defined radio

The topics covered in the past chapters lie in the realm of baseband signal processing shown at the right of Figure 10.2 such as timing synchronization, equalization and carrier recovery. In this chapter, we are progressing towards the left of this figure. We will study only the digital frontend since this text exclusively deals with discrete-time processing. Interestingly, the magic of DSP manifests itself nowhere more than this digital frontend part. Finally, instead of going into the details of the design of a Digital Downconverter (DDC) or a Digital Upconverter (DUC), we will emphasize on the basic principles of multirate signal processing that play a crucial role in their designs.

We start with the architecture of a modulator for a linearly modulated scheme such

as QAM. I highly recommend reading Section 7.13 for understanding polyphase partitions of a mother filter before continuing further.

10.1 Tx Architecture

Let us return to the discrete-time modulator studied during the conceptual understanding of a Quadrature Amplitude Modulation (QAM) scheme in Chapter 3 and drawn here in Figure 10.3.

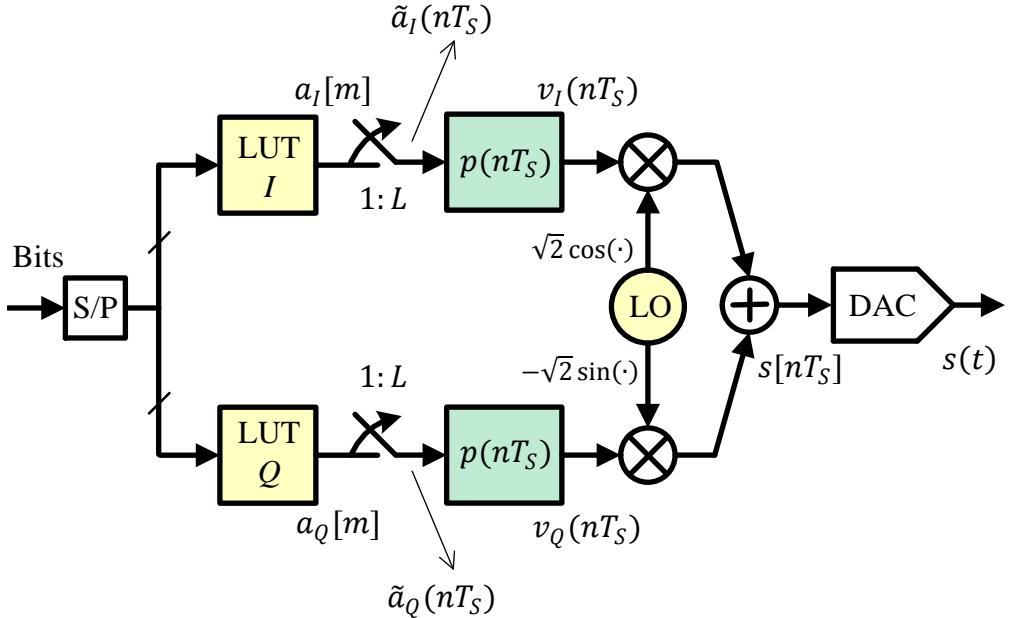


Figure 10.3: A QAM modulator upconverts the signal to an Intermediate Frequency (IF) using simple discrete-time processing

The procedure to produce a QAM waveform in theory is as follows. First, the symbol sequences $a_I[m]$ and $a_Q[m]$ are converted to discrete-time impulse trains $\tilde{a}_I(nT_S)$ and $\tilde{a}_Q(nT_S)$, respectively, in separate arms through upsampling by L , where L is the number of samples per symbol. For a symbol time T_M , $F_S = L/T_M$ and we can write

$$\begin{aligned} I &\rightarrow v_I(nT_S) = \sum_m a_I[m] p(nT_S - mT_M) \\ Q &\uparrow v_Q(nT_S) = \sum_m a_Q[m] p(nT_S - mT_M) \end{aligned} \quad (10.1)$$

Upconversion to a higher frequency, say an Intermediate Frequency (IF), follows this pulse shaping operation in which we multiply $v(nT_S)$ with a complex sinusoid at a frequency k_{IF}/N , where k_{IF}/N corresponds to the IF.

$$\begin{aligned} I &\rightarrow v_I(nT_S) \sqrt{2} \cos 2\pi \frac{k_{\text{IF}}}{N} n - v_Q(nT_S) \sqrt{2} \sin 2\pi \frac{k_{\text{IF}}}{N} n \\ Q &\uparrow v_Q(nT_S) \sqrt{2} \cos 2\pi \frac{k_{\text{IF}}}{N} n + v_I(nT_S) \sqrt{2} \sin 2\pi \frac{k_{\text{IF}}}{N} n \end{aligned} \quad (10.2)$$

Now the signals sent over the air or any medium are always real, so we can choose either of the I or Q arm for transmission. By convention, the I arm is chosen which represents a general QAM waveform as

$$\begin{aligned}s(nT_S) &= v_I(nT_S)\sqrt{2} \cos 2\pi \frac{k_{\text{IF}}}{N} n - v_Q(nT_S)\sqrt{2} \sin 2\pi \frac{k_{\text{IF}}}{N} n \\&= v_I(nT_S)\sqrt{2} \cos 2\pi \frac{F_{\text{IF}}}{F_S} n - v_Q(nT_S)\sqrt{2} \sin 2\pi \frac{F_{\text{IF}}}{F_S} n \\&= v_I(nT_S)\sqrt{2} \cos 2\pi F_{\text{IF}} n T_S - v_Q(nT_S)\sqrt{2} \sin 2\pi F_{\text{IF}} n T_S\end{aligned}\quad (10.3)$$

where we have used the fundamental relation between continuous and discrete frequencies $k/N = F/F_S$. After Digital to Analog Conversion (DAC), the continuous-time signal $s(t)$ can be expressed as

$$s(t) = v_I(t)\sqrt{2} \cos 2\pi F_{\text{IF}} t - v_Q(t)\sqrt{2} \sin 2\pi F_{\text{IF}} t$$

The important point about this architecture is that discrete-time processing is employed to convert the signal to the IF that overcomes many of the disadvantages of carrying out the same operations through analog components. We will discuss these disadvantages in the context of Rx architectures in Section 10.2.

10.1.1 Polyphase Filterbank Implementation

Our first task is to produce the baseband pulse shaped waveforms $v_I(nT_S)$ and $v_Q(nT_S)$ mentioned in Eq (10.1) in an efficient manner. The theory behind a pulse shaping filter was described in detail in Section 3.6. For a symbol rate $1/T_M$, we are going to accomplish this task in two stages:

1. through a pulse shaping filter at a rate $F_{S,1}$, and
2. a further interpolating Low-Pass Filter (LPF) at the actual rate F_S to accommodate the high IF in the digital domain.

Thereafter, we will convert the interpolating Low-Pass Filter (LPF) to an interpolating Band-Pass Filter (BPF) that places the desired signal at a particular IF. Let us begin with the design of the pulse shaping filter.

Pulse Shaping through a Polyphase Filterbank

We begin with $F_{S,1} = 1/T_{S,1}$ and leave the actual F_S for a later description. Coming to initial task of bandlimiting the spectrum, the oversampling factor L is typically chosen as 2 or 4 to accommodate the excess bandwidth α of the pulse shaping filter $p(nT_{S,1})$ where

$$T_{S,1} = \frac{T_M}{L}$$

Pulse shaping is just a simple convolution between the upsampled by L modulated symbols $\tilde{a}(nT_{S,1})$ and the filter $p(nT_{S,1})$ where the summations over the input samples are executed after weighting them by the filter coefficients. The upsampled signal contains $L - 1$ zeros that do not contribute to the weighted sum that is formed at the filter output. We have already covered such a structure in detail in Section 7.13 and briefly touch on a polyphase filterbank below operating at a rate $F_{S,1} = L/T_M$.

Consider the rate $F_{S,1}$ samples of the initial pulse shaping filter $p(nT_{S,1})$ with an arbitrary length, which we call a *mother filter*. Instead of a 1-dimensional array, the coefficients of this mother filter can also be written as a 2-dimensional matrix such that the first L elements are written into the 1st column, the next L elements are written into the 2nd column, and so on. This mapping is illustrated in Table 10.1 for $L = 4$. Now we know that these are called *polyphase partitions* and we learned the logic behind the term ‘polyphase’ in Section 7.13.

Table 10.1: Polyphase partitions of the mother filter $p(nT_{S,1})$ for $L = 4$

Polyphase Partition ↓	Filter Samples			
	→ → → →			
$p_0(mT_M) \rightarrow$	$p(0T_{S,1})$	$p(4T_{S,1})$	$p(8T_{S,1})$...
$p_1(mT_M) \rightarrow$	$p(1T_{S,1})$	$p(5T_{S,1})$	$p(9T_{S,1})$...
$p_2(mT_M) \rightarrow$	$p(2T_{S,1})$	$p(6T_{S,1})$	$p(10T_{S,1})$...
$p_3(mT_M) \rightarrow$	$p(3T_{S,1})$	$p(7T_{S,1})$	$p(11T_{S,1})$...

From Table 10.1, notice that there are $L = 4$ polyphase partitions

$$p_0(mT_M), p_1(mT_M), p_2(mT_M), p_3(mT_M)$$

of the mother filter $p(nT_{S,1})$ all operating at the symbol rate $1/T_M = F_{S,1}/L$. This is because the sampling rate $F_{S,1}$ is L times higher and there are L such partitions. The length of each partition is $1/L$ of the mother filter (e.g., a length of 3 each for a total length of $3L = 12$ in the table).

This set forms a parallel bank of L filters, each of which is a downsampled version of the mother filter. Looking at the first column, the starting index – and hence the timing offset – of each of these partitions is unique taking values from the set $0, 1, \dots, L - 1$, the effect of which was shown in time and frequency domain figures of Section 7.13. It was shown there that due to this partition and the subsequent timing offsets, *only one nonzero sample is available at the output of only one such partition* at each output time $T_{S,1}$. Therefore, we can simply point to the arm that supplies this nonzero sample in a sequential manner, as drawn in Figure 10.4. Notice that the polyphase partitions operate only at the symbol rate $1/T_M$ while the commutator at the output runs at the rate $1/T_{S,1}$, thus producing the modulated and pulse shaped signal ready for the processing of the next stage.

Figure 10.5 presents the spectrum of this signal generated at a rate of $L = 4$ samples/symbol for a Square-Root Raised Cosine pulse with excessive bandwidth $\alpha = 0.5$. Observe that the baseband extends from $-F_{S,1}/2$ to $+F_{S,1}/2$ which becomes

$$\left\{ -\frac{F_{S,1}}{2}, +\frac{F_{S,1}}{2} \right\} = \left\{ -\frac{L}{2T_M}, +\frac{L}{2T_M} \right\} = \{-2, +2\} \quad (10.4)$$

where the values of $L = 4$ and $1/T_M = 1$ are assumed. We will continue using these values throughout this discussion for ease of exposition. For $\alpha = 0.5$, the edges of the

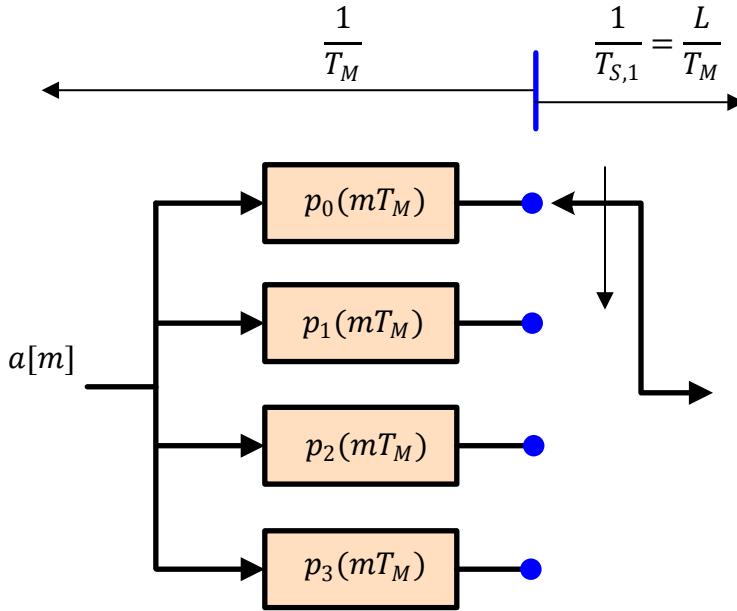


Figure 10.4: A polyphase decomposition of the pulse shaping filter

passband and stopband of this pulse shaping filter are

$$\begin{aligned} F_{\text{pass}} \left\{ p(nT_{S,1}) \right\} &= \frac{1 - \alpha}{2T_M} = 0.25 \\ F_{\text{stop}} \left\{ p(nT_{S,1}) \right\} &= \frac{1 + \alpha}{2T_M} = 0.75 \end{aligned} \quad (10.5)$$

Our next target is to interpolate this signal at a higher rate to prepare it for upconversion to IF.

Interpolating Lowpass Filter

After pulse shaping, there are several options available to us that depend on the Tx architecture. Referring to Figure 10.3 as an example, the waveforms in I and Q arms

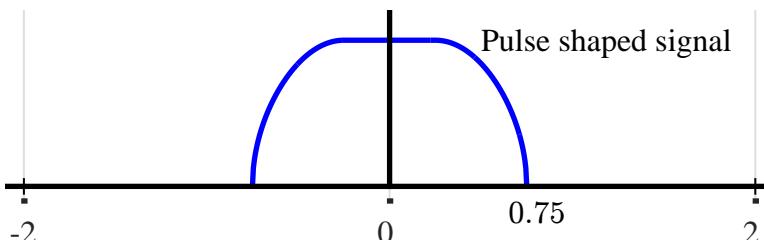


Figure 10.5: Spectrum at the output of the pulse shaping filter at $L = 4$ samples/symbol for a Square-Root Raised Cosine pulse with excessive bandwidth $\alpha = 0.5$. A unit symbol rate $1/T_M$ is assumed

can be converted to analog signals through a pair of DACs which can then be upconverted to IF through analog signal processing. However, all this requires considerable hardware such as two DACs, two lowpass filters, two mixers and a quadrature oscillator. Furthermore, each such pair needs to be matched to avoid impairments arising from their imbalance in the Tx signal. This cost can be avoided if the DAC sample rate can be further increased by a substantial amount and this upconversion to IF is done in discrete domain.

Then the problem we face is that if a low upsampling factor of $L = 4$ dictates the DAC sample rate F_S , the spectral replicas are really close to each other and there is little room left for accommodating a passband signal at the IF. For example, for a unit symbol rate $1/T_M$, the baseband according to Eq (10.4) extends from -2 to $+2$ and typical values of IF cannot be accommodated. On the other hand, if the sample rate F_S is increased by a factor of P to, say $LP = 4P$, through an interpolating lowpass filter, then this makes enough room for the baseband signal to be upconverted to the IF in discrete domain.

The price to pay for this choice is a higher clock rate at the DAC resulting in more power consumption. This is because switching the logic gates from high to low or low to high at a faster rate draws more current per unit time and hence consumes more power. This is in addition to the heat dissipation arising from parasitic components and circuit imperfections occurring at the faster rate. Nonetheless, we can reduce the cost of a higher sample rate by implementing the interpolating filter through a polyphase partition as well.

Many radios implement this interpolation in multiple stages that include a cascade of filters with none or minimal number of multiplication operations. Instead of focusing on a specific architecture, our main purpose here is to understand how multirate signal processing results in an efficient system realization. Therefore, to keep the discussion simple, we can either assume a simple pulse shaping filter oversampled by LP or we can take a single interpolating lowpass filter that upsamples the pulse shaped signal by a factor of P and denote it by $h(nT_S)$, where

$$T_S = \frac{T_{S,1}}{P} = \frac{T_M}{LP}$$

We choose the latter route in this development. Other discrete-time implementations can be easily grasped through the extension of the concepts studied here. This is shown in Figure 10.6 where we denote the output of this cascade as $v(nT_S)$ which was the input to the digital mixers in Figure 10.3 and the rest of this text. From here onwards, we will assume that $P = 8$.

With this setting in place, we assign the passband of the interpolating lowpass filter $h(nT_S)$ at the edge of the stopband of $p(nT_{S,1})$ given in Eq (10.5) and its stopband at the edge of the stopband of the next replica as follows.

$$\begin{aligned} F_{\text{pass}} \{ h(nT_S) \} &= F_{\text{stop}} \{ p(nT_{S,1}) \} = 0.75 \\ F_{\text{stop}} \{ h(nT_S) \} &= L - F_{\text{stop}} \{ p(nT_{S,1}) \} = 3.25 \end{aligned} \tag{10.6}$$

The specifications for this interpolating lowpass filter are drawn in Figure 10.7 where the wide transition band of the lowpass filter can be observed.

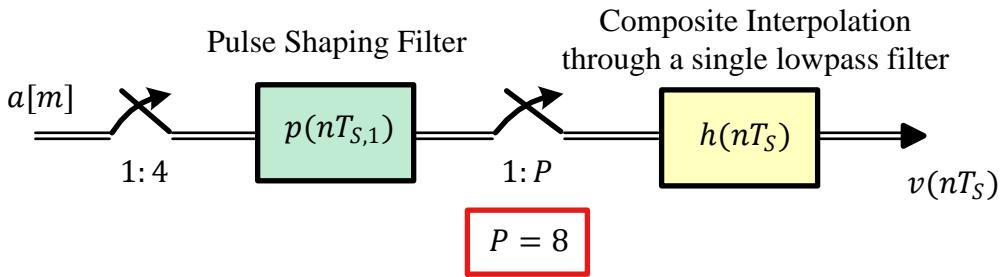


Figure 10.6: An interpolator (or a few stages of efficient interpolation) follows the initial pulse shaped baseband signal. For simplicity of explanation, we assume a single interpolating by $P = 8$ lowpass filter

Exercise 10.1

From the passband and stopband frequencies derived in Eq (10.6), we design the low-pass interpolating FIR filter through Parks-McClellan algorithm that meets the following specifications.

$$\text{Sample rate } F_S = LP = 4(8) = 32$$

$$\text{Passband edge } F_{\text{pass}} = 0.75$$

$$\text{Stopband edge } F_{\text{stop}} = 3.25$$

$$\text{Passband ripple } \delta_{\text{pass,dB}} = 0.1 \text{ dB}$$

$$\text{Stopband ripple } \delta_{\text{stop,dB}} = -50 \text{ dB}$$

A procedure for this design was explained in Section 2.5 where a Matlab function `firpm()` returns the coefficients of a length $N + 1$ linear phase FIR filter. By trial and error, we find that $N = 40$ approximately satisfies these requirements.

$$\begin{aligned} h(nT_S) &= \text{firpm}(N - 1, [0 \quad F_{\text{pass}} \quad F_{\text{stop}} \quad 0.5F_S]/(0.5F_S), [1 \quad 1 \quad 0 \quad 0]) \\ &= \text{firpm}(39, [0 \quad 0.75 \quad 3.25 \quad 16]/16, [1 \quad 1 \quad 0 \quad 0]) \end{aligned}$$

In this function, the vector $[1 \quad 1 \quad 0 \quad 0]$ specifies the desired amplitude at band edges of passband and stopband, respectively. Finally, a normalization by $F_S/2$ instead of F_S is a (confusing) Matlab convention.

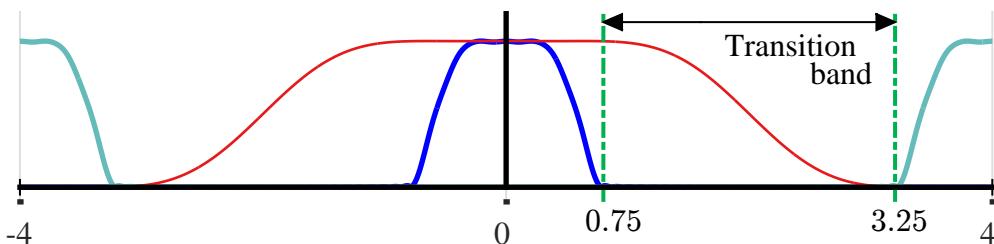


Figure 10.7: Pulse shaping filter and its spectral replicas for $L = 4$, $1/T_M = 1$ and the transition band of the interpolating lowpass filter

The frequency response $20 \log_{10} |H(F)|$ of the final filter is drawn in Figure 10.8 overlaid on a square-root Nyquist filter having $L = 4$ and excess bandwidth $\alpha = 0.5$. Designing a square-root Nyquist filter was explained in Section 3.6. A Square-Root Raised Cosine was not utilized due to its high sidelobes up to -40 dB which would have violated the stopband specification of -50 dB.

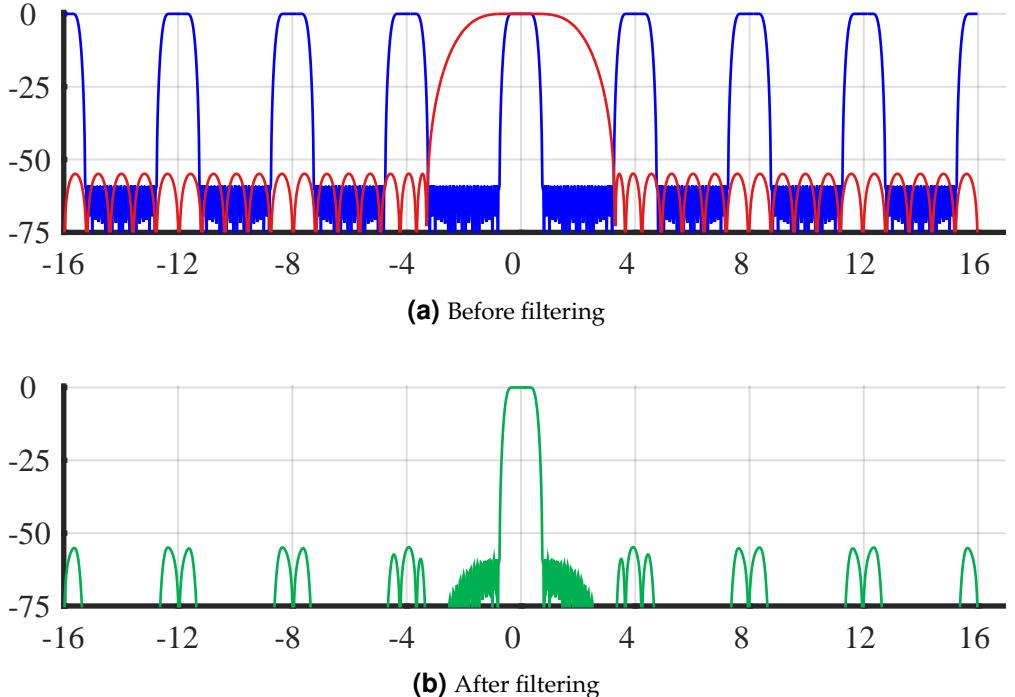


Figure 10.8: Spectrum of the zero-packed square-root Nyquist filter for $\alpha = 0.5$ and $L = 4$ as well as that of $1 : 8$ interpolating *lowpass* filter

It is evident from Figure 10.8a that the transition bandwidth of $1 : 8$ interpolating filter specified according to the passband and stopband requirements of 0.75 and 3.25 , respectively, is prepared to suppress the spectral replicas of the shaping filter arising as a result of $1 : 8$ upsampling process. From Figure 10.8b, the result of this lowpass filtering has sidelobes attenuation of slightly more than 50 dB and a similar attenuation of the adjacent spectra. Some additional attenuation in digital filters with equiripple sidelobes is desirable to alleviate the effects of finite arithmetic and sidelobe integration.

This lowpass filter is now partitioned into $P = 8$ polyphase arms with 5 taps each since $8 \times 5 = 40$.

$$h_0(nT_{S,1}), h_1(nT_{S,1}), \dots, h_7(nT_{S,1})$$

Observe that $T_{S,1}$ signifies the fact that these arms are operating at the lower rate $F_{S,1} = F_S/P$. We can increase the filter order for better suppression and in doing so, we need to realize that $N = 48$ is a suitable option so that the polyphase arms later can all have 6 taps in 8 stages. Another option is to design the filter with a certain N and then add zero padding to make it a multiple of $P = 8$. We do not describe

the polyphase partition here since the procedure is very similar to the shaping filters studied above and in Section 7.13.

Next, we discuss the upconversion procedure to the IF.

From Lowpass to Bandpass Filter

By now, we have accomplished implementing a QAM modulator in an efficient manner up till Eq (10.1). To generate the signal $s(nT_S)$ at IF, we need to actually mix the signal $v(nT_S)$ with a complex sinusoid with frequency k_{IF}/N and retain the inphase part. From Figure 10.6, recall that we are denoting the interpolating filter output as $v(nT_S)$ instead of the pulse shaping filter output.

The simplest method to accomplish this task is to emulate the analog mixer in discrete domain through sample-by-sample product between the signal and the carrier. We reproduce the expression for $s(nT_S)$ from Eq (10.3) below.

$$s(nT_S) = v_I(nT_S)\sqrt{2} \cos 2\pi \frac{F_{\text{IF}}}{F_S} n - v_Q(nT_S)\sqrt{2} \sin 2\pi \frac{F_{\text{IF}}}{F_S} n \quad (10.7)$$

This product takes place at the high sample rate $F_S = LP$ at each time instant n . The first hint about a parallel approach comes from the fact that the pulse shaping and interpolating lowpass filters were implemented through polyphase partitions. There must be some technique to embed the upconversion into the same architecture.

Looking at Figure 10.8a earlier, notice that we isolated the spectral copy at baseband through lowpass filtering. *That signal already had many spectral replica at higher frequencies!* As long as the IF coincides with one of the normalized frequencies 4, 8, 12 or 16, we can simply upconvert the lowpass filter $h(nT_S)$ to the desired upper band beforehand to make it a Band-Pass Filter (BPF) $h_{\text{BP}}(nT_S)$ and process the zero-packed symbols with this filter instead.

$I \rightarrow$ $Q \uparrow$	$h_{\text{BP},I}(nT_S) = h(nT_S) \cos 2\pi \frac{k_{\text{IF}}}{P} n$ $h_{\text{BP},Q}(nT_S) = h(nT_S) \sin 2\pi \frac{k_{\text{IF}}}{P} n$
---------------------------------	---

(10.8)

where k_{IF} refers to the length- P DFT index corresponding to the intermediate frequency F_{IF} (this is a slight abuse of notation since k_{IF} was originally defined as the index in a length- N DFT but then we had to include the factor L from the shaping filter too). Instead of the interpolating lowpass filter, when this interpolating bandpass filter is applied to the output of the pulse shaping filter, say for a normalized IF equal to 12, we get the upconverted signal $s(nT_S)$ as plotted in Figure 10.9 for a frequency domain view.

Here, the spectrum at the normalized frequency of 12 is passed through the interpolator while all the adjacent channels, including the one at baseband, have been suppressed by the filtering operation. Recall that real signals have a symmetric Fourier Transform and hence this one-sided spectrum represents a complex signal. In fact, this corresponds to the signal described in Eq (10.2). Our desired signal $s(nT_S)$ originates from the inphase (real) part of the time domain signal corresponding to this passband spectrum. Furthermore, although the pulse shaping filter shown in Figure 10.9a is real and hence symmetric, the original QAM symbols are complex valued

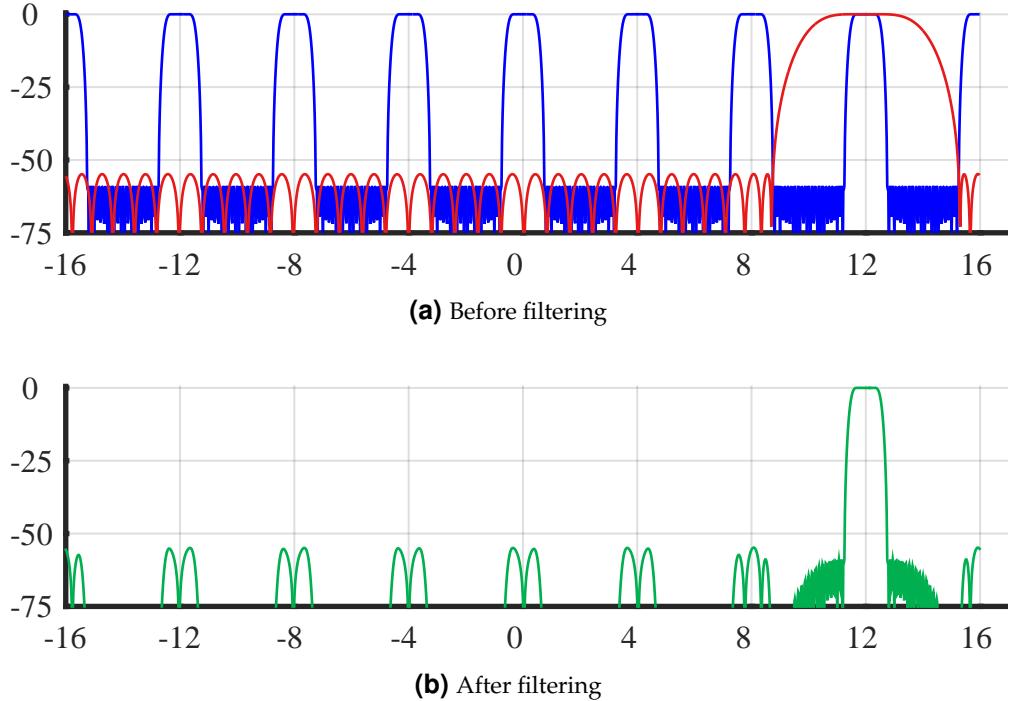


Figure 10.9: Spectrum of the zero-packed square-root Nyquist filter for $\alpha = 0.5$ and $L = 4$ as well as that of 1 : 8 interpolating *bandpass* filter

and hence represent a complex time series $v(nT_S)$ after pulse shaping. Consequently, this 1 complex convolution requires 4 real convolutions which is simplified to 2 convolutions due to the fact that only the real part of the resultant signal is needed for transmission. This is similar to the multiplication rule of complex signals $I \cdot I - Q \cdot Q$.

$$s(nT_S) = v_I(nT_S) * h_{BP,I}(nT_S) - v_Q(nT_S) * h_{BP,Q}(nT_S)$$

To derive the final architecture embedding the spectral translation in multirate signal processing, usually a z-Transform approach[†] is employed. Since we do not have

[†]We can simply replace $h(nT_S)$ with $h_{BP}(nT_S)$ in the definition of z-Transform when

$$h_{BP}(nT_S) = h(nT_S) \exp\left(j2\pi \frac{k}{P} n\right)$$

Here, we take help from Eq (7.73) in regards to the filter decomposition.

$$\begin{aligned} H_{BP}(z) &= \sum_n h(nT_S) \exp\left(j2\pi \frac{k}{P} n\right) z^{-n} = \sum_n h(nT_S) \left\{ z \cdot \exp\left(-j2\pi \frac{k}{P}\right) \right\}^{-n} \\ &= \sum_{i=0}^{P-1} z^{-i} \exp\left(j2\pi \frac{k}{P} i\right) \sum_n h(nP + i) z^{-nP} \underbrace{\exp\left(j2\pi \frac{k}{P} nP\right)}_{=1} \\ &= \sum_{i=0}^{P-1} z^{-i} \exp\left(j2\pi \frac{k}{P} i\right) \sum_n h(nP + i) z^{-nP} \end{aligned} \quad (10.9)$$

where the simplification in the second line follows from the polyphase partition in Eq (7.73). What this equations states is that for forming a bandpass filter, the polyphase partition of the lowpass filter $h(nT_S)$

this liberty, we follow another approach by utilizing the following two facts.

- Coming to the first $P = 8$ multiplications, i.e., for $n = 0, 1, \dots, P - 1$, first observe from $s(nT_S)$ in Eq (10.7) that the multiplication is taking place at the highest rate $LP = 32$ in a conventional serial mixing. As an example, 5 cycles of a complex sinusoid for $k = 1$ are drawn at the top of Figure 10.10. In actuality, the period and frequency of this sinusoid represent the IF F_{IF} .
- Then, we imagine the polyphase decomposition of the interpolating bandpass filter (which will be similar to that of the pulse shaping filter in Figure 10.4) and notice that the output samples will also be taken at the higher rate of LP . Therefore, one such complex multiplication can be associated with each polyphase arm from top to bottom.

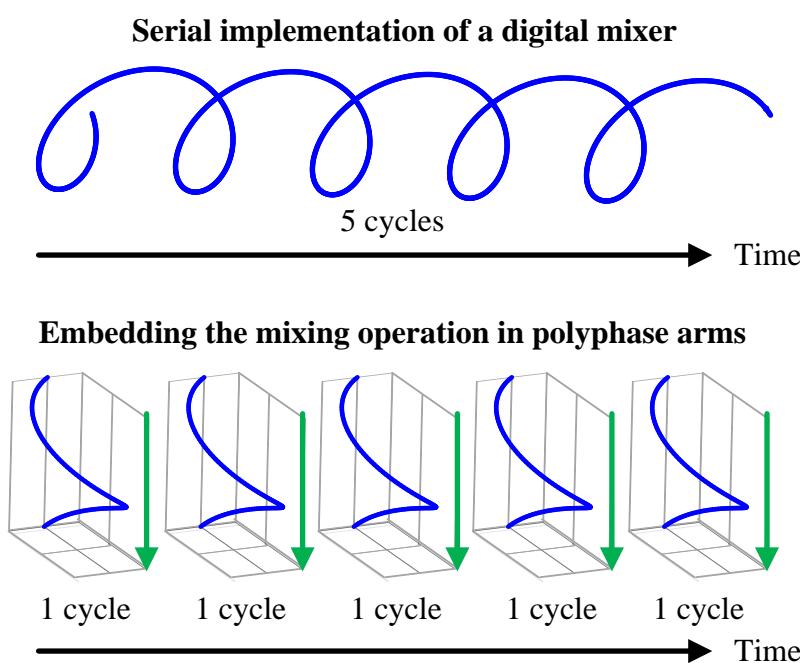


Figure 10.10: A visualization of the mixing process for a serial implementation as well as its embedded form in polyphase arms of the bandpass interpolating filter is shown for $k = 1$

- For values of n after $n = P - 1$, the complex sinusoid just repeats itself due to its periodicity with $P = 8$. This can be seen for $n = P$ as

$$\begin{array}{ll} I & \rightarrow \\ Q & \uparrow \end{array} \quad \begin{aligned} \cos 2\pi \frac{k}{P} P &= \cos 2\pi k = 1 \\ \sin 2\pi \frac{k}{P} P &= \sin 2\pi k = 0 \end{aligned}$$

stays intact while each delay is replaced by a delay and a complex heterodyne that is a function of i in each polyphase arm and *not* a function of n .

which is the same as for $n = 0$. Similarly, for $n = P + 1$, we can use the identities $\cos(A + B) = \cos A \cos B - \sin A \sin B$ and $\sin(A + B) = \sin A \cos B + \cos A \sin B$.

$$\begin{array}{ll} I & \rightarrow \quad \cos 2\pi \frac{k}{P} (P+1) = \cos 2\pi \left(k + \frac{k}{P} \right) = \cos 2\pi \frac{k}{P} (1) \\ Q & \uparrow \quad \sin 2\pi \frac{k}{P} (P+1) = \sin 2\pi \left(k + \frac{k}{P} \right) = \sin 2\pi \frac{k}{P} (1) \end{array}$$

which is the same as for $n = 1$. This means that the complex multiplications from $n = 0$ to $P - 1$ can be associated with each polyphase arm of the interpolator *even for the input samples arriving after $n = P - 1$* . A visualization of this approach is drawn at the bottom of Figure 10.10 for $L = 4$, $P = 8$ and an example $k = 1$. This becomes more clear when we draw the final block diagram for the efficient implementation of a QAM Tx.

Final Architecture

With this intuition at hand and employing a polyphase architecture for the interpolating bandpass filter, we construct the final Tx architecture shown in Figure 10.11. Some comments regarding this figure are in order.

- The multirate system consists of 3 different sample rates: $1/T_M$ for generating the QAM symbols, $4/T_M$ for pulse shaping and $32/T_M$ for interpolation. Thanks to the polyphase partitions, the shaping occurs at the lower rate $1/T_M$ and interpolation at $4/T_M$.
- Here, two polyphase partitions in I and Q arms are drawn, although a complex convolution is composed of 4 real convolutions between two complex signals, namely the pulse shaped QAM symbols and the bandpass interpolating filter. As discussed before, this is because only the inphase part of the final upconverted signal is required to generate $s(nT_S)$.
- Instead of employing the bandpass filter with complex coefficients, the same real lowpass filter is used for both convolutions and the spectral translation is left to the complex sinusoid at the end.
- The negative sign in the final summer comes from the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ for the real part carried over to convolution.
- Since the argument of the complex sinusoid is a constant as far as *each individual polyphase arm* is concerned, it can be placed at either side of the interpolating lowpass filter.
- Following the example in Figure 10.9, the value of k_{IF} is 3 in a length-8 DFT, i.e., 12 in a length-32 DFT.

On a final note, we saw in Section 2.7.2 that since an upsampled signal with a similar amplitude as the original signal has P times more samples, the filter should be designed with a gain P so that it maintains the same time domain amplitude in the upsampled signal as before while attenuating the extra $P - 1$ spectral replicas.

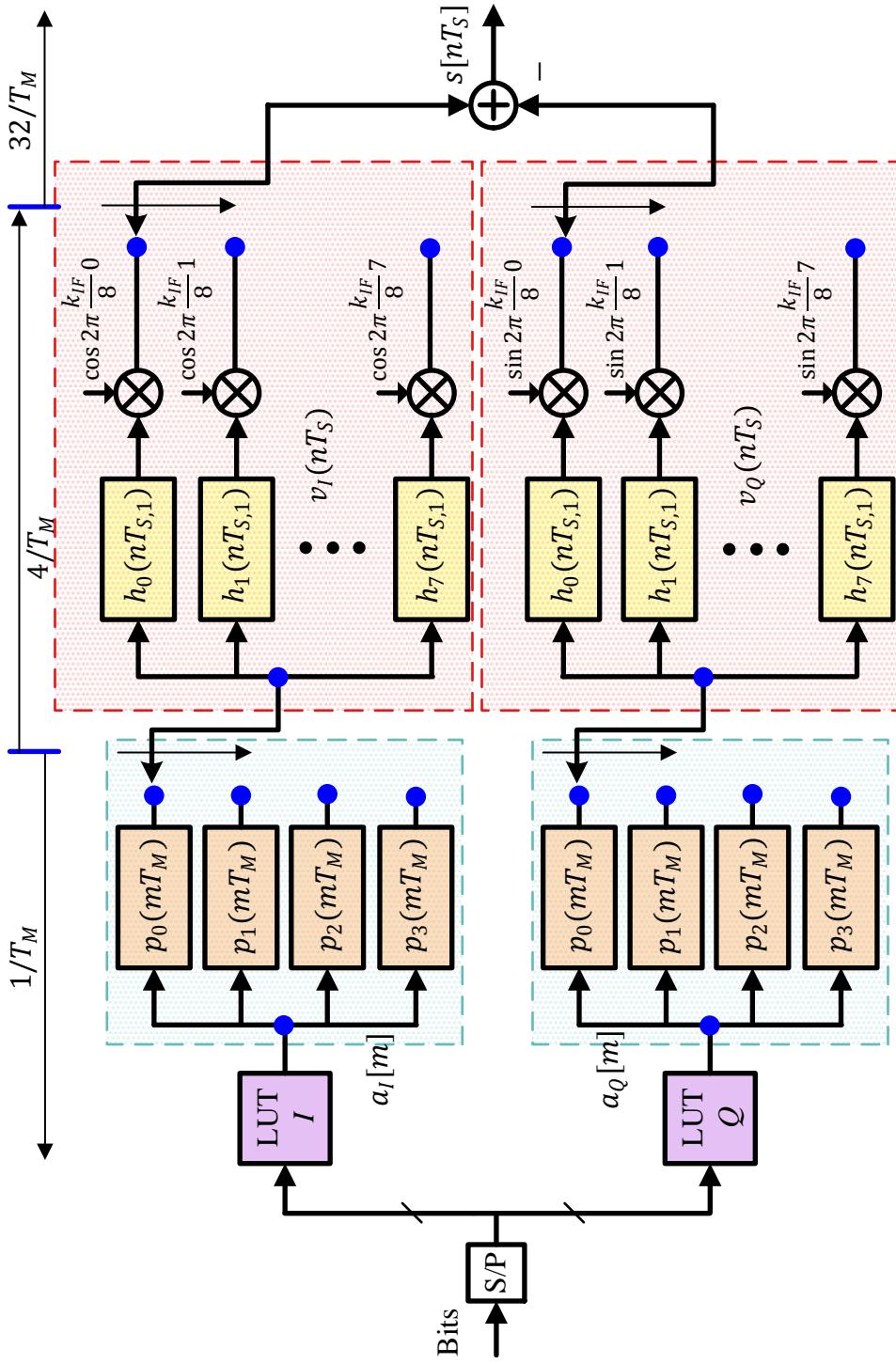


Figure 10.11: A QAM Tx architecture using efficient multirate signal processing for $L = 4$ and $P = 8$

After digital to analog conversion, the signal at the intermediate frequency can be upconverted to the carrier frequency in the analog frontend.

Next, we describe a digital frontend example for a Microchip radio transceiver AT86RF215. The AT86RF215 is a multi-band radio transceiver for various sub-1GHz bands and the 2.4GHz band specially designed for smart metering and applications IEEE Std 802.15.4gTM-2012, ETSI TS 102 887-1, IEEE Std 802.15.4TM-2015.

Example 10.1

According to its datasheet [48], the Transmitter Digital Frontend (TX_DFE) performs discrete time sampling rate conversion of the complex baseband signal. For this conversion, a zero-intermediate frequency (zero-IF) architecture is applied. The I/Q data sampling frequency f_s is to be configured according to

$$f_s = \frac{4}{SR} \text{ MHz}$$

where SR is defined by the register TXDFE.SR. The signal processing flow is depicted in Figure 10.12. The baseband signal at sampling frequency f_s can be pre-filtered at block PRE_FLT. A selection of suitable normalized cut-off frequencies f_{cut} (TXDFE.RCUT) is configurable. This filter stage can be utilized to relax interpolation complexity of the baseband processor (suppression of unwanted images). For $f_{cut} = 1$, the block is bypassed, reducing group delay. Up-sampling from f_s to 4MHz is accomplished by UP_SRC(1:SR), employing a linear phase FIR interpolation filter, with normalized cut-off frequency 1/SR. Finally, up-sampling to the DAC sampling frequency of 32MHz is accomplished by UP_SRC(1:8), employing three stages of linear phase FIR filters with normalized cut-off frequency 1/2.

As far as the Tx analog frontend for Microchip AT86RF215 is concerned [48], the transmitter is based on a direct up-conversion architecture[†]. After the digital to analog converter (DAC) the I-signals and Q-signals are passed through analog 2nd order low-pass filters (LPF).

We now turn our attention towards Rx architectures.

[†]Direct conversion is the same as zero-IF as we will see soon in Section 10.2.

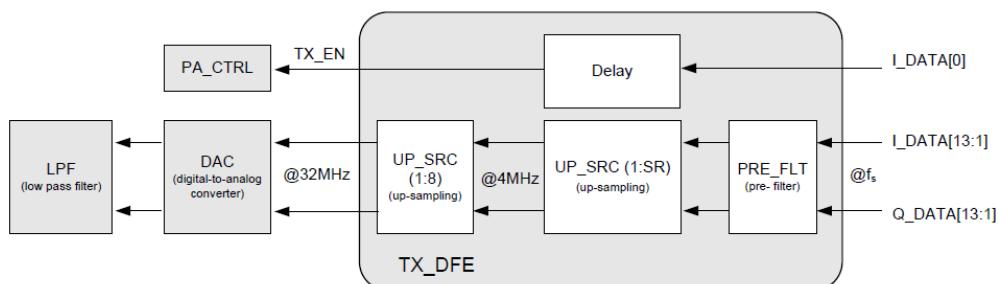


Figure 10.12: Tx upsampling and filtering in Microchip AT86RF215 [48], reproduced with permission

10.2 Rx Architectures

The discussion on different kinds of receiver architectures in this section is based on the excellent references [2], [49] and [50]. Let us begin with the fundamentals of frequency domain representation of real and complex signals as well as their frequency translations. In Section 2.9, we discovered that real time domain signals are conjugate symmetric in frequency domain which implies that

$$S[N - k] = S^*[k] \quad (10.10)$$

In terms of I and Q arms,

$$\begin{aligned} I &\rightarrow S_I[N - k] = S_I[k] \\ Q &\uparrow S_Q[N - k] = -S_Q[k] \end{aligned} \quad (10.11)$$

which is equivalent to

$$\begin{aligned} |S[N - k]| &= |S[k]| \\ \angle S[N - k] &= -\angle S[k] \end{aligned} \quad (10.12)$$

The DFT of a real time domain signal is drawn in Figure 10.13 to show the even symmetric I part and odd symmetric Q part. While this kind of symmetry is evident in 3D figure, the 2D figures are plotted to further highlight this concept.

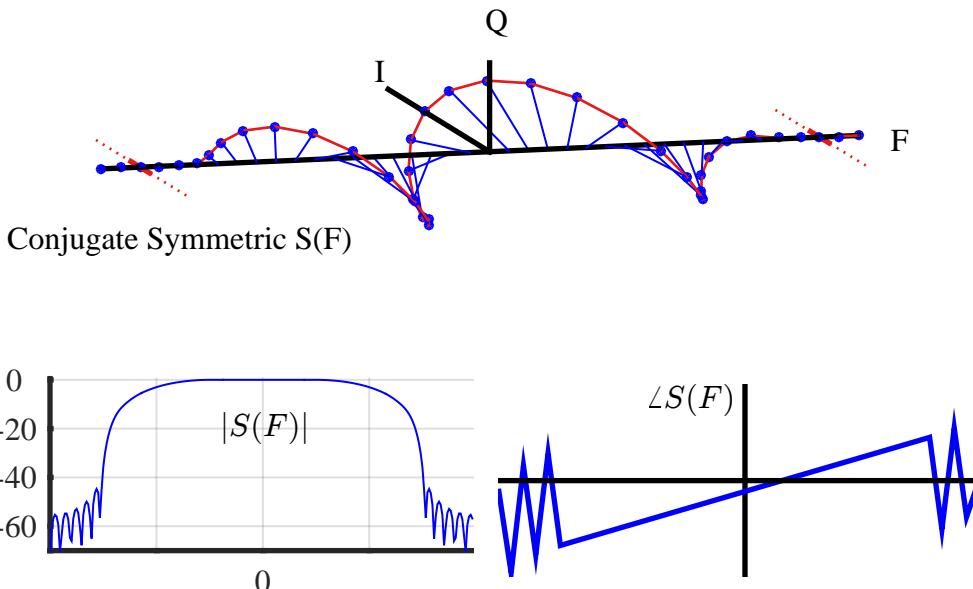


Figure 10.13: A real time domain signal possesses conjugate symmetry, i.e., it has an even symmetric I part (as well as magnitude) and odd symmetric Q part (as well as phase)

In regards to the radio architectures, selectivity and sensitivity play the most significant roles which are defined as follows.

Selectivity: Frequency selectivity is a measure of the receiver's capability to separate a desired band around a carrier frequency from the neighbouring signals.

Sensitivity: A receiver's sensitivity is the minimum power of the input signal with enough SNR to be successfully demodulated at the Rx.

The primary criteria while selecting a particular Rx architecture depends on the cost, complexity, power dissipation and the number of external components that cannot be integrated on the chip. With the advancements in Analog to Digital Converters (ADC) and the supremacy of digital signal processing, the architectures based on pushing the ADC towards the antenna are gradually dominating the radio frontend.

It is also important to distinguish between a *band* and a *channel*. The band includes the entire spectrum in which a radio following a particular standard is allowed to operate. For example, a WiFi Rx and a Bluetooth Rx both should be able to communicate in the band between 2.4 GHz to 2.5 GHz. The channel, however, refers to the signal bandwidth of one specific user within the band.

10.2.1 Tuned Radio Frequency (TRF) Rx

The tasks of the Rx to demodulate the Tx signal begin with selecting the signal within a specific bandwidth at a desired frequency, commonly known as a particular *channel*. To avoid interference from the neighbouring channels, the most straightforward approach is to filter out the spectral contents outside this channel. This was one of the earliest techniques employed for building radio receivers known as *Tuned Radio Frequency (TRF)* systems and consists of an adjustable, or tunable, Bandpass Filter (BPF) around the desired frequency, i.e., *the main idea is to move the filter to the signal*.

For example, the signal at frequency $F_{C,2}$ is selected through tuning the bandpass filter to frequency $F_{C,2}$ that removes all other spectral contents, as illustrated in Figure 10.14. If an adjacent channel were to be chosen, the tuning control mechanism consisting of circuit elements like variable capacitors and inductors just altered the center frequency of this filter. The major problem with a TRF is designing tunable bandpass filter with constant bandwidth and sufficient frequency selectivity over the entire tunable range.

Suffering from poor selectivity and low sensitivity, TRF receivers quickly became obsolete within the early years of radio. In today's age of fast digital electronics, there would have been no need to even mention them as a category for Rx architectures. Surprisingly, however, they do have a reincarnation in the realm of digital signal processing where it is straightforward to move the filter to the channel through appropriate conversion of a lowpass filter to a bandpass filter. Such an example was shown in Figure 10.9 in the context of an upconversion process which also has a counterpart in the downconversion procedure described later.

10.2.2 Heterodyne Rx

During World War I, Edwin Howard Armstrong invented the superheterodyne Rx as an alternative to the TRF receivers to overcome their limitations in regards to selectivity and sensitivity. Instead of employing a tunable bandpass filter that moves the filter to the signal, the concept of a superheterodyne Rx is to design a tunable Local Oscillator

(LO) operating at F_{LO} that *moves the signal to a fixed bandpass filter*, as drawn in Figure 10.15. This filter operates at an explicit frequency known as an *Intermediate Frequency (IF)*.

The advantage of such an approach is that regardless of the selected channel, most of the amplification and filtering operations are performed at the fixed intermediate frequency where it is relatively easier to design high gain amplifiers and filters exhibiting sharp transition bandwidths.

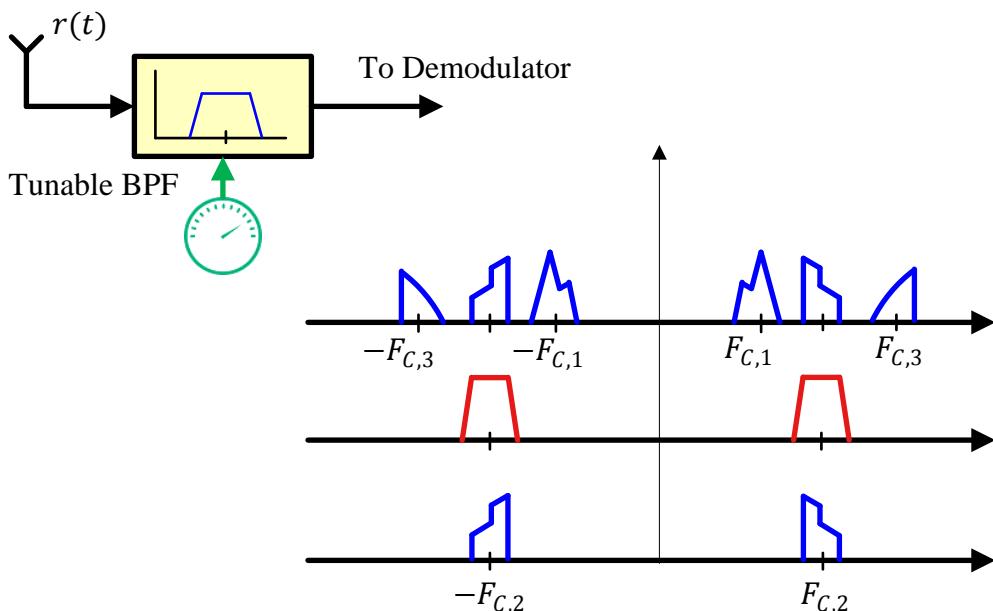


Figure 10.14: A tuned radio frequency receiver selects the desired channel through a bandpass filter

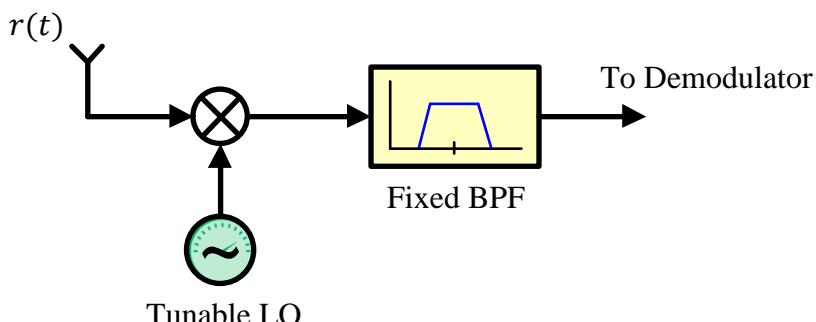


Figure 10.15: A tunable Local Oscillator (LO) selects a desired channel that is filtered through a fixed bandpass filter

The Heterodyne Principle

To understand its principle of operation, recall from Section 1.4 that the spectrum of a real sinusoid $\cos 2\pi F_C t$ at a carrier frequency F_C consists of two impulses, one at $+F_C$ and the other at $-F_C$. At the Tx, this sinusoid is mixed (i.e., multiplied) with the modulated signal $v(t)$ as

$$s(t) = v(t) \cdot \cos 2\pi F_C t$$

In frequency domain, a convolution takes place between those two impulses and $V(F)$. This convolution results in the spectrum $V(F)$ shifted to two frequencies, namely $\pm F_C$, thus producing the passband signal $S(F)$.

$$S(F) = \frac{1}{2} [S(F + F_C) + S(F - F_C)]$$

This resulting signal is drawn in the first row of Figure 10.16. At the Rx side, $S(F)$ is mixed with a tunable LO with a sinusoid $\cos 2\pi F_{LO} t$ at frequency F_{LO} .

$$x(t) = s(t) \cdot \cos 2\pi F_{LO} t = v(t) \cdot \cos 2\pi F_C t \cdot \cos 2\pi F_{LO} t$$

Using the identity $2 \cos A \cos B = \cos(A + B) + \cos(A - B)$, these two resulting real sinusoids at the Rx in time domain imply four impulses in frequency domain. The convolution occurs again, this time generating the copies of $V(F)$ at the following four frequencies.

$$\begin{array}{ll} +F_C + F_{LO} & -F_C - F_{LO} \\ +F_C - F_{LO} & -F_C + F_{LO} \end{array} \quad (10.13)$$

This principle of spectral translations through convolution with F_{LO} is plotted in Figure 10.16. Since the bandpass filter at the Rx is located at the IF, one of the above spectral replicas must fall at the same frequency. Assuming that this copy is $F_C - F_{LO}$ out of the four shown in Figure 10.16, the signal is downconverted to an IF equal to

$$F_{IF} = +F_C - F_{LO}$$

For a fixed F_{IF} and variable F_{LO} , we can capture any channel by tuning F_{LO} according to the above relation.

$$F_{LO} = F_C - F_{IF} \quad (10.14)$$

Since $F_{LO} < F_C$, this kind of mixing is known as *low side injection*. The other option is high side injection in which $F_{LO} > F_C$. Next, we investigate the image frequency problem in a heterodyne Rx.

The Image Frequency

While explaining the superheterodyne principle in Figure 10.16, we made an assumption that the whole spectrum only consists of our desired signal. In reality, the very concept of a spectrum is based on dividing the users in frequency domain by assigning them different frequencies, commonly known as Frequency Division Multiplexing (FDM). For this reason, much of the empty spectrum in Figure 10.16 is occupied by other transmissions.

This fact proves harmful for a Rx working with real sinusoids due to the following reason. Our spectral translation brings the desired signal $s(t)$ at carrier frequency F_C

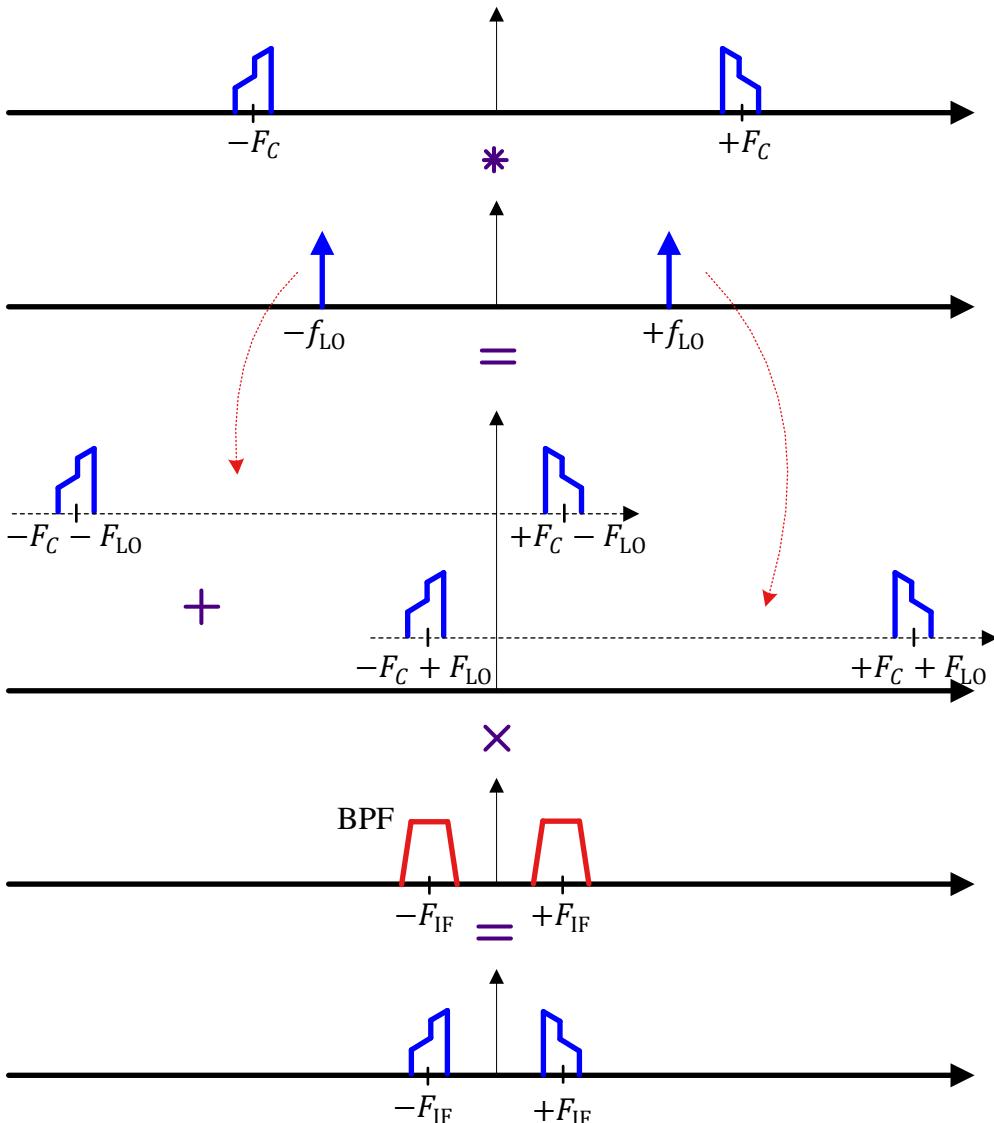


Figure 10.16: Principle of spectral translations in a heterodyne Rx

into the passband of the filter at intermediate frequency F_{IF} . As plotted in Figure 10.17, this is through convolution with the impulse at $-F_{LO}$. The same translation also brings another spectrum to F_{IF} due to the convolution with the impulse at F_{LO} . From inspection of this figure, it becomes clear that the spectrum interfering with the desired signal comes from a frequency located F_{IF} away from F_{LO} . To understand this point, measure the frequency difference between F_C and F_{LO} , and then between F_{LO} and F_{image} in Figure 10.17.

The above fact helps us in calculating the location of the image frequency F_{image}

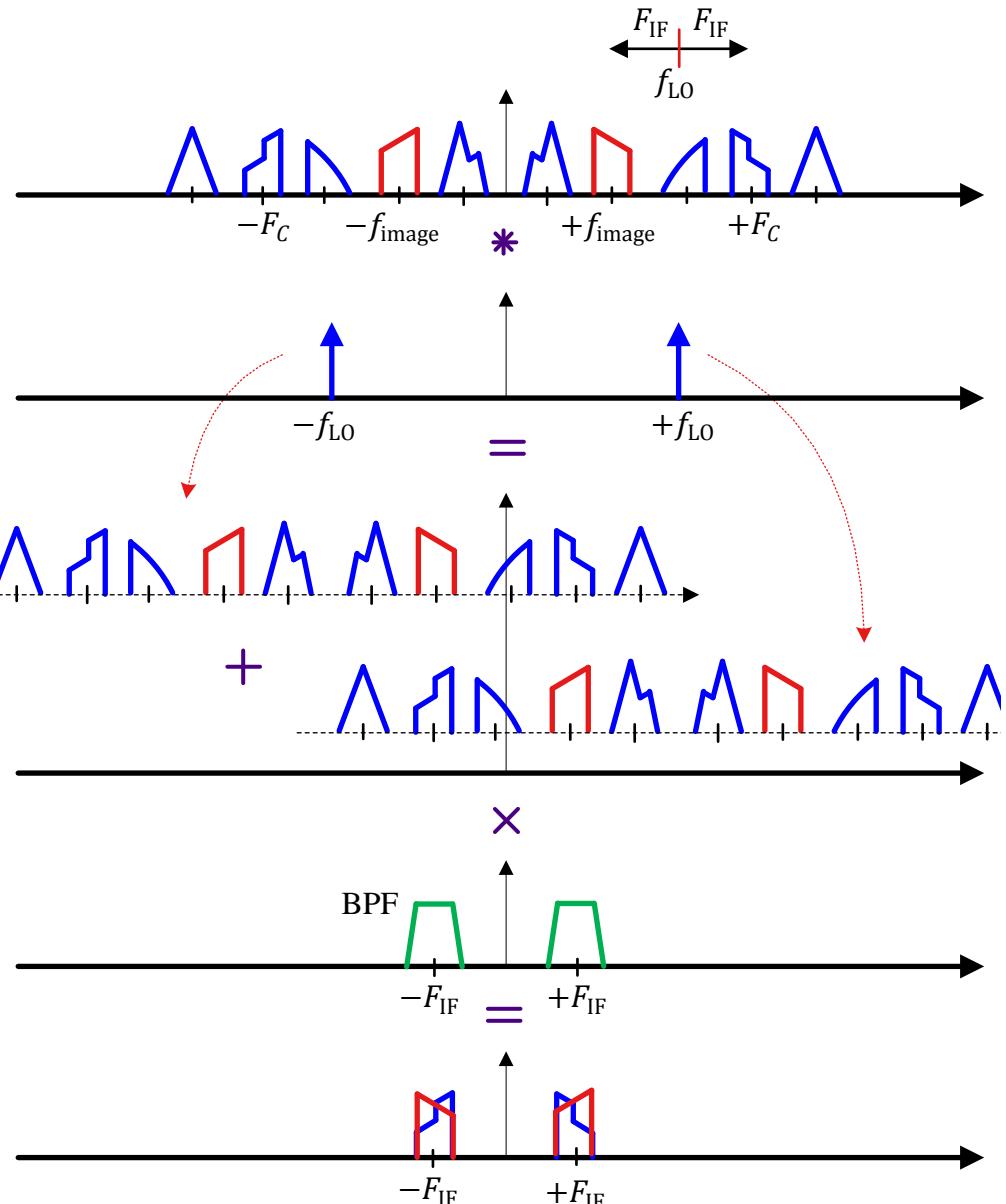


Figure 10.17: The image frequency problem of a superheterodyne Rx

for low side injection as

$$F_{\text{image}} = F_C - 2F_{\text{IF}} \quad (10.15)$$

Intuitively, this relation makes sense because keeping in mind the addition and subtraction of the sinusoidal frequencies, F_{IF} is reached from one side by the signal at F_C and from the other side by a signal that is at a further frequency difference of F_{IF} , as

shown at the top of Figure 10.17. Once the image frequency is in the mixer, there is no way to remove it since it is now heterodyned into the same IF band as the desired station.

The Superheterodyne Architecture

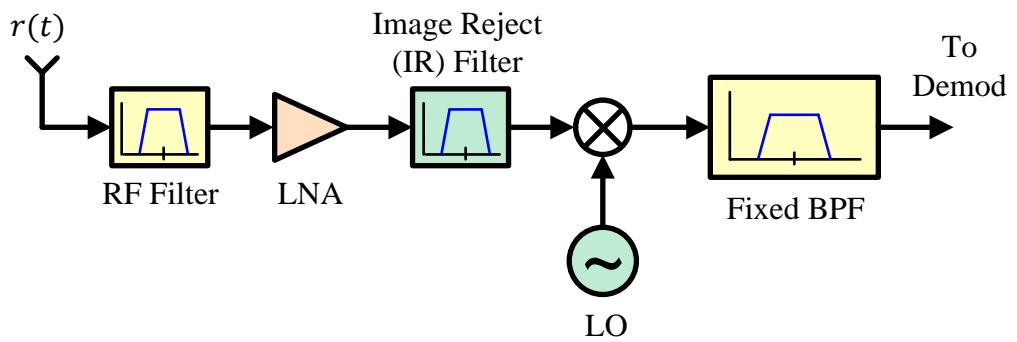
A superheterodyne Rx solves the image frequency issue by inserting an *Image Reject (IR) filter* prior to the mixer. This leads to a superheterodyne Rx architecture drawn in Figure 10.18a which works in the following stages.

- An RF preselection filter serves the purpose of removing out of band signal energy as well as partially suppressing the signal located at the image frequency.
- The signal is subsequently amplified by a Low Noise Amplifier (LNA).
- Next, the image frequency signal is cleaned up by an Image Reject (IR) filter. Whether the IR filter is fixed or tunable depends on the band of desired signals that determines where the image frequencies are located. In any case, the requirements on an IR filter are much relaxer than the TRF Rx since its only purpose is to filter out the image signal (as opposed to filtering out everything around the desired band) and the image frequencies lie away from the center frequency. This results in a large transition bandwidth and low cost for the IR filter.
- The signal at the IR filter output is multiplied or mixed with the output of a tunable Local Oscillator (LO) to downconvert the desired band to a fixed Intermediate Frequency (IF).
- Eventually the output can be shifted directly to baseband from here or further downconverted to lower IFs before final demodulation. If another stage of downconversion is utilized, such an architecture is known as a *dual-IF* receiver.

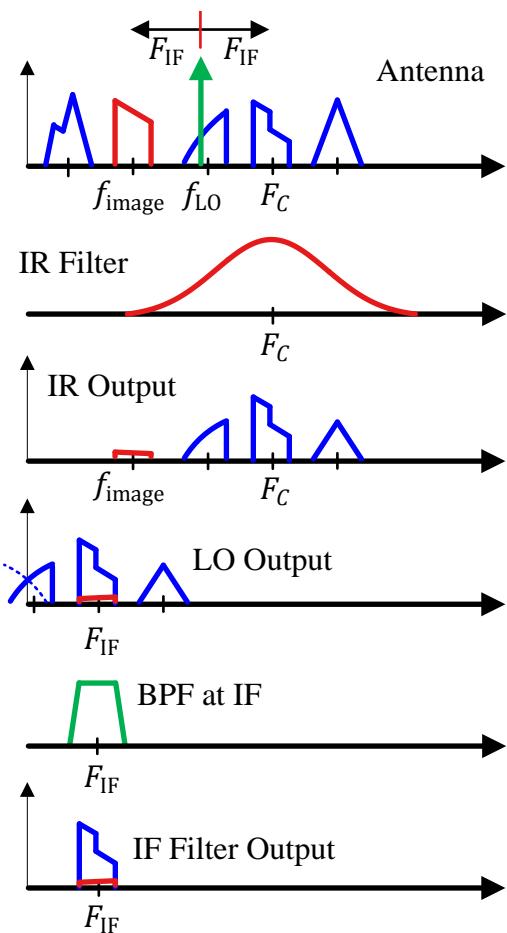
At this stage, it is imperative to have a look at the output signals at various stages during this process to get an insight into the design of an IR filter. This is illustrated in Figure 10.18b in which the difference between the transition bands of the IR filter and the fixed BPF is of particular note.

We have seen in Eq (10.15) that the desired signal and the image signal are separated by twice the IF. Recalling that a larger transition bandwidth puts relaxer constraints on a filter, *it is attractive to choose a high IF* so that the frequency spacing between the desired signal and the image signal is as large as possible. This is shown by the top arrows and a wide IR filter in Figure 10.18b. On the other hand, *a low IF allows utilization of high quality channel select filters* with better selectivity or out-of-band rejection. Consequently, the choice of IF depends on the tradeoff between image rejection and channel selection qualities. Since the image falls directly over the intended channel, it deteriorates the sensitivity of the Rx. This then translates into the more familiar sensitivity-selectivity tradeoff in a Rx.

Over the years, a heterodyne architecture has been widely used in communication receivers due to its good performance achieved by striking a balance in the tradeoffs mentioned above. Nevertheless, it requires more external (to the integrated chip) components due to multiple conversion stages and consequently occupies a larger form factor. Next, we find the root cause of the image frequency problem and solve it through another route that leads to a different Rx architecture.



(a) Block diagram



(b) Signals during different stages

Figure 10.18: A superheterodyne Rx

10.2.3 Zero-IF (Direct Conversion) Rx

Recalling that a real time domain signal has a conjugate symmetric spectrum, a real sinusoid has two impulses in its spectrum, one at a positive frequency $+F_{LO}$ and the other at the negative frequency $-F_{LO}$. A clue for the root cause of the image frequency problem appears when the LO frequency F_{LO} is equal to the carrier frequency F_C . In this scenario, the image band is the same as the band of the desired signal and hence cannot be eliminated by any image reject filter, see Figure 10.17 and put $F_{LO} = F_C$.

One solution to this issue is to employ complex signal processing and mix the signal with a complex sinusoid of frequency $-F_{LO}$ which has a single impulse in its spectrum at that frequency, see Figure 1.29. This operation produces a complex signal whose spectrum is simply a shifted version of the real signal spectrum due to convolution with this single impulse. This is drawn in Figure 10.19 for $F_{LO} = F_C$ where one-sided spectral translation of the real Rx signal from F_C to $F_C - F_{LO} = 0$ eliminates the need of an image reject filter. It must be remembered that the complete elimination happens in a theoretical sense only since imperfections in the analog frontend contribute towards a limited suppression.

With $F_{LO} = F_C$ option available as above, we can downconvert the Rx signal directly to the baseband. There are different terminologies used for this architecture as follows.

Direct conversion There are no intermediate stages and the translation is performed in one step.

Zero-IF The baseband spectrum can also be considered as an IF at $F = 0$.

Homodyne This terminology reflects a conversion philosophy different than a heterodyne architecture.

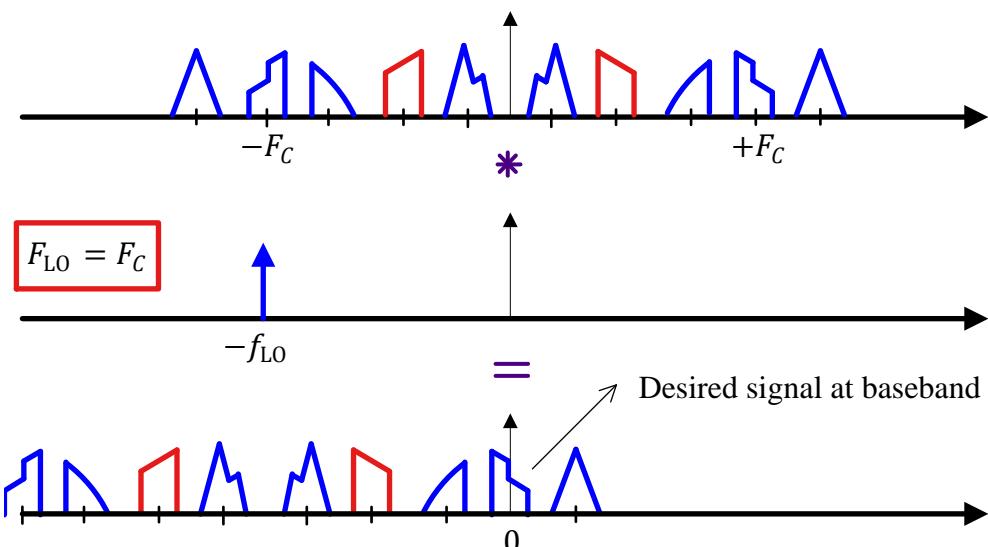


Figure 10.19: Mixing the real Rx signal with a complex sinusoid results in one-sided spectral translation

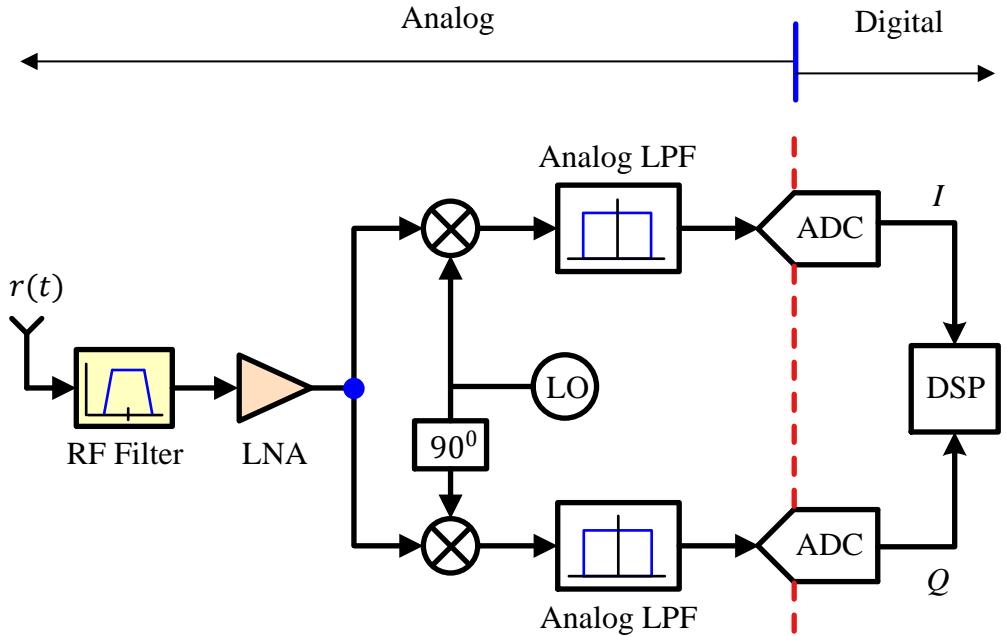


Figure 10.20: A zero-IF architecture downconverts the Rx signal to baseband

A block diagram for a zero-IF architecture is drawn in Figure 10.20. After the pre-selection filter and low noise amplifier, downconversion to baseband is accomplished through a pair of mixers and the quadrature sinusoids forming the complex sinusoid. The complex signal thus produced is filtered by two lowpass filters, one in the I arm and the other in the Q arm, that remove the double frequency term, select the desired band and suppress the adjacent channels. As a result, the lowpass filters need to exhibit sharp cutoff characteristics. Although the bulk of signal processing seems to happen in analog domain, the opposite is true. A direct conversion Rx simply down-converts the passband signal to the baseband and leaves the job of cleaning up all the real-world impairments to the DSP engine, some of which we will shortly see.

Challenges for Zero IF

While the concept of directly downconverting a signal seems the most logical option, there are a number of challenges that need to be overcome for realization of a zero-IF architecture.

DC offset This is the most severe problem at the baseband output signal of a direct conversion Rx. A conversion to the band around zero frequency makes the desired signal vulnerable to offset voltages that can saturate the subsequent stages. To understand this issue, consider that there are two inputs to the mixer, one coming from the LO and the other from the LNA. This results in a phenomenon known as **LO leakage** shown in Figure 10.21 that arises due to the following factors (the corresponding indices are marked on the figure).

1. The isolation between the mixer ports connected to the LO and the **LNA**

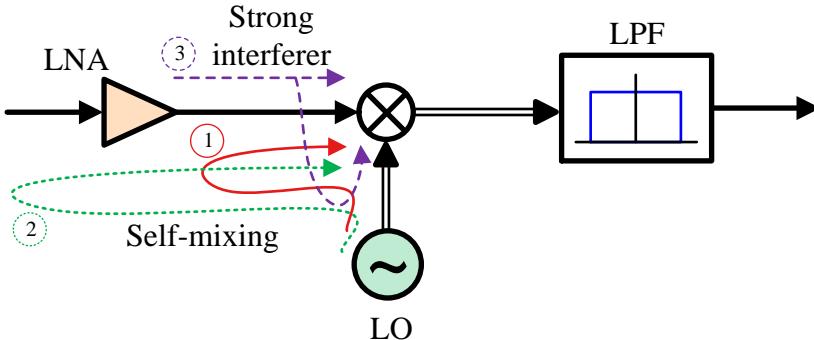


Figure 10.21: Self mixing of the LO in a zero-IF Rx

output is not infinite which causes the LO sinusoid to leak through to the other input of the mixer.

2. A similar argument holds for the *LNA input* as well.

Due to both of these leakages, the LO sinusoid is mixed with a delayed version of itself – known as *self-mixing* – thus producing a DC (Direct Current) component *at the mixer output* entering the lowpass filter. Assuming the LO signal as $A \cos 2\pi(k/N)n$ and the leaked version coming into the other mixer input as $B \cos(2\pi(k/N)n - \phi)$, we have

$$2A \cos\left(2\pi \frac{k}{N}n\right) \cdot B \cos\left(2\pi \frac{k}{N}n - \phi\right) = \underbrace{AB \cos \phi}_{\text{DC offset}} + AB \cos\left(2\pi \frac{2k}{N}n - \phi\right)$$

While the double frequency term is eliminated through the lowpass filter, the DC offset is summed with the desired signal within the baseband and saturates the subsequent electronic stages. From an SDR perspective, this is particularly harmful for the Analog to Digital Converter (ADC) that prepares the digital signal for DSP applications during demodulation. Moreover, this DC offset can vary with time when the LO signal leaks through the antenna, gets radiated outwards and reflected back to the Rx from nearby objects. For rapidly changing reflections, it can become difficult to isolate this time-varying offset.

3. To worsen the situation, a strong in-band interferer within the passband of the RF preselection filter also leaks from the LNA output to the mixer port connected to the LO and hence gets multiplied with itself creating another DC offset. This is also shown in Figure 10.21.

In addition to creating the DC offset issue, the LO sinusoid leaked through the antenna acts as an unmodulated carrier wave that deteriorates the performance of the nearby receivers operating in the same band by creating additional interference. It should be remembered that some of these issues are also present in a heterodyne Rx but to a much lesser extent. Since the desired signal falls at the intermediate frequency F_{IF} , the mixing may only arise from strong interferers within the band and the DC offset thus produced can be easily filtered out.

IQ imbalance In a zero-IF Rx, complex signal processing is employed through analog circuitry to directly shift the target spectrum to baseband as shown in Figure 10.20. For this purpose, both I and Q mixers, LPFs and ADCs must impart identical gain and 90° phase difference across the whole signal bandwidth in their respective I and Q arms, an impossible task to achieve in analog components. A figurative example of IQ imbalance is shown in Figure 10.22 as different sizes of the mixers, LPFs and the ADCs as well as the tilts. As a consequence, amplitude and phase mismatches occur between each pair of parallel sections in I and Q paths. Furthermore, quadrature mixing through a pair of real sinusoids (i.e., a complex sinusoid) is required for downconversion generated by a single LO. This is done by using the relation $\cos(A - 90^\circ) = \sin A$ and hence the cosine output of an LO is phase shifted by 90° to produce the sine output[†] (not shown in that figure). As expected, this phase shift is not exactly 90° due to the manufacturing tolerances.

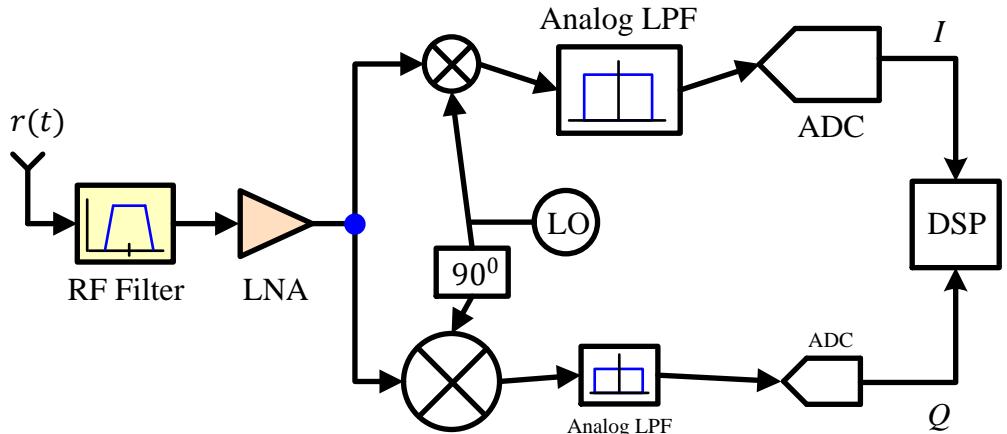


Figure 10.22: A figurative example of IQ imbalance depicting potential sources of mismatches

To see the effect of this IQ imbalance coming through the LO only, denote the amplitude or gain error of the LO as γ_Δ and phase error of the LO as θ_Δ and assume that half of the gain and phase errors lie on the I arm of the LO while the other half on the Q arm.

$$\begin{aligned} I &\rightarrow V_{\text{LO},I}(t) = \sqrt{2} \left(1 + \frac{\gamma_\Delta}{2}\right) \cos \left(2\pi F_C t + \frac{\theta_\Delta}{2}\right) \\ Q &\uparrow V_{\text{LO},Q}(t) = -\sqrt{2} \left(1 - \frac{\gamma_\Delta}{2}\right) \sin \left(2\pi F_C t - \frac{\theta_\Delta}{2}\right) \end{aligned} \quad (10.16)$$

where, as before, the factor $\sqrt{2}$ is used for normalization purpose. Now consider a 4-QAM waveform at passband arriving at the Rx antenna.

$$r(t) = v_I(t)\sqrt{2} \cos 2\pi F_C t - v_Q(t)\sqrt{2} \sin 2\pi F_C t$$

[†]Phase shifting the real RF signal by 90° is also an option which theoretically leads to similar baseband I and Q waveforms.

Next, we multiply the real signal $r(t)$ above with the LO complex output in Eq (10.16). Here, we use the regular trigonometric sum product formulas in addition to low-pass filtering the high frequency components.

$$\begin{aligned} I \rightarrow & \quad x_I(t) = v_I(t) \left(1 + \frac{\gamma_\Delta}{2}\right) \cos\left(\frac{\theta_\Delta}{2}\right) - v_Q(t) \left(1 + \frac{\gamma_\Delta}{2}\right) \sin\left(\frac{\theta_\Delta}{2}\right) \\ Q \uparrow & \quad x_Q(t) = v_Q(t) \left(1 - \frac{\gamma_\Delta}{2}\right) \cos\left(\frac{\theta_\Delta}{2}\right) + v_I(t) \left(1 - \frac{\gamma_\Delta}{2}\right) \sin\left(\frac{\theta_\Delta}{2}\right) \end{aligned}$$

Since $(1 + \gamma_\Delta)/2$ and $(1 - \gamma_\Delta)/2$ are constants (but not equal), notice that the above equations are very similar to those encountered during the study on effect of a phase offset on QAM constellation, see Eq (5.51) and Eq (5.52). Therefore, we can follow a similar derivation on our way to symbol rate sampling and matched filtering which yields the final result similar to Eq (5.3) as

$$\begin{aligned} I \rightarrow & \quad x_I(mT_M) = \left(1 + \frac{\gamma_\Delta}{2}\right) \left\{ a_I[m] \cos\left(\frac{\theta_\Delta}{2}\right) - a_Q[m] \sin\left(\frac{\theta_\Delta}{2}\right) \right\} \\ Q \uparrow & \quad x_Q(mT_M) = \left(1 - \frac{\gamma_\Delta}{2}\right) \left\{ a_Q[m] \cos\left(\frac{\theta_\Delta}{2}\right) + a_I[m] \sin\left(\frac{\theta_\Delta}{2}\right) \right\} \end{aligned}$$

- Notice from the above equation that when the gain error γ_Δ is zero, it reduces to the same equation regarding the phase offset in a QAM constellation.
- On the other hand, when the phase error θ_Δ is zero, the data symbols $a_I[m]$ and $a_Q[m]$ get scaled by different factors.

We have seen the effect of a phase offset θ_Δ in isolation on a QAM constellation. Now let us draw the gain error γ_Δ to view its effect in isolation from the phase offset. For this purpose, Figure 10.23a illustrates a 4-QAM signal constellation for a gain error of $\gamma_\Delta = -0.8$ while Figure 10.23b shows the combined effect of gain and phase errors.

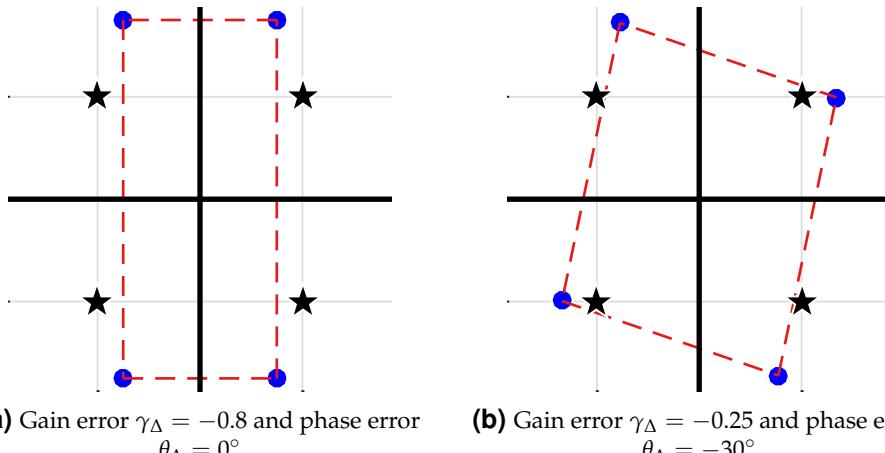


Figure 10.23: Effect of IQ imbalance on a 4-QAM constellation

In addition to an inaccurate mapping on the Rx constellation, such an imbalance affects the performance of the synchronization loops in the demodulator.

Advantages of Zero IF

The challenges mentioned above were significant enough that the zero-IF architecture mostly remained in the shadows in comparison to the superheterodyne Rx. In the past years, there have been a number of developments that contributed towards its widespread adoption and now can be found in most consumer radios. The discussion from here onwards heavily relies on Ref. [51].

Chip integration: With a growing list of applications, a wireless device needs to support multiple bands and more bandwidth. A direct conversion to baseband eliminates the need for an IF filter and IF amplifiers. A pair of lowpass filters at baseband are more easily integrated as active lowpass filters instead of lossy and off-chip fixed-IF devices. These active filters cover a wide range of bandwidth since they can be tuned from hundreds of kHz to hundreds of MHz. A wide range of RF frequencies can also be covered by simply adjusting the LO frequency. This flexibility coming from integrated programmable baseband filters greatly reduces the PCB footprint of a zero-IF radio.

Cost: Through elimination of some components and integration of others, the zero-IF design reduces the system cost as compared to other architectures. This is in addition to cost savings that come from the flexibility of adding new bands without rigorous planning.

Power: Since a zero-IF architecture directly reduces the frequencies of interest to baseband, the analog circuits are designed at the lowest frequency possible thus reducing power consumption.

The problems associated with such radios are usually controlled through analog optimization and digital correction. For both the DC offset and IQ imbalance, tracking algorithms are employed to compensate for their respective distortions. A device like AD9371 [51] is a typical example of a zero-IF transceiver covering a wide range of frequencies from 300 MHz to 6 GHz. Each Tx can cover between 20 MHz and 100 MHz of bandwidth while each Rx bandwidth span is from 5 MHz to 100 MHz.

One important point to remember here is that relatively imprecise tolerances still lead to successful signal recovery for lower-order modulation schemes such as BPSK and 4-QAM. On the other hand, for higher-order modulation schemes such as 64-QAM and 256-QAM, even little inaccuracies in DC offset and *IQ* imbalance add up and corrupt the baseband signal beyond repair. As the manufacturing tolerances cross certain thresholds, no amount of baseband DSP can then recover the data symbols. Nevertheless, as the technology improves in favour of a small form factor with digital calibrations, direct conversion receivers have become a sound choice in many digital radio systems.

10.2.4 Low-IF and Digital-IF Rx

As opposed to a conventional heterodyne and direct conversion receivers, it is also possible to implement some portion of downconversion steps in digital domain. This leads us to low-IF and digital-IF architectures. We begin with the low-IF Rx.

Low-IF Rx

The low-IF Rx combines the advantages of both the heterodyne and zero-IF receivers, a block diagram of which is illustrated at the top of Figure 10.24. As opposed to a zero-IF architecture, there are two downconversion paths but the second downconversion is performed in discrete domain. The important point here is that the *IF is quite low*, sometimes even just one or two times the signal bandwidth. The advantages of this approach are as follows.

- This strategy alleviates the DC offset issue because the desired signal is parked at a low frequency higher than the DC. The second digital mixer – which is just a Numerically Controlled Oscillator (NCO) and does not suffer from a DC offset – then shifts the original zero frequency to an out-of-band negative frequency when it translates the desired spectrum to zero.
- The issues arising from *IQ* imbalance due to quadrature downconversion in the first stage and then sampling the *I* and *Q* channels through two ADCs can be corrected in DSP domain through adaptive algorithms similar to synchronizers and equalizers.

The main disadvantage of this approach is that image and interference signals are present at the ADC input that reduce the Rx sensitivity if not filtered properly. As a result, sampling at IF requires a faster ADC with a higher resolution than that required for a zero-IF Rx to cope with these challenges in digital domain. A higher sample rate for the ADC implies a higher sample rate for subsequent signal processing stages before the downampler which increases the hardware cost.

In summary, the low-IF architecture is an excellent compromise between the superheterodyne and zero-IF architectures since it trades the challenges arising from a non-ideal analog part with the ADC. The improvements in the ADC performance over the past decades made this an attractive approach.

As an example, let us refer back to the analog frontend of Microchip AT86RF215 described in its datasheet [48]. After passing a low-noise amplifier (LNA), the received signal is down-converted to a low intermediate frequency (IF). A variable IF in combination with variable band-pass filtering provides flexibility with respect to channel selection at different channel spacing.

Digital-IF Rx

Instead of quadrature downconversion as in the low-IF Rx which introduces the *IQ* imbalance issue to be corrected later, it is also possible to digitize the incoming real signal in a heterodyne Rx at the IF stage through one ADC. This approach is known as a digital-IF architecture and is drawn in the bottom of Figure 10.24. The main advantages of this approach are the following.

- Just like a low-IF architecture, it eliminates the DC offset problem.
- Due to quadrature downconversion performed in discrete domain, the *IQ* imbalance issue is nonexistent because (a) the discrete-time sinusoids are just a sequence of numbers and hence equal gain and an exact 90° phase difference between them comes down to simply setting the right values in the registers, (b) the analog mixers are replaced by arithmetic multipliers, and (c) there is no

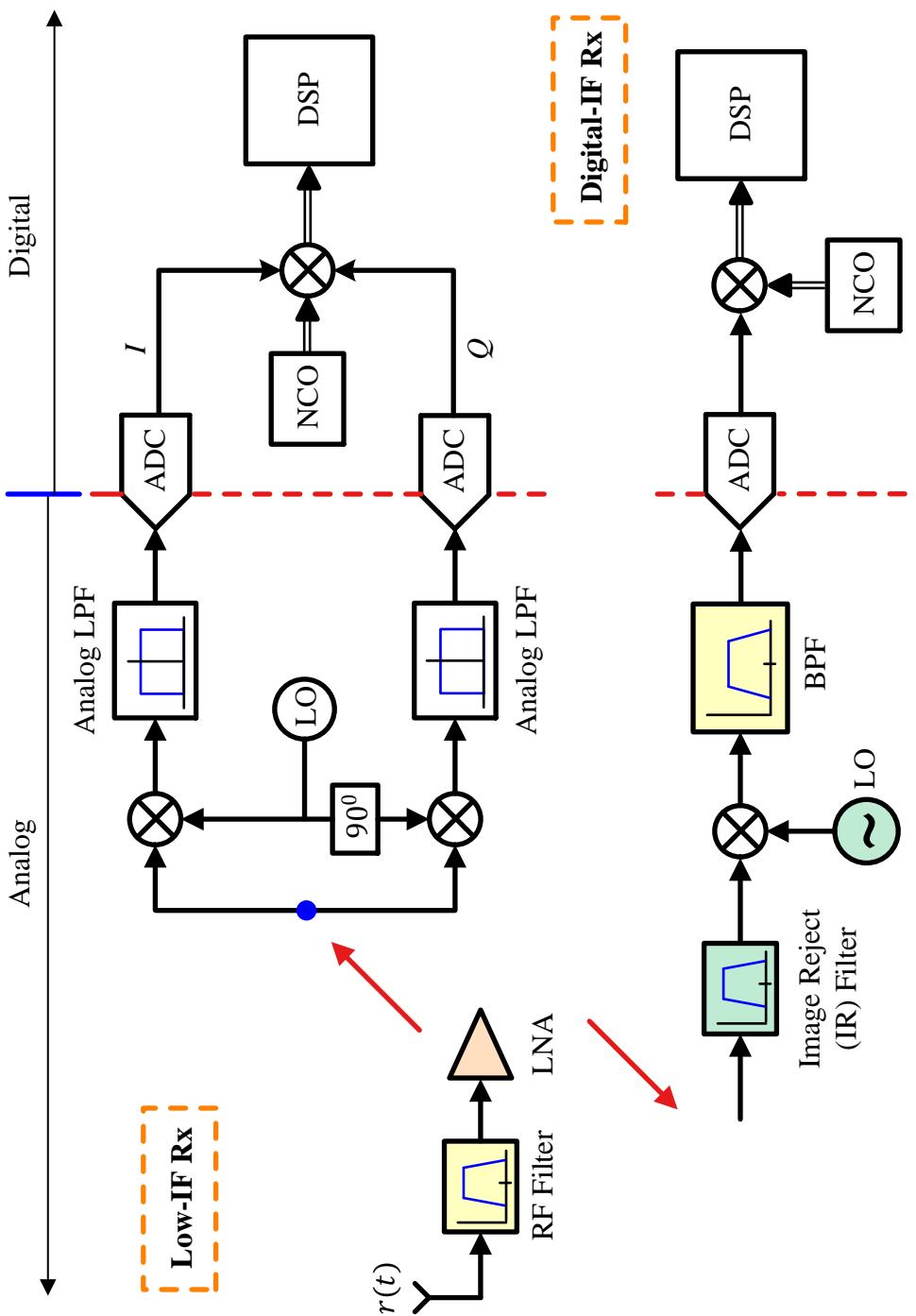


Figure 10.24: Block diagrams of the low-IF and digital-IF Rx architectures

gain difference issue between the ADCs because only one ADC is employed for sampling.

The main drawback is that conflicting ADC requirements in terms of a faster clock rate and a higher dynamic range are challenging to meet, as the sample rate is significantly higher than that required in other architectures. To some extent, we can reduce the cost of a higher sample rate by exploiting the concepts from multirate signal processing.

Example 10.2

Returning to Microchip AT86RF215 [48], the purpose of the Receiver Digital Frontend (RX_DFE) is discrete time sampling rate conversion of the complex baseband signal at the I/Q ADC interface. In addition, Automatic Gain Control (AGC) and estimation of the Received Signal Strength Indicator (RSSI) are provided by the RX_DFE. The signal processing flow is depicted in Figure 10.25. The analog to digital converter (ADC) provides the raw data of the complex low-IF baseband signal at a sampling frequency of 32MHz. At a first stage, the raw data is down-sampled to the sampling frequency of 4MHz (block DOWN_SRC(1:8)) and converted to zero-IF (DOWN_MIX). Depending on the configuration register RXDFE.SR, the signal is further down-sampled to the target receive sampling frequency:

$$f_s = \frac{4}{SR} \text{ MHz}$$

This is accomplished by block DOWN_SRC(1:SR), employing a linear phase FIR decimation filter with normalized cut-off frequency 1/SR. The discrete time baseband signal can be further filtered by block POST_FLT. A selection of suitable normalized cut-off frequencies f_{cut} (RXDFE.RCUT) is available. This filter can be used for attenuation of out-of-band signals (such as adjacent channels).

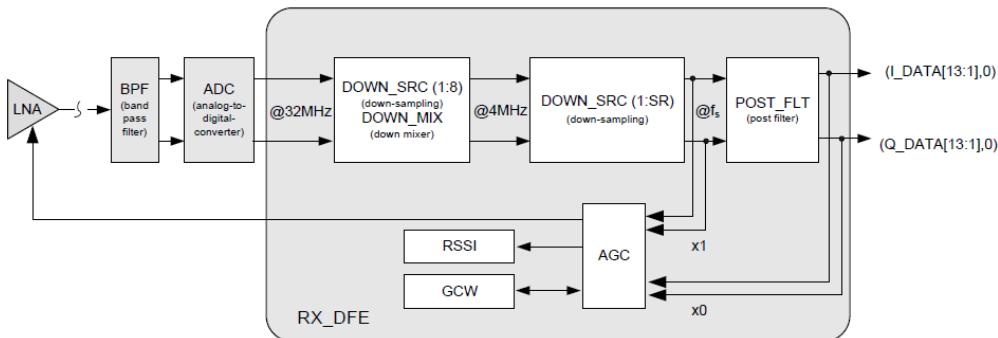


Figure 10.25: Rx downsampling and filtering in Microchip AT86RF215 [48], reproduced with permission

There are several other receiver architectures that are feasible either through slight variations in previous configurations or through advances in integration technology such as a bandpass sampling receiver. Next, we describe how multirate filters bring down the level of complexity in a digital radio by combining filtering and spectral translations in an efficient manner.

10.2.5 Polyphase Filterbank Implementation

A true software radio directly samples the signal at RF after the LNA and all signal processing operations are performed in digital domain. In the architectures described above, some radios downconvert the Rx signal directly to baseband through analog processing where a particular channel needs to be selected from the wide downconverted band. Other radios divide this task between analog and digital stages in which the signal is not sampled at baseband and the final downconversion with accompanying sample rate conversion is executed in digital domain.

Instead of emulating the series of analog signal processing operations or getting into details of a specific architecture, here we focus on how multirate signal processing results in an efficient system realization for any DSP radio. Therefore, from here onwards, we assume the availability of a digitized Rx signal $r(nT_S)$ and the target of the channelizer is to select a channel located at a higher frequency. The discussion here mostly follows Ref. [1] by fred harris but with an ease of exposition.

Consider an input signal at the Rx antenna that is bandlimited by frontend analog filters and sampled according to the Nyquist criterion. Our design target is to select and downconvert a single channel located at a higher frequency k_C/N . For simplicity, assume a unit sample rate $F_S = 1$ and the spectrum of the input signal consists of equally spaced channels at multiples of $F_S/P = 1/P$ with equal bandwidths. This setting places the channels at the frequencies that are integer multiples of $1/P$ as shown in Figure 10.26.

$$\frac{k_C}{N} = \frac{q}{P} \quad \text{for an integer } q$$

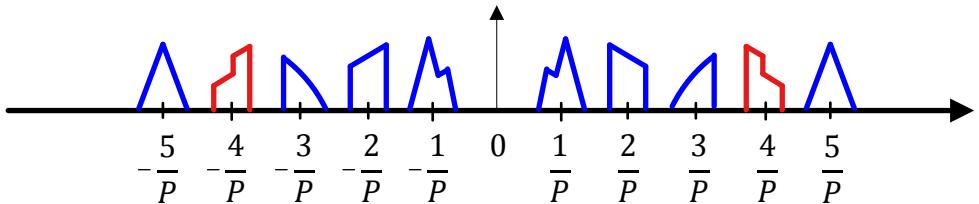


Figure 10.26: Spectrum of the digitized input signal contains multiple channels from which a single channel needs to be selected

Simple Lowpass Filters

When we imitate the analog signal processing operations, they are executed in the following order. A block diagram for this Rx is drawn in Figure 10.27 along with the corresponding signal spectra.

- A selected frequency band is downconverted through a complex heterodyne with a frequency $-q/P$.

$$\begin{aligned} I &\rightarrow & x_I(nT_S) &= r(nT_S) \cos 2\pi \frac{q}{P} n \\ Q &\uparrow & x_Q(nT_S) &= -r(nT_S) \sin 2\pi \frac{q}{P} n \end{aligned} \tag{10.17}$$

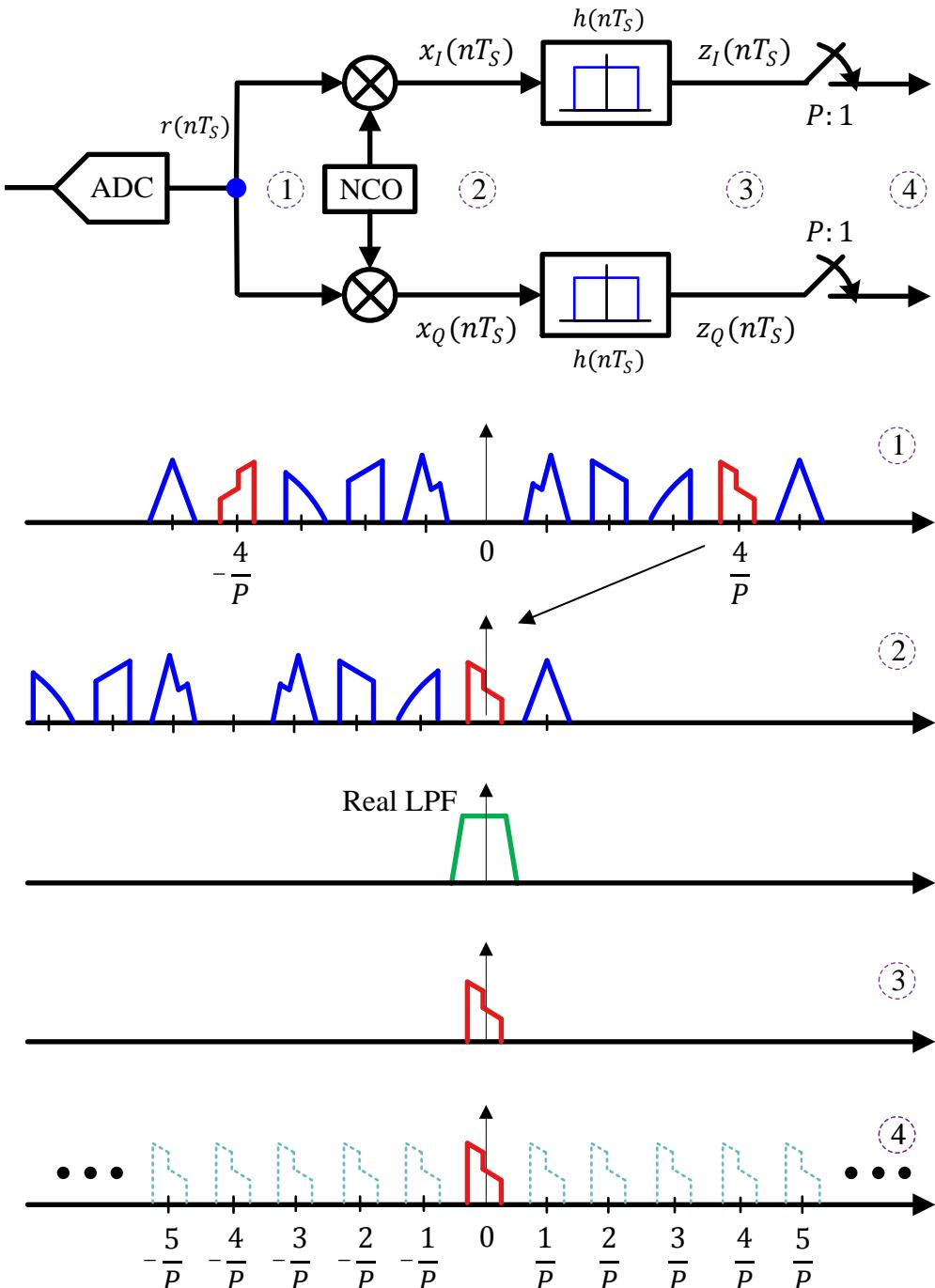


Figure 10.27: A block diagram of a direct conversion Rx implementing the lowpass filters and the signals at the corresponding stages

In Figure 10.27, we have $q = 4$. Furthermore, we ignore the scaling factor of $\sqrt{2}$

to keep the expressions simple.

- The baseband signal is lowpass filtered to match the desired channel bandwidth. Since the signal becomes complex after multiplication with the complex heterodyne, this convolution is carried out by two digital lowpass filters. From Eq (10.17),

$$\begin{aligned} I \rightarrow z_I(nT_S) &= h(nT_S) * x_I(nT_S) = \sum_i h(iT_S) r\{(n-i)T_S\} \cos 2\pi \frac{q}{P}(n-i) \\ Q \uparrow z_Q(nT_S) &= h(nT_S) * x_Q(nT_S) = \sum_i h(iT_S) r\{(n-i)T_S\} \sin 2\pi \frac{q}{P}(n-i) \end{aligned} \quad (10.18)$$

where we have used the identity $-\sin A = \sin(-A)$.

- Finally, a $P : 1$ downsampling in accordance with the channel bandwidth is carried out for reducing the load of the succeeding DSP stages. According to the sampling theorem, this generates spectral replicas of the desired signal at multiples of $\pm 1/P$ as drawn at the bottom of Figure 10.27.

It is clear that the downsampling by $P : 1$ in the final stage necessitates some changes in the preceding stages to make the system more efficient. Why compute the samples earlier that are discarded at the end? For this reason, we proceed towards replacing the lowpass filter with a bandpass filter that directly operates on the Rx signal and then downconvert it to baseband through resampling instead of the heterodyne.

Simple Bandpass Filters

As opposed to the heterodyne principle of moving the signal to the filter, we now apply the principle learned in Tuned Radio Frequency receivers in Section 10.2.1 of moving the filter to the signal. From Figure 10.27, it is straightforward to deduce that the low-pass filter can be transformed into a bandpass filter with a center frequency of q/P and then downconvert the result to baseband. We can also derive this through opening Eq (10.18) by using trigonometric identities $\cos(A - B) = \cos A \cos B + \sin A \sin B$ and $\sin(A - B) = \sin A \cos B - \cos A \sin B$.

$$\begin{aligned} I \rightarrow z_I(nT_S) &= \sum_i h(iT_S) r\{(n-i)T_S\} \left[\cos 2\pi \frac{q}{P} i \cdot \cos 2\pi \frac{q}{P} n + \right. \\ &\quad \left. \sin 2\pi \frac{q}{P} i \cdot \sin 2\pi \frac{q}{P} n \right] \\ Q \uparrow z_Q(nT_S) &= \sum_i h(iT_S) r\{(n-i)T_S\} \left[\sin 2\pi \frac{q}{P} i \cdot \cos 2\pi \frac{q}{P} n - \right. \\ &\quad \left. \cos 2\pi \frac{q}{P} i \cdot \sin 2\pi \frac{q}{P} n \right] \end{aligned}$$

Notice from the multiplication rule of complex numbers $I \cdot I - Q \cdot Q$ and $Q \cdot I + I \cdot Q$ that the above equation contains the product of two complex sinusoids with frequencies $+2\pi(q/P)i$ and $-2\pi(q/P)n$ in which only the former depends on the convolution

index i . Therefore, the term $+2\pi(q/P)i$ can be associated with the filter $h(iT_S)$ to transform it into a complex Band-Pass Filter (BPF)[†].

$$\begin{aligned} I &\rightarrow h_{BP,I}(iT_S) = h(iT_S) \cos 2\pi \frac{q}{P} i \\ Q &\uparrow h_{BP,Q}(iT_S) = h(iT_S) \sin 2\pi \frac{q}{P} i \end{aligned} \quad (10.19)$$

A block diagram depicting the operation of the complex bandpass filter is drawn in Figure 10.28 with the following observations.

- The two identical lowpass filters from Figure 10.27 have been replaced with a bandpass filter with I and Q parts described above. This is not an issue since a digital filter is just a set of coefficients in a processor memory and can be transformed free of cost by just changing the coefficients.
- Moreover, four multipliers and two adders are required instead of just two multipliers before. It seems that the filter translation to passband has resulted in a more complicated architecture. Utilizing the concepts from multirate signal processing, these inefficiencies can be ironed out to arrive at a structure very suitable from a system point of view, as we shortly find out.

Looking at the block diagram in Figure 10.28, we see that the $P : 1$ downsampling happens at the end of the signal processing chain. This means that the effect of discarding $P - 1$ samples out of every P samples needs to be taken into account in *all the blocks before the downampler*. At the top of this figure, we find two blocks before the downampler, namely the complex heterodyne and the bandpass filters, that need to be modified from this perspective. This is what we do next.

Modifying the Complex Heterodyne

There are two angles to look into modifying the complex heterodyne, time domain and frequency domain.

Time Domain View

The downsampling is performed with a ratio of $P : 1$. Then, according to Figure 10.28, the complex heterodyne carries out P complex multiplications and just 1 of them passes through the downampler. Discarding those $P - 1$ complex products is the same as moving the downampler before the complex heterodyne (which implies throwing $P - 1$ samples coming from the bandpass filters) and then generating only 1 complex multiplication.

Since downsampling indicates sampling at every P^{th} instant or $n = iP$, let us simplify the argument of the complex heterodyne.

$$2\pi \frac{q}{P} n \Big|_{n=iP} = 2\pi \frac{q}{P} iP = 2\pi q l$$

[†]This can be seen from modifying Eq (10.18), which displayed heterodyne first and lowpass filtering second, as follows.

$$z[n] = \sum_i h[i] \cdot r[n-i] e^{-j2\pi(q/P)(n-i)} = e^{-j2\pi(q/P)n} \sum_i r[n-i] \underbrace{h[i] e^{j2\pi(q/P)i}}_{\text{Bandpass filter}}$$

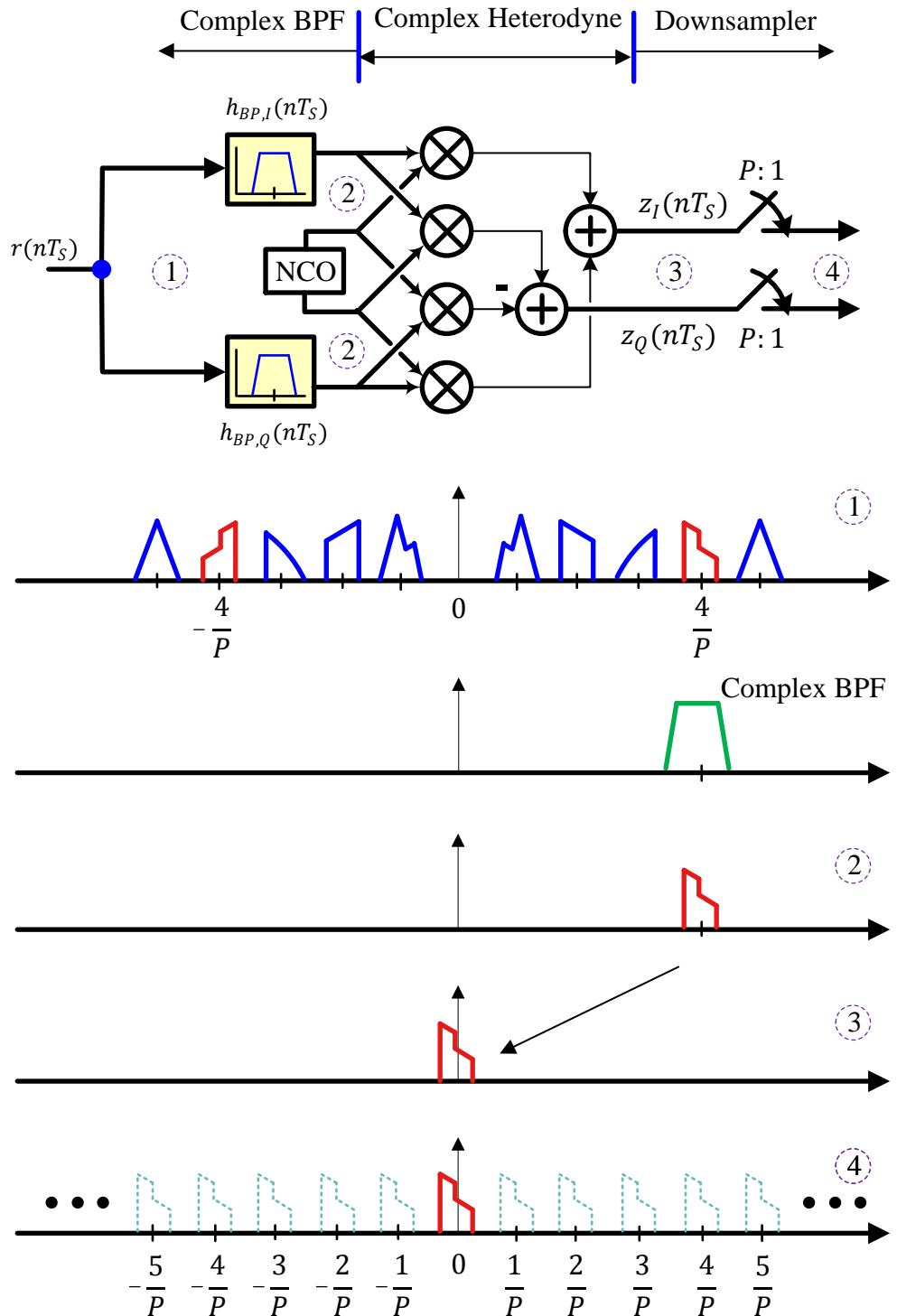


Figure 10.28: Moving the filter to the signal

where both q and l are integers. Clearly, the argument reduces to an integer multiple of 2π and the complex sinusoid degenerates to a constant equal to 1.

$$\cos 2\pi ql = 1 \quad \text{and} \quad \sin 2\pi ql = 0$$

We do not require the complex heterodyne anymore! However, the bandpass filter in Eq (10.19) and Figure 10.28 is complex and has complex phase rotators still associated with it as we see later. We also discover that it is more efficient to implement the system with a real lowpass filter and associate the complex multiplications with the polyphase arms, just like we did before in the Tx architecture.

Frequency Domain View

A hint for modifying the complex heterodyne comes from the signal spectra drawn at the lower half of Figure 10.28. We are reducing the bandwidth by a factor of P which in frequency domain can be taken as a rescaling of the frequency axis by P with the baseband spanning a bandwidth of $1/P$ around zero. As discussed in Section 2.7.1, $P - 1$ extra spectral copies arise as spectral aliases around the new sample rate.

In a general case, the signal center frequency k_C/N is unrelated to the spacing between the aliases *emerged after the downampler*, i.e., $1/P$. However, the signal center frequency is located at $k_C/N = q/P$ here and the spectral aliases are placed at other integer multiples of $1/P$. Then, one alias after downsampling lands right at the baseband (i.e., $0 \times 1/P$), as shown in Figure 10.29.

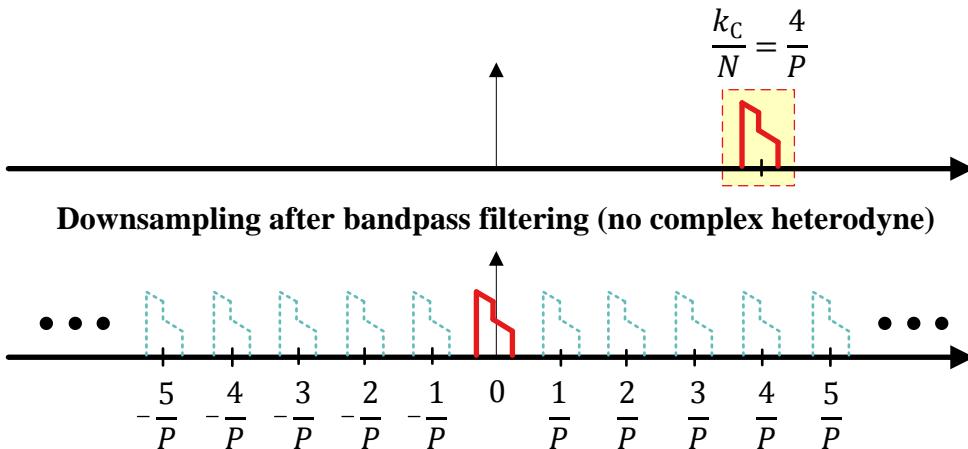


Figure 10.29: When $k_C/N = q/P$ for an integer q , one alias lands right at the baseband

We have shifted the $P : 1$ downampler operation before the complex heterodyne in Figure 10.28. Now we slide this $P : 1$ downampler even further to the left and modify the complex bandpass filter to accommodate the simplification.

Modifying the Lowpass Filter - Time Domain

Now we incorporate the concept of a polyphase filterbank to simplify the bandpass filtering operations. For this purpose, we start with the *original lowpass filter* that was

transformed into the bandpass filter and later extend the same concept later towards the bandpass filter.

We have covered the polyphase partition of a lowpass filter during the discussion about polyphase clock synchronization in Section 7.13 and that of a pulse shaping filter during the Tx architecture in Section 10.1.1. From here onwards, I assume that the reader has covered those topics. In particular, I highly recommend you to go through the material in polyphase clock synchronization before proceeding further.

Table 10.2: Polyphase partitions of the mother lowpass filter $h(nT_S)$ forming a polyphase filterbank

Polyphase Partition ↓	Filter Samples			
	→ → →			
$h_0(nPT_S) \rightarrow$	$h[0T_S]$	$h[PT_S]$	$h[2PT_S]$...
$h_1(nPT_S) \rightarrow$	$h[1T_S]$	$h[(P+1)T_S]$	$h[(2P+1)T_S]$...
⋮
$h_\delta(nPT_S) \rightarrow$	$h[\delta T_S]$	$h[(P+\delta)T_S]$	$h[(2P+\delta)T_S]$...
⋮
$h_{P-1}(nPT_S) \rightarrow$	$h[(P-1)T_S]$	$h[(2P-1)T_S]$	$h[(3P-1)T_S]$...

Instead of a 1-dimensional array, we partition a mother filter by loading the filter taps into a 2-dimensional matrix such that the first P elements are written into the 1st column, the next P elements are written into the 2nd column, and so on. This generates P polyphase partitions corresponding to each row of the table, shown in Table 10.2. Looking at the filter time indices, it is clear that $1/T_S$ is the rate of the input signal as well as the mother filter while $1/PT_S$ is the rate of the partitions.

With this setting in place, we can utilize the polyphase partitions for the mother filter to construct a parallel architecture such that the convolution with each such partition happens at the low sample rate $1/PT_S$. Follow the sequence of operations in Figure 10.30 for $P = 4$ which illustrates the high rate ($1/T_S$) operations from the perspective of polyphase arms. Again, this figure looks like the game CONNECT4 which I play with my son.

The convolution sum at each time nT_S is defined as

$$\sum_i r(iT_S)h(nT_S - iT_S) = r(0T_S)h(nT_S) + r(1T_S)h(nT_S - 1T_S) + r(2T_S)h(nT_S - 2T_S) + r(3T_S)h(nT_S - 3T_S) + \dots \quad (10.20)$$

- At time $n = 0$, the first sample of the original time series $r(nT_S)$ arrives from the doorstep of partition 0. From here onwards, we assume that any overlap implies a multiplication operation between the filter coefficient and the input sample

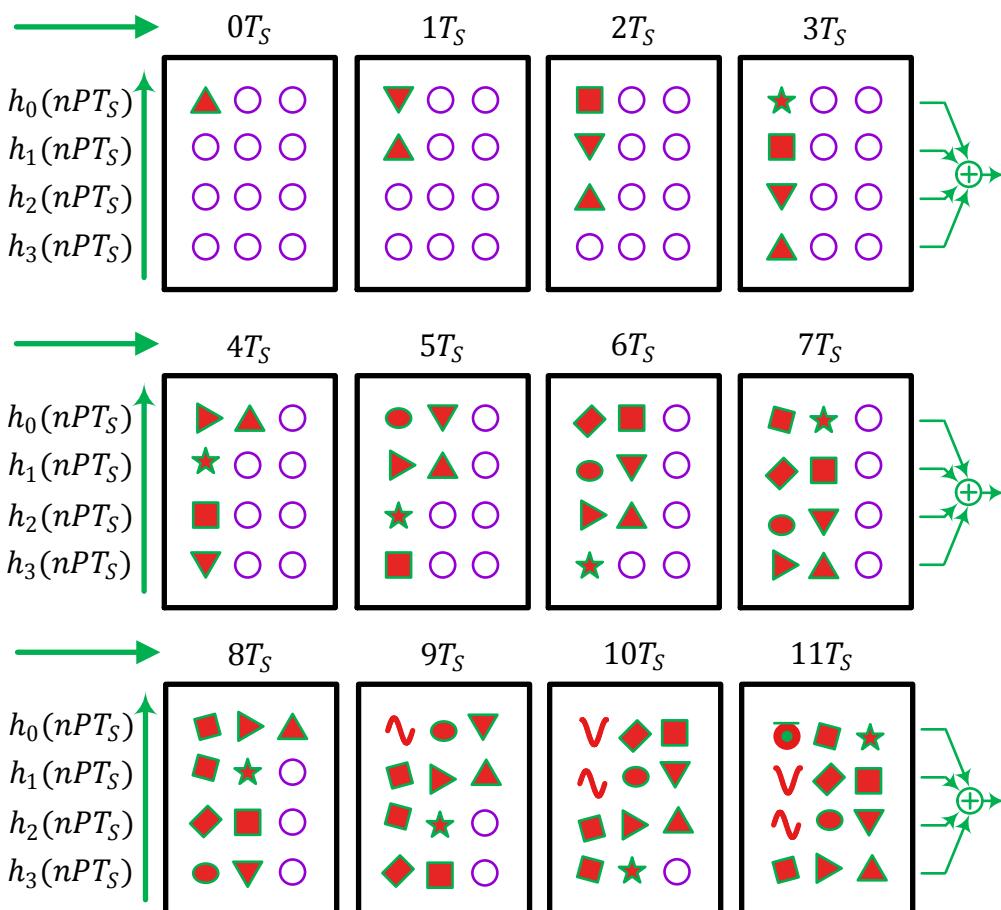
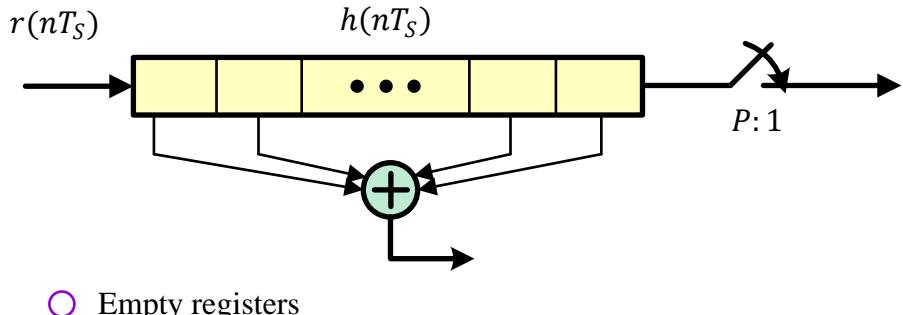


Figure 10.30: A downsampling polyphase architecture loads the input samples by columns at a rate $1/T_S$ and reads the output samples at the summation junction at a rate $1/PT_S$ where $P = 4$. The convolution at each partition happens at the low rate $1/PT_S$

present at the same location. For example, the first input sample $r(0T_S)$ is a little triangle and gets multiplied with $h(0T_S)$ in the top left box at instant $0T_S$ of Figure 10.30.

$$n = 0 \rightarrow r[0]h[0]$$

- At the next instant $n = 1T_S$, this sample moves down to partition 1 and the second input sample (an inverted triangle) moves in. Now two products are formed here but still no summation is carried out.

$$n = 1 \rightarrow r[0]h[1], r[1]h[0]$$

- This process repeats itself as the new samples keep arriving and the products can be formed at each multiple of T_S . For example, at $n = 3T_S$ in the last column of Figure 10.30 where $r(3T_S)$ is a star and overlaps with $h(0T_S)$, four products are formed.

$$n = 3 \rightarrow r[0]h[3], r[1]h[2], r[2]h[1], r[3]h[0]$$

Downsampling implies that the output result is only needed at this instant. So the summation is performed at this very moment. Compare the summation of these terms after plugging $n = 3$ in the convolution expression of Eq (10.20)! They are exactly the same.

Notice that the summation of the products between inputs and overlapping filter coefficients shown at the far right in Figure 10.30 is taken for $P = 4$ at $3T_S, 7T_S, 11T_S, \dots$ while the sum for intermediate instants is not required. Therefore, we can simply load the input values from bottom to top shown by green arrows on the left.

The last column (i.e., the instants spaced by $P = 4$) clearly demonstrates that this convolution is equivalent to the original data samples $r(nT_S)$ successively forming the input to the filterbank from the bottom to the top and the output simply taken from the final summation at rate $1/PT_S$. Such successive loading is said to be performed through a commutator shown as an *upwards* moving pointer in an efficient implementation at the left of Figure 10.31.

In case you are wondering how different is this downsampling architecture from an upsampling architecture, the answer is that *it is just a dual of an upsampling process*. From the construction of flow graphs, we know that a dual of a flow graph is formed by performing the following operations.

- reversing the directions of all arrows,
- reversing the coefficient ordering,
- turning nodes into summing junctions, and
- turning summing junctions into nodes.

When you perform the above replacements in Figure 10.31 starting from the right, the architecture turns into that of the upsampling case drawn in Figure 10.4. While all the other steps make sense, a node turning into a summation is quite puzzling. In time domain, we can see this as the summation part of the convolution, but it will play a beautiful and central role in the frequency domain view of the downsampling process.

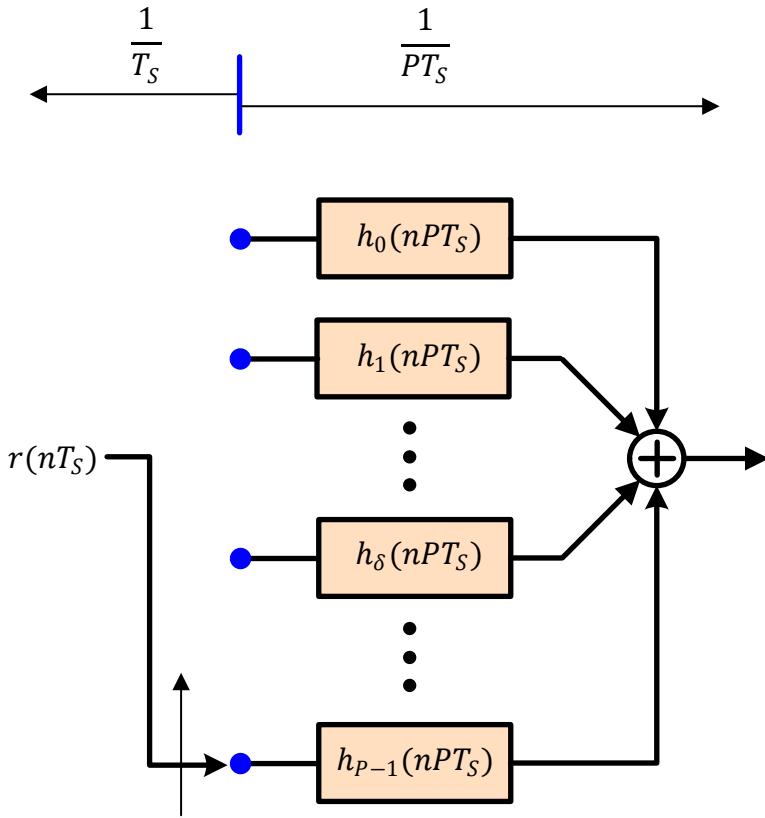


Figure 10.31: Efficient downsampling at a rate $1/PT_S$ through the polyphase children

Modifying the Lowpass Filter - Frequency Domain

For reasons that will become clear soon, we need to revisit the process of creating the polyphase partitions of the mother filter in the context of downsampling procedure. We begin with the introduction to downsampling from Section 2.7.1 and fuse some pieces of information through utilizing the sampling sequence from Section 1.10.4. For this purpose, the length- N mother filter is shown in Figure 10.32a along with its spectrum in Figure 10.32b.

Child Filter 0

To begin, the polyphase child $h_0(nPT_S)$ comes from a simple downsample operation as follows.

- A $P : 1$ downsampling operation can be *imagined* as multiplying the impulse response $h(nT_S)$ of the original filter with a sampling sequence of period P shown in Figure 10.32c and then throwing off the intermediate zero samples. In practice, such zero values are not computed through multiplication, as they are immediately going to be discarded anyway.

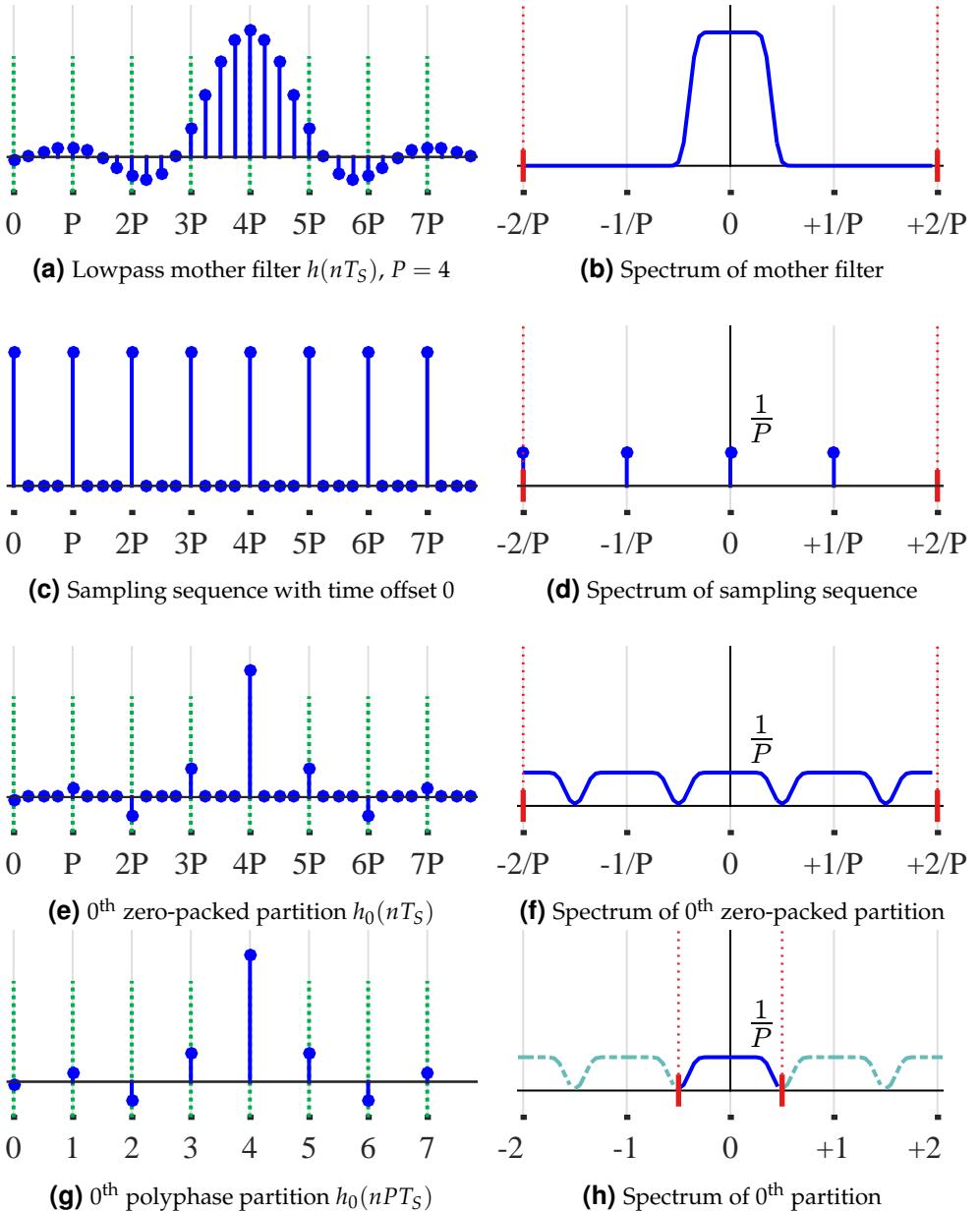


Figure 10.32: A figurative process to generate 0th polyphase partition that proves useful in deriving its spectrum

- This multiplication in time induces a convolution in frequency between their spectra.
- We all know how the spectrum $H[k]$ of a lowpass filter looks like. On the other hand, we saw in Figure 2.28 that the spectrum of a sampling sequence consists of P impulses at frequency bins that are integer multiples of $k = \pm N/P$. Since

we are concerned with the discrete frequencies k/N instead of their indices k here, we normalize these values by $1/N$ and hence the impulses are located at integer multiples of $\pm 1/P$. Furthermore, these spectral replicas are scaled down by a factor of $1/P$ and drawn in Figure 10.32d.

- Combining these with the fact that convolution of a signal with a unit impulse is the signal itself, the result of this spectral convolution between signals in Figure 10.32b and Figure 10.32d is the emergence of P spectral copies of $H[k]$ at $\pm \text{integer} \times 1/P$, all having a spectral amplitude of $1/P$. This is drawn in Figure 10.32f and this is the spectrum of the product between the two sequences. This can also be viewed as *a downsampled but zero-packed version of the mother filter with time offset 0*. This spectrum is the reason why we took this route as we shortly see how this takes a central role in the understanding of downsampling a signal through a polyphase filterbank.
- When the zeroed values are discarded from the zero-packed signal, we are left with the 0^{th} child filter $h_0(nPT_S)$ which is shown in time and frequency domains at the bottom of Figure 10.32. This step reduces the bandwidth by a factor of P , which in frequency domain can be taken as a rescaling of the frequency axis by P with the baseband around $k = 0$. As a result, $P - 1$ extra spectral copies now act as spectral aliases around the new sample rate at integer multiples of $1/P$. That is why these are drawn with shaded line in Figure 10.32h.

Next, we examine the product of the mother filter with a unity shifted sampling sequence to generate zero-packed time series $h_1(nT_S)$. Later, this zero-packed sequence is downsampled by P to create the child filter 1, i.e., $h_1(nPT_S)$.

Child Filter 1

As for the rest of the partitions $h_\delta(nPT_S)$ where $\delta = 1, 2, \dots, P - 1$, they are also $P : 1$ downsampled versions of the mother filter with a little difference: *the sampling sequence is shifted right by δ each time*. In time domain, the effect is that the *timing offset* of each such partition is right shifted by δ samples as compared to the time base of the 0^{th} zero-packed partition of Figure 10.32e. To see the effect in frequency domain, recall from Section 1.10.4 that the spectrum of a δ -shifted sampling sequence is phase rotated for each k by a complex sinusoid with inverse period given by δ .

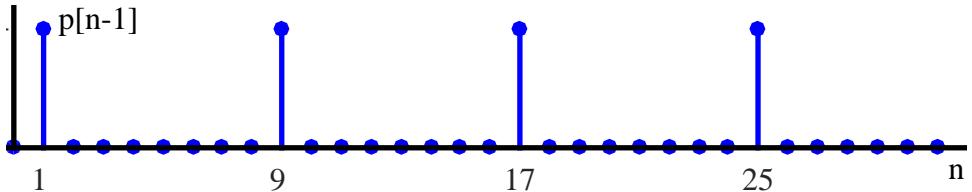
$$\text{phase rotation} = 2\pi \frac{k}{N}(-\delta) = -2\pi \frac{q}{P}(\delta) \quad (10.21)$$

While we continue our study with $P = 4$ in the subsequent discussion, let us consider an example spectrum for $\delta = -1$ and $P = 8$ just because the resultant spectrum looks more beautiful. The shifted sampling sequence and its spectrum are drawn in Figure 10.33 for a right shift of 1. Here, the phase shift for each such spectral impulse is determined as

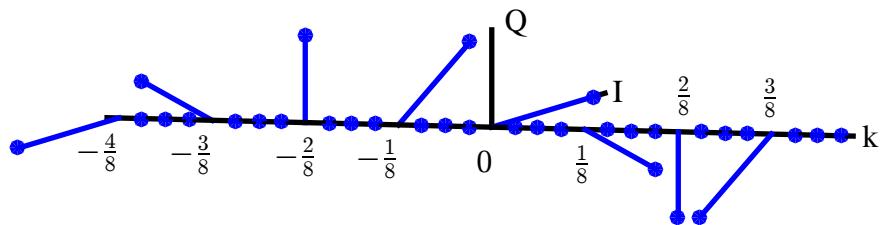
$$\text{phase rotation} = 2\pi \frac{q}{8}(-1) = q \times (-45^\circ)$$

You can spot these phase shifts in Figure 10.33b as -45° for the impulse at $1/8$, -90° for the impulse at $2/8$ and so on.

$$\begin{aligned} \text{phase rotation} &= \pi, +\frac{3\pi}{4}, +\frac{\pi}{2}, +\frac{\pi}{4}, 0, -\frac{\pi}{4}, -\frac{\pi}{2}, -\frac{3\pi}{4} \\ &= 180^\circ, +135^\circ, +90^\circ, +45^\circ, 0, -45^\circ, -90^\circ, -135^\circ \end{aligned}$$



(a) Sampling sequence shifted by 1



(b) Spectrum contains a phase shift of $\pm 2\pi(q/P)(-1)$ radians for each q

Figure 10.33: Spectrum of a sampling sequence shifted in time domain consists of phase rotated impulses

In time domain, this sequence in Figure 10.33a is multiplied with the mother filter that necessitates a convolution in frequency domain. When the mother filter spectrum is convolved with the spectrum of the shifted sampling sequence of Figure 10.33b, this raises $P - 1$ spectral replicas of the mother filter at integer multiples of $1/P$, just like in the case of $h_0(nT_S)$ of Figure 10.32f (where P was chosen as 4). Nonetheless, *the crucial difference is that these replicas are rotated by the phases* defined by the above expression as multiples of -45° . I have drawn the spectrum of $h_1(nT_S)$ in Figure 10.34 for a clear demonstration of the rotated spectra. The top plot shows a regular 3D view while the bottom plot illustrates a sideways angle to clearly distinguish the phases involved. Notice that $P = 8$ spectral copies within the baseband inherit their respective phases from the spectral impulses of Figure 10.33b.

Reader is reminded that this is the spectrum of the zero-packed series $h_1(nT_S)$, and not of $h_1(nPT_S)$ as we have not discarded the intermediate zeros yet. Something that will help later is that for each spectrum with a certain angle, there is a corresponding spectrum with an opposite angle. This fact is nowhere more evident than the bottom plot of Figure 10.34. These are said to be lying at *P roots of unity*. Many students encounter this concept in the form of cube roots of unity learned at school which is a discontinuous jump in their so far linear mathematical progression.

Narrowband Beamforming

Let us go back to plotting the spectra for the remaining zero-packed partitions for $P = 4$. Following the same procedure, this is shown in Figure 10.35 for $h_\delta(nT_S)$, where

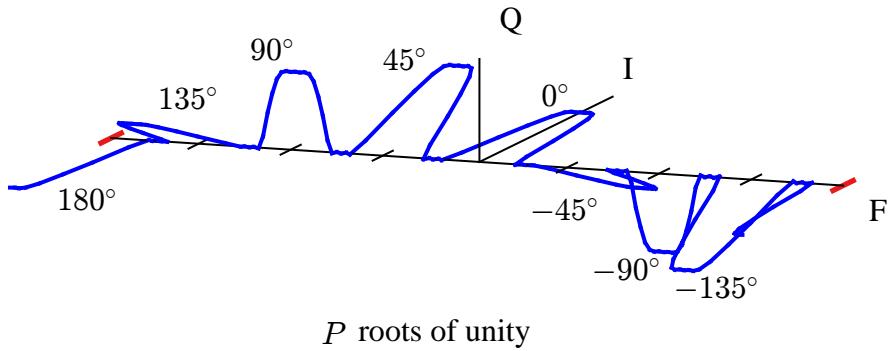


Figure 10.34: Spectrum for $\delta = 1$ and $P = 8$ before discarding the zeros – i.e., the spectrum of $h_1(nT_S)$ (and not of $h_1(nPT_S)$)! Both plots demonstrate from different angles the same eight phases given by multiples of $2\pi/8 = 45^\circ$

$\delta = 0, 1, 2, 3$. Observe that the spectral phases are given by Eq (10.21) as follows.

$$\begin{aligned}
 \text{Phases for } h_0(nT_S) &\rightarrow \text{Integer multiples of } \frac{2\pi}{4}(0) = 0 \\
 \text{Phases for } h_1(nT_S) &\rightarrow \text{Integer multiples of } \frac{2\pi}{4}(-1) = -90^\circ \\
 \text{Phases for } h_2(nT_S) &\rightarrow \text{Integer multiples of } \frac{2\pi}{4}(-2) = -180^\circ \\
 \text{Phases for } h_3(nT_S) &\rightarrow \text{Integer multiples of } \frac{2\pi}{4}(-3) = -270^\circ = +90^\circ
 \end{aligned}$$

In this figure, I have marked the angles with dotted lines and deliberately left extra gap between the spectral replicas. Notice that the spectra for $h_0(nT_S)$ and $h_1(nT_S)$ follow from what was explained before in Figure 10.32f and Figure 10.34, respectively.

Something interesting happens when the spectra are overlaid on top of one another. All these spectra at integer multiples of output sample rate overlap in opposite directions except the one at baseband. This is drawn in Figure 10.36a and is a direct result of their placement at P roots of unity in Figure 10.34. The final summation at

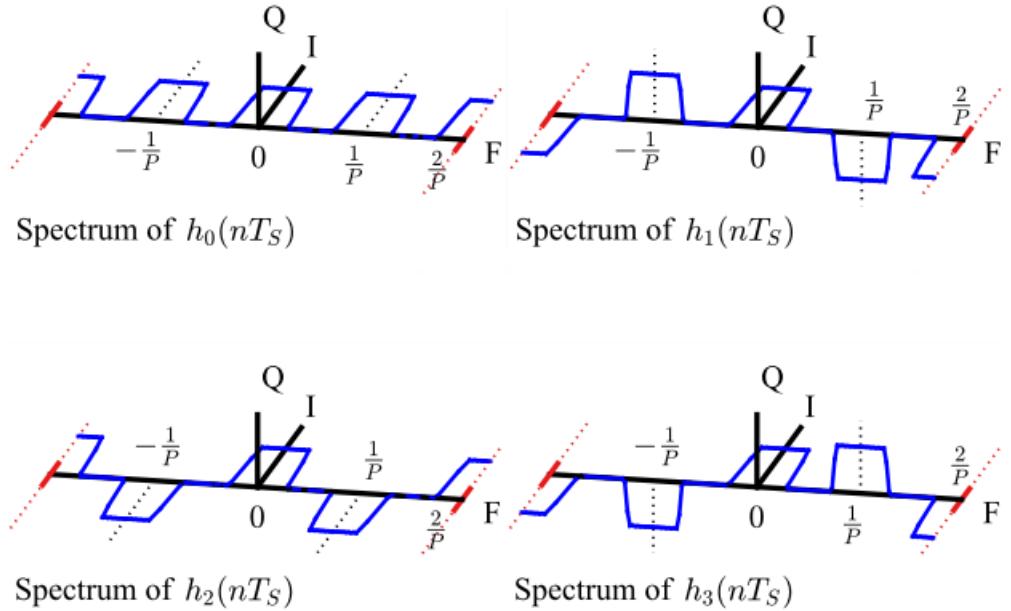


Figure 10.35: Spectrum for all values of δ and $P = 4$ before discarding the zeros – i.e., the spectrum of $h_\delta(nT_S)$ (and not of $h_\delta(nPT_S)$)! Extra gap between spectral replicas is deliberately left for clear demonstration

the end cancels all of them out and only the middle spectrum survives, as drawn in Figure 10.36b.

Let us turn our attention to the input sequence now. Referring to the efficient polyphase architecture of Figure 10.31, there are P time series at the ports of the input commutator which are $P : 1$ downsampled versions of the input sequence. Neglect the downsampling for a moment and concentrate on the effect of a delay of 0 to $P - 1$ from the top to bottom on the input sequence. Then, observe in the input spectrum shown in Figure 10.37 that there is a matching phase profile for the middle channel spectrum in all those downsampled time series. However, *for every other center frequency* (e.g.,

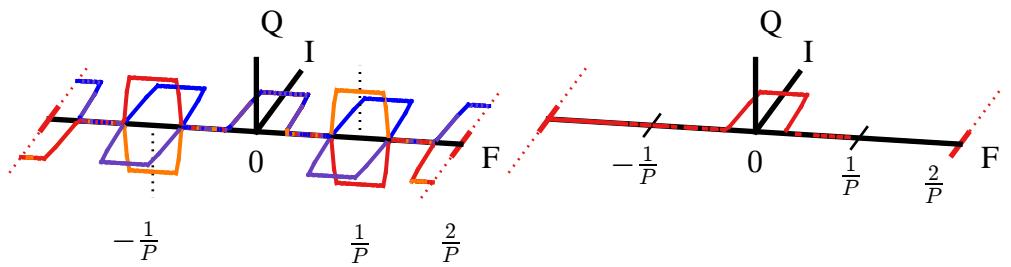


Figure 10.36: Spectral cancellation due to P roots of unity

$\pm 1/P, \pm 2/P, \dots$), a unique phase profile exists that consists of opposite phases from the top to the bottom. This is shown in Figure 10.37 where $x = -2\pi \cdot 1/P = -90^\circ$, see Eq (10.21).

$$\begin{aligned} -2x, -1x, 0x, 1x, 2x &= 180^\circ, 90^\circ, 0^\circ, -90^\circ, -180^\circ && \text{at partition 1} \\ -4x, -2x, 0x, 2x, 4x &= 360^\circ, 180^\circ, 0^\circ, -180^\circ, -360^\circ && \text{at partition 2} \end{aligned}$$

\vdots \vdots

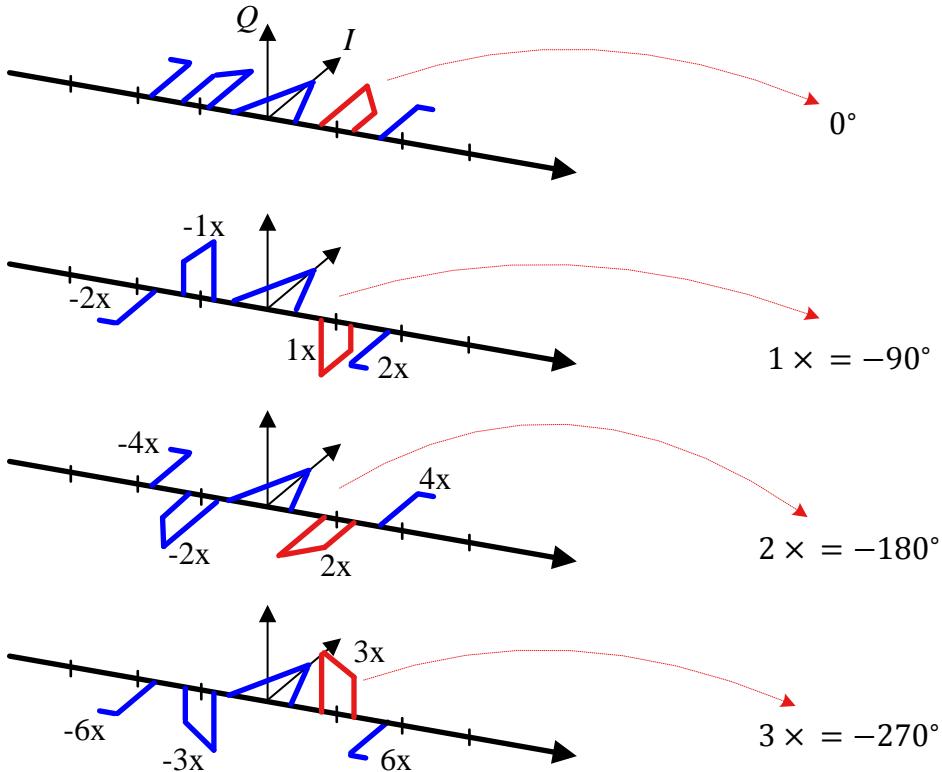


Figure 10.37: Rotations of the input spectra at each successive child filter before downsampling where $x = -90^\circ$

Accounting for $P : 1$ downsampling now, the input spectrum on each port suffers from aliasing coming from the multiples of the output sample rate $1/P$. This is because the whole input spectrum gets replicated at $0, \pm 1/P, \pm 2/P$, and so on. When this happens, the channel spectra overlap in a manner such that the middle spectrum has the same profile from top to bottom while all the other channels lie on P roots of unity. This is drawn in Figure 10.38 for $q = 0/P$ only for clarity of visualization. Here, the arrows indicate the alias that contributes that particular channel spectrum at baseband. This all looks quite disordered but the final summer at the output of the polyphase architecture decimates them all keeping only the channel centered at baseband! This is known as *narrowband beamforming*.

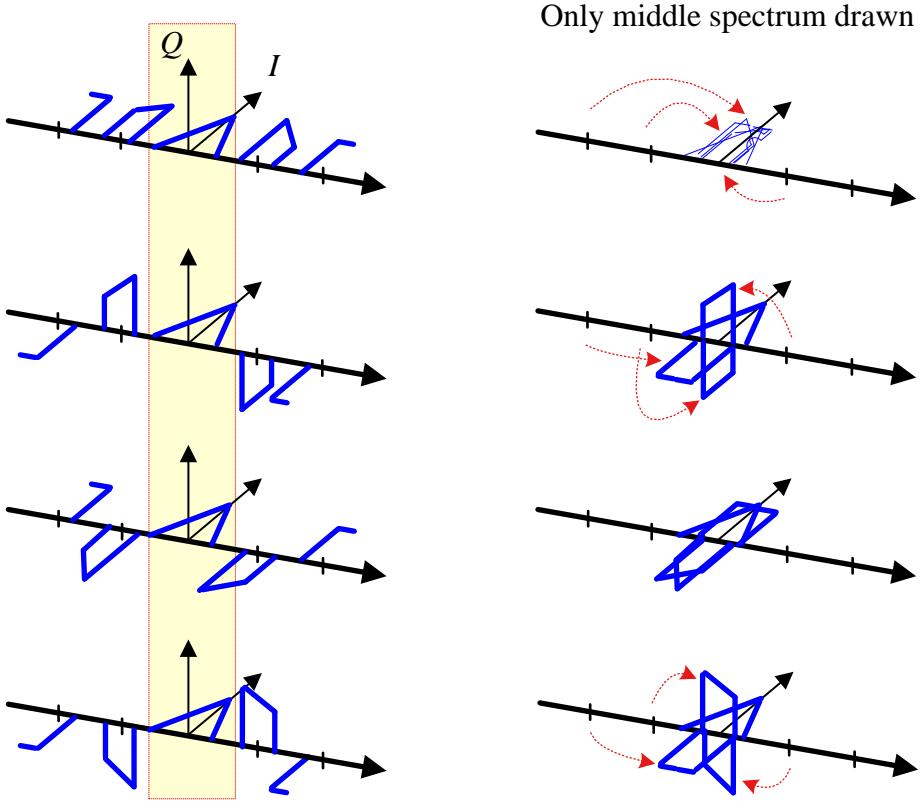


Figure 10.38: Aliasing of all the input channels is shown for baseband only for clarity of visualization. The arrows indicate the alias from which each channel spectrum comes from

Time Delay Beamforming

This kind of spectral cancellation is only true for narrowband signals, i.e., spectral impulses (time domain complex sinusoids) which form opposing phases as the roots of unity. Figure 10.37 is incomplete because it only displays the rotations of band centers which is our narrowband beamforming perspective described earlier. We must take into account the whole spectrum, the delays appearing into the input and the left shifts by δ in the polyphase partitions. These delays have a profound impact on the role played by the polyphase filterbank in filtering the input signal. This is what we see next.

Remember the lesson learned in Section 1.9 that a delay in time domain induces a multiplication with a complex sinusoid in frequency domain. So first, let us remove the leading δ zeros from $h_\delta(nT_S)$ for each δ in a polyphase partition. This is the same as shifting the zero-packed partition impulse response to the left by δ .

Next, we discard $P - 1$ zeros between every two samples to generate P polyphase

child filters given by

$$\begin{aligned} h_\delta(nPT_S) &= h(nPT_S + \delta \cdot T_S) \quad \delta = 0, 1, \dots, P-1 \\ &= h\left(\left(n + \frac{\delta}{P}\right) PT_S\right) \end{aligned} \tag{10.22}$$

There is a slight abuse of notation here since a different index should be used for the child filter. Finally, denote the Fourier Transforms as

$$\begin{aligned} h_0(nPT_S) &\xrightarrow{\mathcal{F}} H_0(F) \\ h_\delta(nPT_S) &\xrightarrow{\mathcal{F}} H_\delta(F) \end{aligned}$$

Just a difference of a time shift implies that magnitudes of all child filters are the same but their phases are a function of this time shift.

$$\begin{aligned} |H_\delta(F)| &= |H_0(F)| \\ \angle H_\delta(F) &= 2\pi \frac{F}{1/(PT_S)} \cdot \frac{\delta}{P} = 2\pi \frac{F}{F_S} \delta \end{aligned}$$

where $\angle H_0(F)$ is zero due to the impulse response being real and even while $1/(PT_S)$ is the sample rate of the child filters. We draw the magnitude and phase spectra of the polyphase partitions in Figure 10.39 for a lowpass filter impulse response of length 40 with $P = 4$ partitions. Notice that their magnitude spectra are the same unity gain approximations of an all-pass filter because the mother filter is a lowpass filter which exhibits an all-pass behaviour within the reduced bandwidth. On the other hand, their phase spectra demonstrate a linear phase profile with different slopes arising due to their respective time origins. These slopes differ by a scaling factor only, see the equation above. It is due to these distinct phase profiles that these filters combined are known as a *polyphase filterbank*!

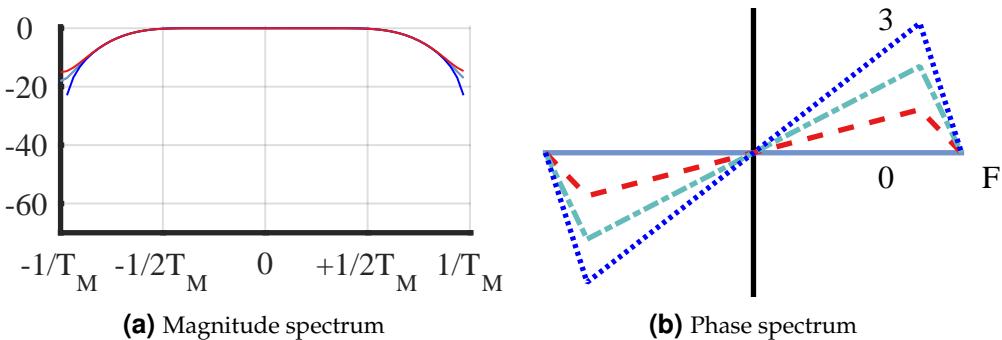


Figure 10.39: While the magnitude spectra of the polyphase partitions are very similar, their phase spectra demonstrate a linear phase profile with different slopes due to their respective time origins

Recall from the polyphase architecture in Figure 10.31 that successive loading is said to be performed through a commutator shown as an *upwards* moving pointer.

With the help of this commutator, the following inputs meet their respective child filters in preparation for convolution.

$$\begin{aligned} r(nPT_S - 0T_S) &\rightarrow h_0(nPT_S) = h(nPT_S + 0T_S) \\ r(nPT_S - 1T_S) &\rightarrow h_1(nPT_S) = h(nPT_S + 1T_S) \\ r(nPT_S - 2T_S) &\rightarrow h_2(nPT_S) = h(nPT_S + 2T_S) \\ r(nPT_S - 3T_S) &\rightarrow h_3(nPT_S) = h(nPT_S + 3T_S) \end{aligned}$$

This is illustrated in Figure 10.40 where I have removed the commutator for clarity. Now recall Eq (1.61) where we said that

Shift in one domain \longrightarrow Multiplication by a complex sinusoid
in the other domain

The spectrum of the input signal $r(nT_S)$ is rotated by a frequency domain complex sinusoid with different inverse period $-\delta$ at each polyphase arm and is given by

$$\begin{aligned} I &\rightarrow R_{\delta,I}(F) = R(F) \cos 2\pi \frac{F}{F_S} \cdot \delta \\ Q &\uparrow R_{\delta,Q}(F) = -R(F) \sin 2\pi \frac{F}{F_S} \cdot \delta \end{aligned}$$

On the other hand, the corresponding rotation for the child filter in each arm is positive.

$$\begin{aligned} I &\rightarrow H_{\delta,I}(F) = H_0(F) \cos 2\pi \frac{F}{F_S} \cdot \delta \\ Q &\uparrow H_{\delta,Q}(F) = H_0(F) \sin 2\pi \frac{F}{F_S} \cdot \delta \end{aligned}$$

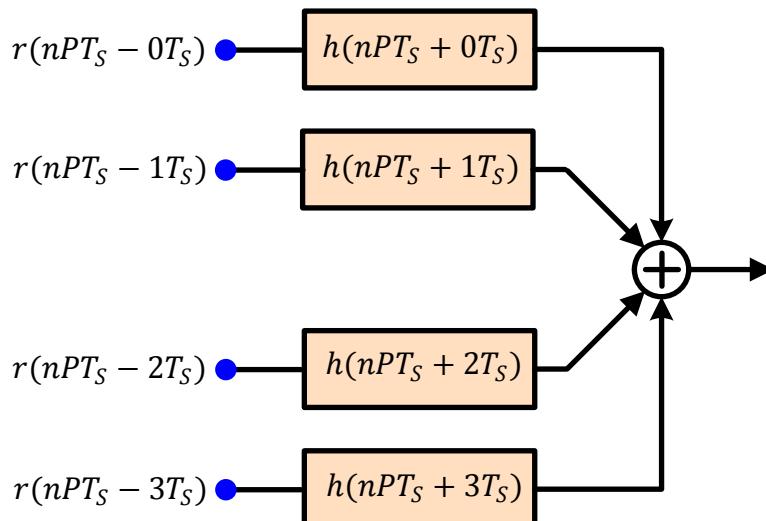


Figure 10.40: Commutator delivers the delayed inputs to their respective polyphase filters

These are the complex sinusoids that rotate the original spectra of Figure 10.35. For this reason, while each child filter $h_\delta(nPT_S) = h(nPT_S + \delta T_S)$ undergoes a convolution operation with its respective delayed input $r(nPT_S - \delta T_S)$, it also provides the correct rotation for the spectral aliases *over the signal bandwidth* so that they can cancel each other at the output of the final summation.

A visual summary of this operation is drawn in Figure 10.41 which plots the spectrum of the delayed input signal $r(nPT_S - 3T_S)$ as well as that of the child filter $h_3(nPT_S)$. A similar response with different delays δ is manifested in other arms. This kind of spectral modification should have been incorporated in Figure 10.37 as well but it would have made the figure too confusing at that instant.

Spectral multiplication at polyphase arm 3

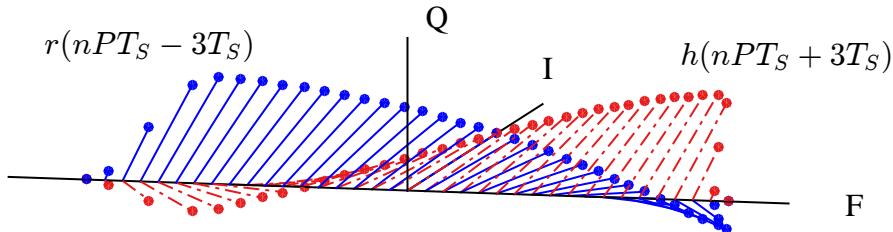


Figure 10.41: Delayed input signal $r(nPT_S - 3T_S)$ shown with child filter 3 frequency response

Due to the equal delays in the opposite directions, the filter treats the signal as if it was not delayed *in any polyphase arm*. A final summation at the end produces the cumulative convolution output as if the input was filtered by the mother filter $h(nT_S)$ itself – right out of the definition of the convolution. The spectral aliases from right and left are destroyed after this coherent summation, similar to what happened in Figure 10.37 but each full bandwidth spectrum rotated by its corresponding δ . If I had plotted the input signal phases (arising from the delays) along with the polyphase filters in Figure 10.39b, they would have been exactly the same but with corresponding *downward* slopes. This is why these polyphase filters are said to provide linear phase shifts producing the time delays that perform a *time-delay beamforming* task: at their outputs, they align the separate time series delivered to their inputs sequentially with successive time offsets[†].

Modifying the Bandpass Filter

Just like we did in the case of a QAM modulator in Section 10.1.1, we can convert the above lowpass filter into a bandpass filter to select any frequency located at q/P where q is an integer. We have already seen that the spectral copy at baseband had many replicas at higher frequencies and we can simply upconvert the lowpass filter $h(nT_S)$ to the desired upper band beforehand to make it a Band-Pass Filter (BPF)

[†]Refer back to Figure 7.77 for a visual description of this process in time domain.

$h_{BP}(nT_S)$ and process the input signal with this filter instead.

$I \rightarrow$

$$h_{BP,I}(nT_S) = h(nT_S) \cos 2\pi \frac{q}{P} n$$

$Q \uparrow$

$$h_{BP,Q}(nT_S) = h(nT_S) \sin 2\pi \frac{q}{P} n$$

Before we investigate how this process works, remember from the downsampling lowpass filter that a z-Transform approach is usually employed to derive the final architecture embedding the spectral translation through the bandpass filter, see Eq (10.9). There, we found that for forming a bandpass filter, the polyphase partitions of the low-pass filter $h(nT_S)$ stay intact while each delay is replaced by a delay and a complex heterodyne in each polyphase arm. A block diagram for this approach is drawn in Figure 10.42.

What the extra complex heterodyne in the end accomplishes is the following.

- There are P spectral aliases appearing at baseband due to $P : 1$ downsampling and each of them possesses a unique phase profile according to Figure 10.37.
- As we progress up the successive ports of the commutator at the input, each alias at frequency q/P undergoes successive phase increments of $2\pi q/P$, i.e., starting from the top of the commutator and coming down to port δ , the phase angles of the alias at center frequency q/P are $-2\pi(q/P)\delta$.
- On the other hand, the phase rotators shown in Figure 10.42 exhibit angles of $2\pi(q/P)\delta$ thus de-rotating and aligning the phase shifts from the center frequency q/P (instead of $q = 0$).
- These spectral replicas at q/P for each polyphase arm add up in phase while the spectral replicas from all other center frequencies are destructively cancelled out. This is drawn in Figure 10.43 for $q/P = 1/P$. Observe the alignment of the replicas at $1/P$ and the phase angles for all center frequencies are shown on the right side of the figure as follows.

$0/P$	\rightarrow	$0^\circ, 90^\circ, 180^\circ, 270^\circ$
$1/P$	\rightarrow	$0^\circ, 0^\circ, 0^\circ, 0^\circ$
$-1/P$	\rightarrow	$0^\circ, 180^\circ, 360^\circ, 540^\circ$
$2/P$	\rightarrow	$0^\circ, 270^\circ, 540^\circ, 810^\circ$

Clearly, the sum of all the above phases for those remaining center frequencies is zero. This is why these phase shifters are said to provide the phase rotations that perform a *narrowband beamforming* task.

In summary, there is a negative phase rotation for each center frequency induced by the time delays at the polyphase filterbank input which is compensated by a positive phase rotation *only for the q^{th} band* from the phase rotators at the polyphase filterbank output.

Also notice in Figure 10.42 that the complex scalars are fitted at the right of the polyphase partitions, although they could have been attached to the left as well. In the

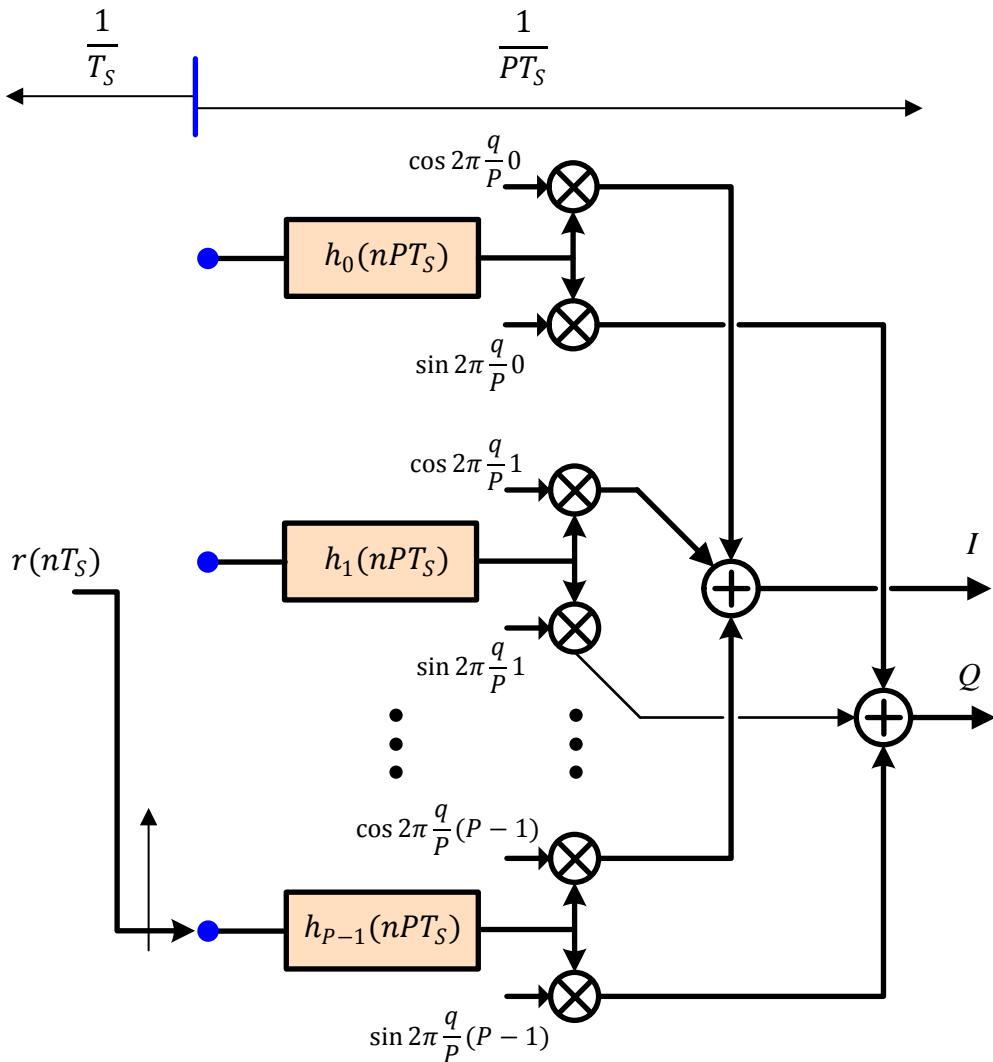


Figure 10.42: A block diagram for a $P : 1$ downampler and downconverter using multirate signal processing

latter approach, the signal becomes complex after leaving the multipliers and hence two lowpass filters are needed, one for the I arm and the other for the Q arm. On the other hand, the advantage of the particular arrangement in Figure 10.42 is that the lowpass filter is real and hence one filter provides the convolution output to I and Q multipliers to generate the complex output only at the end of the chain. A word of caution: in this respect, Figure 10.43 is not actually correct since it displays the rotations before $P : 1$ downsampling operation. In reality, however, these phase adjustments act on the aliased spectra quite similar to what we saw in Figure 10.38.

The real attraction of this architecture comes from the need to downconvert multiple channels at once. For a small number of channels, a separate set of phase rotators can be attached to the filterbank output, one for each q . However, when the number

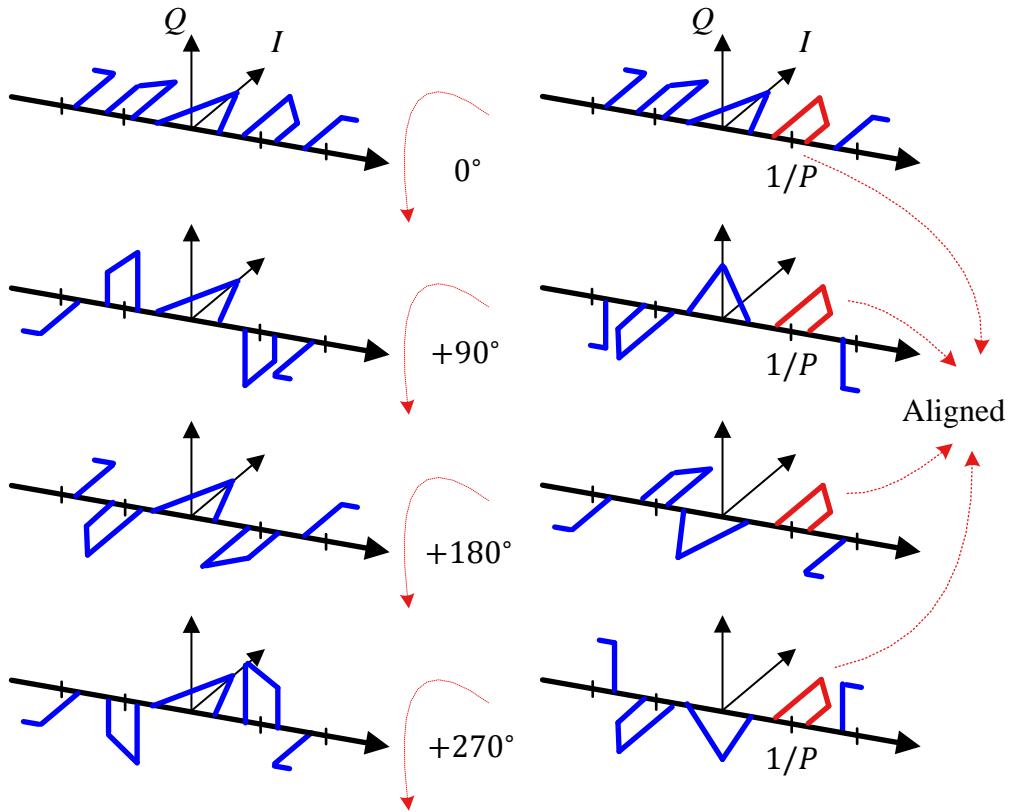


Figure 10.43: Spectral replicas for $q/P = 1/P$ add up in phase while those at rest of the center frequencies cancel out

of channels increases, we invoke the similarity between these phase rotators and the iDFT definition and realize that a P -point iDFT (actually, an iFFT) at the filterbank output simultaneously downconverts all the channels located on integer multiples of $1/P$. This is because a P -point iFFT correlates the input signal with all such P complex sinusoids.

As a final remark, one might think that a time domain view of the downsampling operation can be enough for our understanding and frequency domain view is not really needed. Nevertheless, we learned above that a P -channel downconverter can be constructed simply by using a polyphase partition of a lowpass filter followed by a P -point iFFT. Think about the computational savings when P is large. It would have been quite difficult to come up with such an architecture thinking solely in time domain.

10.3 The Origins of Magic

From the theory of digital transmission and practice of system implementation we have described so far, one wonders why the field of DSP is so versatile as compared to analog signal processing. Where does this magic come from?

The answer is in the storage of signals. Since most processes executed by nature

are real-time, storage enables us to break through the limitations of these natural systems and enter into completely new realms. This is supported by plenty of examples in the history of humankind.

- The advent of agriculture around 10000 BC actually enabled us to store food and grain which opened the opportunity (for some) to pursue interests unrelated to gathering or hunting for food leading to rapid advancement of civilization.
- The first system of writing was invented around 3000 BC by the ancient Sumerians of Mesopotamia. This storage broke the data processing limits of the human brain and also made it possible for us to convey our thoughts in both space (with far-flung areas) and time (with future generations). Probably this is why Newton said: “If I have seen further, it is by standing on the shoulders of giants.”

In general, a paradigm shift occurs whenever a new kind of storage arises. Thus, unlike the analog domain, unique techniques can be exploited in DSP due to the storage of samples in the processor memory, as we saw in this text. Further examples are also available in communication theory. Viterbi for instance, in his famous 1991 paper [52], summarized the essence of Shannon theory for digital communications in the form of three lessons. The first lesson was the following.

“Never discard information prematurely that may be useful in making a decision until after all decisions related to that information have been completed.”

Andrew Viterbi

The applications of this lesson lead to soft decision decoding in convolutional codes and maximum-likelihood sequence estimation in high-quality wireline modems. The inventors of Turbo Codes also exploited the storage of log-likelihood ratios for just a little extra time (according to the number of iterations) that revolutionized the decoding methodology of error control codes forever.

Expanding on this theme, storage alone is not enough for the magic to succeed. We must accumulate the knowledge of how to exploit the stored signals in an efficient manner. Since the early days of progress, humans always had three resources available: raw materials, energy and knowledge. The traditional focus was on increasing the raw materials and energy resources while gaining more knowledge was the lowest priority. For example, the world's coal is estimated to run out before the middle of next century. By burning billions of tonnes of coal every year, we are emptying the reserves all over the world. On the other hand, we can discover more efficient methods of producing energy from a renewable resource like sunlight or wind and this knowledge will fulfill our needs for generations to come.

In a similar manner, the tradeoffs traditionally available to a communication systems designer were bandwidth and power. With the rise of DSP, cost-effective implementation of algorithms has become possible adding a third dimension to our arsenal with the following equivalents.

- Bandwidth ↔ raw material
- Power ↔ energy
- DSP ↔ knowledge

This is shown in Figure 10.44 where DSP refers to the computational complexity of the algorithms. Here, more DSP knowledge can be traded-off with less bandwidth and/or power [21].

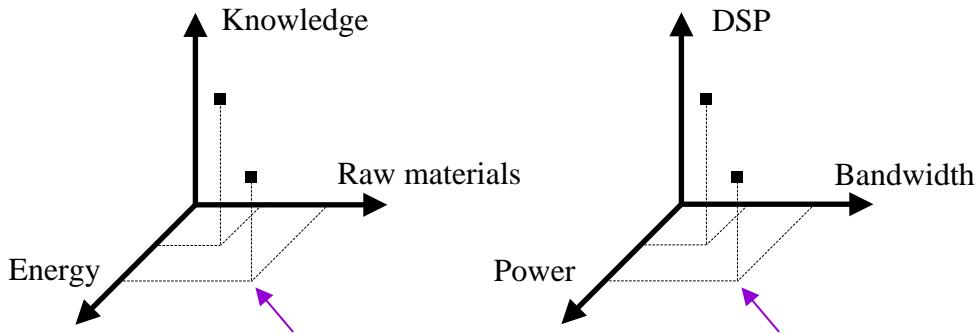


Figure 10.44: Bandwidth, power and DSP correspond to the traditional trio of raw materials, energy and knowledge

The advent of DSP has revolutionized wireless communications and enabled us to pack and transmit more information within the same bandwidth than the early wireless pioneers could ever hope for. Some of the most straightforward examples are as below.

- Audio and video compression eliminate statistical redundancy in the stored bit-stream which results in less bandwidth required for the same information.
- On the other end, redundancy can be smartly introduced in the bitstream that allows the decoding algorithms at the Rx to pull the intended message out of noise which results in less energy (SNR) required for data transmission at the expense of bandwidth. However, Gottfried Ungerboeck invented Trellis Coded Modulation (TCM) in 1970s that combines modulation and coding to improve the error performance within the same bandwidth.
- We have seen many examples of DSP applications in this text, none more important than simplifying the signal equalization at high data rates through transforming the problem via OFDM.
- Thanks to DSP algorithms, Multiple Input Multiple Output (MIMO) systems can employ several independent spatial streams (according to the number of Tx and Rx antennas) over the same channel at the same time within the same bandwidth.
- Full-duplex wireless communication builds on the knowledge from equalization concepts in which radios can now transmit and receive at the same time through applying smart self-interference cancellation techniques.

The current research is focused on applying machine learning techniques to wireless communications. The radio waves are not much different than the sound waves. As predicted by some, if the devices do become intelligent at some point in the future, transmitters and receivers along with the antennas give them the opportunity to talk among themselves, just like we use a vocal box along with vocal cords to transmit our

voice and ears to receive the generated vibrations. Since digital communication is all mathematics at its heart and no one is better at computations, there is a possibility that we will have entirely new ways of communicating through a wireless channel devised by the machines themselves and unbeknownst to us.

10.4 The Small Picture

A polyphase filterbank accomplishes the following tasks for downconverting a signal.

Filter passband: The mother filter is a lowpass filter exhibiting a passband width of $1/P$ while its $P : 1$ downsampled children are approximations to all-pass filters exhibiting a unity gain over the new low bandwidth, see Figure 10.39.

Time delay beamforming: The phase shifts of these filters provide the time delays required for de-rotating (over the whole new bandwidth) the input signal spectrum that was originally rotated by the complex sinusoid arising from the delays at the input commutator. They are said to perform a time-delay beamforming task.

Narrowband beamforming: The phase rotators following the filters in Figure 10.42 perform alignment of the band center for each spectral alias to prepare all but one of them for destructive cancellation as a result of summation. They are said to perform a frequency domain beamforming task.

THE END

One Page Summary for Rx Algorithms

Benefitting mostly from the small picture sections described at the end of some chapters, a one page summary about the crux of Rx algorithms is presented at the next page. These are the signals that are hidden in the Rx waveform and need to be extracted through particular signal processing operations. Reader is advised to go through the respective chapters to fully appreciate how this page summarizes the understanding and design of wireless communication systems.

One Page Summary for Rx Algorithms

Parameter	Time Domain	Frequency Domain
Carrier phase sync: a complex constant		
Carrier frequency sync: a complex sinusoid		
Timing sync: Real sinusoid at symbol rate		
Wireless channel		
OFDM		

Bibliography

- [1] fred harris. *Multirate Signal Processing for Communication Systems*. Prentice Hall, 2004.
- [2] Michael Rice. *Digital Communications - A Discrete-Time Approach*. Prentice Hall, 2009.
- [3] J. Proakis. *Digital Communications (4th Edition)*. McGraw-Hill, 2001.
- [4] R. Lyons. *Understanding Digital Signal Processing (3rd Edition)*. Prentice Hall, 2010.
- [5] B.P. Lathi and Zhi Ding. *Modern Digital and Analog Communication Systems (5th Edition)*. Oxford University Press, 2018.
- [6] S. Smith. *The Scientist & Engineer's Guide to Digital Signal Processing (1st Edition)*. California Technical Pub., 1997.
- [7] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27, 1948.
- [8] E. Jacobsen. Pulse shaping in single-carrier communication systems. <https://www.dsprelated.com/showarticle/60.php>, 2008.
- [9] H. Nyquist. Certain topics in telegraph transmission theory. *Trans. AIEE*, 47, 1928.
- [10] fred harris. An alternate design technique for square-root nyquist shaping filters. In *WInnComm*, 2015.
- [11] C. Rapp. Coded and uncoded m-psk and cpm signals on nonlinear bandpass channels: An approach to a fair comparison. In *IEEE Proc. of the Global Telecommunications Conference*, 1990.
- [12] H. Bellescize. La réception synchrone. *L'Onde Electrique*, 11, 1922.
- [13] S. Strogatz. *Sync: How Order Emerges from Chaos in the Universe, Nature, and Daily Life*. Hachette Books, 2004.
- [14] R. Best. *Phase Locked Loops - Design, Simulation and Applications (6th Edition)*. McGraw Hill, 2007.
- [15] T. Rondeau. Control loop gain values. <http://www.trondeau.com/blog/2011/8/13/control-loop-gain-values.html>, 2011.

- [16] J. Costas. Synchronous communications. *Proceedings of the IRE*, 44, 1956.
- [17] J. Cham and D. Whiteson. *We Have No Idea: A Guide to the Unknown Universe*. Riverhead Books, 2017.
- [18] U. Mengali and A. D'Andrea. *Synchronization Techniques for Digital Receivers*. Wiley, 1997.
- [19] M. Fitz. Planar filtered techniques for burst mode carrier synchronization. In *IEEE Proc. of the Global Telecommunications Conference*, 1991.
- [20] M. Luise and R. Reggiannini. Carrier frequency recovery in all-digital modems for burst-mode transmissions. *IEEE Transactions on Communications*, 43, 1995.
- [21] H. Meyr, M. Moeneclaey, and S. Fechtel. *Digital Communication Receivers, Volume 2: Synchronization, Channel Estimation, and Signal Processing*. Wiley, 1998.
- [22] M. Oerder and H. Meyr. Digital filter and square timing recovery. *IEEE Transactions on Communications*, 36(5), 1988.
- [23] F. Gardner. A bpsk/qpsk timing-error detector for sampled receivers. *IEEE Transactions on Communications*, 34(5), 1986.
- [24] W. Lindsey and M. Simon. *Telecommunication Systems Engineering*. Prentice Hall, 1973.
- [25] A. D'Andrea and M. Luise. Optimization of symbol timing recovery for qam data demodulators. *IEEE Transactions on Communications*, 44(3), 1996.
- [26] F. Gardner. Interpolation in digital modems – part i: Fundamentals. *IEEE Transactions on Communications*, 41(3), 1993.
- [27] K. Mueller and M. Muller. Timing recovery in digital synchronous data receivers. *IEEE Transactions on Communications*, 24(5), 1976.
- [28] J. Treichler, M. Larimore, and J. Harp. Practical blind demodulators for high-order qam signals. *Proceedings of the IEEE*, 86(10), 1998.
- [29] fred harris and M. Rice. Multirate digital filters for symbol timing synchronization in software defined radios. *IEEE Journal on Selected Areas in Communications*, 19(12), 2001.
- [30] C. Dick, B. Egg, and fred harris. Architecture and simulation of timing synchronization circuits for the fpga implementation of narrowband waveforms. In *SDR Technical Conference and Product Exposition*, 2006.
- [31] D. Falconer. History of equalization 1860-1980. *IEEE Communications Magazine*, 49(10), 2011.
- [32] Wikipedia. The cold equations. https://en.wikipedia.org/wiki/The_Cold_Equations.
- [33] A. Molisch. *Wireless Communications (2nd Edition)*. Wiley-IEEE Press, 2010.

- [34] H. Arslan and G. Bottomley. Channel estimation in narrowband wireless communication systems. *Wireless Communications and Mobile Computing*, 2001.
- [35] R. Brown and P. Hwang. *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises (4th Edition)*. Wiley, 2012.
- [36] S. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, 73(9), 1985.
- [37] S. Weinstein and P. Ebert. Data transmission by frequency-division multiplexing using the discrete fourier transform. *IEEE Transactions on Commun. Technol.*, COM-19(5), 1971.
- [38] J. Bingham. Multicarrier modulation for data transmisson: An idea whose time has come. *IEEE Communications Magazine*, 28(5), 1990.
- [39] J. Mitola. The software radio. In *Proceedings of IEEE National Telesystems Conference*, 1992.
- [40] H. Sari, G. Karam, and I. Jeanclaude. Transmission techniques for digital terres-trial tv broadcasting. *IEEE Communications Magazine*, 33(2), 1995.
- [41] Keysight Technologies. Ofdm raised cosine windowing. http://rfmw.em.keysight.com/rfcomms/n4010a/n4010aWLAN/onlineguide/default.htm#ofdm_raised_cosine_windowing.htm.
- [42] M. Speth et al. Optimum receiver design for wireless broad-band systems using ofdm-part i. *IEEE Transactions on Communications*, 47(11), 1999.
- [43] M. Speth et al. Optimum receiver design for wireless broad-band systems using ofdm-part ii. *IEEE Transactions on Communications*, 49(4), 2001.
- [44] T. Schmidl and D. Cox. Robust frequency and timing synchronization for ofdm. *IEEE Transactions on Communications*, 45(12), 1997.
- [45] H. Minn, V. Bhargava, and K. Letaief. A robust timing and frequency synchroniza-tion for ofdm systems. *IEEE Transactions on Wireless Communications*, 2(4), 2003.
- [46] Clem Schmidt. Qirx software defined radio. <http://www.softsyst.com/qirx>, 2018.
- [47] M. Ozdemir and H. Arslan. Channel estimation for wireless ofdm systems. *IEEE Communications Surveys & Tutorials*, 9(2), 2007.
- [48] Microchip Technology Incorporated. *AT86RF215 / AT86RF215IQ / AT86RF215M DATASHEET*, 10 2008. Rev. 1.2.
- [49] B. Razavi. *RF Microelectronics*. Prentice Hall, 1998.
- [50] S. Mirabbasi and K. Martin. Classical and modern receiver architectures. *IEEE Communications Magazine*, 38(11), 2000.
- [51] B. Brannon. Where zero-if wins: 50% smaller pcb footprint at 1/3 the cost. *Analog Dialogue*, 50(09), 2016.
- [52] A. Viterbi. Wireless digital communication: A view based on three lessons learned. *IEEE Communications Magazine*, 29(9), 1991.

About the Author



Qasim Chaudhari is an independent consultant and trainer with expertise in wireless communications, Software Defined Radio (SDR) and Digital Signal Processing (DSP). As the founder of <https://wirelesspi.com>, he focuses on explaining concepts through beautiful figures, simple mathematics and intuitive reasoning. After obtaining a PhD in Electrical Engineering from Texas A&M University, he worked in different research labs on DSP algorithms development for real world demonstrations of wireless systems such as a MIMO-OFDM testbed, low-SNR receivers and phase of arrival based localization. You can reach him at info@wirelesspi.com.

