Fundamentals of Signal Processing

Collection Editor:
Minh N. Do
# Table of Contents

**Introduction to Fundamentals of Signal Processing** .................................................. 1

## 1 Foundations

1.1 Signals Represent Information ............................................................................. 3
1.2 Introduction to Systems ....................................................................................... 6
1.3 Discrete-Time Signals and Systems ..................................................................... 8
1.4 Linear Time-Invariant Systems ........................................................................... 11
1.5 Discrete Time Convolution ................................................................................ 11
1.6 Review of Linear Algebra .................................................................................... 18
1.7 Hilbert Spaces ..................................................................................................... 28
1.8 Signal Expansions ............................................................................................... 29
1.9 Introduction to Fourier Analysis ......................................................................... 33
1.10 Continuous Time Fourier Transform (CTFT) ..................................................... 34
1.11 Discrete Time Fourier Transform (DTFT) ......................................................... 38
1.12 DFT as a Matrix Operation ................................................................................ 41
1.13 The FFT Algorithm .......................................................................................... 44
Solutions .................................................................................................................. 48

## 2 Sampling and Frequency Analysis

2.1 Introduction ......................................................................................................... 49
2.2 Proof .................................................................................................................... 51
2.3 Illustrations .......................................................................................................... 54
2.4 Sampling and Reconstruction with Matlab ....................................................... 58
2.5 Systems View of Sampling and Reconstruction ................................................ 59
2.6 Sampling CT Signals: A Frequency Domain Perspective ................................... 60
2.7 The DFT: Frequency Domain with a Computer Analysis .................................. 63
2.8 Discrete-Time Processing of CT Signals ............................................................. 73
2.9 Short Time Fourier Transform .......................................................................... 78
2.10 Spectrograms .................................................................................................... 91
2.11 Filtering with the DFT ....................................................................................... 96
2.12 Image Restoration Basics ................................................................................ 104
Solutions .................................................................................................................. 107

## 3 Digital Filtering

3.1 Difference Equation ............................................................................................ 109
3.2 The Z Transform: Definition ............................................................................ 114
3.3 Table of Common z-Transforms ........................................................................ 119
3.4 Understanding Pole/Zero Plots on the Z-Plane .................................................. 120
3.5 Filtering in the Frequency Domain .................................................................... 126
3.6 Linear-Phase FIR Filters ................................................................................... 130
3.7 Filter Structures ................................................................................................. 134
3.8 Overview of Digital Filter Design ...................................................................... 134
3.9 Window Design Method ..................................................................................... 135
3.10 Frequency Sampling Design Method for FIR Filters ....................................... 136
3.11 Parks-McClellan FIR Filter Design ................................................................... 138
3.12 FIR Filter Design using MATLAB .................................................................... 145
3.13 MATLAB FIR Filter Design Exercise ............................................................... 146
Solutions .................................................................................................................. 147

## 4 Multirate Signal Processing

4.1 Upsampling ........................................................................................................ 149
Introduction to Fundamentals of Signal Processing

What is Digital Signal Processing?

To understand what is Digital Signal Processing (DSP) let’s examine what does each of its words mean. “Signal” is any physical quantity that carries information. “Processing” is a series of steps or operations to achieve a particular end. It is easy to see that Signal Processing is used everywhere to extract information from signals or to convert information-carrying signals from one form to another. For example, our brain and ears take input speech signals, and then process and convert them into meaningful words. Finally, the word “Digital” in Digital Signal Processing means that the process is done by computers, microprocessors, or logic circuits.

The field DSP has expanded significantly over that last few decades as a result of rapid developments in computer technology and integrated-circuit fabrication. Consequently, DSP has played an increasingly important role in a wide range of disciplines in science and technology. Research and development in DSP are driving advancements in many high-tech areas including telecommunications, multimedia, medical and scientific imaging, and human-computer interaction.

To illustrate the digital revolution and the impact of DSP, consider the development of digital cameras. Traditional film cameras mainly rely on physical properties of the optical lens, where higher quality requires bigger and larger system, to obtain good images. When digital cameras were first introduced, their quality were inferior compared to film cameras. But as microprocessors become more powerful, more sophisticated DSP algorithms have been developed for digital cameras to correct optical defects and improve the final image quality. Thanks to these developments, the quality of consumer-grade digital cameras has now surpassed the equivalence in film cameras. As further developments for digital cameras attached to cell phones (cameraphones), where due to small size requirements of the lenses, these cameras rely on DSP power to provide good images. Essentially, digital camera technology uses computational power to overcome physical limitations. We can find the similar trend happens in many other applications of DSP such as digital communications, digital imaging, digital television, and so on.

In summary, DSP has foundations on Mathematics, Physics, and Computer Science, and can provide the key enabling technology in numerous applications.

Overview of Key Concepts in Digital Signal Processing

The two main characters in DSP are signals and systems. A signal is defined as any physical quantity that varies with one or more independent variables such as time (one-dimensional signal), or space (2-D or 3-D signal). Signals exist in several types. In the real-world, most of signals are continuous-time or analog signals that have values continuously at every value of time. To be processed by a computer, a continuous-time signal has to be first sampled in time into a discrete-time signal so that its values at a discrete set of time instants can be stored in computer memory locations. Furthermore, in order to be

\[ \text{This content is available online at } \text{http://cnx.org/content/m13673/1.1/}. \]

Available for free at Connexions <http://cnx.org/content/coll0360/1.4>
processed by logic circuits, these signal values have to be quantized into a set of discrete values, and the final result is called a digital signal. When the quantization effect is ignored, the terms discrete-time signal and digital signal can be used interchangeability.

In signal processing, a system is defined as a process whose input and output are signals. An important class of systems is the class of linear time-invariant (or shift-invariant) systems. These systems have a remarkable property is that each of them can be completely characterized by an impulse response function (sometimes is also called as point spread function), and the system is defined by a convolution (also referred to as a filtering) operation. Thus, a linear time-invariant system is equivalent to a (linear) filter. Linear time-invariant systems are classified into two types, those that have finite-duration impulse response (FIR) and those that have an infinite-duration impulse response (IIR).

A signal can be viewed as a vector in a vector space. Thus, linear algebra provides a powerful framework to study signals and linear systems. In particular, given a vector space, each signal can be represented (or expanded) as a linear combination of elementary signals. The most important signal expansions are provided by the Fourier transforms. The Fourier transforms, as with general transforms, are often used effectively to transform a problem from one domain to another domain where it is much easier to solve or analyze. The two domains of a Fourier transform have physical meaning and are called the time domain and the frequency domain.

Sampling, or the conversion of continuous-domain real-life signals to discrete numbers that can be processed by computers, is the essential bridge between the analog and the digital worlds. It is important to understand the connections between signals and systems in the real world and inside a computer. These connections are convenient to analyze in the frequency domain. Moreover, many signals and systems are specified by their frequency characteristics.

Because any linear time-invariant system can be characterized as a filter, the design of such systems boils down to the design the associated filters. Typically, in the filter design process, we determine the coefficients of an FIR or IIR filter that closely approximates the desired frequency response specifications. Together with Fourier transforms, the z-transform provides an effective tool to analyze and design digital filters.

In many applications, signals are conveniently described via statistical models as random signals. It is remarkable that optimum linear filters (in the sense of minimum mean-square error), so called Wiener filters, can be determined using only second-order statistics (autocorrelation and crosscorrelation functions) of a stationary process. When these statistics cannot be specified beforehand or change over time, we can employ adaptive filters, where the filter coefficients are adapted to the signal statistics. The most popular algorithm to adaptively adjust the filter coefficients is the least-mean square (LMS) algorithm.
Chapter 1

Foundations

1.1 Signals Represent Information

Whether analog or digital, information is represented by the fundamental quantity in electrical engineering: the **signal**. Stated in mathematical terms, a **signal is merely a function**. Analog signals are continuous-valued; digital signals are discrete-valued. The independent variable of the signal could be time (speech, for example), space (images), or the integers (denoting the sequencing of letters and numbers in the football score).

1.1.1 Analog Signals

Analog signals are usually signals defined over continuous independent variable(s). Speech\(^2\) is produced by your vocal cords exciting acoustic resonances in your vocal tract. The result is pressure waves propagating in the air, and the speech signal thus corresponds to a function having independent variables of space and time and a value corresponding to air pressure: \(s(x,t)\) (Here we use vector notation \(x\) to denote spatial coordinates). When you record someone talking, you are evaluating the speech signal at a particular spatial location, \(x_0\) say. An example of the resulting waveform \(s(x_0, t)\) is shown in this figure (Figure 1.1: Speech Example).

\(^1\)This content is available online at <http://cnx.org/content/m0001/2.27/>.

\(^2\)“Modeling the Speech Signal” <http://cnx.org/content/m0049/latest/>
Figure 1.1: A speech signal's amplitude relates to tiny air pressure variations. Shown is a recording of the vowel "e" (as in "speech").

Photographs are static, and are continuous-valued signals defined over space. Black-and-white images have only one value at each point in space, which amounts to its optical reflection properties. In Figure 1.2 (Lena), an image is shown, demonstrating that it (and all other images as well) are functions of two independent spatial variables.
Figure 1.2: On the left is the classic Lena image, which is used ubiquitously as a test image. It contains straight and curved lines, complicated texture, and a face. On the right is a perspective display of the Lena image as a signal: a function of two spatial variables. The colors merely help show what signal values are about the same size. In this image, signal values range between 0 and 255; why is that?

Color images have values that express how reflectivity depends on the optical spectrum. Painters long ago found that mixing together combinations of the so-called primary colors—red, yellow and blue—can produce very realistic color images. Thus, images today are usually thought of as having three values at every point in space, but a different set of colors is used: How much of red, green and blue is present. Mathematically, color pictures are multivalued-vector-valued signals: \( s(x) = (r(x), g(x), b(x))^T \).

Interesting cases abound where the analog signal depends not on a continuous variable, such as time, but on a discrete variable. For example, temperature readings taken every hour have continuous-analog-values, but the signal’s independent variable is (essentially) the integers.

### 1.1.2 Digital Signals

The word "digital" means discrete-valued and implies the signal has an integer-valued independent variable. Digital information includes numbers and symbols (characters typed on the keyboard, for example). Computers rely on the digital representation of information to manipulate and transform information. Symbols do not have a numeric value, and each is represented by a unique number. The ASCII character code has the upper- and lowercase characters, the numbers, punctuation marks, and various other symbols represented by a seven-bit integer. For example, the ASCII code represents the letter a as the number 97 and the letter A as 65. Table 1.1: ASCII Table shows the international convention on associating characters with integers.

**ASCII Table**

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Table 1.1: The ASCII translation table shows how standard keyboard characters are represented by integers. In pairs of columns, this table displays first the so-called 7-bit code (how many characters in a seven-bit code?), then the character the number represents. The numeric codes are represented in hexadecimal (base-16) notation. Mnemonic characters correspond to control characters, some of which may be familiar (like \texttt{cr} for carriage return) and some not (\texttt{bel} means a "bell").

1.2 Introduction to Systems$^a$

Signals are manipulated by systems. Mathematically, we represent what a system does by the notation $y(t) = S(x(t))$, with $x$ representing the input signal and $y$ the output signal.

![Definition of a system](image)

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{system_definition.png}
\caption{The system depicted has input $x(t)$ and output $y(t)$. Mathematically, systems operate on function(s) to produce other function(s). In many ways, systems are like functions, rules that yield a value for the dependent variable (our output signal) for each value of its independent variable (its input signal). The notation $y(t) = S(x(t))$ corresponds to this block diagram. We term $S(\cdot)$ the input-output relation for the system.}
\end{figure}

\footnotesize$^a$This content is available online at <http://cnx.org/content/m0005/2.19/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
This notation mimics the mathematical symbology of a function: A system’s input is analogous to an independent variable and its output the dependent variable. For the mathematically inclined, a system is a **functional**: a function of a function (signals are functions).

Simple systems can be connected together—one system’s output becomes another’s input—to accomplish some overall design. Interconnection topologies can be quite complicated, but usually consist of weaves of three basic interconnection forms.

### 1.2.1 Cascade Interconnection

![Cascade Interconnection Diagram](image)

**Figure 1.4:** The most rudimentary ways of interconnecting systems are shown in the figures in this section. This is the cascade configuration.

The simplest form is when one system’s output is connected only to another’s input. Mathematically, 

\[ w(t) = S_1(x(t)) \quad \text{and} \quad y(t) = S_2(w(t)) \]

with the information contained in \( x(t) \) processed by the first, then the second system. In some cases, the ordering of the systems matter, in others it does not. For example, in the fundamental model of communication \(^4\) the ordering most certainly matters.

### 1.2.2 Parallel Interconnection

![Parallel Interconnection Diagram](image)

**Figure 1.5:** The parallel configuration.

A signal \( x(t) \) is routed to two (or more) systems, with this signal appearing as the input to all systems simultaneously and with equal strength. Block diagrams have the convention that signals going to more

\(^4\)“Structure of Communication Systems”, Figure 1: Fundamental model of communication

<http://cnx.org/content/m0002/latest/#commsys>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
than one system are not split into pieces along the way. Two or more systems operate on $x(t)$ and their outputs are added together to create the output $y(t)$. Thus, $y(t) = S_1(x(t)) + S_2(x(t))$, and the information in $x(t)$ is processed separately by both systems.

1.2.3 Feedback Interconnection

![Figure 1.6: The feedback configuration.](image)

The subtlest interconnection configuration has a system's output also contributing to its input. Engineers would say the output is "fed back" to the input through system 2, hence the terminology. The mathematical statement of the feedback interconnection (Figure 1.6: feedback) is that the feed-forward system produces the output: $y(t) = S_1(e(t))$. The input $e(t)$ equals the input signal minus the output of some other system's output to $y(t)$: $e(t) = x(t) - S_2(y(t))$. Feedback systems are omnipresent in control problems, with the error signal used to adjust the output to achieve some condition defined by the input (controlling) signal. For example, in a car's cruise control system, $x(t)$ is a constant representing what speed you want, and $y(t)$ is the car's speed as measured by a speedometer. In this application, system 2 is the identity system (output equals input).

1.3 Discrete-Time Signals and Systems

Mathematically, analog signals are functions having as their independent variables continuous quantities, such as space and time. Discrete-time signals are functions defined on the integers; they are sequences. As with analog signals, we seek ways of decomposing discrete-time signals into simpler components. Because this approach leads to a better understanding of signal structure, we can exploit that structure to represent information (create ways of representing information with signals) and to extract information (retrieve the information thus represented). For symbolic-valued signals, the approach is different: We develop a common representation of all symbolic-valued signals so that we can embody the information they contain in a unified way. From an information representation perspective, the most important issue becomes, for both real-valued and symbolic-valued signals, efficiency: what is the most parsimonious and compact way to represent information so that it can be extracted later.

1.3.1 Real- and Complex-valued Signals

A discrete-time signal is represented symbolically as $s(n)$, where $n = \{\ldots, -1, 0, 1, \ldots\}$.

---

5This content is available online at [http://cnx.org/content/m10342/2.16/](http://cnx.org/content/m10342/2.16/).

Available for free at Connexions [http://cnx.org/content/col10360/1.4](http://cnx.org/content/col10360/1.4)
Figure 1.7: The discrete-time cosine signal is plotted as a stem plot. Can you find the formula for this signal?

We usually draw discrete-time signals as stem plots to emphasize the fact they are functions defined only on the integers. We can delay a discrete-time signal by an integer just as with analog ones. A signal delayed by $m$ samples has the expression $s(n - m)$.

1.3.2 Complex Exponentials

The most important signal is, of course, the complex exponential sequence.

$$s(n) = e^{i2\pi fn} \quad (1.1)$$

Note that the frequency variable $f$ is dimensionless and that adding an integer to the frequency of the discrete-time complex exponential has no effect on the signal’s value.

$$e^{i2\pi (f + m)n} = e^{i2\pi fn} e^{i2\pi mn} = e^{i2\pi fn} \quad (1.2)$$

This derivation follows because the complex exponential evaluated at an integer multiple of $2\pi$ equals one. Thus, we need only consider frequency to have a value in some unit-length interval.

1.3.3 Sinusoids

Discrete-time sinusoids have the obvious form $s(n) = A \cos(2\pi fn + \phi)$. As opposed to analog complex exponentials and sinusoids that can have their frequencies be any real value, frequencies of their discrete-time counterparts yield unique waveforms only when $f$ lies in the interval $(-\frac{1}{2}, \frac{1}{2})$. This choice of frequency interval is arbitrary; we can also choose the frequency to lie in the interval $[0, 1)$. How to choose a unit-length interval for a sinusoid’s frequency will become evident later.

1.3.4 Unit Sample

The second-most important discrete-time signal is the unit sample, which is defined to be

$$\delta(n) = \begin{cases} 
1 & \text{if } n = 0 \\
0 & \text{otherwise} 
\end{cases} \quad (1.3)$$
Chapter 1. Foundations

Unit sample

![Figure 1.8: The unit sample.](image)

Examination of a discrete-time signal's plot, like that of the cosine signal shown in Figure 1.7 (Cosine), reveals that all signals consist of a sequence of delayed and scaled unit samples. Because the value of a sequence at each integer \( m \) is denoted by \( s(m) \) and the unit sample delayed to occur at \( m \) is written \( \delta(n-m) \), we can decompose any signal as a sum of unit samples delayed to the appropriate location and scaled by the signal value.

\[
s(n) = \sum_{m=-\infty}^{\infty} s(m) \delta(n-m) \tag{1.4}
\]

This kind of decomposition is unique to discrete-time signals, and will prove useful subsequently.

1.3.5 Unit Step

The unit step in discrete-time is well-defined at the origin, as opposed to the situation with analog signals.

\[
u(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases} \tag{1.5}
\]

1.3.6 Symbolic Signals

An interesting aspect of discrete-time signals is that their values do not need to be real numbers. We do have real-valued discrete-time signals like the sinusoid, but we also have signals that denote the sequence of characters typed on the keyboard. Such characters certainly aren’t real numbers, and as a collection of possible signal values, they have little mathematical structure other than that they are members of a set. More formally, each element of the symbolic-valued signal \( s(n) \) takes on one of the values \( \{a_1, \ldots, a_K\} \) which comprise the alphabet \( A \). This technical terminology does not mean we restrict symbols to being members of the English or Greek alphabet. They could represent keyboard characters, bytes (8-bit quantities), integers that convey daily temperature. Whether controlled by software or not, discrete-time systems are ultimately constructed from digital circuits, which consist entirely of analog circuit elements. Furthermore, the transmission and reception of discrete-time signals, like e-mail, is accomplished with analog signals and systems. Understanding how discrete-time and analog signals and systems intertwine is perhaps the main goal of this course.

1.3.7 Discrete-Time Systems

Discrete-time systems can act on discrete-time signals in ways similar to those found in analog signals and systems. Because of the role of software in discrete-time systems, many more different systems can be envisioned and "constructed" with programs than can be with analog signals. In fact, a special class of analog signals can be converted into discrete-time signals, processed with software, and converted back into...
an analog signal, all without the incursion of error. For such signals, systems can be easily produced in software, with equivalent analog realizations difficult, if not impossible, to design.

1.4 Linear Time-Invariant Systems

A discrete-time signal \( s(n) \) is delayed by \( n_0 \) samples when we write \( s(n - n_0) \), with \( n_0 > 0 \). Choosing \( n_0 \) to be negative advances the signal along the integers. As opposed to analog delays\(^7\), discrete-time delays can only be integer valued. In the frequency domain, delaying a signal corresponds to a linear phase shift of the signal’s discrete-time Fourier transform: \( s(n - n_0) \leftrightarrow e^{-(i2\pi fn_0)}S(e^{i2\pi f}). \)

Linear discrete-time systems have the superposition property.

Superposition

\[
S(a_1x_1(n) + a_2x_2(n)) = a_1S(x_1(n)) + a_2S(x_2(n))
\] (1.6)

A discrete-time system is called shift-invariant (analogous to time-invariant analog systems) if delaying the input delays the corresponding output.

Shift-Invariant

If \( S(x(n)) = y(n) \), Then \( S(x(n - n_0)) = y(n - n_0) \) (1.7)

We use the term shift-invariant to emphasize that delays can only have integer values in discrete-time, while in analog signals, delays can be arbitrarily valued.

We want to concentrate on systems that are both linear and shift-invariant. It will be these that allow us the full power of frequency-domain analysis and implementations. Because we have no physical constraints in "constructing" such systems, we need only a mathematical specification. In analog systems, the differential equation specifies the input-output relationship in the time-domain. The corresponding discrete-time specification is the difference equation.

The Difference Equation

\[
y(n) = a_1y(n - 1) + \cdots + a_py(n - p) + b_0x(n) + b_1x(n - 1) + \cdots + b_qx(n - q)
\] (1.8)

Here, the output signal \( y(n) \) is related to its past values \( y(n - l) \), \( l = \{1, \ldots, p\} \), and to the current and past values of the input signal \( x(n) \). The system’s characteristics are determined by the choices for the number of coefficients \( p \) and \( q \) and the coefficients’ values \( \{a_1, \ldots, a_p\} \) and \( \{b_0, b_1, \ldots, b_q\} \).

Aside: There is an asymmetry in the coefficients: where is \( a_0 \) ? This coefficient would multiply the \( y(n) \) term in the difference equation (1.8. The Difference Equation). We have essentially divided the equation by it, which does not change the input-output relationship. We have thus created the convention that \( a_0 \) is always one.

As opposed to differential equations, which only provide an implicit description of a system (we must somehow solve the differential equation), difference equations provide an explicit way of computing the output for any input. We simply express the difference equation by a program that calculates each output from the previous output values, and the current and previous inputs.

1.5 Discrete Time Convolution

1.5.1 Introduction

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output a system produces for a given input signal. It can be shown that a linear time invariant system is

\(^6\)This content is available online at http://cnx.org/content/m05608/2.7/.

\(^7\)Simple Systems\(^6\): Section Delay <http://cnx.org/content/m0006/latest/#delay>

\(^8\)This content is available online at <http://cnx.org/content/m10087/2.30/>.
completely characterized by its impulse response. The sifting property of the discrete time impulse function tells us that the input signal to a system can be represented as a sum of scaled and shifted unit impulses. Thus, by linearity, it would seem reasonable to compute of the output signal as the sum of scaled and shifted unit impulse responses. That is exactly what the operation of convolution accomplishes. Hence, convolution can be used to determine a linear time invariant system’s output from knowledge of the input and the impulse response.

1.5.2 Convolution and Circular Convolution

1.5.2.1 Convolution

1.5.2.1.1 Operation Definition

Discrete time convolution is an operation on two discrete time signals defined by the integral

\[(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k] g[n - k] \tag{1.9}\]

for all signals \(f, g\) defined on \(\mathbb{Z}\). It is important to note that the operation of convolution is commutative, meaning that

\[f * g = g * f \tag{1.10}\]

for all signals \(f, g\) defined on \(\mathbb{Z}\). Thus, the convolution operation could have been just as easily stated using the equivalent definition

\[(f * g)[n] = \sum_{k=-\infty}^{\infty} f[n - k] g[k] \tag{1.11}\]

for all signals \(f, g\) defined on \(\mathbb{Z}\). Convolution has several other important properties not listed here but explained and derived in a later module.

1.5.2.1.2 Definition Motivation

The above operation definition has been chosen to be particularly useful in the study of linear time invariant systems. In order to see this, consider a linear time invariant system \(H\) with unit impulse response \(h\). Given a system input signal \(x\) we would like to compute the system output signal \(H(x)\). First, we note that the input can be expressed as the convolution

\[x[n] = \sum_{k=-\infty}^{\infty} x[k] \delta[n - k] \tag{1.12}\]

by the sifting property of the unit impulse function. By linearity

\[H(x[n]) = \sum_{k=-\infty}^{\infty} x[k] H(\delta[n - k]) \tag{1.13}\]

Since \(H(\delta[n - k])\) is the shifted unit impulse response \(h[n - k]\), this gives the result

\[H(x[n]) = \sum_{k=-\infty}^{\infty} x[k] h[n - k] = (x * h)[n] \tag{1.14}\]

Hence, convolution has been defined such that the output of a linear time invariant system is given by the convolution of the system input with the system unit impulse response.
1.5.2.1.3 Graphical Intuition

It is often helpful to be able to visualize the computation of a convolution in terms of graphical processes. Consider the convolution of two functions \( f, g \) given by

\[
(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k] g[n-k] = \sum_{k=-\infty}^{\infty} f[n-k] g[k].
\]

(1.15)

The first step in graphically understanding the operation of convolution is to plot each of the functions. Next, one of the functions must be selected, and its plot reflected across the \( k = 0 \) axis. For each real \( n \), that same function must be shifted left by \( n \). The point-wise product of the two resulting plots is then computed, and then all of the values are summed.

**Example 1.1**

Recall that the impulse response for a discrete time echoing feedback system with gain \( a \) is

\[
h[n] = a^n u[n],
\]

(1.16)

and consider the response to an input signal that is another exponential

\[
x[n] = b^n u[n].
\]

(1.17)

We know that the output for this input is given by the convolution of the impulse response with the input signal

\[
y[n] = x[n] * h[n].
\]

(1.18)

We would like to compute this operation by beginning in a way that minimizes the algebraic complexity of the expression. However, in this case, each possible choice is equally simple. Thus, we would like to compute

\[
y[n] = \sum_{k=-\infty}^{\infty} a^k u[k] b^{n-k} u[n-k].
\]

(1.19)

The step functions can be used to further simplify this sum. Therefore,

\[
y[n] = 0
\]

for \( n < 0 \) and

\[
y[n] = \sum_{k=0}^{n} (ab)^k
\]

(1.21)

for \( n \geq 0 \). Hence, provided \( ab \neq 1 \), we have that

\[
y[n] = \begin{cases} 
0 & n < 0 \\
\frac{1-(ab)^{n+1}}{1-(ab)} & n \geq 0
\end{cases}
\]

(1.22)
1.5.2.2 Circular Convolution

Discrete time circular convolution is an operation on two finite length or periodic discrete time signals defined by the sum

\[(f \circledast g)[n] = \sum_{k=0}^{N-1} f[k] \cdot \hat{g}[n-k] (1.23)\]

for all signals \(f, g\) defined on \(Z[0, N-1]\) where \(\hat{f}, \hat{g}\) are periodic extensions of \(f\) and \(g\). It is important to note that the operation of circular convolution is commutative, meaning that

\[f \circledast g = g \circledast f (1.24)\]

for all signals \(f, g\) defined on \(Z[0, N-1]\). Thus, the circular convolution operation could have been just as easily stated using the equivalent definition

\[(f \circledast g)[n] = \sum_{k=0}^{N-1} f[n-k] \cdot \hat{g}[k] (1.25)\]

for all signals \(f, g\) defined on \(Z[0, N-1]\) where \(\hat{f}, \hat{g}\) are periodic extensions of \(f\) and \(g\). Circular convolution has several other important properties not listed here but explained and derived in a later module.

Alternatively, discrete time circular convolution can be expressed as the sum of two summations given by

\[(f \circledast g)[n] = \sum_{k=0}^{n} f[k] \cdot \hat{g}[n-k] + \sum_{k=n+1}^{N-1} f[k] \cdot \hat{g}[n-k + N] (1.26)\]

for all signals \(f, g\) defined on \(Z[0, N-1]\).

Meaningful examples of computing discrete time circular convolutions in the time domain would involve complicated algebraic manipulations dealing with the wrap around behavior, which would ultimately be more confusing than helpful. Thus, none will be provided in this section. Of course, example computations in the time domain are easy to program and demonstrate. However, discrete time circular convolutions are more easily computed using frequency domain tools as will be shown in the discrete time Fourier series section.

1.5.2.2.1 Definition Motivation

The above operation definition has been chosen to be particularly useful in the study of linear time invariant systems. In order to see this, consider a linear time invariant system \(H\) with unit impulse response \(h\). Given a periodic system input signal \(x\) we would like to compute the system output signal \(H(x)\). First, we note that the input can be expressed as the circular convolution

\[x[n] = \sum_{k=0}^{N-1} x[k] \cdot \hat{\delta}[n-k] (1.27)\]

by the sifting property of the unit impulse function. By linearity,

\[H(x[n]) = \sum_{k=0}^{N-1} x[k] \cdot H(\hat{\delta}[n-k]) . (1.28)\]
Since $H(\delta[n-k])$ is the shifted unit impulse response $h[n-k]$, this gives the result

$$H(x[n]) = \sum_{k=0}^{N-1} x[k] h[n-k] = (x \ast h)[n].$$

Hence, circular convolution has been defined such that the output of a linear time invariant system is given by the convolution of the system input with the system unit impulse response.

### 1.5.2.2 Graphical Intuition

It is often helpful to be able to visualize the computation of a circular convolution in terms of graphical processes. Consider the circular convolution of two finite length functions $f, g$ given by

$$(f \circledast g)[n] = \sum_{k=0}^{N-1} \hat{f}[k] \hat{g}[n-k] = \sum_{k=0}^{N-1} f[n-k] g[k].$$

The first step in graphically understanding the operation of convolution is to plot each of the periodic extensions of the functions. Next, one of the functions must be selected, and its plot reflected across the $k = 0$ axis. For each $n \in \mathbb{Z}[0, N-1]$, that same function must be shifted left by $n$. The point-wise product of the two resulting plots is then computed, and finally all of these values are summed.
1.5.3 Interactive Element

Figure 1.9: Interact (when online) with the Mathematica CDF demonstrating Discrete Linear Convolution. To download, right click and save as .cdf. Available for free at Connexions [http://cnx.org/content/col10360/1.4]
1.5.4 Convolution Summary

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output signal of a linear time invariant system for a given input signal with knowledge of the system’s unit impulse response. The operation of discrete time convolution is defined such that it performs this function for infinite length discrete time signals and systems. The operation of discrete time circular convolution is defined such that it performs this function for finite length and periodic discrete time signals. In each case, the output of the system is the convolution or circular convolution of the input signal with the unit impulse response.
1.6 Review of Linear Algebra

Vector spaces are the principal object of study in linear algebra. A vector space is always defined with respect to a field of scalars.

1.6.1 Fields

A field is a set \( F \) equipped with two operations, addition and multiplication, and containing two special members 0 and 1 \((0 \neq 1)\), such that for all \( \{a, b, c\} \in F \):

1. \( (a + b) \in F \)
   \[ a + b = b + a \]
   \[ (a + b) + c = a + (b + c) \]
   \[ a + 0 = a \]
   e. there exists \(-a\) such that \(a + (-a) = 0\)
2. \( ab \in F \)
   \[ ab = ba \]
   \[ (ab)c = a(bc) \]
   \[ a \cdot 1 = a \]
   e. there exists \(a^{-1}\) such that \(aa^{-1} = 1\)
3. \( a(b + c) = ab + ac \)

More concisely

1. \( F \) is an abelian group under addition
2. \( F \) is an abelian group under multiplication
3. multiplication distributes over addition

1.6.1.1 Examples

\( \mathbb{Q}, \mathbb{R}, \mathbb{C} \)

1.6.2 Vector Spaces

Let \( F \) be a field, and \( V \) a set. We say \( V \) is a vector space over \( F \) if there exist two operations, defined for all \( a \in F, u \in V \) and \( v \in V \):

- vector addition: \((u, v) \rightarrow (u + v) \in V\)
- scalar multiplication: \((a, v) \rightarrow av \in V\)

and if there exists an element denoted \( 0 \in V \), such that the following hold for all \( a \in F, b \in F, \) and \( u \in V, v \in V, \) and \( w \in V \):

1. \( u + (v + w) = (u + v) + w \)
   \[ u + v = v + u \]
   \[ u + 0 = u \]
   d. there exists \(-u\) such that \(u + (-u) = 0\)
2. \( a(u + v) = au + av \)
   \[ (a + b)u = au + bu \]
   \[ (ab)u = a(bu) \]
   d. \( 1 \cdot u = u \)

More concisely,

\(^9\)This content is available online at <http://cnx.org/content/m11948/1.2/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
1. $V$ is an abelian group under plus
2. Natural properties of scalar multiplication

### 1.6.2.1 Examples
- $\mathbb{R}^N$ is a vector space over $\mathbb{R}$
- $\mathbb{C}^N$ is a vector space over $\mathbb{C}$
- $\mathbb{C}^N$ is a vector space over $\mathbb{R}$
- $\mathbb{R}^N$ is not a vector space over $\mathbb{C}$

The elements of $V$ are called **vectors**.

### 1.6.3 Euclidean Space
Throughout this course we will think of a signal as a vector

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}^T$$

The samples $\{x_i\}$ could be samples from a finite duration, continuous time signal, for example.

A signal will belong to one of two vector spaces:

#### 1.6.3.1 Real Euclidean space
$x \in \mathbb{R}^N$ (over $\mathbb{R}$)

#### 1.6.3.2 Complex Euclidean space
$x \in \mathbb{C}^N$ (over $\mathbb{C}$)

### 1.6.4 Subspaces
Let $V$ be a vector space over $F$.

A subset $S \subseteq V$ is called a **subspace** of $V$ if $S$ is a vector space over $F$ in its own right.

**Example 1.2**
$V = \mathbb{R}^2$, $F = \mathbb{R}$, $S$ = any line though the origin.

**Figure 1.10:** $S$ is any line through the origin.

Are there other subspaces?

**Theorem 1.1:**
$S \subseteq V$ is a subspace if and only if for all $a \in F$ and $b \in F$ and for all $s \in S$ and $t \in S$, $(as + bt) \in S$
1.6.5 Linear Independence

Let $u_1, \ldots, u_k \in V$.

We say that these vectors are **linearly dependent** if there exist scalars $a_1, \ldots, a_k \in F$ such that

$$\sum_{i=1}^{k} a_i u_i = 0$$  \hspace{1cm} (1.31)

and at least one $a_i \neq 0$.

If (1.31) only holds for the case $a_1 = \cdots = a_k = 0$, we say that the vectors are **linearly independent**.

**Example 1.3**

$$\begin{bmatrix} 1 & -1 & 2 \end{bmatrix} - 2 \begin{bmatrix} -2 & 3 & 0 \end{bmatrix} + 1 \begin{bmatrix} -5 & 7 & -2 \end{bmatrix} = 0$$

so these vectors are linearly dependent in $\mathbb{R}^3$.

1.6.6 Spanning Sets

Consider the subset $S = \{v_1, v_2, \ldots, v_k\}$. Define the **span** of $S$

$$< S > \equiv \text{span}(S) \equiv \left\{ \sum_{i=1}^{k} a_i v_i \mid a_i \in F \right\}$$

**Fact:** $< S >$ is a subspace of $V$.

**Example 1.4**

$$V = \mathbb{R}^3, F = \mathbb{R}, S = \{v_1, v_2\}, v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow < S > = \text{xy-plane}.$$
Figure 1.11: \(<S>\) is the xy-plane.

1.6.6.1 Aside

If \(S\) is infinite, the notions of linear independence and span are easily generalized:

We say \(S\) is linearly independent if, for every finite collection \(u_1, \ldots, u_k \in S\), \((k\) arbitrary) we have

\[
\left( \sum_{i=1}^{k} a_i u_i = 0 \right) \Rightarrow \forall i : (a_i = 0)
\]

The span of \(S\) is

\[
< S > = \left\{ \sum_{i=1}^{k} a_i u_i \mid a_i \in F \land u_i \in S \land (k < \infty) \right\}
\]

**Note:** In both definitions, we only consider finite sums.

1.6.7 Bases

A set \(B \subseteq V\) is called a **basis** for \(V\) over \(F\) if and only if

1. \(B\) is linearly independent
2. \(<B> = V\)

Bases are of fundamental importance in signal processing. They allow us to decompose a signal into building blocks (basis vectors) that are often more easily understood.
Chapter 1. Foundations

Example 1.5

\( V = \text{(real or complex) Euclidean space, } \mathbb{R}^N \text{ or } \mathbb{C}^N. \)

\[ B = \{e_1, \ldots, e_N\} \equiv \text{standard basis} \]

\[ e_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \]

where the 1 is in the \( i \)th position.

Example 1.6

\( V = \mathbb{C}^N \text{ over } \mathbb{C}. \)

\[ B = \{u_1, \ldots, u_N\} \]

which is the DFT basis.

\[ u_k = \begin{pmatrix} 1 \\ e^{-i(\pi \frac{k}{N})} \\ \vdots \\ e^{-i(\pi \frac{k}{N}(N-1))} \end{pmatrix} \]

where \( i = \sqrt{-1}. \)

1.6.7.1 Key Fact

If \( B \) is a basis for \( V \), then every \( v \in V \) can be written uniquely (up to order of terms) in the form

\[ v = \sum_{i=1}^{N} a_i v_i \]

where \( a_i \in F \) and \( v_i \in B \).

1.6.7.2 Other Facts

- If \( S \) is a linearly independent set, then \( S \) can be extended to a basis.
- If \( \langle S \rangle = V \), then \( S \) contains a basis.

1.6.8 Dimension

Let \( V \) be a vector space with basis \( B \). The dimension of \( V \), denoted \( \dim(V) \), is the cardinality of \( B \).

**Theorem 1.2:**

Every vector space has a basis.

**Theorem 1.3:**

Every basis for a vector space has the same cardinality.

\( \Rightarrow \dim(V) \) is well-defined.

If \( \dim(V) < \infty \), we say \( V \) is finite dimensional.
1.6.8.1 Examples

<table>
<thead>
<tr>
<th>vector space</th>
<th>field of scalars</th>
<th>dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{R}^N )</td>
<td>( \mathbb{R} )</td>
<td></td>
</tr>
<tr>
<td>( \mathbb{C}^N )</td>
<td>( \mathbb{C} )</td>
<td></td>
</tr>
<tr>
<td>( \mathbb{C}^N )</td>
<td>( \mathbb{R} )</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2

Every subspace is a vector space, and therefore has its own dimension.

**Example 1.7**
Suppose \( (S = \{ u_1, \ldots, u_k \}) \subseteq V \) is a linearly independent set. Then

\[
\dim (\langle S \rangle) =
\]

**Facts**
- If \( S \) is a subspace of \( V \), then \( \dim (S) \leq \dim (V) \).
- If \( \dim (S) = \dim (V) < \infty \), then \( S = V \).

1.6.9 Direct Sums

Let \( V \) be a vector space, and let \( S \subseteq V \) and \( T \subseteq V \) be subspaces.

We say \( V \) is the **direct sum** of \( S \) and \( T \), written \( V = S \oplus T \), if and only if for every \( v \in V \), there exist unique \( s \in S \) and \( t \in T \) such that \( v = s + t \).

If \( V = S \oplus T \), then \( T \) is called a **complement** of \( S \).

**Example 1.8**

\[
V = C' = \{ f : \mathbb{R} \to \mathbb{R} | f \text{ is continuous} \}
\]

\[
S = \text{even functions in } C'
\]

\[
T = \text{odd functions in } C'
\]

\[
f(t) = \frac{1}{2} (f(t) + f(-t)) + \frac{1}{2} (f(t) - f(-t))
\]

If \( f = g + h = g' + h' \), \( g \in S \) and \( g' \in S \), \( h \in T \) and \( h' \in T \), then \( g - g' = h' - h \) is odd and even, which implies \( g = g' \) and \( h = h' \).

1.6.9.1 Facts
1. Every subspace has a complement
2. \( V = S \oplus T \) if and only if
   a. \( S \cap T = \{0\} \)
   b. \( \langle S, T \rangle = V \)
3. If \( V = S \oplus T \), and \( \dim (V) < \infty \), then \( \dim (V) = \dim (S) + \dim (T) \)

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
1.6.9.2 Proofs

Invoke a basis.

1.6.10 Norms

Let $V$ be a vector space over $F$. A norm is a mapping $V \rightarrow F$, denoted by $\| \cdot \|$, such that for all $u \in V$, $v \in V$, and $\lambda \in F$

1. $\| u \| > 0$ if $u \neq 0$
2. $\| \lambda u \| = |\lambda| \| u \|$
3. $\| u + v \| \leq \| u \| + \| v \|$

1.6.10.1 Examples

Euclidean norms:

$x \in \mathbb{R}^N$:

$$\| x \| = \left( \sum_{i=1}^{N} x_i^2 \right)^{\frac{1}{2}}$$

$x \in \mathbb{C}^N$:

$$\| x \| = \left( \sum_{i=1}^{N} (|x_i|^2) \right)^{\frac{1}{2}}$$

1.6.10.2 Induced Metric

Every norm induces a metric on $V$

$$d(u, v) \equiv \| u - v \|$$

which leads to a notion of "distance" between vectors.

1.6.11 Inner products

Let $V$ be a vector space over $F$, $F = \mathbb{R}$ or $\mathbb{C}$. An inner product is a mapping $V \times V \rightarrow F$, denoted $\langle \cdot, \cdot \rangle$, such that

1. $\langle v, v \rangle \geq 0$, and $\langle v, v \rangle = 0 \iff v = 0$
2. $\langle u, v \rangle = \langle v, u \rangle$
3. $\langle au + bv, w \rangle = a \langle u, w \rangle + b \langle v, w \rangle$

1.6.11.1 Examples

$\mathbb{R}^N$ over $\mathbb{R}$:

$$\langle x, y \rangle = (x^T y) = \sum_{i=1}^{N} x_i y_i$$

$\mathbb{C}^N$ over $\mathbb{C}$:

$$\langle x, y \rangle = (x^H y) = \sum_{i=1}^{N} \overline{x_i} y_i$$
If \( x = (x_1, \ldots, x_N)^T \in \mathbb{C} \), then
\[
x^H \equiv \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}^T
\]
is called the "Hermitian," or "conjugate transpose" of \( x \).

### 1.6.12 Triangle Inequality

If we define \( \| u \| = \langle u, u \rangle \), then
\[
\| u + v \| \leq \| u \| + \| v \|
\]
Hence, every inner product induces a norm.

### 1.6.13 Cauchy-Schwarz Inequality

For all \( u \in V \), \( v \in V \),
\[
| \langle u, v \rangle | \leq \| u \| \| v \|
\]
In inner product spaces, we have a notion of the angle between two vectors:
\[
\left( \angle (u, v) = \arccos \left( \frac{\langle u, v \rangle}{\| u \| \| v \|} \right) \right) \in [0, 2\pi)
\]

### 1.6.14 Orthogonality

\( u \) and \( v \) are **orthogonal** if
\[
\langle u, v \rangle = 0
\]
Notation: \( u \perp v \).

If in addition \( \| u \| = \| v \| = 1 \), we say \( u \) and \( v \) are **orthonormal**.

In an orthogonal (orthonormal) set, each pair of vectors is orthogonal (orthonormal).

---

**Figure 1.12**: Orthogonal vectors in \( \mathbb{R}^2 \).

---

### 1.6.15 Orthonormal Bases

An Orthonormal basis is a basis \( \{v_i\} \) such that
\[
\langle v_i, v_j \rangle = \delta_{ij} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j 
\end{cases}
\]
Example 1.9
The standard basis for $\mathbb{R}^N$ or $\mathbb{C}^N$

Example 1.10
The normalized DFT basis

$$u_k = \frac{1}{\sqrt{N}} \begin{pmatrix}
1 \\
e^{-\left(\frac{2\pi}{N}\right)} \\
\vdots \\
e^{-\left(\frac{2\pi}{N}(N-1)\right)}
\end{pmatrix}$$

1.6.16 Expansion Coefficients
If the representation of $v$ with respect to $\{v_i\}$ is

$$v = \sum_i a_i v_i$$

then

$$a_i = \langle v_i, v \rangle$$

1.6.17 Gram-Schmidt
Every inner product space has an orthonormal basis. Any (countable) basis can be made orthogonal by the Gram-Schmidt orthogonalization process.

1.6.18 Orthogonal Compliments
Let $S \subseteq V$ be a subspace. The orthogonal compliment $S$ is

$$S^\perp = \{ u \mid u \in V \land \langle u, v \rangle = 0 \land \forall v : (v \in S) \}$$

$S^\perp$ is easily seen to be a subspace.
If $\dim (v) < \infty$, then $V = S \oplus S^\perp$.

ASIDE: If $\dim (v) = \infty$, then in order to have $V = S \oplus S^\perp$ we require $V$ to be a Hilbert Space.

1.6.19 Linear Transformations
Loosely speaking, a linear transformation is a mapping from one vector space to another that preserves vector space operations.

More precisely, let $V, W$ be vector spaces over the same field $F$. A linear transformation is a mapping $T : V \rightarrow W$ such that

$$T(au + bv) = aT(u) + bT(v)$$

for all $a \in F$, $b \in F$ and $u \in V$, $v \in V$.

In this class we will be concerned with linear transformations between (real or complex) Euclidean spaces, or subspaces thereof.

1.6.20 Image

$$\text{image} (T) = \{ w \mid w \in W \land T(v) = w \text{ for some } v \}$$

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
1.6.21 Nullspace

Also known as the kernel:

\[ \ker(T) = \{ v \mid v \in V \land (T(v) = 0) \} \]

Both the image and the nullspace are easily seen to be subspaces.

1.6.22 Rank

\[ \text{rank}(T) = \dim(\text{image}(T)) \]

1.6.23 Nullity

\[ \text{null}(T) = \dim(\ker(T)) \]

1.6.24 Rank plus nullity theorem

\[ \text{rank}(T) + \text{null}(T) = \dim(V) \]

1.6.25 Matrices

Every linear transformation \( T \) has a **matrix representation.** If \( T : E^N \rightarrow E^M, E = \mathbb{R} \text{ or } \mathbb{C} \), then \( T \) is represented by an \( M \times N \) matrix

\[
A = \begin{pmatrix}
a_{11} & \cdots & a_{1N} \\
\vdots & \ddots & \vdots \\
a_{M1} & \cdots & a_{MN}
\end{pmatrix}
\]

where \((a_{11}, \ldots, a_{M1})^T = T(e_i) \) and \( e_i = (0, \ldots, 1, \ldots, 0)^T \) is the \( i \)th **standard basis** vector.

**Aside:** A linear transformation can be represented with respect to any bases of \( E^N \) and \( E^M \), leading to a different \( A \). We will always represent a linear transformation using the standard bases.

1.6.26 Column span

\[ \text{colspan}(A) = \langle A \rangle = \text{image}(A) \]

1.6.27 Duality

If \( A : \mathbb{R}^N \rightarrow \mathbb{R}^M \), then

\[ \ker^\perp(A) = \text{image}(A^T) \]
If $A : \mathbb{C}^N \rightarrow \mathbb{C}^M$, then

$$
\ker(A) \perp (A) = \text{image} (A^H)
$$

1.6.28 Inverses

The linear transformation/matrix $A$ is invertible if and only if there exists a matrix $B$ such that $AB = BA = I$ (identity).

Only square matrices can be invertible.

**Theorem 1.4:**

Let $A : F^N \rightarrow F^N$ be linear, $F = \mathbb{R}$ or $\mathbb{C}$. The following are equivalent:

1. $A$ is invertible (nonsingular)
2. rank $(A) = N$
3. null $(A) = 0$
4. $\det A \neq 0$
5. The columns of $A$ form a basis.

If $A^{-1} = A^T$ (or $A^H$ in the complex case), we say $A$ is orthogonal (or unitary).

1.7 Hilbert Spaces$^{10}$

1.7.1 Hilbert Spaces

A vector space $S$ with a valid inner product$^{11}$ defined on it is called an inner product space, which is also a normed linear space. A Hilbert space is an inner product space that is complete with respect to the norm defined using the inner product. Hilbert spaces are named after David Hilbert$^{12}$, who developed this idea through his studies of integral equations. We define our valid norm using the inner product as:

$$
\| x \| = \sqrt{\langle x, x \rangle}
$$

---

$^{10}$This content is available online at <http://cnx.org/content/m10840/2.7/>.

$^{11}$"Inner Products" <http://cnx.org/content/m10755/latest/>

$^{12}$http://www-history.mcs.st-andrews.ac.uk/history/Mathematicians/Hilbert.html

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Hilbert spaces are useful in studying and generalizing the concepts of Fourier expansion, Fourier transforms, and are very important to the study of quantum mechanics. Hilbert spaces are studied under the functional analysis branch of mathematics.

1.7.1.1 Examples of Hilbert Spaces

Below we will list a few examples of Hilbert spaces. You can verify that these are valid inner products at home.

- For \( \mathbb{C}^n \),

\[
\langle x, y \rangle = y^T x = \begin{pmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} x_i y_i
\]

- Space of finite energy complex functions: \( L^2(\mathbb{R}) \)

\[
\langle f, g \rangle = \int_{-\infty}^{\infty} f(t) \overline{g(t)} dt
\]

- Space of square-summable sequences: \( \ell^2(\mathbb{Z}) \)

\[
\langle x, y \rangle = \sum_{i=-\infty}^{\infty} x[i] \overline{y[i]}
\]

1.8 Signal Expansions

1.8.1 Main Idea

When working with signals many times it is helpful to break up a signal into smaller, more manageable parts. Hopefully by now you have been exposed to the concept of eigenvectors and there use in decomposing a signal into one of its possible basis. By doing this we are able to simplify our calculations of signals and systems through eigenfunctions of LTI systems.

Now we would like to look at an alternative way to represent signals, through the use of orthonormal basis. We can think of orthonormal basis as a set of building blocks we use to construct functions. We will build up the signal/vector as a weighted sum of basis elements.

Example 1.11

The complex sinusoids \( \frac{1}{\sqrt{T}} e^{j\omega_0 nt} \) for all \(-\infty < n < \infty\) form an orthonormal basis for \( L^2([0, T]) \).

In our Fourier series equation, \( f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega_0 nt} \), the \( \{c_n\} \) are just another representation of \( f(t) \).

Note: For signals/vectors in a Hilbert Space, the expansion coefficients are easy to find.
1.8.2 Alternate Representation

Recall our definition of a basis: A set of vectors \( \{b_i\} \) in a vector space \( S \) is a basis if

1. The \( b_i \) are linearly independent.
2. The \( b_i \) span \( S \). That is, we can find \( \{\alpha_i\} \), where \( \alpha_i \in \mathbb{C} \) (scalars) such that

\[
\forall x, x \in S : \left( x = \sum_i \alpha_i b_i \right) \tag{1.33}
\]

where \( x \) is a vector in \( S \), \( \alpha \) is a scalar in \( \mathbb{C} \), and \( b \) is a vector in \( S \).

Condition 2 in the above definition says we can decompose any vector in terms of the \( \{b_i\} \). Condition 1 ensures that the decomposition is unique (think about this at home).

**Example 1.12**

Let us look at simple example in \( \mathbb{R}^2 \), where we have the following vector:

\[
x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}
\]

Standard Basis: \( \{e_0, e_1\} = \{(1, 0)^T, (0, 1)^T\} \)

\[
x = e_0 + 2e_1
\]

Alternate Basis: \( \{h_0, h_1\} = \{(1, 1)^T, (1, -1)^T\} \)

\[
x = \frac{3}{2} h_0 + \frac{-1}{2} h_1
\]

In general, given a basis \( \{b_0, b_1\} \) and a vector \( x \in \mathbb{R}^2 \), how do we find the \( \alpha_0 \) and \( \alpha_1 \) such that

\[
x = \alpha_0 b_0 + \alpha_1 b_1 \tag{1.34}
\]

1.8.3 Finding the Coefficients

Now let us address the question posed above about finding \( \alpha_i \)'s in general for \( \mathbb{R}^2 \). We start by rewriting (1.34) so that we can stack our \( b_i \)'s as columns in a \( 2 \times 2 \) matrix.

\[
\begin{pmatrix} x \\ \end{pmatrix} = \alpha_0 \begin{pmatrix} b_0 \\ \end{pmatrix} + \alpha_1 \begin{pmatrix} b_1 \\ \end{pmatrix} \tag{1.35}
\]

\[
\begin{pmatrix} \vdots & \vdots \\ b_0 & b_1 \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \tag{1.36}
\]

---

188”Linear Algebra: The Basics”: Section Span <http://cnx.org/content/m10734/latest/#span_sec>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Example 1.13
Here is a simple example, which shows a little more detail about the above equations.

\[
\begin{pmatrix}
  x[0] \\
  x[1]
\end{pmatrix} =
\begin{pmatrix}
  \alpha_0 \\
  \alpha_1
\end{pmatrix}
+ \begin{pmatrix}
  b_0[0] \\
  b_0[1]
\end{pmatrix} + \begin{pmatrix}
  b_1[0] \\
  b_1[1]
\end{pmatrix}
= \begin{pmatrix}
  \alpha_0 b_0[0] + \alpha_1 b_1[0] \\
  \alpha_0 b_0[1] + \alpha_1 b_1[1]
\end{pmatrix}
\]

(1.37)

\[
\begin{pmatrix}
  x[0] \\
  x[1]
\end{pmatrix} =
\begin{pmatrix}
  b_0[0] & b_1[0] \\
  b_0[1] & b_1[1]
\end{pmatrix}
\begin{pmatrix}
  \alpha_0 \\
  \alpha_1
\end{pmatrix}
\]

(1.38)

1.8.3.1 Simplifying our Equation
To make notation simpler, we define the following two items from the above equations:

- **Basis Matrix:**
  \[
  B =
  \begin{pmatrix}
    \vdots & \vdots \\
    b_0 & b_1 \\
    \vdots & \vdots
  \end{pmatrix}
  \]

- **Coefficient Vector:**
  \[
  \alpha =
  \begin{pmatrix}
    \alpha_0 \\
    \alpha_1
  \end{pmatrix}
  \]

This gives us the following, concise equation:

\[
x = B\alpha
\]

(1.39)

which is equivalent to \( x = \sum_{i=0}^{1} \alpha_i b_i \).

Example 1.14
Given a standard basis, \( \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \), then we have the following basis matrix:

\[
B =
\begin{pmatrix}
  0 & 1 \\
  1 & 0
\end{pmatrix}
\]

To get the \( \alpha_i \)'s, we solve for the coefficient vector in (1.39)

\[
\alpha = B^{-1} x
\]

(1.40)

Where \( B^{-1} \) is the inverse matrix\(^{19}\) of \( B \).

\(^{19}\) "Matrix Inversion" <http://cnx.org/content/m2113/latest/>
1.8.3.2 Examples

Example 1.15
Let us look at the standard basis first and try to calculate $\alpha$ from it.

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

Where $I$ is the identity matrix. In order to solve for $\alpha$ let us find the inverse of $B$ first (which is obviously very trivial in this case):

$$B^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Therefore we get,

$$\alpha = B^{-1}x = x$$

Example 1.16
Let us look at a ever-so-slightly more complicated basis of

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\} = \{h_0, h_1\}$$

Then our basis matrix and inverse basis matrix becomes:

$$B = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$B^{-1} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

and for this example it is given that

$$x = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

Now we solve for $\alpha$

$$\alpha = B^{-1}x = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 0.5 \end{pmatrix}$$

and we get

$$x = 2.5h_0 + 0.5h_1$$

Exercise 1.8.1
(Solution on p. 48.)

Now we are given the following basis matrix and $x$:

$$\left\{ b_0, b_1 \right\} = \left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \end{pmatrix} \right\}$$

$$x = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

For this problem, make a sketch of the bases and then represent $x$ in terms of $b_0$ and $b_1$. 
A change of basis simply looks at $x$ from a "different perspective." $B^{-1}$ transforms $x$ from the standard basis to our new basis, $\{b_0, b_1\}$. Notice that this is a totally mechanical procedure.

### 1.8.4 Extending the Dimension and Space

We can also extend all these ideas past just $\mathbb{R}^2$ and look at them in $\mathbb{R}^n$ and $\mathbb{C}^n$. This procedure extends naturally to higher ($>2$) dimensions. Given a basis $\{b_0, b_1, \ldots, b_{n-1}\}$ for $\mathbb{R}^n$, we want to find $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ such that

$$x = \alpha_0 b_0 + \alpha_1 b_1 + \cdots + \alpha_{n-1} b_{n-1}$$

(1.41)

Again, we will set up a basis matrix

$$B = \begin{pmatrix} b_0 & b_1 & b_2 & \ldots & b_{n-1} \end{pmatrix}$$

where the columns equal the basis vectors and it will always be an $n \times n$ matrix (although the above matrix does not appear to be square since we left terms in vector notation). We can then proceed to rewrite (1.39)

$$x = \begin{pmatrix} b_0 & b_1 & \ldots & b_{n-1} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} = B\alpha$$

and

$$\alpha = B^{-1}x$$

### 1.9 Introduction to Fourier Analysis

#### 1.9.1 Fourier’s Daring Leap

Fourier postulated around 1807 that any periodic signal (equivalently finite length signal) can be built up as an infinite linear combination of harmonic sinusoidal waves.

#### 1.9.1.1

i.e. Given the collection

$$B = \{e^{j\frac{2\pi}{T}nt}\}_{n=-\infty}^{\infty}$$

(1.42)

any

$$f(t) \in L^2[0, T)$$

(1.43)

can be approximated arbitrarily closely by

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{j\frac{2\pi}{T}nt}.$$ 

(1.44)

Now, The issue of exact convergence did bring Fourier\(^{21}\) much criticism from the French Academy of Science (Laplace, Lagrange, Monge and LaCroix comprised the review committee) for several years after its

---

20 This content is available online at [http://cnx.org/content/m10096/2.13/].
21 [http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html](http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html)

Available for free at Connexions [http://cnx.org/content/col10360/1.4]
Fourier analysis is fundamental to understanding the behavior of signals and systems. This is a result of the fact that sinusoids are Eigenfunctions \(^{22}\) of linear, time-invariant (LTI) \(^{23}\) systems. This is to say that if we pass any particular sinusoid through a LTI system, we get a scaled version of that same sinusoid on the output. Then, since Fourier analysis allows us to redefine the signals in terms of sinusoids, all we need to do is determine how any given system affects all possible sinusoids (its transfer function \(^{24}\)) and we have a complete understanding of the system. Furthermore, since we are able to define the passage of sinusoids through a system as multiplication of that sinusoid by the transfer function at the same frequency, we can convert the passage of any signal through a system from convolution \(^{25}\) (in time) to multiplication (in frequency). These ideas are what give Fourier analysis its power.

Now, after hopefully having sold you on the value of this method of analysis, we must examine exactly what we mean by Fourier analysis. The four Fourier transforms that comprise this analysis are the Fourier Series \(^{26}\), Continuous-Time Fourier Transform (Section 1.10), Discrete-Time Fourier Transform (Section 1.11) and Discrete Fourier Transform \(^{27}\). For this document, we will view the Laplace Transform \(^{28}\) and Z-Transform (Section 3.3) as simply extensions of the CTFT and DTFT respectively. All of these transforms act essentially the same way, by converting a signal in time to an equivalent signal in frequency (sinusoids). However, depending on the nature of a specific signal i.e. whether it is finite- or infinite-length and whether it is discrete- or continuous-time) there is an appropriate transform to convert the signal into the frequency domain. Below is a table of the four Fourier transforms and when each is appropriate. It also includes the relevant convolution for the specified space.

### Table of Fourier Representations

<table>
<thead>
<tr>
<th>Transform</th>
<th>Time Domain</th>
<th>Frequency Domain</th>
<th>Convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous-Time Fourier Series</td>
<td>(L^2 ([0, T]))</td>
<td>(l^2 (Z))</td>
<td>Continuous-Time Circular</td>
</tr>
<tr>
<td>Continuous-Time Fourier Transform</td>
<td>(L^2 (\mathbb{R}))</td>
<td>(L^2 (\mathbb{R}))</td>
<td>Continuous-Time Linear</td>
</tr>
<tr>
<td>Discrete-Time Fourier Transform</td>
<td>(l^2 (Z))</td>
<td>(L^2 ([0, 2\pi]))</td>
<td>Discrete-Time Linear</td>
</tr>
<tr>
<td>Discrete Fourier Transform</td>
<td>(l^2 ([0, N-1]))</td>
<td>(l^2 ([0, N-1]))</td>
<td>Discrete-Time Circular</td>
</tr>
</tbody>
</table>

Table 1.3

### 1.10 Continuous Time Fourier Transform (CTFT) \(^{29}\)

#### 1.10.1 Introduction

In this module, we will derive an expansion for any arbitrary continuous-time function, and in doing so, derive the Continuous Time Fourier Transform (CTFT).

---

\(^{22}\)"Eigenfunctions of LTI Systems" <http://cnx.org/content/m10500/latest/>  
\(^{23}\)"System Classifications and Properties" <http://cnx.org/content/m10064/latest/>  
\(^{24}\)"Transfer Functions" <http://cnx.org/content/m0028/latest/>  
\(^{25}\)"Properties of Continuous Time Convolution" <http://cnx.org/content/m10088/latest/>  
\(^{26}\)"Continuous-Time Fourier Series (CTFS)" <http://cnx.org/content/m10097/latest/>  
\(^{27}\)"Discrete Fourier Transform" <http://cnx.org/content/m0502/latest/>  
\(^{28}\)"The Laplace Transform" <http://cnx.org/content/m10110/latest/>  
\(^{29}\)This content is available online at <http://cnx.org/content/m10098/2.17/>.  

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Since complex exponentials\(^{30}\) are eigenfunctions of linear time-invariant (LTI) systems\(^{31}\), calculating the output of an LTI system \(H\) given \(e^{st}\) as an input amounts to simple multiplication, where \(H(s) \in \mathbb{C}\) is the eigenvalue corresponding to \(s\). As shown in the figure, a simple exponential input would yield the output

\[
y(t) = H(s) e^{st}
\]  

\(^{(1.45)}\)

\section*{Image not finished}

Using this and the fact that \(H\) is linear, calculating \(y(t)\) for combinations of complex exponentials is also straightforward.

\[
c_1 e^{s_1 t} + c_2 e^{s_2 t} \rightarrow c_1 H(s_1) e^{s_1 t} + c_2 H(s_2) e^{s_2 t}
\]

\[
\sum_n c_n e^{s_n t} \rightarrow \sum_n c_n H(s_n) e^{s_n t}
\]

The action of \(H\) on an input such as those in the two equations above is easy to explain. \(H\) independently scales each exponential component \(e^{s_n t}\) by a different complex number \(H(s_n) \in \mathbb{C}\). As such, if we can write a function \(f(t)\) as a combination of complex exponentials it allows us to easily calculate the output of a system.

Now, we will look to use the power of complex exponentials to see how we may represent arbitrary signals in terms of a set of simpler functions by superposition of a number of complex exponentials. Below we will present the Continuous-Time Fourier Transform (CTFT), commonly referred to as just the Fourier Transform (FT). Because the CTFT deals with nonperiodic signals, we must find a way to include all real frequencies in the general equations. For the CTFT we simply utilize integration over real numbers rather than summation over integers in order to express the aperiodic signals.

\subsection*{1.10.2 Fourier Transform Synthesis}

Joseph Fourier\(^{32}\) demonstrated that an arbitrary \(s(t)\) can be written as a linear combination of harmonic complex sinusoids

\[
s(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega_0 nt}
\]  

\(^{(1.46)}\)

where \(\omega_0 = \frac{2\pi}{T}\) is the fundamental frequency. For almost all \(s(t)\) of practical interest, there exists \(c_n\) to make \((1.46)\) true. If \(s(t)\) is finite energy (\(s(t) \in L^2[0,T]\)), then the equality in \((1.46)\) holds in the sense of energy convergence; if \(s(t)\) is continuous, then \((1.46)\) holds pointwise. Also, if \(s(t)\) meets some mild conditions (the Dirichlet conditions), then \((1.46)\) holds pointwise everywhere except at points of discontinuity.

The \(c_n\) - called the Fourier coefficients - tell us "how much" of the sinusoid \(e^{j\omega_0 nt}\) is in \(s(t)\). The formula shows \(s(t)\) as a sum of complex exponentials, each of which is easily processed by an LTI system (since it is an eigenfunction of every LTI system). Mathematically, it tells us that the set of complex exponentials \(\{\forall n,n \in \mathbb{Z}: (e^{j\omega_0 nt})\}\) form a basis for the space of T-periodic continuous time functions.

\subsection*{1.10.2.1 Equations}

Now, in order to take this useful tool and apply it to arbitrary non-periodic signals, we will have to delve deeper into the use of the superposition principle. Let \(s_T(t)\) be a periodic signal having period \(T\). We want to consider what happens to this signal’s spectrum as the period goes to infinity. We denote the spectrum for any assumed value of the period by \(c_n(T)\). We calculate the spectrum according to the Fourier formula

\(^{30}\)“Continuous Time Complex Exponential” <http://cnx.org/content/m10060/latest/>  
\(^{31}\)“Eigenfunctions of LTI Systems” <http://cnx.org/content/m10900/latest/>  
\(^{32}\)http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
for a periodic signal, known as the Fourier Series (for more on this derivation, see the section on Fourier Series.)

\[
c_n = \frac{1}{T} \int_0^T \! s(t) \exp(-j\omega_0 t) \, dt
\] (1.47)

where \( \omega_0 = \frac{2\pi}{T} \) and where we have used a symmetric placement of the integration interval about the origin for subsequent derivational convenience. We vary the frequency index \( n \) proportionally as we increase the period. Define

\[
S_T(f) \equiv T c_n = \frac{1}{T} \int_0^T \! (S_T(f) \exp(j\omega_0 t) \, dt
\] (1.48)

making the corresponding Fourier Series

\[
s_T(t) = \sum_{n=-\infty}^{\infty} f(t) \exp(j\omega_0 t) \frac{1}{T}
\] (1.49)

As the period increases, the spectral lines become closer together, becoming a continuum. Therefore,

\[
\lim_{T \to \infty} s_T(t) \equiv s(t) = \int_{-\infty}^{\infty} \! S(f) \exp(j\omega_0 t) \, df
\] (1.50)

with

\[
S(f) = \int_{-\infty}^{\infty} \! s(t) \exp(-j\omega_0 t) \, dt
\] (1.51)

Continuous-Time Fourier Transform

\[
\mathcal{F}(\Omega) = \int_{-\infty}^{\infty} \! f(t) e^{-j\Omega t} \, dt
\] (1.52)

Inverse CTFT

\[
f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \! \mathcal{F}(\Omega) e^{j\Omega t} \, d\Omega
\] (1.53)

WARNING: It is not uncommon to see the above formula written slightly different. One of the most common differences is the way that the exponential is written. The above equations use the radial frequency variable \( \Omega \) in the exponential, where \( \Omega = 2\pi f \), but it is also common to include the more explicit expression, \( i2\pi ft \), in the exponential. Click here\textsuperscript{33} for an overview of the notation used in Connexion’s DSP modules.

\textbf{Example 1.17}

We know from Euler’s formula that \( \cos(\omega t) + \sin(\omega t) = \frac{1-j}{2} e^{j\omega t} + \frac{1+j}{2} e^{-j\omega t} \).
1.10.3 CTFT Definition Demonstration

![CTFT Definition](image)

**Figure 1.14:** Interact (when online) with a Mathematica CDF demonstrating Continuous Time Fourier Transform. To Download, right-click and save as .cdf.

1.10.4 Example Problems

**Exercise 1.10.1**

Find the Fourier Transform (CTFT) of the function

\[ f(t) = \begin{cases} e^{-\alpha t} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \]  

(Solution on p. 48.)

**Exercise 1.10.2**

Find the inverse Fourier transform of the ideal lowpass filter defined by

\[ X(\Omega) = \begin{cases} 1 & \text{if } |\Omega| \leq M \\ 0 & \text{otherwise} \end{cases} \]  

(Solution on p. 48.)

1.10.5 Fourier Transform Summary

Because complex exponentials are eigenfunctions of LTI systems, it is often useful to represent signals using a set of complex exponentials as a basis. The continuous time Fourier series synthesis formula expresses a
CHAPTER 1. FOUNDATIONS

continuous time, periodic function as the sum of continuous time, discrete frequency complex exponentials.

\[ f(t) = \sum_{n=\infty}^{\infty} c_n e^{j\omega_0 nt} \quad (1.56) \]

The continuous time Fourier series analysis formula gives the coefficients of the Fourier series expansion.

\[ c_n = \frac{1}{T} \int_{0}^{T} f(t) e^{-j\omega_0 nt} \, dt \quad (1.57) \]

In both of these equations \( \omega_0 = \frac{2\pi}{T} \) is the fundamental frequency.

1.11 Discrete Time Fourier Transform (DTFT)\(^3^4\)

1.11.1 Introduction

In this module, we will derive an expansion for arbitrary discrete-time functions, and in doing so, derive the Discrete Time Fourier Transform (DTFT).

Since complex exponentials\(^3^5\) are eigenfunctions of linear time-invariant (LTI) systems\(^3^6\), calculating the output of an LTI system \( H \) given \( e^{j\omega n} \) as an input amounts to simple multiplication, where \( \omega_0 = \frac{2\pi k}{N} \), and where \( H[k] \in \mathbb{C} \) is the eigenvalue corresponding to \( k \). As shown in the figure, a simple exponential input would yield the output

\[ y[n] = H[k] e^{j\omega n} \quad (1.58) \]

![Figure 1.15: Simple LTI system.](image)

Using this and the fact that \( H \) is linear, calculating \( y[n] \) for combinations of complex exponentials is also straightforward.

\[ c_1 e^{j\omega_1 n} + c_2 e^{j\omega_2 n} \rightarrow c_1 H[k_1] e^{j\omega_1 n} + c_2 H[k_2] e^{j\omega_2 n} \]

\[ \sum_l c_l e^{j\omega_l n} \rightarrow \sum_l c_l H[k_l] e^{j\omega_l n} \]

The action of \( H \) on an input such as those in the two equations above is easy to explain. \( H \) independently scales each exponential component \( e^{j\omega_l n} \) by a different complex number \( H[k_l] \in \mathbb{C} \). As such, if we can write a function \( y[n] \) as a combination of complex exponentials it allows us to easily calculate the output of a system.

\(^3^4\)This content is available online at <http://cnx.org/content/m10108/2.18/>.

\(^3^5\)“Continuous Time Complex Exponential” <http://cnx.org/content/m10060/latest/>

\(^3^6\)“Eigenfunctions of LTI Systems” <http://cnx.org/content/m10500/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Now, we will look to use the power of complex exponentials to see how we may represent arbitrary signals in terms of a set of simpler functions by superposition of a number of complex exponentials. Below we will present the **Discrete-Time Fourier Transform** (DTFT). Because the DTFT deals with nonperiodic signals, we must find a way to include all real frequencies in the general equations. For the DTFT we simply utilize summation over all real numbers rather than summation over integers in order to express the aperiodic signals.

### 1.11.2 DTFT synthesis

It can be demonstrated that an arbitrary Discrete Time-periodic function \( f[n] \) can be written as a linear combination of harmonic complex sinusoids

\[
f[n] = \sum_{k=0}^{N-1} c_k e^{i\omega_0 kn}
\]  

where \( \omega_0 = \frac{2\pi}{N} \) is the fundamental frequency. For almost all \( f[n] \) of practical interest, there exists \( c_n \) to make (1.59) true. If \( f[n] \) is finite energy ( \( f[n] \in L^2[0,N] \)), then the equality in (1.59) holds in the sense of energy convergence; with discrete-time signals, there are no concerns for divergence as there are with continuous-time signals.

The \( c_n \) - called the Fourier coefficients - tell us "how much" of the sinusoid \( e^{i\omega_0 kn} \) is in \( f[n] \). The formula shows \( f[n] \) as a sum of complex exponentials, each of which is easily processed by an LTI system (since it is an eigenfunction of every LTI system). Mathematically, it tells us that the set of complex exponentials \( \{\forall k, k \in \mathbb{Z} : (e^{i\omega_0 kn})\} \) form a basis for the space of N-periodic discrete time functions.

#### 1.11.2.1 Equations

Now, in order to take this useful tool and apply it to arbitrary non-periodic signals, we will have to delve deeper into the use of the superposition principle. Let \( s_T(t) \) be a periodic signal having period \( T \). We want to consider what happens to this signal’s spectrum as the period goes to infinity. We denote the spectrum for any assumed value of the period by \( c_n(T) \). We calculate the spectrum according to the Fourier formula for a periodic signal, known as the Fourier Series (for more on this derivation, see the section on **Fourier Series**.)

\[
c_n = \frac{1}{T} \int_0^T s(t) \exp(-j\omega_0 t) \, dt
\]

where \( \omega_0 = \frac{2\pi}{T} \) and where we have used a symmetric placement of the integration interval about the origin for subsequent derivational convenience. We vary the frequency index \( n \) proportionally as we increase the period. Define

\[
S_T(f) = T c_n = \frac{1}{T} \int_0^T (S_T(f) \exp(j\omega_0 t)) \, dt
\]

making the corresponding Fourier Series

\[
s_T(t) = \sum_{-\infty}^{\infty} f(t) \exp(j\omega_0 t) \frac{1}{T}
\]

As the period increases, the spectral lines become closer together, becoming a continuum. Therefore,

\[
\lim_{T \to \infty} s_T(t) \equiv s(t) = \int_{-\infty}^{\infty} S(f) \exp(j\omega_0 t) \, df
\]
with

\[ S(f) = \int_{-\infty}^{\infty} s(t) \exp(-j\omega_0 t) \, dt \quad (1.64) \]

**Discrete-Time Fourier Transform**

\[ \mathcal{F}(\omega) = \sum_{n=-\infty}^{\infty} f[n] e^{-j\omega n} \quad (1.65) \]

**Inverse DTFT**

\[ f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathcal{F}(\omega) e^{j\omega n} \, d\omega \quad (1.66) \]

**Warning:** It is not uncommon to see the above formula written slightly different. One of the most common differences is the way that the exponential is written. The above equations use the radial frequency variable \( \omega \) in the exponential, where \( \omega = 2\pi f \), but it is also common to include the more explicit expression, \( i2\pi ft \), in the exponential. Sometimes DTFT notation is expressed as \( \mathcal{F}(e^{j\omega}) \), to make it clear that it is not a CTFT (which is denoted as \( \mathcal{F}(\Omega) \)). Click here\(^{37}\) for an overview of the notation used in Connexion’s DSP modules.

### 1.11.3 DTFT Definition demonstration

![DTFT Definition](image)

**Figure 1.16:** Click on the above thumbnail image (when online) to download an interactive Mathematica Player demonstrating Discrete Time Fourier Transform. To Download, right-click and save target as .cdf.

\(^{37}\)“DSP notation” \(<http://cnx.org/content/m10361/latest/>\)

Available for free at Connexions \(<http://cnx.org/content/col10360/1.4/>\)
1.11.4 DTFT Summary

Because complex exponentials are eigenfunctions of LTI systems, it is often useful to represent signals using a set of complex exponentials as a basis. The discrete time Fourier transform synthesis formula expresses a discrete time, aperiodic function as the infinite sum of continuous frequency complex exponentials.

\[ \mathcal{F}(\omega) = \sum_{n=-\infty}^{\infty} f[n] e^{-(i\omega n)} \]  

(1.67)

The discrete time Fourier transform analysis formula takes the same discrete time domain signal and represents the signal in the continuous frequency domain.

\[ f[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathcal{F}(\omega) e^{i\omega n} d\omega \]  

(1.68)

1.12 DFT as a Matrix Operation\[38\]

\[\text{This content is available online at } \langle \text{http://cnx.org/content/m10962/2.5/} \rangle.\]

Available for free at Connexions \(<\text{http://cnx.org/content/col10360/1.4}>\)
1.12.1 Matrix Review

Recall:

- Vectors in $\mathbb{R}^N$:
  \[
  \forall x_i, x \in \mathbb{R} : x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}
  \]

- Vectors in $\mathbb{C}^N$:
  \[
  \forall x_i, x \in \mathbb{C} : x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}
  \]

- Transposition:
  a. transpose:
  \[
  x^T = \begin{pmatrix} x_0 & x_1 & \cdots & x_{N-1} \end{pmatrix}
  \]
  b. conjugate:
  \[
  x^H = \begin{pmatrix} x_0^* & x_1^* & \cdots & x_{N-1}^* \end{pmatrix}
  \]

- Inner product\(^{39}\):
  a. real:
  \[
  x^T y = \sum_{i=0}^{N-1} x_i y_i
  \]
  b. complex:
  \[
  x^H y = \sum_{i=0}^{N-1} x_i^* y_i
  \]

- Matrix Multiplication:
  \[
  Ax = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0,N-1} \\ a_{10} & a_{11} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}
  \]
  \[
  y_k = \sum_{n=0}^{N-1} a_{kn} x_n
  \]

- Matrix Transposition:
  \[
  A^T = \begin{pmatrix} a_{00} & a_{10} & \cdots & a_{N-1,0} \\ a_{01} & a_{11} & \cdots & a_{N-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{0,N-1} & a_{1,N-1} & \cdots & a_{N-1,N-1} \end{pmatrix}
  \]

\(^{39}\)“Inner Products” <http://cnx.org/content/m10755/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Matrix transposition involved simply swapping the rows with columns.

\[ A^H = A^T \]

The above equation is Hermitian transpose.

\[ A^T_{k,n} = A_{n,k} \]

\[ A^H_{k,n} = \bar{A}_{n,k} \]

### 1.12.2 Representing DFT as Matrix Operation

Now let’s represent the DFT\(^{40}\) in vector-matrix notation.

\[
x = \begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{pmatrix}
\]

\[
X = \begin{pmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{pmatrix} \in \mathbb{C}^N
\]

Here \(x\) is the vector of time samples and \(X\) is the vector of DFT coefficients. How are \(x\) and \(X\) related:

\[
X[k] = \sum_{n=0}^{N-1} x[n] e^{-\left(i \frac{2\pi}{N} kn\right)}
\]

where

\[
a_{kn} = \left(e^{-\left(i \frac{2\pi}{N}\right)}\right)^{kn} = W_N^{kn}
\]

so

\[
X = Wx
\]

where \(X\) is the DFT vector, \(W\) is the matrix and \(x\) the time domain vector.

\[
W_{k,n} = \left(e^{-\left(i \frac{2\pi}{N}\right)}\right)^{kn}
\]

\[
X = W \begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{pmatrix}
\]

IDFT:

\[
x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \left(e^{i \frac{2\pi}{N} nk}\right)
\]

---

\(^{40}\)“Discrete Fourier Transform (DFT)" [http://cnx.org/content/m10249/latest/]

Available for free at Connexions [http://cnx.org/content/col10360/1.4]
where
\[
\left( e^{\frac{2\pi}{N}} \right)^{nk} = W_N^{nk}
\]

\(W_N^{nk}\) is the matrix Hermitian transpose. So,
\[
x = \frac{1}{N} W^H X
\]
where \(x\) is the time vector, \(\frac{1}{N} W^H\) is the inverse DFT matrix, and \(X\) is the DFT vector.

1.13 The FFT Algorithm\(^{41}\)

**Definition 1.1: FFT**
(Fast Fourier Transform) An efficient computational algorithm for computing the DFT\(^{42}\).

1.13.1 The Fast Fourier Transform FFT
DFT can be expensive to compute directly
\[
\forall k, 0 \leq k \leq N - 1 : \left( X [k] = \sum_{n=0}^{N-1} x [n] e^{-i2\pi \frac{k}{N} n} \right)
\]
For each \(k\), we must execute:
- \(N\) complex multiplies
- \(N - 1\) complex adds

The total cost of direct computation of an \(N\)-point DFT is
- \(N^2\) complex multiplies
- \(N (N - 1)\) complex adds

How many adds and multiplies of real numbers are required?
This "\(O(N^2)\)" computation rapidly gets out of hand, as \(N\) gets large:

\[
\begin{array}{cccccc}
N & 1 & 10 & 100 & 1000 & 10^6 \\
N^2 & 1 & 100 & 10,000 & 10^6 & 10^{12} \\
\end{array}
\]

Table 1.4

The FFT provides us with a much more efficient way of computing the DFT. The FFT requires only "\(O(N \log N)\)" computations to compute the \(N\)-point DFT.

\(^{41}\)This content is available online at <http://cnx.org/content/m10964/2.6/>.

\(^{42}\)"Discrete Fourier Transform (DFT)" <http://cnx.org/content/m10249/latest/>
The FFT and digital computers revolutionized DSP (1960 - 1980).

1.13.2 How does the FFT work?

- The FFT exploits the symmetries of the complex exponentials $W_N^{kn} = e^{-i\frac{2\pi}{N} kn}$
- $W_N^{kn}$ are called "twiddle factors"

**Rule 1.1**: Complex Conjugate Symmetry

$$W_N^{k(N-n)} = W_N^{-(kn)} = W_N^{-kn}$$

$$e^{-i\left(2\pi \frac{k}{N} (N-n)\right)} = e^{i2\pi \frac{k}{N} n} = e^{-i\left(2\pi \frac{k}{N} n\right)}$$

**Rule 1.2**: Periodicity in n and k

$$W_N^{kn} = W_N^{k(N+n)} = W_N^{(k+N)n}$$

$$e^{-i\frac{2\pi}{N} kn} = e^{-i\frac{2\pi}{N} k(N+n)} = e^{-i\frac{2\pi}{N} (k+N)n}$$

$$W_N = e^{-i\frac{2\pi}{N} k}$$

1.13.3 Decimation in Time FFT

- Just one of many different FFT algorithms
- The idea is to build a DFT out of smaller and smaller DFTs by decomposing $x[n]$ into smaller and smaller subsequences.
- Assume $N = 2^m$ (a power of 2)
1.13.3.1 Derivation

$N$ is even, so we can complete $X[k]$ by separating $x[n]$ into two subsequences each of length $\frac{N}{2}$.

\[
x[n] \rightarrow \begin{cases} 
\frac{N}{2} & \text{if } n = \text{even} \\
\frac{N}{2} & \text{if } n = \text{odd}
\end{cases}
\]

\[
\forall k, 0 \leq k \leq N - 1 : \left( X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \right)
\]

\[
X[k] = \sum_{n=2r} x[n] W_N^{kn} + \sum_{n=2r+1} x[n] W_N^{kn}
\]

where $0 \leq r \leq \frac{N}{2} - 1$. So

\[
X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_N^{k2r} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_N^{k(2r+1)}
\]

\[
= \sum_{r=0}^{\frac{N}{2}-1} x[2r] (W_N^2)^{kr} + W_N^{k} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] (W_N^2)^{kr}
\]

(1.69)

where $W_N^2 = e^{-(i \frac{\pi}{2})} = W_{\frac{N}{2}}$. So

\[
X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_\frac{N}{2}^{k2r} + W_\frac{N}{2}^{k} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_\frac{N}{2}^{k2r}
\]

where $\sum_{r=0}^{\frac{N}{2}-1} x[2r] W_\frac{N}{2}^{k2r}$ is $\frac{N}{2}$-point DFT of even samples ( $G[k]$) and $\sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_\frac{N}{2}^{k2r}$ is $\frac{N}{2}$-point DFT of odd samples ( $H[k]$).

\[
\forall k, 0 \leq k \leq N - 1 : \left( X[k] = G[k] + W_N^{k} H[k] \right)
\]

Decomposition of an $N$-point DFT as a sum of 2 $\frac{N}{2}$-point DFTs.

Why would we want to do this? **Because it is more efficient!**

**NOTE:** Cost to compute an $N$-point DFT is approximately $N^2$ complex mults and adds.

But decomposition into 2 $\frac{N}{2}$-point DFTs + combination requires only

\[
\left( \frac{N}{2} \right)^2 + \left( \frac{N}{2} \right)^2 + N = \frac{N^2}{2} + N
\]

where the first part is the number of complex mults and adds for $\frac{N}{2}$-point DFT, $G[k]$. The second part is the number of complex mults and adds for $\frac{N}{2}$-point DFT, $H[k]$. The third part is the number of complex mults and adds for combination. And the total is $\frac{N^2}{2} + N$ complex mults and adds.

**Example 1.18: Savings**

*For $N = 1000,*

\[
N^2 = 10^6
\]

\[
\frac{N^2}{2} + N = \frac{10^6}{2} + 1000
\]
Because 1000 is small compared to 500,000,
\[
\frac{N^2}{2} + N \simeq \frac{10^6}{2}
\]
So why stop here?! Keep decomposing. Break each of the \(\frac{N}{2}\)-point DFTs into two \(\frac{N}{4}\)-point DFTs, etc., ....
We can keep decomposing:
\[
\frac{N}{2^1} = \left\{ \frac{N}{2}, \frac{N}{4}, \frac{N}{8}, \ldots, \frac{N}{2^{m-1}}, \frac{N}{2^m} \right\} = 1
\]
where
\[
m = \log_2 N = \text{times}
\]
Computational cost: \(N\)-pt DFT \(\rightarrow\) two \(\frac{N}{2}\)-pt DFTs. The cost is \(N^2 \rightarrow 2\left(\frac{N}{2}\right)^2 + N\). So replacing each \(\frac{N}{2}\)-pt DFT with two \(\frac{N}{4}\)-pt DFTs will reduce cost to
\[
2 \left( 2 \left( \frac{N}{4} \right)^2 + \frac{N}{2} \right) + N = 4 \left( \frac{N}{4} \right)^2 + 2N = \frac{N^2}{2^2} + 2N = \frac{N^2}{2^p} + pN
\]
As we keep going \(p = \{3, 4, \ldots, m\}\), where \(m = \log_2 N\). We get the cost
\[
\frac{N^2}{2^{\log_2 N}} + N\log_2 N = \frac{N^2}{N} + N\log_2 N = N + N\log_2 N
\]
\(N + N\log_2 N\) is the total number of complex adds and mults.
For large \(N\), cost \(\simeq N\log_2 N\) or "\(O(N\log_2 N)\)" since \(N\log_2 N \gg N\) for large \(N\).

---

**Image not finished**

**Figure 1.19:** \(N = 8\) point FFT. Summing nodes \(W_n^k\) twiddle multiplication factors.

---

**NOTE:** Weird order of time samples

---

**Image not finished**

**Figure 1.20:** This is called "butterflies."

---

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Solutions to Exercises in Chapter 1

Solution to Exercise 1.8.1 (p. 32)
In order to represent $x$ in terms of $b_0$ and $b_1$ we will follow the same steps we used in the above example.

$$B = \begin{pmatrix} 1 & 2 \\ 3 & 0 \end{pmatrix}$$

$$B^{-1} = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{3} & -\frac{1}{6} \end{pmatrix}$$

$$\alpha = B^{-1} x = \begin{pmatrix} 1 \\ \frac{2}{3} \end{pmatrix}$$

And now we can write $x$ in terms of $b_0$ and $b_1$.

$$x = b_0 + \frac{2}{3} b_1$$

And we can easily substitute in our known values of $b_0$ and $b_1$ to verify our results.

Solution to Exercise 1.10.1 (p. 37)
In order to calculate the Fourier transform, all we need to use is (1.52) (Continuous-Time Fourier Transform), complex exponentials, and basic calculus.

$$F(\Omega) = \int_{-\infty}^{\infty} f(t) e^{-i\Omega t} dt$$

$$= \int_{0}^{\infty} e^{-(\alpha t)} e^{-i\Omega t} dt$$

$$= \int_{0}^{\infty} e^{-(t(\alpha + i\Omega))} dt$$

$$= 0 - \frac{1}{\alpha + i\Omega}$$

$$F(\Omega) = \frac{1}{\alpha + i\Omega}$$

Solution to Exercise 1.10.2 (p. 37)
Here we will use (1.53) (Inverse CTFT) to find the inverse FT given that $t \neq 0$.

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} x(\Omega) e^{i\Omega t} d\Omega$$

$$= \frac{1}{2\pi} e^{i\Omega t} \bigg|_{\Omega=\omega} d\Omega$$

$$= \frac{1}{\pi} \sin(Mt)$$

$$x(t) = \frac{M}{\pi} \left( \frac{\sin(Mt/\pi)}{\pi} \right)$$

---

43”Continuous Time Complex Exponential” <http://cnx.org/content/m10060/latest/>  
Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Chapter 2

Sampling and Frequency Analysis

2.1 Introduction\(^1\)

Contents of Sampling chapter

- Introduction (Current module)
- Proof (Section 2.2)
- Illustrations (Section 2.3)
- Matlab Example (Section 2.4)
- Hold operation\(^2\)
- System view (Section 2.5)
- Aliasing applet\(^3\)
- Exercises\(^4\)
- Table of formulas\(^5\)

2.1.1 Why sample?

This section introduces sampling. Sampling is the necessary fundament for all digital signal processing and communication. Sampling can be defined as the process of measuring an analog signal at distinct points.

Digital representation of analog signals offers advantages in terms of

- robustness towards noise, meaning we can send more bits/s
- use of flexible processing equipment, in particular the computer
- more reliable processing equipment
- easier to adapt complex algorithms

\(^1\)This content is available online at <http://cnx.org/content/m11419/1.29/>.
\(^2\)“Hold operation” <http://cnx.org/content/m11458/latest/>
\(^3\)“Aliasing Applet” <http://cnx.org/content/m11448/latest/>
\(^4\)“Exercises” <http://cnx.org/content/m11442/latest/>
\(^5\)“Table of Formulas” <http://cnx.org/content/m11450/latest/>
2.1.2 Claude E. Shannon

Figure 2.1: Claude Elwood Shannon (1916-2001)

Claude Shannon\(^6\) has been called the father of information theory, mainly due to his landmark papers on the "Mathematical theory of communication"\(^7\). Harry Nyquist\(^8\) was the first to state the sampling theorem in 1928, but it was not proven until Shannon proved it 21 years later in the paper "Communications in the presence of noise"\(^9\).

2.1.3 Notation

In this chapter we will be using the following notation

- Original analog signal \( x(t) \)
- Sampling frequency \( F_s \)
- Sampling interval \( T_s \) (Note that: \( F_s = \frac{1}{T_s} \))
- Sampled signal \( x_s(n) \). (Note that \( x_s(n) = x(nT_s) \))
- Real angular frequency \( \Omega \)
- Digital angular frequency \( \omega \). (Note that: \( \omega = \Omega T_s \))

2.1.4 The Sampling Theorem

**NOTE:** When sampling an analog signal the sampling frequency must be greater than twice the highest frequency component of the analog signal to be able to reconstruct the original signal from the sampled version.

\(^6\)http://www.research.att.com/~njas/doc/ces5.html
\(^7\)http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
\(^8\)http://www.wikipedia.org/wiki/Harry_Nyquist
\(^9\)http://www.stanford.edu/class/ee104/shannonpaper.pdf

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
2.1.5
Finished? Have at look at: Proof (Section 2.2); Illustrations (Section 2.3); Matlab Example (Section 2.4); Aliasing applet\textsuperscript{10}; Hold operation\textsuperscript{11}; System view (Section 2.5); Exercises\textsuperscript{12}

2.2 Proof\textsuperscript{a}

NOTE: In order to recover the signal $x(t)$ from its samples exactly, it is necessary to sample $x(t)$ at a rate greater than twice its highest frequency component.

2.2.1 Introduction
As mentioned earlier (p. 49), sampling is the necessary fundament when we want to apply digital signal processing on analog signals.

Here we present the proof of the sampling theorem. The proof is divided in two. First we find an expression for the spectrum of the signal resulting from sampling the original signal $x(t)$. Next we show that the signal $x(t)$ can be recovered from the samples. Often it is easier using the frequency domain when carrying out a proof, and this is also the case here.

Key points in the proof

- We find an equation (2.8) for the spectrum of the sampled signal
- We find a simple method to reconstruct (2.14) the original signal
- The sampled signal has a periodic spectrum...
- ...and the period is $2 \times \pi F_s$

2.2.2 Proof part 1 - Spectral considerations

By sampling $x(t)$ every $T_s$ second we obtain $x_s(n)$. The inverse fourier transform of this time discrete signal\textsuperscript{14} is

$$x_s(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_s(e^{i\omega}) e^{i\omega n} d\omega \quad (2.1)$$

For convenience we express the equation in terms of the real angular frequency $\Omega$ using $\omega = \Omega T_s$. We then obtain

$$x_s(n) = \frac{T_s}{2\pi} \int_{-\pi T_s}^{\pi T_s} X_s(e^{i\Omega}) e^{i\Omega T_s n} d\Omega \quad (2.2)$$

The inverse fourier transform of a continuous signal is

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\Omega) e^{i\Omega t} d\Omega \quad (2.3)$$

From this equation we find an expression for $x(nT_s)$

$$x(nT_s) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(i\Omega) e^{i\Omega nT_s} d\Omega \quad (2.4)$$

\textsuperscript{10}“Aliasing Applet” <http://cnx.org/content/m11448/latest/>
\textsuperscript{11}“Hold operation” <http://cnx.org/content/m11458/latest/>
\textsuperscript{12}“Exercises” <http://cnx.org/content/m11442/latest/>
\textsuperscript{13}This content is available online at <http://cnx.org/content/m11423/1.27/>.
\textsuperscript{14}“Discrete time signals” <http://cnx.org/content/m11476/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
To account for the difference in region of integration we split the integration in (2.4) into subintervals of length $\frac{2\pi}{T_s}$ and then take the sum over the resulting integrals to obtain the complete area.

$$x(nT_s) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \int_{\frac{2k\pi}{T_s}}^{\frac{2(k+1)\pi}{T_s}} X(i\Omega) e^{i\Omega nT_s} d\Omega$$

(2.5)

Then we change the integration variable, setting $\Omega = \eta + \frac{2\pi k}{T_s}$

$$x(nT_s) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \int_{\frac{2k\pi}{T_s}}^{\frac{2(k+1)\pi}{T_s}} X(i\left(\eta + \frac{2\pi k}{T_s}\right)) e^{i(\eta + \frac{2\pi k}{T_s})nT_s} d\eta$$

(2.6)

We obtain the final form by observing that $e^{i2\pi kn} = 1$, reinserting $\eta = \Omega$ and multiplying by $\frac{T_s}{2\pi}$

$$x(nT_s) = \frac{T_s}{2\pi} \sum_{k=-\infty}^{\infty} \int_{\frac{2k\pi}{T_s}}^{\frac{2(k+1)\pi}{T_s}} X(i\Omega + \frac{2\pi k}{T_s}) e^{i\Omega nT_s} d\Omega$$

(2.7)

To make $x_s(n) = x(nT_s)$ for all values of $n$, the integrands in (2.2) and (2.7) have to agree, that is

$$X_s(e^{i\Omega T_s}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X\left(i\left(\Omega + \frac{2\pi k}{T_s}\right)\right)$$

(2.8)

This is a central result. We see that the digital spectrum consists of a sum of shifted versions of the original, analog spectrum. Observe the periodicity!

We can also express this relation in terms of the digital angular frequency $\omega = \Omega T_s$

$$X_s(e^{i\omega}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X\left(i\left(\omega + \frac{2\pi k}{T_s}\right)\right)$$

(2.9)

This concludes the first part of the proof. Now we want to find a reconstruction formula, so that we can recover $x(t)$ from $x_s(n)$.

### 2.2.3 Proof part II - Signal reconstruction

For a bandlimited (Figure 2.3) signal the inverse fourier transform is

$$x(t) = \frac{1}{2\pi} \int_{\frac{-\pi}{T_s}}^{\frac{\pi}{T_s}} X(i\Omega) e^{i\Omega t} d\Omega$$

(2.10)

In the interval we are integrating we have: $X_s(e^{i\Omega T_s}) = \frac{X(i\Omega)}{T_s}$. Substituting this relation into (2.10) we get

$$x(t) = \frac{T_s}{2\pi} \int_{\frac{-\pi}{T_s}}^{\frac{\pi}{T_s}} X_s(e^{i\Omega T_s}) e^{i\Omega t} d\Omega$$

(2.11)

Using the DTFT relation for $X_s(e^{i\Omega T_s})$ we have

$$x(t) = \frac{T_s}{2\pi} \int_{\frac{-\pi}{T_s}}^{\frac{\pi}{T_s}} \sum_{n=-\infty}^{\infty} x_s(n) e^{-(i\Omega n T_s)} e^{i\Omega t} d\Omega$$

(2.12)

---

15"Table of Formulas" <http://cnx.org/content/m11450/latest/>
Interchanging integration and summation (under the assumption of convergence) leads to

\[ x(t) = \frac{T_s}{2\pi} \sum_{n=-\infty}^{\infty} x_s(n) \int_{\frac{-\pi}{T_s}}^{\frac{\pi}{T_s}} e^{i\Omega(t-nT_s)} d\Omega \]  \hspace{1cm} (2.13)

Finally we perform the integration and arrive at the important reconstruction formula

\[ x(t) = \sum_{n=-\infty}^{\infty} x_s(n) \frac{\sin \left( \frac{\pi}{T_s} (t - nT_s) \right)}{\frac{\pi}{T_s} (t - nT_s)} \]  \hspace{1cm} (2.14)

(Thanks to R. Loos for pointing out an error in the proof.)
2.2.4 Summary

\[ X_s(e^{i\Omega T_s}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X\left(i\left(\Omega + \frac{2\pi k}{T_s}\right)\right) \]

\[ x(t) = \sum_{n=-\infty}^{\infty} x_s(n) \sin\left(\frac{\pi}{T_s}(t-nT_s)\right) \]

2.2.5

Go to Introduction (Section 2.1); Illustrations (Section 2.3); Matlab Example (Section 2.4); Hold operation\(^{16}\); Aliasing applet\(^{17}\); System view (Section 2.5); Exercises\(^{18}\) ?

2.3 Illustrations\(^{19}\)

In this module we illustrate the processes involved in sampling and reconstruction. To see how all these processes work together as a whole, take a look at the system view (Section 2.5). In Sampling and reconstruction with Matlab (Section 2.4) we provide a Matlab script for download. The matlab script shows the process of sampling and reconstruction live.

2.3.1 Basic examples

**Example 2.1**
To sample an analog signal with 3000 Hz as the highest frequency component requires sampling at 6000 Hz or above.

**Example 2.2**
The sampling theorem can also be applied in two dimensions, i.e. for image analysis. A 2D sampling theorem has a simple physical interpretation in image analysis: Choose the sampling interval such that it is less than or equal to half of the smallest interesting detail in the image.

2.3.2 The process of sampling

We start off with an analog signal. This can for example be the sound coming from your stereo at home or your friend talking.

The signal is then sampled uniformly. Uniform sampling implies that we sample every \( T_s \) seconds. In Figure 2.2 we see an analog signal. The analog signal has been sampled at times \( t = nT_s \).

\(^{16}\)“Hold operation” <http://cnx.org/content/m11458/latest/>  
\(^{17}\)“Aliasing Applet” <http://cnx.org/content/m11448/latest/>  
\(^{18}\)“Exercises” <http://cnx.org/content/m11442/latest/>  
\(^{19}\)This content is available online at <http://cnx.org/content/m11443/1.33/>.
In signal processing it is often more convenient and easier to work in the frequency domain. So let’s look at the signal in frequency domain, Figure 2.3. For illustration purposes we take the frequency content of the signal as a triangle. (If you Fourier transform the signal in Figure 2.2 you will not get such a nice triangle.)

Notice that the signal in Figure 2.3 is bandlimited. We can see that the signal is bandlimited because
CHAPTER 2. SAMPLING AND FREQUENCY ANALYSIS

$X(i\Omega)$ is zero outside the interval $[-\Omega_g, \Omega_g]$. Equivalently we can state that the signal has no angular frequencies above $\Omega_g$, corresponding to no frequencies above $F_s = \frac{\Omega_g}{2\pi}$.

Now let’s take a look at the sampled signal in the frequency domain. While proving (Section 2.2) the sampling theorem we found the the spectrum of the sampled signal consists of a sum of shifted versions of the analog spectrum. Mathematically this is described by the following equation:

$$X_s(e^{i\Omega T_s}) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} X\left(i\left(\Omega + \frac{2\pi k}{T_s}\right)\right) \tag{2.15}$$

2.3.2.1 Sampling fast enough

In Figure 2.4 we show the result of sampling $x(t)$ according to the sampling theorem (Section 2.1.4: The Sampling Theorem). This means that when sampling the signal in Figure 2.2/Figure 2.3 we use $F_s \geq 2F_g$. Observe in Figure 2.4 that we have the same spectrum as in Figure 2.3 for $\Omega \in [-\Omega_g, \Omega_g]$, except for the scaling factor $\frac{1}{T_s}$. This is a consequence of the sampling frequency. As mentioned in the proof (Key points in the proof, p. 51) the spectrum of the sampled signal is periodic with period $2\pi F_s = \frac{2\pi}{T_s}$.

![Figure 2.4: The spectrum $X_s$. Sampling frequency is OK.](http://cnx.org/content/m11448/latest/)

So now we are, according to the sample theorem (Section 2.1.4: The Sampling Theorem), able to reconstruct the original signal exactly. How we can do this will be explored further down under reconstruction (Section 2.3.3: Reconstruction). But first we will take a look at what happens when we sample too slowly.

2.3.2.2 Sampling too slowly

If we sample $x(t)$ too slowly, that is $F_s < 2F_g$, we will get overlap between the repeated spectra, see Figure 2.5. According to (2.15) the resulting spectra is the sum of these. This overlap gives rise to the concept of aliasing.

**NOTE:** If the sampling frequency is less than twice the highest frequency component, then frequencies in the original signal that are above half the sampling rate will be "aliased" and will appear in the resulting signal as lower frequencies.

The consequence of aliasing is that we cannot recover the original signal, so aliasing has to be avoided. Sampling too slowly will produce a sequence $x_s(n)$ that could have originated from a number of signals. So there is no chance of recovering the original signal. To learn more about aliasing, take a look at this module\textsuperscript{20}.

\textsuperscript{20}Available for free at Connexions <http://cnx.org/content/col10360/1.4>
To avoid aliasing we have to sample fast enough. But if we can’t sample fast enough (possibly due to costs) we can include an Anti-Aliasing filter. This will not allow us to get an exact reconstruction but can still be a good solution.

**Note:** Typically a low-pass filter that is applied before sampling to ensure that no components with frequencies greater than half the sample frequency remain.

**Example 2.3**

The stagecoach effect

In older western movies you can observe aliasing on a stagecoach when it starts to roll. At first the spokes appear to turn forward, but as the stagecoach increase its speed the spokes appear to turn backward. This comes from the fact that the sampling rate, here the number of frames per second, is too low. We can view each frame as a sample of an image that is changing continuously in time. (Applet illustrating the stagecoach effect\textsuperscript{21})

\subsection{2.3.3 Reconstruction}

Given the signal in Figure 2.4 we want to recover the original signal, but the question is how? When there is no overlapping in the spectrum, the spectral component given by $k = 0$ (see (2.15)), is equal to the spectrum of the analog signal. This offers an opportunity to use a simple reconstruction process. Remember what you have learned about filtering. What we want is to change signal in Figure 2.4 into that of Figure 2.3. To achieve this we have to remove all the extra components generated in the sampling process. To remove the extra components we apply an ideal analog low-pass filter as shown in Figure 2.6. As we see the ideal filter is rectangular in the frequency domain. A rectangle in the frequency domain corresponds to a sinc\textsuperscript{22} function in time domain (and vice versa).

\textsuperscript{21}http://flowers.ofthenight.org/wagonWheel/wagonWheel.html

\textsuperscript{22}http://ccrma-www.stanford.edu/~jos/Interpolation/sinc_function.html
CHAPTER 2. SAMPLING AND FREQUENCY ANALYSIS

Figure 2.6: $H(i\Omega)$ The ideal reconstruction filter.

Then we have reconstructed the original spectrum, and as we know \textbf{if two signals are identical in the frequency domain, they are also identical in the time domain}. End of reconstruction.

2.3.4 Conclusions

The Shannon sampling theorem requires that the input signal prior to sampling is band-limited to at most half the sampling frequency. Under this condition the samples give an exact signal representation. It is truly remarkable that such a broad and useful class signals can be represented that easily!

We also looked into the problem of reconstructing the signals form its samples. Again the simplicity of the principle is striking: linear filtering by an ideal low-pass filter will do the job. However, the ideal filter is impossible to create, but that is another story...

2.3.5

Go to? Introduction (Section 2.1); Proof (Section 2.2); Illustrations (Section 2.3); Matlab Example (Section 2.4); Aliasing applet\textsuperscript{23}; Hold operation\textsuperscript{24}; System view (Section 2.5); Exercises\textsuperscript{25}

2.4 Sampling and Reconstruction with Matlab\textsuperscript{26}

2.4.1 Matlab files

Samprecon.m\textsuperscript{27}

2.4.2

Introduction (Section 2.1); Proof (Section 2.2); Illustrations (Section 2.3); Aliasing applet\textsuperscript{28}; Hold operation (Section 2.4); System view (Section 2.5); Exercises\textsuperscript{29}

\textsuperscript{23}"Aliasing Applet" <http://cnx.org/content/m11448/latest/>  
\textsuperscript{24}"Hold operation" <http://cnx.org/content/m11458/latest/>  
\textsuperscript{25}"Exercises" <http://cnx.org/content/m11442/latest/>  
\textsuperscript{26}This content is available online at <http://cnx.org/content/m11549/1.9/>.  
\textsuperscript{27}http://cnx.rice.edu/content/m11549/latest/Samprecon.m  
\textsuperscript{28}"Aliasing Applet" <http://cnx.org/content/m11448/latest/>  
\textsuperscript{29}"Exercises" <http://cnx.org/content/m11442/latest/>
2.5 Systems View of Sampling and Reconstruction

2.5.1 Ideal reconstruction system

Figure 2.7 shows the ideal reconstruction system based on the results of the Sampling theorem proof (Section 2.2).

Figure 2.7 consists of a sampling device which produces a time-discrete sequence \( x_s(n) \). The reconstruction filter, \( h(t) \), is an ideal analog sinc filter, with \( h(t) = \text{sinc} \left( \frac{t}{T_s} \right) \). We can’t apply the time-discrete sequence \( x_s(n) \) directly to the analog filter \( h(t) \). To solve this problem we turn the sequence into an analog signal using delta functions. Thus we write \( x_s(t) = \sum_{n=-\infty}^{\infty} x_s(n) \delta(t-nT) \).

But when will the system produce an output \( \hat{x}(t) = x(t) \)? According to the sampling theorem (Section 2.1.4: The Sampling Theorem) we have \( \hat{x}(t) = x(t) \) when the sampling frequency, \( F_s \), is at least twice the highest frequency component of \( x(t) \).

2.5.2 Ideal system including anti-aliasing

To be sure that the reconstructed signal is free of aliasing it is customary to apply a lowpass filter, an anti-aliasing filter (p. 57), before sampling as shown in Figure 2.8.

Again we ask the question of when the system will produce an output \( \hat{x}(t) = s(t) \)? If the signal is entirely confined within the passband of the lowpass filter we will get perfect reconstruction if \( F_s \) is high enough.

But if the anti-aliasing filter removes the "higher" frequencies, (which in fact is the job of the anti-aliasing filter), we will never be able to exactly reconstruct the original signal, \( s(t) \). If we sample fast enough we can reconstruct \( x(t) \), which in most cases is satisfying.

The reconstructed signal, \( \hat{x}(t) \), will not have aliased frequencies. This is essential for further use of the signal.

---

30 This content is available online at <http://cnx.org/content/m11465/1.20/>.
32 "Table of Formulas" <http://cnx.org/content/m11450/latest/>
2.5.3 Reconstruction with hold operation

To make our reconstruction system realizable there are many things to look into. Among them are the fact that any practical reconstruction system must input finite length pulses into the reconstruction filter. This can be accomplished by the hold operation\(^{33}\). To alleviate the distortion caused by the hold operator we apply the output from the hold device to a compensator. The compensation can be as accurate as we wish, this is cost and application consideration.

By the use of the hold component the reconstruction will not be exact, but as mentioned above we can get as close as we want.

2.5.4

Introduction (Section 2.1); Proof (Section 2.2); Illustrations (Section 2.3); Matlab example (Section 2.4); Hold operation\(^{35}\); Aliasing applet\(^{36}\); Exercises\(^{37}\)

2.6 Sampling CT Signals: A Frequency Domain Perspective\(^{38}\)

2.6.1 Understanding Sampling in the Frequency Domain

We want to relate \(x_c(t)\) directly to \(x[n]\). Compute the CTFT of

\[
x_s(t) = \sum_{n=-\infty}^{\infty} x_c(nT) \delta(t - nT)
\]

\[
X_s(\Omega) = \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x_c(nT) \delta(t - nT) e^{-i\Omega t} \, dt
\]

\[
= \sum_{n=-\infty}^{\infty} x_c(nT) \int_{-\infty}^{\infty} \delta(t - nT) e^{-i\Omega t} \, dt
\]

\[
= \sum_{n=-\infty}^{\infty} x[n] e^{-i\Omega nT}
\]

\[
= \sum_{n=-\infty}^{\infty} x[n] e^{-i\omega n}
\]

\[
= X(\omega)
\]

where \(\omega \equiv \Omega T\) and \(X(\omega)\) is the DTFT of \(x[n]\).

**NOTE:**

\[
X_s(\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s)
\]

---

\(^{33}\)"Hold operation" <http://cnx.org/content/m11458/latest/>

\(^{34}\)"Hold operation" <http://cnx.org/content/m11458/latest/>

\(^{35}\)"Hold operation" <http://cnx.org/content/m11458/latest/>

\(^{36}\)"Aliasing Applet" <http://cnx.org/content/m11448/latest/>

\(^{37}\)"Exercises" <http://cnx.org/content/m11442/latest/>

\(^{38}\)This content is available online at <http://cnx.org/content/ml0994/2.2/>.
\[ X(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(\frac{\omega-2\pi k}{T}\right) \]  

where this last part is 2π-periodic.

### 2.6.1.1 Sampling

![Diagram showing CTFT and DTFT](image)

**Figure 2.10**

**Example 2.4: Speech**

Speech is intelligible if bandlimited by a CT lowpass filter to the band ±4 kHz. We can sample speech as slowly as _____?
2.6.2 Relating $x[n]$ to sampled $x(t)$

Recall the following equality:

$$x_s(t) = \sum_{n} x(nT) \delta(t - nT)$$

Figure 2.11

Note that there is no mention of $T$ or $\Omega_s$!

Figure 2.12
Recall the CTFT relation:

\[ x(at) \leftrightarrow \frac{1}{\alpha}X\left(\frac{\Omega}{\alpha}\right) \] (2.18)

where \( \alpha \) is a scaling of time and \( \frac{1}{\alpha} \) is a scaling in frequency.

\[ X_s(\Omega) \equiv X(\Omega T) \] (2.19)

2.7 The DFT: Frequency Domain with a Computer Analysis\(^{39}\)

2.7.1 Introduction

We just covered ideal (and non-ideal) (time) sampling of CT signals (Section 2.6). This enabled DT signal processing solutions for CT applications (Figure 2.14):

\(^{39}\)This content is available online at <http://cnx.org/content/m10992/2.3/>.
Much of the theoretical analysis of such systems relied on frequency domain representations. How do we carry out these frequency domain analysis on the computer? Recall the following relationships:

\[ x[n] \xrightarrow{\text{DTFT}} X(\omega) \]
\[ x(t) \xrightarrow{\text{CTFT}} X(\Omega) \]

where \( \omega \) and \( \Omega \) are continuous frequency variables.

### 2.7.1.1 Sampling DTFT

Consider the DTFT of a discrete-time (DT) signal \( x[n] \). Assume \( x[n] \) is of finite duration \( N \) (i.e., an \( N \)-point signal).

\[ X(\omega) = \sum_{n=0}^{N-1} x[n] e^{-i\omega n} \quad (2.20) \]

where \( X(\omega) \) is the continuous function that is indexed by the real-valued parameter \(-\pi \leq \omega \leq \pi\). The other function, \( x[n] \), is a discrete function that is indexed by integers.

We want to work with \( X(\omega) \) on a computer. Why not just sample \( X(\omega) \)?

\[ X[k] = X\left(\frac{2\pi}{N}k\right) = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}kn} \quad (2.21) \]

In (2.21) we sampled at \( \omega = \frac{2\pi}{N}k \) where \( k = \{0, 1, \ldots, N-1\} \) and \( X[k] \) for \( k = \{0, \ldots, N-1\} \) is called the **Discrete Fourier Transform (DFT)** of \( x[n] \).

**Example 2.5**

**Finite Duration DT Signal**

*Image not finished*

![Image not finished](http://cnx.org/content/col10360/1.4)

The DTFT of the image in Figure 2.15 (Finite Duration DT Signal) is written as follows:

\[ X(\omega) = \sum_{n=0}^{N-1} x[n] e^{-i\omega n} \quad (2.22) \]
where $\omega$ is any $2\pi$-interval, for example $-\pi \leq \omega \leq \pi$.

**Sample $X(\omega)$**

**Image not finished**

Figure 2.16

where again we sampled at $\omega = \frac{2\pi}{N}k$ where $k = \{0, 1, \ldots, M - 1\}$. For example, we take $M = 10$.

In the following section (Section 2.7.1.1.1: Choosing $M$) we will discuss in more detail how we should choose $M$, the number of samples in the $2\pi$ interval.

(This is precisely how we would plot $X(\omega)$ in Matlab.)

2.7.1.1 Choosing $M$

2.7.1.1.1 Case 1

Given $N$ (length of $x[n]$), choose $M \gg N$ to obtain a dense sampling of the DTFT (Figure 2.17):

![Figure 2.17](http://cnx.org/content/col10360/1.4)

2.7.1.1.2 Case 2

Choose $M$ as small as possible (to minimize the amount of computation).

In general, we require $M \geq N$ in order to represent all information in

$$\forall n, n = \{0, \ldots, N - 1\} : (x[n])$$

Let’s concentrate on $M = N$:

$$x[n] \xrightarrow{\text{DFT}} X[k]$$

for $n = \{0, \ldots, N - 1\}$ and $k = \{0, \ldots, N - 1\}$

numbers $\leftrightarrow$ $N$ numbers

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
2.7.2 Discrete Fourier Transform (DFT)

Define

\[ X[k] \equiv X\left(\frac{2\pi k}{N}\right) \]  

where \( N = \text{length} (x[n]) \) and \( k = \{0, \ldots, N-1\} \). In this case, \( M = N \).

**DFT**

\[ X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi \frac{k}{N} n} \]  

**Inverse DFT (IDFT)**

\[ x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi \frac{k}{N} n} \]  

2.7.2.1 Interpretation

Represent \( x[n] \) in terms of a sum of \( N \) complex sinusoids\(^40\) of amplitudes \( X[k] \) and frequencies

\[ \forall k, k \in \{0, \ldots, N-1\} : \left(\omega_k = \frac{2\pi k}{N}\right) \]

**Note:** Fourier Series with fundamental frequency \( \frac{2\pi}{N} \)

2.7.2.1.1 Remark 1

IDFT treats \( x[n] \) as though it were \( N \)-periodic.

\[ x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi \frac{k}{N} n} \]  

where \( n \in \{0, \ldots, N-1\} \)

**Exercise 2.7.1**

What about other values of \( n \)?  

(Solution on p. 107.)

2.7.2.1.2 Remark 2

Proof that the IDFT inverts the DFT for \( n \in \{0, \ldots, N-1\} \)

\[ \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi \frac{k}{N} n} = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{m=0}^{N-1} x[m] e^{(-i)2\pi \frac{k}{N} m} e^{i2\pi \frac{k}{N} n} = ???
\]

**Example 2.6: Computing DFT**

Given the following discrete-time signal (Figure 2.18) with \( N = 4 \), we will compute the DFT using two different methods (the DFT Formula and Sample DTFT):

\(^{40}\)Continuous Time Complex Exponential" <http://cnx.org/content/m10060/latest/>
1. DFT Formula

\[
X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi \frac{k}{N} n} = 1 + e^{(-i)2\pi \frac{k}{4}} + e^{(-i)2\pi \frac{k}{2}} + e^{(-i)2\pi \frac{k}{4}} \quad (2.28)
\]

Using the above equation, we can solve and get the following results:

\[x[0] = 4\]
\[x[1] = 0\]
\[x[2] = 0\]
\[x[3] = 0\]

2. Sample DTFT. Using the same figure, Figure 2.18, we will take the DTFT of the signal and get the following equations:

\[
X(\omega) = \sum_{n=0}^{3} e^{(-i)\omega n} = \frac{1 - e^{-i3\omega}}{1 - e^{-i\omega}} \quad (2.29)
\]

Our sample points will be:

\[\omega_k = \frac{2\pi k}{4} = \frac{\pi}{2} k\]

where \(k = \{0, 1, 2, 3\}\) (Figure 2.19).
2.7.3 Periodicity of the DFT

DFT $X[k]$ consists of samples of DTFT, so $X(\omega)$, a $2\pi$-periodic DTFT signal, can be converted to $X[k]$, an $N$-periodic DFT.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi \frac{k}{N} n}$$

(2.30)

where $e^{-i2\pi \frac{k}{N} n}$ is an $N$-periodic basis function (See Figure 2.20).

Also, recall,

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{i2\pi \frac{k}{N} n}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{i2\pi \frac{k}{N} (n+mN)}$$

(2.31)
Example 2.7: Illustration

Figure 2.21

**NOTE:** When we deal with the DFT, we need to remember that, in effect, this treats the signal as an $N$-periodic sequence.

### 2.7.4 A Sampling Perspective

Think of sampling the continuous function $X(\omega)$, as depicted in Figure 2.22. $S(\omega)$ will represent the sampling function applied to $X(\omega)$ and is illustrated in Figure 2.22 as well. This will result in our discrete-time sequence, $X[k]$. 

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
CHAPTER 2. SAMPLING AND FREQUENCY ANALYSIS

Figure 2.22

NOTE: Remember the multiplication in the frequency domain is equal to convolution in the time domain!

2.7.4.1 Inverse DTFT of $S(\omega)$

\[
\sum_{k=-\infty}^{\infty} \delta \left( \omega - \frac{2\pi k}{N} \right) \quad (2.32)
\]

Given the above equation, we can take the DTFT and get the following equation:

\[
N \sum_{m=-\infty}^{\infty} \delta \left[ n - mN \right] \equiv S[n] \quad (2.33)
\]

**Exercise 2.7.2**

Why does (2.33) equal $S[n]$?  

*(Solution on p. 107.)*
So, in the time-domain we have (Figure 2.23):
2.7.5 Connections

Combine signals in Figure 2.24 to get signals in Figure 2.25.
2.8 Discrete-Time Processing of CT Signals

2.8.1 DT Processing of CT Signals

2.8.1.1 Analysis

\[ Y_c(\Omega) = H_{\text{LP}}(\Omega) Y(\Omega T) \]  \hspace{1cm} (2.34)

where we know that \( Y(\omega) = X(\omega) G(\omega) \) and \( G(\omega) \) is the frequency response of the DT LTI system. Also, remember that

\[ \omega \equiv \Omega T \]

So,

\[ Y_c(\Omega) = H_{\text{LP}}(\Omega) G(\Omega T) X(\Omega T) \]  \hspace{1cm} (2.35)

where \( Y_c(\Omega) \) and \( H_{\text{LP}}(\Omega) \) are CTFTs and \( G(\Omega T) \) and \( X(\Omega T) \) are DTFTs.

**NOTE:**

\[ X(\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c(\omega - 2\pi k) \]

OR

\[ X(\Omega T) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s) \]

Therefore our final output signal, \( Y_c(\Omega) \), will be:

\[ Y_c(\Omega) = H_{\text{LP}}(\Omega) G(\Omega T) \left( \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} X_c(\Omega - k\Omega_s) \right) \]  \hspace{1cm} (2.36)

---

This content is available online at [http://cnx.org/content/m10993/2.2/].

Available for free at Connexions [http://cnx.org/content/col10360/1.4].
Now, if $X_c(\Omega)$ is bandlimited to $[-\frac{\Omega_s}{2}, \frac{\Omega_s}{2}]$ and we use the usual lowpass reconstruction filter in the D/A, Figure 2.27:

\[ Y_c(\Omega) = \begin{cases} 
G(\Omega T) X_c(\Omega) & \text{if } |\Omega| < \frac{\Omega_s}{2} \\
0 & \text{otherwise}
\end{cases} \] (2.37)
2.8.1.2 Summary

For bandlimited signals sampled at or above the Nyquist rate, we can relate the input and output of the DSP system by:

\[ Y_c(\Omega) = G_{\text{eff}}(\Omega) X_c(\Omega) \]  

(2.38)

where

\[ G_{\text{eff}}(\Omega) = \begin{cases} 
G(\Omega T) & \text{if } |\Omega| < \frac{\Omega_s}{2} \\
0 & \text{otherwise}
\end{cases} \]

Figure 2.28

2.8.1.2.1 Note

\( G_{\text{eff}}(\Omega) \) is LTI if and only if the following two conditions are satisfied:

1. \( G(\omega) \) is LTI (in DT).
2. \( X_c(T) \) is bandlimited and sampling rate equal to or greater than Nyquist. For example, if we had a simple pulse described by

\[ X_c(t) = u(t - T_0) - u(t - T_1) \]

where \( T_1 > T_0 \). If the sampling period \( T > T_1 - T_0 \), then some samples might "miss" the pulse while others might not be "missed." This is what we term **time-varying behavior**.

Example 2.8

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
If $\frac{2\pi}{T} > 2B$ and $\omega_1 < BT$, determine and sketch $Y_c(\Omega)$ using Figure 2.29.

2.8.2 Application: 60Hz Noise Removal

Unfortunately, in real-world situations electrodes also pick up ambient 60 Hz signals from lights, computers, etc. In fact, usually this "60 Hz noise" is much greater in amplitude than the EKG signal shown in Figure 2.30. Figure 2.31 shows the EKG signal; it is barely noticeable as it has become overwhelmed by noise.
Figure 2.31: Our EKG signal, $y(t)$, is overwhelmed by noise.

2.8.2.1 DSP Solution

2.8.2.2 Sampling Period/Rate

First we must note that $|Y(\Omega)|$ is **bandlimited** to ±60 Hz. Therefore, the minimum rate should be 120 Hz. In order to get the best results we should set

$$f_s = 240\text{Hz}$$
\[ \Omega_s = 2\pi \left( \frac{240 \text{ rad}}{s} \right) \]

**Figure 2.34**

#### 2.8.2.3 Digital Filter

Therefore, we want to design a digital filter that will remove the 60Hz component and preserve the rest.

**Figure 2.35**

#### 2.9 Short Time Fourier Transform

**2.9.1 Short Time Fourier Transform**

The Fourier transforms (FT, DTFT, DFT, etc.) do not clearly indicate how the frequency content of a signal changes over time.

That information is hidden in the phase - it is not revealed by the plot of the magnitude of the spectrum.

**NOTE:** To see how the frequency content of a signal changes over time, we can cut the signal into blocks and compute the spectrum of each block.

To improve the result,

1. blocks are overlapping

---

42This content is available online at <http://cnx.org/content/m10570/2.4/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
2. each block is multiplied by a window that is tapered at its endpoints.

Several parameters must be chosen:

- Block length, \( R \).
- The type of window.
- Amount of overlap between blocks. (Figure 2.36 (STFT: Overlap Parameter))
- Amount of zero padding, if any.
The short-time Fourier transform is defined as

\[ X(\omega, m) = \text{STFT}(x(n)) := \text{DTFT}(x(n - m)w(n)) \]

\[ = \sum_{n=-\infty}^{\infty} x(n - m)w(n) e^{-i\omega n} \]

\[ = \sum_{n=0}^{R-1} x(n - m)w(n) e^{-i\omega n} \]  

where \( w(n) \) is the window function of length \( R \).

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
1. The STFT of a signal \( x(n) \) is a function of two variables: time and frequency.
2. The block length is determined by the support of the window function \( w(n) \).
3. A graphical display of the magnitude of the STFT, \( |X(\omega, m)| \), is called the spectrogram of the signal. It is often used in speech processing.
4. The STFT of a signal is invertible.
5. One can choose the block length. A long block length will provide higher frequency resolution (because the main-lobe of the window function will be narrow). A short block length will provide higher time resolution because less averaging across samples is performed for each STFT value.
6. A narrow-band spectrogram is one computed using a relatively long block length \( R \), (long window function).
7. A wide-band spectrogram is one computed using a relatively short block length \( R \), (short window function).

### 2.9.1.1 Sampled STFT

To numerically evaluate the STFT, we sample the frequency axis \( \omega \) in \( N \) equally spaced samples from \( \omega = 0 \) to \( \omega = 2\pi \).

\[
\forall k, 0 \leq k \leq N - 1: \left( \omega_k = \frac{2\pi}{N} k \right) \tag{2.40}
\]

We then have the discrete STFT,

\[
X^d(k, m) := X\left(\frac{2\pi}{N} k, m\right) = \sum_{n=0}^{R-1} x(n - m) w(n) e^{-i\omega_k n} = \sum_{n=0}^{R-1} x(n - m) w(n) W_N^{-(kn)} = DFT_N \left( x(n - m) w(n) |_{n=0}^{R-1} \right) \tag{2.41}
\]

where \( 0, \ldots, 0 \) is \( N - R \).

In this definition, the overlap between adjacent blocks is \( R - 1 \). The signal is shifted along the window one sample at a time. That generates more points than is usually needed, so we also sample the STFT along the time direction. That means we usually evaluate

\[
X^d(k, Lm)
\]

where \( L \) is the time-skip. The relation between the time-skip, the number of overlapping samples, and the block length is

\[
\text{Overlap} = R - L
\]

**Exercise 2.9.1**

Match each signal to its spectrogram in Figure 2.37.

*(Solution on p. 107.)*

Available for free at Connexions [http://cnx.org/content/col10360/1.4]
Figure 2.37
2.9.1.2 Spectrogram Example

Figure 2.38
The matlab program for producing the figures above (Figure 2.38 and Figure 2.39).

```matlab
% LOAD DATA
load mtlb;
x = mtlb;

figure(1), clf
plot(0:4000,x)
xlabel('n')
ylabel('x(n)')

% SET PARAMETERS
R = 256;       % R: block length
window = hamming(R); % window function of length R
N = 512;   % N: frequency discretization
L = 35;   % L: time lapse between blocks
fs = 7418;   % fs: sampling frequency
```

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
overlap = R - L;

% COMPUTE SPECTROGRAM
[B,f,t] = specgram(x,N,fs,window,overlap);

% MAKE PLOT
figure(2), clf
imagesc(t,f,log10(abs(B)));
colormap('jet')
axis xy
xlabel('time')
ylabel('frequency')
title('SPECTROGRAM, R = 256')
2.9.1.3 Effect of window length $R$

Narrow-band spectrogram: better frequency resolution

Figure 2.40
Wide-band spectrogram: better time resolution

Figure 2.41

Here is another example to illustrate the frequency/time resolution trade-off (See figures - Figure 2.40 (Narrow-band spectrogram: better frequency resolution), Figure 2.41 (Wide-band spectrogram: better time resolution), and Figure 2.42 (Effect of Window Length R)).
CHAPTER 2. SAMPLING AND FREQUENCY ANALYSIS

**Effect of Window Length R**

![Effect of Window Length R](image)

2.9.1.4 Effect of L and N

A spectrogram is computed with different parameters:

\[ L \in \{1, 10\} \]

\[ N \in \{32, 256\} \]

- \( L \) = time lapse between blocks.
- \( N \) = FFT length (Each block is zero-padded to length \( N \).)

In each case, the block length is 30 samples.

**Exercise 2.9.2** *(Solution on p. 107.)*

For each of the four spectrograms in Figure 2.43 can you tell what \( L \) and \( N \) are?
$L$ and $N$ do not effect the time resolution or the frequency resolution. They only affect the ‘pixelation’.

2.9.1.5 Effect of $R$ and $L$

Shown below are four spectrograms of the same signal. Each spectrogram is computed using a different set of parameters.

$$R \in \{120, 256, 1024\}$$

$$L \in \{35, 250\}$$

where

- $R =$ block length
- $L =$ time lapse between blocks.

**Exercise 2.9.3**

*(Solution on p. 107.)*

For each of the four spectrograms in Figure 2.44, match the above values of $L$ and $R$.
If you like, you may listen to this signal with the `soundsc` command; the data is in the file: `stft_data.m`. Here (Figure 2.45) is a figure of the signal.
2.10 Spectrograms

We know how to acquire analog signals for digital processing (pre-filtering, sampling, and A/D conversion) and to compute spectra of discrete-time signals (using the FFT algorithm), let’s put these various components together to learn how the spectrogram shown in Figure 2.46 (Speech Spectrogram), which is used to analyze speech, is calculated. The speech was sampled at a rate of 11.025 kHz and passed through a 16-bit A/D converter.

**Point of Interest:** Music compact discs (CDs) encode their signals at a sampling rate of 44.1 kHz. We’ll learn the rationale for this number later. The 11.025 kHz sampling rate for the speech is 1/4 of the CD sampling rate, and was the lowest available sampling rate commensurate with speech signal bandwidths available on my computer.

**Exercise 2.10.1** *(Solution on p. 107.)*

Looking at Figure 2.46 (Speech Spectrogram) the signal lasted a little over 1.2 seconds. How long was the sampled signal (in terms of samples)? What was the datarate during the sampling process in bps (bits per second)? Assuming the computer storage is organized in terms of bytes (8-bit quantities), how many bytes of computer memory does the speech consume?

---

43 This content is available online at [http://cnx.org/content/m0505/2.21/].
44 "The Sampling Theorem" [http://cnx.org/content/m0050/latest/]
45 "The Sampling Theorem" [http://cnx.org/content/m0050/latest/]
46 "Amplitude Quantization" [http://cnx.org/content/m0051/latest/]
47 "Fast Fourier Transform (FFT)" [http://cnx.org/content/m10250/latest/]
48 "Modeling the Speech Signal" [http://cnx.org/content/m0049/latest/]

Available for free at Connexions [http://cnx.org/content/col10360/1.4]
The resulting discrete-time signal, shown in the bottom of Figure 2.46 (Speech Spectrogram), clearly changes its character with time. To display these spectral changes, the long signal was sectioned into frames: comparatively short, contiguous groups of samples. Conceptually, a Fourier transform of each frame is calculated using the FFT. Each frame is not so long that significant signal variations are retained within a frame, but not so short that we lose the signal’s spectral character. Roughly speaking, the speech signal’s spectrum is evaluated over successive time segments and stacked side by side so that the $x$-axis corresponds to time and the $y$-axis frequency, with color indicating the spectral amplitude.

An important detail emerges when we examine each framed signal (Figure 2.47 (Spectrogram Hanning vs. Rectangular)).
Figure 2.47: The top waveform is a segment 1024 samples long taken from the beginning of the "Rice University" phrase. Computing Figure 2.46 (Speech Spectrogram) involved creating frames, here demarked by the vertical lines, that were 256 samples long and finding the spectrum of each. If a rectangular window is applied (corresponding to extracting a frame from the signal), oscillations appear in the spectrum (middle of bottom row). Applying a Hanning window gracefully tapers the signal toward frame edges, thereby yielding a more accurate computation of the signal’s spectrum at that moment of time.

At the frame’s edges, the signal may change very abruptly, a feature not present in the original signal. A transform of such a segment reveals a curious oscillation in the spectrum, an artifact directly related to this sharp amplitude change. A better way to frame signals for spectrograms is to apply a window: Shape the signal values within a frame so that the signal decays gracefully as it nears the edges. This shaping is accomplished by multiplying the framed signal by the sequence \( w(n) \). In sectioning the signal, we essentially applied a rectangular window: \( w(n) = 1, 0 \leq n \leq N - 1 \). A much more graceful window is the Hanning window; it has the cosine shape \( w(n) = \frac{1}{2} \left(1 - \cos \left( \frac{2\pi n}{N} \right) \right) \). As shown in Figure 2.47 (Spectrogram Hanning vs. Rectangular), this shaping greatly reduces spurious oscillations in each frame’s spectrum. Considering the spectrum of the Hanning windowed frame, we find that the oscillations resulting from applying the rectangular window obscured a formant (the one located at a little more than half the Nyquist frequency).

Exercise 2.10.2 (Solution on p. 107.)

What might be the source of these oscillations? To gain some insight, what is the length-2N discrete Fourier transform of a length-N pulse? The pulse emulates the rectangular window, and certainly has edges. Compare your answer with the length-2N transform of a length-N Hanning window.
Non-overlapping windows

Figure 2.48: In comparison with the original speech segment shown in the upper plot, the non-overlapped Hanning windowed version shown below it is very ragged. Clearly, spectral information extracted from the bottom plot could well miss important features present in the original.

If you examine the windowed signal sections in sequence to examine windowing’s effect on signal amplitude, we see that we have managed to amplitude-modulate the signal with the periodically repeated window (Figure 2.48 (Non-overlapping windows)). To alleviate this problem, frames are overlapped (typically by half a frame duration). This solution requires more Fourier transform calculations than needed by rectangular windowing, but the spectra are much better behaved and spectral changes are much better captured.

The speech signal, such as shown in the speech spectrogram (Figure 2.46: Speech Spectrogram), is sectioned into overlapping, equal-length frames, with a Hanning window applied to each frame. The spectra of each of these is calculated, and displayed in spectrograms with frequency extending vertically, window time location running horizontally, and spectral magnitude color-coded. Figure 2.49 (Overlapping windows for computing spectrograms) illustrates these computations.
Overlap windows for computing spectrograms

Figure 2.49: The original speech segment and the sequence of overlapping Hanning windows applied to it are shown in the upper portion. Frames were 256 samples long and a Hanning window was applied with a half-frame overlap. A length-512 FFT of each frame was computed, with the magnitude of the first 257 FFT values displayed vertically, with spectral amplitude values color-coded.

Exercise 2.10.3 (Solution on p. 107.)
Why the specific values of 256 for N and 512 for K? Another issue is how was the length-512 transform of each length-256 windowed frame computed?
2.11 Filtering with the DFT$^{49}$

2.11.1 Introduction

\[ y[n] = x[n] \ast h[n] \]
\[ = \sum_{k=-\infty}^{\infty} x[k] h[n-k] \]  \hspace{1cm} (2.42)

\[ Y(\omega) = X(\omega) H(\omega) \]  \hspace{1cm} (2.43)

Assume that \( H(\omega) \) is specified.

**Exercise 2.11.1**  \hspace{1cm} (*Solution on p. 107.*)

How can we implement \( X(\omega) H(\omega) \) in a computer?

Recall that the DFT treats \( N \)-point sequences as if they are periodically extended (Figure 2.51):

$^{49}$This content is available online at <http://cnx.org/content/m11022/2.3/>. 

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
2.11.2 Compute IDFT of $Y[k]$

$$\tilde{y}[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y[k] e^{i2\pi \frac{k}{N} n}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X[k] H[k] e^{i2\pi \frac{k}{N} n}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{m=0}^{N-1} x[m] e^{-i2\pi \frac{k}{N} (n-m)} H[k] e^{i2\pi \frac{k}{N} m}$$

$$= \sum_{m=0}^{N-1} x[m] \left( \frac{1}{N} \sum_{k=0}^{N-1} H[k] e^{i2\pi \frac{k}{N} (n-m)} \right)$$

$$= \sum_{m=0}^{N-1} x[m] h[((n-m))_N]$$

(2.44)

And the IDFT periodically extends $h[n]$:

$$\tilde{h}[n-m] = h[((n-m))_N]$$

This computes as shown in Figure 2.52:

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
is called circular convolution and is denoted by Figure 2.53.

\[ \tilde{y}[n] = \sum_{m=0}^{N-1} x[m] h[((n-m)\mod N)] \]  

(2.45)

Figure 2.53: The above symbol for the circular convolution is for an \( N \)-periodic extension.

2.11.2.1 DFT Pair

\[ x[n] \odot h[n] \overset{\text{DFT}}{\longrightarrow} (X[k]) (H[k]) \]

Figure 2.54

Note that in general:
Example 2.9: Regular vs. Circular Convolution

To begin with, we are given the following two length-3 signals:

\[ x[n] = \{1, 2, 3\} \]

\[ h[n] = \{1, 0, 2\} \]

We can zero-pad these signals so that we have the following discrete sequences:

\[ x[n] = \{\ldots, 0, 1, 2, 3, 0, \ldots\} \]

\[ h[n] = \{\ldots, 0, 1, 0, 2, 0, \ldots\} \]

where \( x[0] = 1 \) and \( h[0] = 1 \).

- Regular Convolution:

\[ y[n] = \sum_{m=0}^{2} x[m] h[n-m] \quad (2.46) \]

Using the above convolution formula (refer to the link if you need a review of convolution (Section 1.5)), we can calculate the resulting value for \( y[0] \) to \( y[4] \). Recall that because we have two length-3 signals, our convolved signal will be length-5.

- \( n = 0 \)

\[ \{\ldots, 0, 0, 0, 1, 2, 3, 0, \ldots\} \]

\[ \{\ldots, 0, 2, 0, 1, 0, 0, \ldots\} \]

\[ y[0] = 1 \times 1 + 2 \times 0 + 3 \times 0 = 1 \quad (2.47) \]

- \( n = 1 \)

\[ \{\ldots, 0, 0, 1, 2, 3, 0, \ldots\} \]

\[ \{\ldots, 0, 2, 0, 1, 0, 0, \ldots\} \]

\[ y[1] = 1 \times 0 + 2 \times 1 + 3 \times 0 = 2 \quad (2.48) \]
\[ y[2] = 1 \times 2 + 2 \times 0 + 3 \times 1 = 5 \]  
\[ \text{(2.49)} \]

\[ y[3] = 4 \]  
\[ \text{(2.50)} \]

\[ y[4] = 6 \]  
\[ \text{(2.51)} \]

**Regular Convolution Result**

\[ \tilde{y}[n] = \sum_{m=0}^{2} x[m] h \left[ (n - m)_{N} \right] \]  
\[ \text{(2.52)} \]

And now with circular convolution our \( h[n] \) changes and becomes a periodically extended signal:

\[ h \left[ ((n))_{N} \right] = \{ \ldots, 1, 0, 2, 1, 0, 2, 1, 0, 2, \ldots \} \]  
\[ \text{(2.53)} \]

\[ \tilde{y}[0] = 1 \times 1 + 2 \times 2 + 3 \times 0 = 5 \]  
\[ \text{(2.54)} \]
\[ n = 1 \quad \{ \ldots, 0, 0, 1, 2, 3, 0, \ldots \} \]
\[ \{ \ldots, 0, 1, 2, 0, 1, 2, 0, \ldots \} \]
\[ \tilde{y}[1] = 1 \times 1 + 2 \times 1 + 3 \times 2 = 8 \quad (2.55) \]
\[ n = 2 \quad \tilde{y}[2] = 5 \quad (2.56) \]
\[ n = 3 \quad \tilde{y}[3] = 5 \quad (2.57) \]
\[ n = 4 \quad \tilde{y}[4] = 8 \quad (2.58) \]

Circular Convolution Result

Figure 2.57: Result is 3-periodic.

Figure 2.58 (Circular Convolution from Regular) illustrates the relationship between circular convolution and regular convolution using the previous two figures:

Circular Convolution from Regular

Figure 2.58: The left plot (the circular convolution results) has a "wrap-around" effect due to periodic extension.
2.11.2.2 Regular Convolution from Periodic Convolution

1. "Zero-pad" $x[n]$ and $h[n]$ to avoid the overlap (wrap-around) effect. We will zero-pad the two signals to a length-5 signal (5 being the duration of the regular convolution result):

   $x[n] = \{1, 2, 3, 0, 0\}$

   $h[n] = \{1, 0, 2, 0, 0\}$

2. Now take the DFTs of the zero-padded signals:

   $\tilde{y}[n] = \frac{1}{N} \sum_{k=0}^{4} X[k] H[k] e^{j2\pi \frac{5}{N} n}$

   $= \sum_{m=0}^{4} x[m] h[5((n-m)_{5})] \quad (2.59)$

Now we can plot this result (Figure 2.59):

---

**Figure 2.59:** The sequence from 0 to 4 (the underlined part of the sequence) is the regular convolution result. From this illustration we can see that it is 5-periodic!

---

**NOTE:** We can compute the regular convolution result of a convolution of an $M$-point signal $x[n]$ with an $N$-point signal $h[n]$ by padding each signal with zeros to obtain two $M + N - 1$ length sequences and computing the circular convolution (or equivalently computing the IDFT of $H[k]X[k]$, the product of the DFTs of the zero-padded signals) (Figure 2.60).
2.11.3 DSP System

No Image.

Figure 2.60: Note that the lower two images are simply the top images that have been zero-padded.

1. Sample finite duration continuous-time input $x(t)$ to get $x[n]$ where $n = \{0, \ldots, M - 1\}$.
2. Zero-pad $x[n]$ and $h[n]$ to length $M + N - 1$.
3. Compute DFTs $X[k]$ and $H[k]$
4. Compute IDFTs of $X[k]H[k]$

$$y[n] = \tilde{y}[n]$$

where $n = \{0, \ldots, M + N - 1\}$.
5. Reconstruct $y(t)$
2.12 Image Restoration Basics

2.12.1 Image Restoration

In many applications (e.g., satellite imaging, medical imaging, astronomical imaging, poor-quality family portraits) the imaging system introduces a slight distortion. Often images are slightly blurred and image restoration aims at deblurring the image.

The blurring can usually be modeled as an LSI system with a given PSF $h[m,n]$.

The observed image is

$$g[m,n] = h[m,n] * f[m,n]$$  \hspace{1cm} (2.60)

$$G(u,v) = H(u,v) F(u,v)$$  \hspace{1cm} (2.61)

$$F(u,v) = \frac{G(u,v)}{H(u,v)}$$  \hspace{1cm} (2.62)

Example 2.10: Image Blurring

Above we showed the equations for representing the common model for blurring an image. In Figure 2.63 we have an original image and a PSF function that we wish to apply to the image in order to model a basic blurred image.

Once we apply the PSF to the original image, we receive our blurred image that is shown in Figure 2.64:
2.12.1.1 Frequency Domain Analysis

Example 2.10 (Image Blurring) looks at the original images in its typical form; however, it is often useful to look at our images and PSF in the frequency domain. In Figure 2.65, we take another look at the image blurring example above and look at how the images and results would appear in the frequency domain if we applied the Fourier transforms.
Figure 2.65
Solutions to Exercises in Chapter 2

Solution to Exercise 2.7.1 (p. 66)

\[ x[n + N] = ??? \]

Solution to Exercise 2.7.2 (p. 70)

\( S[n] \) is \( N \)-periodic, so it has the following Fourier Series\(^{51}\):

\[
c_k = \frac{1}{N} \int_{-N/2}^{N/2} \delta[n] e^{(-i)2\pi \frac{k}{N} n} \, dn
\]

\[
= \frac{1}{N} \quad (2.63)
\]

\[
S[n] = \sum_{k=-\infty}^{\infty} e^{(-i)2\pi \frac{k}{N} n} \quad (2.64)
\]

where the DTFT of the exponential in the above equation is equal to \( \delta(\omega - \frac{2\pi k}{N}) \).

Solution to Exercise 2.9.1 (p. 81)

Solution to Exercise 2.9.2 (p. 88)

Solution to Exercise 2.9.3 (p. 89)

Solution to Exercise 2.10.1 (p. 91)

Number of samples equals \( 1.2 \times 11025 = 13230 \). The datarate is \( 11025 \times 16 = 176.4 \) kbps. The storage required would be 26460 bytes.

Solution to Exercise 2.10.2 (p. 93)

The oscillations are due to the boxcar window’s Fourier transform, which equals the sinc function.

Solution to Exercise 2.10.3 (p. 95)

These numbers are powers-of-two, and the FFT algorithm can be exploited with these lengths. To compute a longer transform than the input signal’s duration, we simply zero-pad the signal.

Solution to Exercise 2.11.1 (p. 96)

Discretize (sample) \( X(\omega) \) and \( H(\omega) \). In order to do this, we should take the DFTs of \( x[n] \) and \( h[n] \) to get \( X[k] \) and \( X[k] \). Then we will compute

\[
\tilde{y}[n] = \text{IDFT}(X[k] H[k])
\]

Does \( \tilde{y}[n] = y[n] \)?

\(^{51}\)“Fourier Series: Eigenfunction Approach” <http://cnx.org/content/m10496/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Chapter 3

Digital Filtering

3.1 Difference Equation

3.1.1 Introduction

One of the most important concepts of DSP is to be able to properly represent the input/output relationship to a given LTI system. A linear constant-coefficient difference equation (LCCDE) serves as a way to express just this relationship in a discrete-time system. Writing the sequence of inputs and outputs, which represent the characteristics of the LTI system, as a difference equation help in understanding and manipulating a system.

Definition 3.1: difference equation
An equation that shows the relationship between consecutive values of a sequence and the differences among them. They are often rearranged as a recursive formula so that a system's output can be computed from the input signal and past outputs.

Example

\[ y[n] + 7y[n - 1] + 2y[n - 2] = x[n] - 4x[n - 1] \]  \hspace{1cm} (3.1)

3.1.2 General Formulas for the Difference Equation

As stated briefly in the definition above, a difference equation is a very useful tool in describing and calculating the output of the system described by the formula for a given sample \( n \). The key property of the difference equation is its ability to help easily find the transform, \( H(z) \), of a system. In the following two subsections, we will look at the general form of the difference equation and the general conversion to a z-transform directly from the difference equation.

3.1.2.1 Difference Equation

The general form of a linear, constant-coefficient difference equation (LCCDE), is shown below:

\[ \sum_{k=0}^{N} a_k y[n - k] = \sum_{k=0}^{M} b_k x[n - k] \]  \hspace{1cm} (3.2)

---

1This content is available online at <http://cnx.org/content/m10595/2.7/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
We can also write the general form to easily express a recursive output, which looks like this:

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

(3.3)

From this equation, note that \( y[n-k] \) represents the outputs and \( x[n-k] \) represents the inputs. The value of \( N \) represents the **order** of the difference equation and corresponds to the memory of the system being represented. Because this equation relies on past values of the output, in order to compute a numerical solution, certain past outputs, referred to as the **initial conditions**, must be known.

### 3.1.2.2 Conversion to Z-Transform

Using the above formula, (3.2), we can easily generalize the **transfer function**, \( H(z) \), for any difference equation. Below are the steps taken to convert any difference equation into its transfer function, i.e. z-transform. The first step involves taking the Fourier Transform\(^2\) of all the terms in (3.2). Then we use the linearity property to pull the transform inside the summation and the time-shifting property of the z-transform to change the time-shifting terms to exponentials. Once this is done, we arrive at the following equation: \( a_0 = 1 \).

\[
Y(z) = - \sum_{k=1}^{N} a_k Y(z) z^{-k} + \sum_{k=0}^{M} b_k X(z) z^{-k}
\]  

(3.4)

\[
H(z) = \frac{Y(z)}{X(z)} = \frac{1 + \sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}
\]  

(3.5)

### 3.1.2.3 Conversion to Frequency Response

Once the z-transform has been calculated from the difference equation, we can go one step further to define the frequency response of the system, or filter, that is being represented by the difference equation.

**NOTE:** Remember that the reason we are dealing with these formulas is to be able to aid us in filter design. A LCCDE is one of the easiest ways to represent FIR filters. By being able to find the frequency response, we will be able to look at the basic properties of any filter represented by a simple LCCDE.

Below is the general formula for the frequency response of a z-transform. The conversion is simple a matter of taking the z-transform formula, \( H(z) \), and replacing every instance of \( z \) with \( e^{jw} \).

\[
H(w) = H(z) |_{z = e^{jw}} = \frac{1 + \sum_{k=0}^{M} b_k e^{-jwk}}{1 + \sum_{k=1}^{N} a_k e^{-jwk}}
\]  

(3.6)

Once you understand the derivation of this formula, look at the module concerning Filter Design from the Z-Transform\(^3\) for a look into how all of these ideas of the Z-transform (Section 3.2), Difference Equation, and Pole/Zero Plots (Section 3.4) play a role in filter design.

\(^2\)“Derivation of the Fourier Transform” <http://cnx.org/content/m0046/latest/>

\(^3\)“Discrete Time Filter Design” <http://cnx.org/content/m10548/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
3.1.3 Example

Example 3.1: Finding Difference Equation

Below is a basic example showing the opposite of the steps above: given a transfer function one can easily calculate the systems difference equation.

\[
H(z) = \frac{(z + 1)^2}{(z - \frac{1}{2})(z + \frac{3}{4})}
\]  

(3.7)

Given this transfer function of a time-domain filter, we want to find the difference equation. To begin with, expand both polynomials and divide them by the highest order \(z\).

\[
H(z) = \frac{(z+1)(z+1)}{(z-\frac{1}{2})(z+\frac{3}{4})}
\]

= \[
\frac{z^2+2z+1}{z^2+2z+1-\frac{1}{2}}
\]

= \[
\frac{1+2z^{-1}+z^{-2}}{1+\frac{1}{2}z^{-1}}
\]

(3.8)

From this transfer function, the coefficients of the two polynomials will be our \(a_k\) and \(b_k\) values found in the general difference equation formula, (3.2). Using these coefficients and the above form of the transfer function, we can easily write the difference equation:

\[
x[n] + 2x[n - 1] + x[n - 2] = y[n] + \frac{1}{4}y[n - 1] - \frac{3}{8}y[n - 2]
\]

(3.9)

In our final step, we can rewrite the difference equation in its more common form showing the recursive nature of the system.

\[
y[n] = x[n] + 2x[n - 1] + x[n - 2] + \frac{-1}{4}y[n - 1] + \frac{3}{8}y[n - 2]
\]

(3.10)

3.1.4 Solving a LCCDE

In order for a linear constant-coefficient difference equation to be useful in analyzing a LTI system, we must be able to find the systems output based upon a known input, \(x(n)\), and a set of initial conditions. Two common methods exist for solving a LCCDE: the direct method and the indirect method, the later being based on the z-transform. Below we will briefly discuss the formulas for solving a LCCDE using each of these methods.

3.1.4.1 Direct Method

The final solution to the output based on the direct method is the sum of two parts, expressed in the following equation:

\[
y(n) = y_h(n) + y_p(n)
\]

(3.11)

The first part, \(y_h(n)\), is referred to as the homogeneous solution and the second part, \(y_p(n)\), is referred to as particular solution. The following method is very similar to that used to solve many differential equations, so if you have taken a differential calculus course or used differential equations before then this should seem very familiar.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
3.1.4.1.1 Homogeneous Solution

We begin by assuming that the input is zero, \( x(n) = 0 \). Now we simply need to solve the homogeneous difference equation:

\[
\sum_{k=0}^{N} a_k y[n - k] = 0 \tag{3.12}
\]

In order to solve this, we will make the assumption that the solution is in the form of an exponential. We will use lambda, \( \lambda \), to represent our exponential terms. We now have to solve the following equation:

\[
\sum_{k=0}^{N} a_k \lambda^{n-k} = 0 \tag{3.13}
\]

We can expand this equation out and factor out all of the lambda terms. This will give us a large polynomial in parenthesis, which is referred to as the characteristic polynomial. The roots of this polynomial will be the key to solving the homogeneous equation. If there are all distinct roots, then the general solution to the equation will be as follows:

\[
y_h(n) = C_1 \lambda_1^n + C_2 \lambda_2^n + \cdots + C_N \lambda_N^n \tag{3.14}
\]

However, if the characteristic equation contains multiple roots then the above general solution will be slightly different. Below we have the modified version for an equation where \( \lambda_1 \) has \( K \) multiple roots:

\[
y_h(n) = C_1 \lambda_1^n + C_1 n \lambda_1^n + C_1 n^2 \lambda_1^n + \cdots + C_1 n^{K-1} \lambda_1^n + C_2 \lambda_2^n + \cdots + C_N \lambda_N^n \tag{3.15}
\]

3.1.4.1.2 Particular Solution

The particular solution, \( y_p(n) \), will be any solution that will solve the general difference equation:

\[
\sum_{k=0}^{N} a_k y_p(n - k) = \sum_{k=0}^{M} b_k x(n - k) \tag{3.16}
\]

In order to solve, our guess for the solution to \( y_p(n) \) will take on the form of the input, \( x(n) \). After guessing at a solution to the above equation involving the particular solution, one only needs to plug the solution into the difference equation and solve it out.

3.1.4.2 Indirect Method

The indirect method utilizes the relationship between the difference equation and z-transform, discussed earlier (Section 3.1.2: General Formulas for the Difference Equation), to find a solution. The basic idea is to convert the difference equation into a z-transform, as described above (Section 3.1.2.2: Conversion to Z-Transform), to get the resulting output, \( Y(z) \). Then by inverse transforming this and using partial-fraction expansion, we can arrive at the solution.

\[
Z\{y(n+1) - y(n)\} = z Y(z) - y(0) \tag{3.17}
\]

This can be interatively extended to an arbitrary order derivative as in Equation (3.18).

\[
Z\{- \sum_{m=0}^{N-1} y(n-m)\} = z^n Y(z) - \sum_{m=0}^{N-1} z^{n-m-1} y^{(m)}(0) \tag{3.18}
\]
Now, the Laplace transform of each side of the differential equation can be taken

\[ Z\left\{ \sum_{k=0}^{N} a_k \left[ y(n-m+1) - \sum_{m=0}^{N-1} y(n-m) y(n) \right] \right\} = Z\{x(n)\} \]  

(3.19)

which by linearity results in

\[ \sum_{k=0}^{N} a_k Z\{y(n-m+1) - \sum_{m=0}^{N-1} y(n-m) y(n)\} = Z\{x(n)\} \]  

(3.20)

and by differentiation properties in

\[ \sum_{k=0}^{N} a_k \left( z^k Z\{y(n)\} - \sum_{m=0}^{N-1} z^{k-m-1} y^{(m)}(0) \right) = Z\{x(n)\} . \]  

(3.21)

Rearranging terms to isolate the Laplace transform of the output,

\[ Z\{y(n)\} = \frac{Z\{x(n)\} + \sum_{k=0}^{N} \sum_{m=0}^{k-1} a_k z^{k-m-1} y^{(m)}(0)}{\sum_{k=0}^{N} a_k z^k} . \]  

(3.22)

Thus, it is found that

\[ Y(z) = X(z) + \sum_{k=0}^{N} \sum_{m=0}^{k-1} a_k z^{k-m-1} y^{(m)}(0) . \]  

(3.23)

In order to find the output, it only remains to find the Laplace transform \( X(z) \) of the input, substitute the initial conditions, and compute the inverse Z-transform of the result. Partial fraction expansions are often required for this last step. This may sound daunting while looking at (3.23), but it is often easy in practice, especially for low order difference equations. (3.23) can also be used to determine the transfer function and frequency response.

As an example, consider the difference equation

\[ y[n-2] + 4y[n-1] + 3y[n] = \cos(n) \]  

(3.24)

with the initial conditions \( y'(0) = 1 \) and \( y(0) = 0 \) Using the method described above, the Z transform of the solution \( y[n] \) is given by

\[ Y[z] = \frac{z}{[z^2 + 1][z + 1][z + 3]} + \frac{1}{[z + 1][z + 3]} . \]  

(3.25)

Performing a partial fraction decomposition, this also equals

\[ Y[z] = \frac{.25}{z + 1} - \frac{.35}{z + 3} + \frac{.1}{z^2 + 1} + \frac{.2}{z^2 + 1} . \]  

(3.26)

Computing the inverse Laplace transform,

\[ y(n) = (.25z^{-n} - .35z^{-3n} + .1\cos(n) + .2\sin(n))u(n) . \]  

(3.27)

One can check that this satisfies that this satisfies both the differential equation and the initial conditions.
3.2 The Z Transform: Definition

3.2.1 Basic Definition of the Z-Transform

The z-transform of a sequence is defined as

\[ X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \quad (3.28) \]

Sometimes this equation is referred to as the bilateral z-transform. At times the z-transform is defined as

\[ X(z) = \sum_{n=0}^{\infty} x[n] z^{-n} \quad (3.29) \]

which is known as the unilateral z-transform.

There is a close relationship between the z-transform and the Fourier transform of a discrete time signal, which is defined as

\[ X(e^{i\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-i\omega n} \quad (3.30) \]

Notice that when the \( z^{-n} \) is replaced with \( e^{-i\omega n} \) the z-transform reduces to the Fourier Transform. When the Fourier Transform exists, \( z = e^{i\omega} \), which is to have the magnitude of \( z \) equal to unity.

3.2.2 The Complex Plane

In order to get further insight into the relationship between the Fourier Transform and the Z-Transform it is useful to look at the complex plane or z-plane. Take a look at the complex plane:

\footnote{This content is available online at \(<\text{http://cnx.org/content/m10549/2.10/>}\).}
The Z-plane is a complex plane with an imaginary and real axis referring to the complex-valued variable $z$. The position on the complex plane is given by $re^{i\omega}$, and the angle from the positive, real axis around the plane is denoted by $\omega$. $X(z)$ is defined everywhere on this plane. $X(e^{i\omega})$ on the other hand is defined only where $|z| = 1$, which is referred to as the unit circle. So for example, $\omega = 1$ at $z = 1$ and $\omega = \pi$ at $z = -1$. This is useful because, by representing the Fourier transform as the z-transform on the unit circle, the periodicity of Fourier transform is easily seen.

### 3.2.3 Region of Convergence

The region of convergence, known as the ROC, is important to understand because it defines the region where the z-transform exists. The ROC for a given $x[n]$, is defined as the range of $z$ for which the z-transform converges. Since the z-transform is a power series, it converges when $x[n]z^{-n}$ is absolutely summable. Stated differently,

$$\sum_{n=-\infty}^{\infty} |x[n]z^{-n}| < \infty$$  \hfill (3.31)

must be satisfied for convergence. This is best illustrated by looking at the different ROC’s of the z-transforms of $\alpha^n u[n]$ and $\alpha^n u[n-1]$.

**Example 3.2**

For

$$x[n] = \alpha^n u[n]$$  \hfill (3.32)
This sequence is an example of a right-sided exponential sequence because it is nonzero for \( n \geq 0 \). It only converges when \( |\alpha z^{-1}| < 1 \). When it converges,

\[
X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n} = \sum_{n=-\infty}^{\infty} \alpha^n u[n] z^{-n} = \sum_{n=0}^{\infty} \alpha^n z^{-n} = \sum_{n=0}^{\infty} (\alpha z^{-1})^n
\]

This sequence is an example of a right-sided exponential sequence because it is nonzero for \( n \geq 0 \). It only converges when \( |\alpha z^{-1}| < 1 \). When it converges,

\[
X(z) = \frac{1}{1-\alpha z^{-1}} = \frac{z}{z-\alpha}
\]

If \( |\alpha z^{-1}| \geq 1 \), then the series, \( \sum_{n=0}^{\infty} (\alpha z^{-1})^n \) does not converge. Thus the ROC is the range of values where

\[
|\alpha z^{-1}| < 1
\]

or, equivalently,

\[
|z| > |\alpha|
\]
Figure 3.3: ROC for $x[n] = \alpha^n u[n]$ where $\alpha = 0.5$

Example 3.3

For

$$x[n] = (-\alpha^n) u[(-n) - 1]$$

(3.37)
Figure 3.4: \( x[n] = (-\alpha^n) u[(-n) - 1] \) where \( \alpha = 0.5 \).

\[
X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n} = \sum_{n=-\infty}^{\infty} (-\alpha^n) u[-n - 1] z^{-n} = -\sum_{n=-\infty}^{-1} \alpha^n z^{-n} = -\sum_{n=-\infty}^{-1} (\alpha^{-1} z)^{-n} = 1 - \sum_{n=1}^{\infty} (\alpha^{-1} z)^n = 1 - \sum_{n=0}^{\infty} (\alpha^{-1} z)^n
\]

The ROC in this case is the range of values where

\[|\alpha^{-1} z| < 1\] (3.39)

or, equivalently,

\[|z| < |\alpha|\] (3.40)

If the ROC is satisfied, then

\[
X(z) = 1 - \frac{1}{1-\alpha^{-1} z} = \frac{z}{z-\alpha}
\]

Available for free at Connexions: <http://cnx.org/content/col10360/1.4>
Figure 3.5: ROC for $x[n] = (-a^n) u[(-n) - 1]$

3.3 Table of Common $z$-Transforms

The table below provides a number of unilateral and bilateral $z$-transforms (Section 3.2). The table also specifies the region of convergence.

NOTE: The notation for $z$ found in the table below may differ from that found in other tables. For example, the basic $z$-transform of $u[n]$ can be written as either of the following two expressions, which are equivalent:

$$\frac{z}{z-1} = \frac{1}{1-z^{-1}}$$

---

5This content is available online at <http://cnx.org/content/m10119/2.14/>.

6"Region of Convergence for the Z-transform" <http://cnx.org/content/m10622/latest/>
Signal | Z-Transform | ROC
--- | --- | ---
$\delta[n - k]$ | $z^{-k}$ | All $(z)$
$u[n]$ | $\frac{z}{z-1}$ | $|z| > 1$
$-u[(-n) - 1]$ | $\frac{z}{z-1}$ | $|z| < 1$
$nu[n]$ | $\frac{(z-1)^{n}}{(z-1)^{2}}$ | $|z| > 1$
$n^2u[n]$ | $\frac{z(z+1)}{(z-1)^{3}}$ | $|z| > 1$
$n^3u[n]$ | $\frac{z(z+1)(z+2)}{(z-1)^{4}}$ | $|z| > 1$
$(-\alpha^n)u[(-n) - 1]$ | $\frac{z}{z-\alpha}$ | $|z| < |\alpha|$
$\alpha^n u[n]$ | $\frac{z}{z-\alpha}$ | $|z| > |\alpha|$
$na^n u[n]$ | $\frac{\alpha z}{(z-\alpha)^{2}}$ | $|z| > |\alpha|$
$n^2\alpha^n u[n]$ | $\frac{\alpha z(z+\alpha)}{(z-\alpha)^{4}}$ | $|z| > |\alpha|$
$\prod_{m=1}^{n-k+1} \frac{\alpha^{n-m} z}{(z-\alpha)^{m!}}$ | $\frac{z}{(z-\alpha)^{n}}$ | $|z| > |\alpha|$
$\gamma^n \cos (\alpha n) u[n]$ | $\frac{z(z-\gamma \cos (\alpha))}{z^2-(2 \cos (\alpha))z+\gamma^2}$ | $|z| > |\gamma|$
$\gamma^n \sin (\alpha n) u[n]$ | $\frac{z \sin (\alpha) z^2 - (2 \cos (\alpha))z + \gamma^2}$ | $|z| > |\gamma|$

Table 3.1

3.4 Understanding Pole/Zero Plots on the Z-Plane

3.4.1 Introduction to Poles and Zeros of the Z-Transform

It is quite difficult to qualitatively analyze the Laplace transform and Z-transform (Section 3.2), since mappings of their magnitude and phase or real part and imaginary part result in multiple mappings of 2-dimensional surfaces in 3-dimensional space. For this reason, it is very common to examine a plot of a transfer function’s poles and zeros to try to gain a qualitative idea of what a system does.

Once the Z-transform of a system has been determined, one can use the information contained in function’s polynomials to graphically represent the function and easily observe many defining characteristics. The Z-transform will have the below structure, based on Rational Functions:

$$X(z) = \frac{P(z)}{Q(z)} \quad (3.43)$$

The two polynomials, $P(z)$ and $Q(z)$, allow us to find the poles and zeros of the Z-Transform.

**Definition 3.2: zeros**
1. The value(s) for $z$ where $P(z) = 0$.
2. The complex frequencies that make the overall gain of the filter transfer function zero.

**Definition 3.3: poles**
1. The value(s) for $z$ where $Q(z) = 0$.
2. The complex frequencies that make the overall gain of the filter transfer function infinite.

---

7This content is available online at <http://cnx.org/content/m10556/2.13/>.
8"The Laplace Transform" <http://cnx.org/content/m10110/latest/>
9"Transfer Functions" <http://cnx.org/content/m0028/latest/>
10"Rational Functions and the Z-Transform" <http://cnx.org/content/m10593/latest/>
11"Poles and Zeros" <http://cnx.org/content/m10112/latest/>
Example 3.4
Below is a simple transfer function with the poles and zeros shown below it.

\[ H(z) = \frac{z + 1}{(z - \frac{1}{2})(z + \frac{3}{4})} \]

The zeros are: \(\{-1\}\)
The poles are: \(\{\frac{1}{2}, -\frac{3}{4}\}\)

3.4.2 The Z-Plane
Once the poles and zeros have been found for a given Z-Transform, they can be plotted onto the Z-Plane. The Z-plane is a complex plane with an imaginary and real axis referring to the complex-valued variable \(z\). The position on the complex plane is given by \(re^{j\theta}\) and the angle from the positive, real axis around the plane is denoted by \(\theta\). When mapping poles and zeros onto the plane, poles are denoted by an "x" and zeros by an "o". The below figure shows the Z-Plane, and examples of plotting zeros and poles onto the plane can be found in the following section.

3.4.3 Examples of Pole/Zero Plots
This section lists several examples of finding the poles and zeros of a transfer function and then plotting them onto the Z-Plane.

Example 3.5: Simple Pole/Zero Plot

\[ H(z) = \frac{z}{(z - \frac{1}{2})(z + \frac{3}{4})} \]

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
The zeros are: \{0\}
The poles are: \(\{\frac{1}{2}, -\frac{3}{2}\}\)

Pole/Zero Plot

**Figure 3.7:** Using the zeros and poles found from the transfer function, the one zero is mapped to zero and the two poles are placed at \(\frac{1}{2}\) and \(-\frac{3}{2}\)

Example 3.6: Complex Pole/Zero Plot

\[
H(z) = \frac{(z-i)(z+i)}{(z-(1\frac{1}{2}-\frac{1}{2}i))(z-1\frac{1}{2}+\frac{1}{2}i)}
\]

The zeros are: \(\{i, -i\}\)
The poles are: \(\{-1, \frac{1}{2} + \frac{1}{2}i, \frac{1}{2} - \frac{1}{2}i\}\)
Figure 3.8: Using the zeros and poles found from the transfer function, the zeros are mapped to $\pm (i)$, and the poles are placed at $-1, \frac{1}{2} + \frac{1}{2}i$ and $\frac{1}{2} - \frac{1}{2}i$.

Example 3.7: Pole-Zero Cancellation

An easy mistake to make with regards to poles and zeros is to think that a function like $\frac{(s+3)(s-1)}{s-1}$ is the same as $s+3$. In theory they are equivalent, as the pole and zero at $s = 1$ cancel each other out in what is known as pole-zero cancellation. However, think about what may happen if this were a transfer function of a system that was created with physical circuits. In this case, it is very unlikely that the pole and zero would remain in exactly the same place. A minor temperature change, for instance, could cause one of them to move just slightly. If this were to occur a tremendous amount of volatility is created in that area, since there is a change from infinity at the pole to zero at the zero in a very small range of signals. This is generally a very bad way to try to eliminate a pole. A much better way is to use control theory to move the pole to a better place.

**Note:** It is possible to have more than one pole or zero at any given point. For instance, the discrete-time transfer function $H(z) = z^2$ will have two zeros at the origin and the continuous-time function $H(s) = \frac{1}{s^2}$ will have 25 poles at the origin.

**MATLAB** - If access to MATLAB is readily available, then you can use its functions to easily create pole/zero plots. Below is a short program that plots the poles and zeros from the above example onto the Z-Plane.

```matlab
% Set up vector for zeros
z = [j ; -j];

% Set up vector for poles
p = [-1 ; .5+.5j ; .5-.5j];

figure(1);
```

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
3.4.4 Interactive Demonstration of Poles and Zeros

Figure 3.9: Interact (when online) with a Mathematica CDF demonstrating Pole/Zero Plots. To Download, right-click and save target as .cdf.
3.4.5 Applications for pole-zero plots

3.4.5.1 Stability and Control theory

Now that we have found and plotted the poles and zeros, we must ask what it is that this plot gives us. Basically what we can gather from this is that the magnitude of the transfer function will be larger when it is closer to the poles and smaller when it is closer to the zeros. This provides us with a qualitative understanding of what the system does at various frequencies and is crucial to the discussion of stability\(^{12}\).

3.4.5.2 Pole/Zero Plots and the Region of Convergence

The region of convergence (ROC) for \(X(z)\) in the complex \(Z\)-plane can be determined from the pole/zero plot. Although several regions of convergence may be possible, where each one corresponds to a different impulse response, there are some choices that are more practical. A ROC can be chosen to make the transfer function causal and/or stable depending on the pole/zero plot.

Filter Properties from ROC

- If the ROC extends outward from the outermost pole, then the system is causal.
- If the ROC includes the unit circle, then the system is stable.

Below is a pole/zero plot with a possible ROC of the \(Z\)-transform in the Simple Pole/Zero Plot (Example 3.5: Simple Pole/Zero Plot) discussed earlier. The shaded region indicates the ROC chosen for the filter. From this figure, we can see that the filter will be both causal and stable since the above listed conditions are both met.

**Example 3.8**

\[
H(z) = \frac{z}{(z - \frac{1}{2})(z + \frac{1}{2})}
\]

**Region of Convergence for the Pole/Zero Plot**

![Figure 3.10: The shaded area represents the chosen ROC for the transfer function.](http://cnx.org/content/m10113/latest/)

\(^{12}\) "BIBO Stability of Continuous Time Systems" <http://cnx.org/content/m10113/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
3.4.5.3 Frequency Response and Pole/Zero Plots

The reason it is helpful to understand and create these pole/zero plots is due to their ability to help us easily design a filter. Based on the location of the poles and zeros, the magnitude response of the filter can be quickly understood. Also, by starting with the pole/zero plot, one can design a filter and obtain its transfer function very easily.

3.5 Filtering in the Frequency Domain

Because we are interested in actual computations rather than analytic calculations, we must consider the details of the discrete Fourier transform. To compute the length-$N$ DFT, we assume that the signal has a duration less than or equal to $N$. Because frequency responses have an explicit frequency-domain specification in terms of filter coefficients, we don’t have a direct handle on which signal has a Fourier transform equaling a given frequency response. Finding this signal is quite easy. First of all, note that the discrete-time Fourier transform of a unit sample equals one for all frequencies. Since the input and output of linear, shift-invariant systems are related to each other by $Y (e^{j2\pi f}) = H (e^{j2\pi f}) X (e^{j2\pi f})$, a unit-sample input, which has $X (e^{j2\pi f}) = 1$, results in the output’s Fourier transform equaling the system’s transfer function.

Exercise 3.5.1 (Solution on p. 147.)

This statement is a very important result. Derive it yourself.

In the time-domain, the output for a unit-sample input is known as the system’s unit-sample response, and is denoted by $h(n)$. Combining the frequency-domain and time-domain interpretations of a linear, shift-invariant system’s unit-sample response, we have that $h(n)$ and the transfer function are Fourier transform pairs in terms of the discrete-time Fourier transform.

\begin{equation}
    h(n) \leftrightarrow H (e^{j2\pi f})
\end{equation}

(3.44)

Returning to the issue of how to use the DFT to perform filtering, we can analytically specify the frequency response, and derive the corresponding length-$N$ DFT by sampling the frequency response.

\begin{equation}
    \forall k, k = \{0, \ldots, N-1\} : \left( H (k) = H \left( e^{j\frac{2\pi k}{N}} \right) \right)
\end{equation}

(3.45)

Computing the inverse DFT yields a length-$N$ signal no matter what the actual duration of the unit-sample response might be. If the unit-sample response has a duration less than or equal to $N$ (it’s a FIR filter), computing the inverse DFT of the sampled frequency response indeed yields the unit-sample response. If, however, the duration exceeds $N$, errors are encountered. The nature of these errors is easily explained by appealing to the Sampling Theorem. By sampling in the frequency domain, we have the potential for aliasing in the time domain (sampling in one domain, be it time or frequency, can result in aliasing in the other) unless we sample fast enough. Here, the duration of the unit-sample response determines the minimal sampling rate that prevents aliasing. For FIR systems — they by definition have finite-duration unit sample responses — the number of required DFT samples equals the unit-sample response’s duration: $N \geq q$.

Exercise 3.5.2 (Solution on p. 147.)

Derive the minimal DFT length for a length-$q$ unit-sample response using the Sampling Theorem. Because sampling in the frequency domain causes repetitions of the unit-sample response in the time domain, sketch the time-domain result for various choices of the DFT length $N$.

---

13 This content is available online at <http://cnx.org/content/m10257/2.18/>.
14 "Discrete-Time Systems in the Frequency Domain", (1) <http://cnx.org/content/m0530/latest/#dtinf>
Exercise 3.5.3
Express the unit-sample response of a FIR filter in terms of difference equation coefficients.
Note that the corresponding question for IIR filters is far more difficult to answer: Consider the example\(^{15}\).

For IIR systems, we cannot use the DFT to find the system’s unit-sample response: aliasing of the unit-sample response will always occur. Consequently, we can only implement an IIR filter accurately in the time domain with the system’s difference equation. **Frequency-domain implementations are restricted to FIR filters.**

Another issue arises in frequency-domain filtering that is related to time-domain aliasing, this time when we consider the output. Assume we have an input signal having duration \(N_x\) that we pass through a FIR filter having a length-\(q+1\) unit-sample response. What is the duration of the output signal? The difference equation for this filter is

\[
y(n) = b_0 x(n) + \cdots + b_q x(n-q)
\]  

This equation says that the output depends on current and past input values, with the input value \(q\) samples previous defining the extent of the filter’s **memory** of past input values. For example, the output at index \(N_x\) depends on \(x(N_x)\) (which equals zero), \(x(N_x-1)\), through \(x(N_x-q)\). Thus, the output returns to zero only after the last input value passes through the filter’s memory. As the input signal’s last value occurs at index \(N_x-1\), the last nonzero output value occurs when \(n-q = N_x-1\) or \(n = q + N_x-1\). Thus, the output signal’s duration equals \(q + N_x\).

Exercise 3.5.4
In words, we express this result as "The output’s duration equals the input’s duration plus the filter’s duration minus one.". Demonstrate the accuracy of this statement.

The main theme of this result is that a filter’s output extends longer than either its input or its unit-sample response. Thus, to avoid aliasing when we use DFTs, the dominant factor is not the duration of input or of the unit-sample response, but of the output. Thus, the number of values at which we must evaluate the frequency response’s DFT must be at least \(q + N_x\) and we must compute the same length DFT of the input. To accommodate a shorter signal than DFT length, we simply **zero-pad** the input: Ensure that for indices extending beyond the signal’s duration that the signal is zero. Frequency-domain filtering, diagrammed in Figure 3.11, is accomplished by storing the filter’s frequency response as the DFT \(H(k)\), computing the input’s DFT \(X(k)\), multiplying them to create the output’s DFT \(Y(k) = H(k) X(k)\), and computing the inverse DFT of the result to yield \(y(n)\).

---

Figure 3.11: To filter a signal in the frequency domain, first compute the DFT of the input, multiply the result by the sampled frequency response, and finally compute the inverse DFT of the product. The DFT’s length must be at least the sum of the input’s and unit-sample response’s duration minus one. We calculate these discrete Fourier transforms using the fast Fourier transform algorithm, of course.

---

\(^{15}\)“Discrete-Time Systems in the Time-Domain”, Example 1 <http://cnx.org/content/m10251/latest/#p0>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
output is always true: \( Y(e^{j2\pi f}) = H(e^{j2\pi f}) \cdot X(e^{j2\pi f}) \). The Fourier transforms in this result are discrete-time Fourier transforms; for example, \( X(e^{j2\pi f}) = \sum_n x(n) e^{-j2\pi fn} \). Unfortunately, using this relationship to perform filtering is restricted to the situation when we have analytic formulas for the frequency response and the input signal. The reason why we had to "invent" the discrete Fourier transform (DFT) has the same origin: The spectrum resulting from the discrete-time Fourier transform depends on the continuous frequency variable \( f \). That's fine for analytic calculation, but computationally we would have to make an uncountably infinite number of computations.

**NOTE:** Did you know that two kinds of infinities can be meaningfully defined? A countably infinite quantity means that it can be associated with a limiting process associated with integers. An uncountably infinite quantity cannot be so associated. The number of rational numbers is countably infinite (the numerator and denominator correspond to locating the rational by row and column; the total number so-located can be counted, voila!); the number of irrational numbers is uncountably infinite. Guess which is "bigger?"

The DFT computes the Fourier transform at a finite set of frequencies — samples the true spectrum — which can lead to aliasing in the time-domain unless we sample sufficiently fast. The sampling interval here is \( \frac{1}{K} \) for a length-\( K \) DFT: faster sampling to avoid aliasing thus requires a longer transform calculation. Since the longest signal among the input, unit-sample response and output is the output, it is that signal’s duration that determines the transform length. We simply extend the other two signals with zeros (zero-pad) to compute their DFTs.

**Example 3.9**
Suppose we want to average daily stock prices taken over last year to yield a running weekly average (average over five trading sessions). The filter we want is a length-5 averager (as shown in the unit-sample response\(^{16}\)), and the input’s duration is 253 (365 calendar days minus weekend days and holidays). The output duration will be \( 253 + 5 - 1 = 257 \), and this determines the transform length we need to use. Because we want to use the FFT, we are restricted to power-of-two transform lengths. We need to choose any FFT length that exceeds the required DFT length. As it turns out, 256 is a power of two (\( 2^8 = 256 \)), and this length just undershoots our required length. To use frequency domain techniques, we must use length-512 fast Fourier transforms.

\(^{16}\)"Discrete-Time Systems in the Time-Domain", Figure 2 \(<http://cnx.org/content/m10251/latest/#fig1002>\)
Figure 3.12: The blue line shows the Dow Jones Industrial Average from 1997, and the red one the length-5 boxcar-filtered result that provides a running weekly of this market index. Note the "edge" effects in the filtered output.

Figure 3.12 shows the input and the filtered output. The MATLAB programs that compute the filtered output in the time and frequency domains are

**Time Domain**

\[
h = [1 1 1 1 1] / 5;
\]

\[
y = \text{filter}(h, 1, [\text{djia} \; \text{zeros}(1, 4)]);
\]

**Frequency Domain**

\[
h = [1 1 1 1 1] / 5;
\]

\[
\text{DJIA} = \text{fft(djia, 512)};
\]

\[
H = \text{fft}(h, 512);
\]

\[
Y = H .* X;
\]

\[
y = \text{ifft}(Y);
\]

**NOTE:** The `filter` program has the feature that the length of its output equals the length of its input. To force it to produce a signal having the proper length, the program zero-pads the input appropriately.

MATLAB’s `fft` function automatically zero-pads its input if the specified transform length (its second argument) exceeds the signal’s length. The frequency domain result will have a small imaginary component — largest value is \(2.2 \times 10^{-11}\) — because of the inherent finite precision nature of computer arithmetic. Because of the unfortunate misfit between signal lengths and favored FFT lengths, the number of arithmetic operations in the time-domain implementation is far less than those required by the frequency domain version: 514 versus 62,271. If the input signal had been one sample shorter, the frequency-domain computations would have been more than a factor of two less (28,696), but far more than in the time-domain implementation.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
An interesting signal processing aspect of this example is demonstrated at the beginning and end of the output. The ramping up and down that occurs can be traced to assuming the input is zero before it begins and after it ends. The filter "sees" these initial and final values as the difference equation passes over the input. These artifacts can be handled in two ways: we can just ignore the edge effects or the data from previous and succeeding years’ last and first week, respectively, can be placed at the ends.

3.6 Linear-Phase FIR Filters\textsuperscript{17}

3.6.1 THE AMPLITUDE RESPONSE

If the real and imaginary parts of $H^f (\omega)$ are given by

$$H^f (\omega) = \Re (\omega) + i \Im (\omega)$$

(3.47)

the magnitude and phase are defined as

$$|H^f (\omega)| = \sqrt{(\Re (\omega))^2 + (\Im (\omega))^2}$$

$$p(\omega) = \arctan \left( \frac{\Im (\omega)}{\Re (\omega)} \right)$$

so that

$$H^f (\omega) = |H^f (\omega)| e^{ip (\omega)}$$

(3.48)

With this definition, $|H^f (\omega)|$ is never negative and $p (\omega)$ is usually discontinuous, but it can be very helpful to write $H^f (\omega)$ as

$$H^f (\omega) = A (\omega) e^{i\theta (\omega)}$$

(3.49)

where $A (\omega)$ can be positive and negative, and $\theta (\omega)$ continuous. $A (\omega)$ is called the amplitude response. Figure 3.13 illustrates the difference between $|H^f (\omega)|$ and $A (\omega)$.

\textsuperscript{17}This content is available online at <http://cnx.org/content/m10705/2.3/>.}
A linear-phase phase filter is one for which the continuous phase \( \theta(\omega) \) is linear.

\[
H_f(\omega) = A(\omega) e^{i\theta(\omega)}
\]

with

\[
\theta(\omega) = M\omega + B
\]

We assume in the following that the impulse response \( h(n) \) is real-valued.

### 3.6.2 WHY LINEAR-PHASE?

If a discrete-time cosine signal

\[
x_1(n) = \cos(\omega_1 n + \phi_1)
\]

is processed through a discrete-time filter with frequency response

\[
H_f(\omega) = A(\omega) e^{i\theta(\omega)}
\]

then the output signal is given by

\[
y_1(n) = A(\omega_1) \cos(\omega_1 n + \phi_1 + \theta(\omega_1))
\]
or

\[ y_1(n) = A(\omega_1) \cos \left( \omega_1 \left( n + \frac{\theta(\omega_1)}{\omega_1} \right) + \phi_1 \right) \]

The LTI system has the effect of scaling the cosine signal and delaying it by \( \frac{\theta(\omega_1)}{\omega_1} \).

Exercise 3.6.1  
(Solution on p. 147.)

When does the system delay cosine signals with different frequencies by the same amount?

The function \( \frac{\theta(\omega)}{\omega} \) is called the phase delay. A linear phase filter therefore has constant phase delay.

3.6.3 WHY LINEAR-PHASE: EXAMPLE

Consider a discrete-time filter described by the difference equation

\[
\begin{align*}
y(n) &= 0.1821x(n) + 0.7865x(n-1) - 0.6804x(n-2) + x(n-3) + 0.6804y(n-1) - 0.7865y(n-2) + 0.1821y(n-3) \\
&= 3.50
\end{align*}
\]

When \( \omega_1 = 0.31\pi \), then the delay is \( \frac{\theta(\omega)}{\omega_1} = 2.45 \). The delay is illustrated in Figure 3.14:

![Figure 3.14](http://cnx.org/content/col10360/1.4)
Notice that the delay is fractional — the discrete-time samples are not exactly reproduced in the output. The fractional delay can be interpreted in this case as a delay of the underlying continuous-time cosine signal.

### 3.6.4 WHY LINEAR-PHASE: EXAMPLE (2)

Consider the same system given on the previous slide, but let us change the frequency of the cosine signal. When $\omega_2 = 0.47\pi$, then the delay is $\frac{-\theta(\omega_2)}{\omega_2} = 0.14$.

![Graph of input and output signals](image)

**Figure 3.15**

**NOTE:** For this example, the delay depends on the frequency, because this system does not have linear phase.

### 3.6.5 WHY LINEAR-PHASE: MORE

From the previous slides, we see that a filter will delay different frequency components of a signal by the same amount if the filter has linear phase (constant phase delay).
In addition, when a narrow band signal (as in AM modulation) goes through a filter, the envelop will be delayed by the group delay or envelop delay of the filter. The amount by which the envelop is delayed is independent of the carrier frequency only if the filter has linear phase.

Also, in applications like image processing, filters with non-linear phase can introduce artifacts that are visually annoying.

### 3.7 Filter Structures

A realizable filter must require only a finite number of computations per output sample. For linear, causal, time-Invariant filters, this restricts one to rational transfer functions of the form

\[ H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \]

Assuming no pole-zero cancellations, \( H(z) \) is FIR if \( \forall i, i > 0 : (a_i = 0) \), and IIR otherwise. Filter structures usually implement rational transfer functions as difference equations.

Whether FIR or IIR, a given transfer function can be implemented with many different filter structures. With infinite-precision data, coefficients, and arithmetic, all filter structures implementing the same transfer function produce the same output. However, different filter structures may produce very different errors with quantized data and finite-precision or fixed-point arithmetic. The computational expense and memory usage may also differ greatly. Knowledge of different filter structures allows DSP engineers to trade off these factors to create the best implementation.

### 3.8 Overview of Digital Filter Design

**Advantages of FIR filters**

1. Straight forward conceptually and simple to implement
2. Can be implemented with fast convolution
3. Always stable
4. Relatively insensitive to quantization
5. Can have linear phase (same time delay of all frequencies)

**Advantages of IIR filters**

1. Better for approximating analog systems
2. For a given magnitude response specification, IIR filters often require much less computation than an equivalent FIR, particularly for narrow transition bands

Both FIR and IIR filters are very important in applications.

**Generic Filter Design Procedure**

1. Choose a desired response, based on application requirements
2. Choose a filter class
3. Choose a quality measure
4. Solve for the filter in class 2 optimizing criterion in 3
3.8.1 Perspective on FIR filtering

Most of the time, people do $L^\infty$ optimal design, using the Parks-McClellan algorithm (Section 3.11). This is probably the second most important technique in "classical" signal processing (after the Cooley-Tukey (radix-2) FFT).

Most of the time, FIR filters are designed to have linear phase. The most important advantage of FIR filters over IIR filters is that they can have exactly linear phase. There are advanced design techniques for minimum-phase filters, constrained $L^2$ optimal designs, etc. (see chapter 8 of text). However, if only the magnitude of the response is important, IIR filters usually require much fewer operations and are typically used, so the bulk of FIR filter design work has concentrated on linear phase designs.

3.9 Window Design Method

The truncate-and-delay design procedure is the simplest and most obvious FIR design procedure.

Exercise 3.9.1
Is it any Good?

3.9.1 L2 optimization criterion

find $\forall n, 0 \leq n \leq M - 1 : (h[n])$, maximizing the energy difference between the desired response and the actual response: i.e., find

$$\min_{h[n]} \left\{ h[n], \int_{-\pi}^{\pi} (|H_d(\omega) - H(\omega)|^2 d\omega) \right\}$$

by Parseval’s relationship.

$$\min_{h[n]} \left\{ h[n], \int_{-\pi}^{\pi} (|H_d(\omega) - H(\omega)|^2 d\omega) \right\} = 2\pi \sum_{n=-\infty}^{\infty} (|h_d[n] - h[n]|)^2 = (3.51)$$

$$2\pi \left( \sum_{n=-\infty}^{-1} (|h_d[n] - h[n]|)^2 + \sum_{n=0}^{M-1} (|h_d[n] - h[n]|)^2 + \sum_{n=M}^{\infty} (|h_d[n] - h[n]|)^2 \right)$$

Since $\forall n, n < 0 \leq M : (h[n])$ this becomes

$$\min_{h[n]} \left\{ h[n], \int_{-\pi}^{\pi} (|H_d(\omega) - H(\omega)|^2 d\omega) \right\} = \sum_{n=-\infty}^{-1} (|h_d[n]|)^2 + \sum_{n=0}^{M-1} (|h_d[n]|)^2 + \sum_{n=M}^{\infty} (|h_d[n]|)^2$$

NOTE: $h[n]$ has no influence on the first and last sums.

The best we can do is let

$$h[n] = \begin{cases} 
    h_d[n] & \text{if } 0 \leq n \leq M - 1 \\
    0 & \text{if else}
\end{cases}$$

Thus $h[n] = h_d[n] w[n]$,

$$w[n] = \begin{cases} 
    1 & \text{if } 0 \leq n (M - 1) \\
    0 & \text{if else}
\end{cases}$$

20 "Decimation-in-time (DIT) Radix-2 FFT" <http://cnx.org/content/m12016/latest/>
21 This content is available online at <http://cnx.org/content/m12790/1.2/>.
22 "Parseval’s Theorem" <http://cnx.org/content/m0047/latest/>
CHAPTER 3. DIGITAL FILTERING

is optimal in a least-total-squared-error (L₂, or energy) sense!

Exercise 3.9.2  
Why, then, is this design often considered undesirable?

For desired spectra with discontinuities, the least-square designs are poor in a minimax (worst-case, or L₂) error sense.

3.9.2 Window Design Method

Apply a more gradual truncation to reduce "ringing" (Gibb's Phenomenon

∀ \( n \leq M - 1 \ h(n) = h(n) \)  

NOTE: \( H(\omega) = H_d(\omega) * W(\omega) \)

The window design procedure (except for the boxcar window) is ad-hoc and not optimal in any usual sense. However, it is very simple, so it is sometimes used for "quick-and-dirty" designs of if the error criterion is itself heuristics.

3.10 Frequency Sampling Design Method for FIR filters

Given a desired frequency response, the frequency sampling design method designs a filter with a frequency response exactly equal to the desired response at a particular set of frequencies \( \omega_k \).

Procedure

∀k, k = [0, 1, ..., N - 1] : \( H_d(\omega_k) = \sum_{n=0}^{M-1} h(n) e^{-i\omega_k n} \)  

(3.52)

NOTE: Desired Response must include linear phase shift (if linear phase is desired)

Exercise 3.10.1  
What is \( H_d(\omega) \) for an ideal lowpass filter, cutoff at \( \omega_c \)?

NOTE: This set of linear equations can be written in matrix form

\[
\begin{pmatrix}
H_d(\omega_0) \\
H_d(\omega_1) \\
\vdots \\
H_d(\omega_{N-1}) \\
\end{pmatrix} =
\begin{pmatrix}
e^{-i\omega_0} & e^{-i\omega_1} & \cdots & e^{-i\omega_{N-1}} \\
e^{-i\omega_0} & e^{-i\omega_1} & \cdots & e^{-i\omega_{N-1}} \\
\vdots & \vdots & \vdots & \vdots \\
e^{-i\omega_0} & e^{-i\omega_1} & \cdots & e^{-i\omega_{N-1}} \\
\end{pmatrix}
\begin{pmatrix}
h(0) \\
h(1) \\
\vdots \\
h(M-1) \\
\end{pmatrix}
\]

or

\[
H_d = Wh
\]

So

\[
h = W^{-1}H_d
\]

NOTE: W is a square matrix for \( N = M \), and invertible as long as \( \omega_i \neq \omega_j + 2\pi l, i \neq j \)

23"Gibbs Phenomena" <http://cnx.org/content/m10092/latest/>  
24This content is available online at <http://cnx.org/content/m12789/1.2/>.  
Available for free at Connexions <http://cnx.org/content/col10360/1.4>
3.10.1 Important Special Case

What if the frequencies are equally spaced between 0 and 2π, i.e. \( \omega_k = \frac{2\pi k}{M} + \alpha \)?

Then

\[
H_d(\omega_k) = \sum_{n=0}^{M-1} h(n) e^{-i(\frac{2\pi k}{M} + \alpha)n} = \sum_{n=0}^{M-1} \left( h(n) e^{-i\alpha n} \right) e^{-i(\frac{2\pi k}{M})n} = \text{DFT!}
\]

so

\[
h(n) e^{-i\alpha n} = \frac{1}{M} \sum_{k=0}^{M-1} H_d(\omega_k) e^{i\frac{2\pi k}{M}n}
\]

or

\[
h[n] = \frac{e^{i\alpha n}}{M} \sum_{k=0}^{M-1} H_d[\omega_k] e^{i\frac{2\pi k}{M}n} = e^{i\alpha n} \text{IDFT} \left[ H_d[\omega_k] \right]
\]

3.10.2 Important Special Case #2

\( h[n] \) symmetric, linear phase, and has real coefficients. Since \( h[n] = h[M-n] \), there are only \( \frac{M}{2} \) degrees of freedom, and only \( \frac{M}{2} \) linear equations are required.

\[
H[\omega_k] = \sum_{n=0}^{M-1} h[n] e^{-i\omega_k n}
\]

\[
= \begin{cases} 
\sum_{n=0}^{\frac{M}{2}-1} h[n] \left( e^{-i(\omega_k n)} + e^{-i(\omega_k (M-n-1))} \right) & \text{if } M \text{ even} \\
\sum_{n=0}^{M-\frac{1}{2}} h[n] \left( e^{-i(\omega_k n)} + e^{-i(\omega_k (M-n-1))} \right) \left( h \left[ \frac{M-1}{2} \right] e^{-i(\omega_k \frac{M-1}{2})} \right) & \text{if } M \text{ odd} \\
\end{cases}
\]

(3.56)

Removing linear phase from both sides yields

\[
A(\omega_k) = \begin{cases} 
2 \sum_{n=0}^{\frac{M}{2}-1} h[n] \cos \omega_k \left( \frac{M-1}{2} - n \right) & \text{if } M \text{ even} \\
2 \sum_{n=0}^{M-\frac{1}{2}} h[n] \cos \omega_k \left( \frac{M-1}{2} - n \right) + h \left[ \frac{M-1}{2} \right] & \text{if } M \text{ odd} \\
\end{cases}
\]

Due to symmetry of response for real coefficients, only \( \frac{M}{2} \omega_k \) on \( \omega \in [0, \pi) \) need be specified, with the frequencies \(-\omega_k\) thereby being implicitly defined also. Thus we have \( \frac{M}{2} \) real-valued simultaneous linear equations to solve for \( h[n] \).

3.10.2.1 Special Case 2a

\( h[n] \) symmetric, odd length, linear phase, real coefficients, and \( \omega_k \) equally spaced: \forall k, 0 \leq k \leq M-1 : (\omega_k = \frac{n\pi}{M})

\[
h[n] = \text{IDFT} \left[ H_d(\omega_k) \right]
\]

\[
= \frac{1}{M} \sum_{k=0}^{M-1} A(\omega_k) e^{-i(\frac{2\pi k}{M})M} e^{i\frac{2\pi k}{M}n} = \frac{1}{M} \sum_{k=0}^{M-1} A(k) e^{i(\frac{2\pi k}{M}(n-\frac{M-1}{2}))}
\]

(3.57)

To yield real coefficients, \( A(\omega) \) must be symmetric

\[
(A(\omega) = A(-\omega)) \Rightarrow (A[k] = A[M-k])
\]
\[ h[n] = \frac{1}{M} \left( A(0) + \sum_{k=1}^{M-1} A[k] \left( e^{i\frac{2\pi k}{M} (n - \frac{M-1}{2})} + e^{-i\frac{2\pi k}{M} (n - \frac{M-1}{2})} \right) \right) \]
\[ = \frac{1}{M} \left( A(0) + 2 \sum_{k=1}^{M-1} A[k] \cos \left( \frac{2\pi k}{M} (n - \frac{M-1}{2}) \right) \right) \]
\[ = \frac{1}{M} \left( A(0) + 2 \sum_{k=1}^{M-1} A[k] (-1)^k \cos \left( \frac{2\pi k}{M} (n + \frac{1}{2}) \right) \right) \]

Similar equations exist for even lengths, anti-symmetric, and \( \alpha = \frac{1}{2} \) filter forms.

3.10.3 Comments on frequency-sampled design

This method is simple conceptually and very efficient for equally spaced samples, since \( h[n] \) can be computed using the IDFT.

\( H(\omega) \) for a frequency sampled design goes exactly through the sample points, but it may be very far from the desired response for \( \omega \neq \omega_k \). This is the main problem with frequency sampled design.

Possible solution to this problem: specify more frequency samples than degrees of freedom, and minimize the total error in the frequency response at all of these samples.

3.10.4 Extended frequency sample design

For the samples \( H(\omega_k) \) where \( 0 \leq k \leq M - 1 \) and \( N > M \), find \( h[n] \), where \( 0 \leq n \leq M - 1 \) minimizing \( \| H_d(\omega_k) - H(\omega_k) \| \)

For \( \| l \|_\infty \) norm, this becomes a linear programming problem (standard packages available!)

Here we will consider the \( \| l \|_2 \) norm.

To minimize the \( \| l \|_2 \) norm; that is, \( \sum_{n=0}^{N-1} |H_d(\omega_k) - H(\omega_k)| \), we have an overdetermined set of linear equations:

\[
\begin{pmatrix}
e^{-i\omega_0} & \ldots & e^{-i\omega_0(M-1)} \\
e^{-i\omega_1} & \ldots & e^{-i\omega_1(M-1)} \\
e^{-i\omega_{N-1}} & \ldots & e^{-i\omega_{N-1}(M-1)}
\end{pmatrix}
\begin{pmatrix}
h[0] \\
h[1] \\
h[N-1]
\end{pmatrix}
= \begin{pmatrix}
H_d(\omega_0) \\
H_d(\omega_1) \\
\vdots \\
H_d(\omega_{N-1})
\end{pmatrix}
\]

or

\[ Wh = H_d \]

The minimum error norm solution is well known to be \( h = (WW)^{-1}WH_d \); \( (WW)^{-1} \) is well known as the pseudo-inverse matrix.

Note: Extended frequency sampled design discourages radical behavior of the frequency response between samples for sufficiently closely spaced samples. However, the actual frequency response may no longer pass exactly through any of the \( H_d(\omega_k) \).

3.11 Parks-McClellan FIR Filter Design

The approximation tolerances for a filter are very often given in terms of the maximum, or worst-case, deviation within frequency bands. For example, we might wish a lowpass filter in a (16-bit) CD player to have no more than \( \frac{1}{2} \)-bit deviation in the pass and stop bands.

\[
H(\omega) = \begin{cases} 
1 - \frac{1}{2^n} & \text{if } \frac{1}{2^n} \leq |H(\omega)| \leq 1 + \frac{1}{2^n} \\
\frac{1}{2^n} & \text{if } \frac{1}{2^n} \leq |H(\omega)| \leq \omega_p \\
\frac{1}{2^n} & \text{if } \omega_p \leq |\omega| \leq \pi 
\end{cases}
\]

\[25\text{This content is available online at <http://cnx.org/content/m12799/1.3/> .}\]

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
The Parks-McClellan filter design method efficiently designs linear-phase FIR filters that are optimal in terms of worst-case (minimax) error. Typically, we would like to have the shortest-length filter achieving these specifications. Figure 3.16 illustrates the amplitude frequency response of such a filter.

![Figure 3.16: The black boxes on the left and right are the passbands, the black boxes in the middle represent the stop band, and the space between the boxes are the transition bands. Note that overshoots may be allowed in the transition bands.](image)

Exercise 3.11.1
Must there be a transition band?  
(Solution on p. 148.)

3.11.1 Formal Statement of the $L_{\infty}$ (Minimax) Design Problem
For a given filter length ($M$) and type (odd length, symmetric, linear phase, for example), and a relative error weighting function $W(\omega)$, find the filter coefficients minimizing the maximum error

$$\arg\min_{h} \arg\max_{\omega \in \Phi} |E(\omega)| = \arg\min_{h} \|E(\omega)\|_{\infty}$$

where

$$E(\omega) = W(\omega) (H_d(\omega) - H(\omega))$$

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
and $F$ is a compact subset of $\omega \in [0, \pi]$ (i.e., all $\omega$ in the passbands and stop bands).

**Note:** Typically, we would often rather specify $\| E(\omega) \|_\infty \leq \delta$ and minimize over $M$ and $h$; however, the design techniques minimize $\delta$ for a given $M$. One then repeats the design procedure for different $M$ until the minimum $M$ satisfying the requirements is found.

We will discuss in detail the design only of odd-length symmetric linear-phase FIR filters. Even-length and anti-symmetric linear phase FIR filters are essentially the same except for a slightly different implicit weighting function. For arbitrary phase, exactly optimal design procedures have only recently been developed (1990).

### 3.11.2 Outline of $L_\infty$ Filter Design

The Parks-McClellan method adopts an indirect method for finding the minimax-optimal filter coefficients.

1. Using results from Approximation Theory, simple conditions for determining whether a given filter is $L_\infty$ (minimax) optimal are found.
2. An iterative method for finding a filter which satisfies these conditions (and which is thus optimal) is developed.

That is, the $L_\infty$ filter design problem is actually solved **indirectly**.

### 3.11.3 Conditions for $L_\infty$ Optimality of a Linear-phase FIR Filter

All conditions are based on Chebyshev’s "Alternation Theorem," a mathematical fact from polynomial approximation theory.

#### 3.11.3.1 Alternation Theorem

Let $F$ be a compact subset on the real axis $x$, and let $P(x)$ be and $L$th-order polynomial

$$P(x) = \sum_{k=0}^{L} a_k x^k$$

Also, let $D(x)$ be a desired function of $x$ that is continuous on $F$, and $W(x)$ a positive, continuous weighting function on $F$. Define the error $E(x)$ on $F$ as

$$E(x) = W(x) (D(x) - P(x))$$

and

$$\| E(x) \|_\infty = \argmax_{x \in F} |E(x)|$$

A necessary and sufficient condition that $P(x)$ is the unique $L$th-order polynomial minimizing $\| E(x) \|_\infty$ is that $E(x)$ exhibits at least $L + 2$ "alternations," that is, there must exist at least $L + 2$ values of $x$, $x_k \in F$, $k = [0, 1, \ldots, L + 1]$, such that $x_0 < x_1 < \cdots < x_{L+2}$ and such that $E(x_k) = -E(x_{k+1}) = \pm (\| E \|_\infty)$

**Exercise 3.11.2**

What does this have to do with linear-phase filter design? (Solution on p. 148.)
3.11.4 Optimality Conditions for Even-length Symmetric Linear-phase Filters

For $M$ even,

$$A(\omega) = \sum_{n=0}^{L} h(L-n) \cos \left( \omega \left( n + \frac{1}{2} \right) \right)$$

where $L = \frac{M}{2} - 1$ Using the trigonometric identity $\cos(\alpha + \beta) = \cos(\alpha - \beta) + 2\cos(\alpha) \cos(\beta)$ to pull out the $\frac{\omega}{2}$ term and then using the other trig identities (p. 148), it can be shown that $A(\omega)$ can be written as

$$A(\omega) = \cos\left( \frac{\omega}{2} \right) \sum_{k=0}^{L} \alpha_k \cos^k(\omega)$$

Again, this is a polynomial in $x = \cos(\omega)$, except for a weighting function out in front.

$$E(\omega) = W(\omega) \left( A_d(\omega) - A(\omega) \right)$$

$$= W(\omega) \left( A_d(\omega) - \cos\left( \frac{\omega}{2} \right) P(\omega) \right)$$

$$= W(\omega) \cos\left( \frac{\omega}{2} \right) \left( \frac{A_d(\omega)}{\cos\left( \frac{\omega}{2} \right)} - P(\omega) \right)$$

which implies

$$E(x) = W'(x) \left( A_d'(x) - P(x) \right)$$

where

$$W'(x) = W \left( (\cos(x))^{-1} \right) \cos\left( \frac{1}{2} (\cos(x))^{-1} \right)$$

and

$$A_d'(x) = \frac{A_d \left( (\cos(x))^{-1} \right)}{\cos\left( \frac{1}{2} (\cos(x))^{-1} \right)}$$

Again, this is a polynomial approximation problem, so the alternation theorem holds. If $E(\omega)$ has at least $L + 2 = \frac{M}{2} + 1$ alternations, the even-length symmetric filter is optimal in an $L^\infty$ sense.

The prototypical filter design problem:

$$W = \begin{cases} 
1 & \text{if } |\omega| \leq \omega_p \\
\frac{\omega}{\delta_p} & \text{if } |\omega_s| \leq |\omega| \end{cases}$$

See Figure 3.17.
3.11.5 \( L_{\infty} \) Optimal Lowpass Filter Design Lemma

1. The maximum possible number of alternations for a lowpass filter is \( L + 3 \). The proof is that the extrema of a polynomial occur only where the derivative is zero: \( \frac{\partial P(x)}{\partial x} = 0 \). Since \( P'(x) \) is an \( (L - 1) \)th-order polynomial, it can have at most \( L - 1 \) zeros. **However**, the mapping \( x = \cos(\omega) \) implies that \( \frac{\partial A(\omega)}{\partial \omega} = 0 \) at \( \omega = 0 \) and \( \omega = \pi \), for two more possible alternation points. **Finally**, the band edges can also be alternations, for a total of \( L - 1 + 2 + 2 = L + 3 \) possible alternations.

2. There must be an alternation at either \( \omega = 0 \) or \( \omega = \pi \).

3. Alternations must occur at \( \omega_p \) and \( \omega_s \). See Figure 3.17.

4. The filter must be equiripple except at possibly \( \omega = 0 \) or \( \omega = \pi \). Again see Figure 3.17.

**Note:** The alternation theorem doesn’t directly suggest a method for computing the optimal filter. It simply tells us how to recognize that a filter is optimal, or isn’t optimal. What we need is an intelligent way of guessing the optimal filter coefficients.
In matrix form, these $L + 2$ simultaneous equations become

$$
\begin{pmatrix}
1 & \cos (\omega_0) & \cos (2\omega_0) & \cdots & \cos (L\omega_0) & \frac{1}{W(\omega_0)}
1 & \cos (\omega_1) & \cos (2\omega_1) & \cdots & \cos (L\omega_1) & \frac{-1}{W(\omega_1)}
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots
1 & \cos (\omega_{L+1}) & \cos (2\omega_{L+1}) & \cdots & \cos (L\omega_{L+1}) & \frac{\pm (1)}{W(\omega_{L+1})}
\end{pmatrix}
\begin{pmatrix}
h (L)
1 & h (L - 1) & \cdots 
h (1) & \cdots 
h (0) & \cdots 
\delta & \cdots 
\end{pmatrix}
= 
\begin{pmatrix}
A_d (\omega_0)
A_d (\omega_1)
\vdots
A_d (\omega_{L+1})
\end{pmatrix}
$$
or

$$
W \begin{pmatrix}
h 
\delta
\end{pmatrix}
= A_d
$$

So, for the given set of $L + 2$ extremal frequencies, we can solve for $h$ and $\delta$ via $(h, \delta)^T = W^{-1}A_d$. Using the FFT, we can compute $A (\omega)$ of $h (n)$, on a dense set of frequencies. If the old $\omega_k$ are, in fact the extremal locations of $A (\omega)$, then the alternation theorem is satisfied and $h (n)$ is optimal. If not, repeat the process with the new extremal locations.

### 3.11.6 Computational Cost

$O (L^3)$ for the matrix inverse and $N \log_2 N$ for the FFT ($N \geq 32L$, typically), per iteration!

This method is expensive computationally due to the matrix inverse.

A more efficient variation of this method was developed by Parks and McClellan (1972), and is based on the Remez exchange algorithm. To understand the Remez exchange algorithm, we first need to understand Lagrange Interpolation.

Now $A (\omega)$ is an $L$th-order polynomial in $x = \cos (\omega)$, so Lagrange interpolation can be used to exactly compute $A (\omega)$ from $L + 1$ samples of $A (\omega_k)$, $k = [0, 1, 2, \ldots, L]$.

Thus, given a set of extremal frequencies and knowing $\delta$, samples of the amplitude response $A (\omega)$ can be computed directly from the

$$
A (\omega_k) = \frac{(-1)^k}{W(\omega_k)} \delta + A_d (\omega_k)
$$

without solving for the filter coefficients!

This leads to computational savings!

Note that (3.61) is a set of $L + 2$ simultaneous equations, which can be solved for $\delta$ to obtain (Rabiner, 1975)

$$
\delta = \frac{\sum_{k=0}^{L+1} \gamma_k A_d (\omega_k)}{\sum_{k=0}^{L+1} \frac{\pm (1)}{W(\omega_k)}}
$$

where

$$
\gamma_k = \prod_{i=0}^{L+1} \frac{1}{\cos (\omega_k) - \cos (\omega_i)}
$$

The result is the Parks-McClellan FIR filter design method, which is simply an application of the Remez exchange algorithm to the filter design problem. See Figure 3.18.
Figure 3.18: The initial guess of extremal frequencies is usually equally spaced in the band. Computing $\delta$ costs $O(L^2)$. Using Lagrange interpolation costs $O(16LL) \simeq O(16L^2)$. Computing $h(n)$ costs $O(L^3)$, but it is only done once!
The cost per iteration is $O(16L^2)$, as opposed to $O(L^3)$; much more efficient for large $L$. Can also interpolate to DFT sample frequencies, take inverse FFT to get corresponding filter coefficients, and zeropad and take longer FFT to efficiently interpolate.

3.12 FIR Filter Design using MATLAB

3.12.1 FIR Filter Design Using MATLAB

3.12.1.1 Design by windowing

The MATLAB function `fir1()` designs conventional lowpass, highpass, bandpass, and bandstop linear-phase FIR filters based on the windowing method. The command

\[
b = \text{fir1}(N, Wn)\]

returns in vector $b$ the impulse response of a lowpass filter of order $N$. The cut-off frequency $Wn$ must be between 0 and 1 with 1 corresponding to the half sampling rate.

The command

\[
b = \text{fir1}(N, Wn, 'high')\]

returns the impulse response of a highpass filter of order $N$ with normalized cutoff frequency $Wn$.

Similarly, $b = \text{fir1}(N, Wn, 'stop')$ with $Wn$ a two-element vector designating the stopband designs a bandstop filter.

Without explicit specification, the Hamming window is employed in the design. Other windowing functions can be used by specifying the windowing function as an extra argument of the function. For example, Blackman window can be used instead by the command $b = \text{fir1}(N, Wn, \text{blackman}(N))$.

3.12.1.2 Parks-McClellan FIR filter design

The MATLAB command

\[
b = \text{remez}(N, F, A)\]

returns the impulse response of the length $N+1$ linear phase FIR filter of order $N$ designed by Parks-McClellan algorithm. $F$ is a vector of frequency band edges in ascending order between 0 and 1 with 1 corresponding to the half sampling rate. $A$ is a real vector of the same size as $F$ which specifies the desired amplitude of the frequency response of the points $(F(k), A(k))$ and $(F(k+1), A(k+1))$ for odd $k$. For odd $k$, the bands between $F(k+1)$ and $F(k+2)$ is considered as transition bands.

\[\text{26This content is available online at } \langle \text{http://cnx.org/content/m10917/2.2/} \rangle.\]
3.13 MATLAB FIR Filter Design Exercise

3.13.1 FIR Filter Design MATLAB Exercise

3.13.1.1 Design by windowing

Exercise 3.13.1 (Solution on p. 148.)
Assuming sampling rate at 48kHz, design an order-40 low-pass filter having cut-off frequency 10kHz by windowing method. In your design, use Hamming window as the windowing function.

3.13.1.2 Parks-McClellan Optimal Design

Exercise 3.13.2 (Solution on p. 148.)
Assuming sampling rate at 48kHz, design an order-40 lowpass filter having transition band 10kHz-11kHz using the Parks-McClellan optimal FIR filter design algorithm.

---

27This content is available online at http://cnx.org/content/m10918/2.2/.

Available for free at Connexions http://cnx.org/content/col10360/1.4/
Solutions to Exercises in Chapter 3

Solution to Exercise 3.5.1 (p. 126)
The DTFT of the unit sample equals a constant (equaling 1). Thus, the Fourier transform of the output equals the transfer function.

Solution to Exercise 3.5.2 (p. 126)
In sampling a discrete-time signal’s Fourier transform $L$ times equally over $[0, 2\pi)$ to form the DFT, the corresponding signal equals the periodic repetition of the original signal.

$$S(k) \leftrightarrow \sum_{i=-\infty}^{\infty} s(n - iL) \quad (3.63)$$

To avoid aliasing (in the time domain), the transform length must equal or exceed the signal’s duration.

Solution to Exercise 3.5.3 (p. 127)
The difference equation for an FIR filter has the form

$$y(n) = \sum_{m=0}^{q} b_m x(n - m) \quad (3.64)$$

The unit-sample response equals

$$h(n) = \sum_{m=0}^{q} b_m \delta(n - m) \quad (3.65)$$

which corresponds to the representation described in a problem28 of a length-$q$ boxcar filter.

Solution to Exercise 3.5.4 (p. 127)
The unit-sample response’s duration is $q + 1$ and the signal’s $N_x$. Thus the statement is correct.

Solution to Exercise 3.6.1 (p. 132)

- $\theta(\omega) = \text{constant}$
- $\theta(\omega) = K\omega$
- The phase is linear.

Solution to Exercise 3.9.1 (p. 135)
Yes; in fact it’s optimal! (in a certain sense)

Solution to Exercise 3.9.2 (p. 136): Gibbs Phenomenon

![Figure 3.19: (a) $A(\omega)$, small $M$ (b) $A(\omega)$, large $M$](http://cnx.org/content/m10251/latest/#ex2001)

---


Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Solution to Exercise 3.10.1 (p. 136)
\[
\begin{cases}
  e^{-(i\omega \frac{M-1}{2})} & \text{if } -\omega_c \leq \omega \leq \omega_c \\
  0 & \text{if } (-\pi \leq \omega < -\omega_c) \lor (\omega_c < \omega \leq \pi)
\end{cases}
\]

Solution to Exercise 3.11.1 (p. 139)
Yes, when the desired response is discontinuous. Since the frequency response of a finite-length filter must be continuous, without a transition band the worst-case error could be no less than half the discontinuity.

Solution to Exercise 3.11.2 (p. 140)
It’s the same problem! To show that, consider an odd-length, symmetric linear phase filter.

\[
H(\omega) = \sum_{n=0}^{M-1} h(n) e^{-(i\omega n)} = \left( e^{-(i\omega \frac{M-1}{2})} \left( h \left( \frac{M-1}{2} \right) + 2 \sum_{n=1}^{L} h \left( \frac{M-1}{2} - n \right) \cos(\omega n) \right) \right)
\]

\[
A(\omega) = h(L) + 2 \sum_{n=1}^{L} h(L - n) \cos(\omega n)
\]

Where \( L = \frac{M-1}{2} \).

Using trigonometric identities (such as \( \cos(n\alpha) = 2\cos((n-1)\alpha) \cos(\alpha) - \cos((n-2)\alpha) \)), we can rewrite \( A(\omega) \) as

\[
A(\omega) = h(L) + 2 \sum_{n=1}^{L} h(L - n) \cos(\omega n) = \sum_{k=0}^{L} \alpha_k \cos^k(\omega)
\]

where the \( \alpha_k \) are related to the \( h(n) \) by a linear transformation. Now, let \( x = \cos(\omega) \). This is a one-to-one mapping from \( x \in [-1, 1] \) onto \( \omega \in [0, \pi] \). Thus \( A(\omega) \) is an \( L \)-th order polynomial in \( x = \cos(\omega) \)!

**NOTE:** The alternation theorem holds for the \( L^\infty \) filter design problem, too!

Therefore, to determine whether or not a length-\( M \), odd-length, symmetric linear-phase filter is optimal in an \( L^\infty \) sense, simply count the alternations in \( E(\omega) = W(\omega)(A_d(\omega) - A(\omega)) \) in the pass and stop bands. If there are \( L + 2 = \frac{M+3}{2} \) or more alternations, \( h(n), 0 \leq n \leq M - 1 \) is the optimal filter!

Solution to Exercise 3.13.1 (p. 146)

\[
b = \text{fir1}(40, 10.0/48.0)
\]

Solution to Exercise 3.13.2 (p. 146)

\[
b = \text{remez}(40, [1 1 0 0], [0 10/48 11/48 1])
\]

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Chapter 4

Multirate Signal Processing

4.1 Upsampling

4.1.1 Upsampling

The operation of upsampling by factor $L \in \mathbb{N}$ describes the insertion of $L - 1$ zeros between every sample of the input signal. This is denoted by "↑ $(L)$" in block diagrams, as in Figure 4.1.

Formally, upsampling can be expressed in the time domain as

$$y[n] = \begin{cases} x \left[ \frac{n}{L} \right] & \text{if } \frac{n}{L} \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}$$

In the $z$-domain,

$$Y(z) = \sum_{n} y[n] z^{-n} = \sum_{n, \frac{n}{L} \in \mathbb{Z}} x \left[ \frac{n}{L} \right] z^{-n} = \sum_{k} x[k] z^{(-k)L} = X(z^L)$$

and substituting $z = e^{j\omega}$ for the DTFT,

$$Y(e^{j\omega}) = X(e^{j\omega L}) \quad (4.1)$$

As shown in Figure 4.2, upsampling compresses the DTFT by a factor of $L$ along with the $\omega$ axis.
4.2 Downsampling

The operation of downsampling by factor $M \in \mathbb{N}$ describes the process of keeping every $M^{th}$ sample and discarding the rest. This is denoted by "$\downarrow (M)^{th}$" in block diagrams, as in Figure 4.3.

Formally, downsampling can be written as

$$y[n] = x[nM]$$

In the $z$ domain,

$$Y(z) = \sum_n y[n] z^{-n}$$

$$= \sum_n x[nM] z^{-n}$$

$$= \sum_m x[m] \left( \frac{1}{M} \sum_{p=0}^{M-1} e^{i \frac{2\pi}{M} pm} \right) z^{-m}$$

(4.2)

---

2This content is available online at $<$http://cnx.org/content/m10441/2.12/$>$.

Available for free at Connexions $<$http://cnx.org/content/col10360/1.4$>$.
where \(\frac{1}{M} \sum_{p=0}^{M-1} e^{\frac{2\pi}{M}pm} = \begin{cases} 1 & \text{if } m \text{ is a multiple of } M \\ 0 & \text{otherwise} \end{cases} \)

\[
Y(z) = \frac{1}{M} \sum_{p=0}^{M-1} \sum_m x[m] \left(e^{-\left(i\frac{2\pi}{M}p\right)z\frac{1}{M}}\right)^m
= \frac{1}{M} \sum_{p=0}^{M-1} X \left(e^{-\left(i\frac{2\pi}{M}p\right)z\frac{1}{M}}\right)
\] (4.3)

Translating to the frequency domain,

\[
Y(e^{i\omega}) = \frac{1}{M} \sum_{p=0}^{M-1} X \left(e^{\frac{2\pi}{M}p}e^{i\omega}\right) \tag{4.4}
\]

As shown in Figure 4.4, downsampling expands each \(2\pi\)-periodic repetition of \(X(e^{i\omega})\) by a factor of \(M\) along the \(\omega\) axis, and reduces the gain by a factor of \(M\). If \(x[m]\) is not bandlimited to \(\frac{\pi}{M}\), aliasing may result from spectral overlap.

**NOTE**: When performing a frequency-domain analysis of systems with up/downsamplers, it is strongly recommended to carry out the analysis in the \(z\)-domain until the last step, as done above. Working directly in the \(e^{i\omega}\)-domain can easily lead to errors.

---

\[x[m] \quad X(e^{i\omega}) \quad \frac{X(e^{i\omega})}{M} \quad Y(e^{i\omega})\]

\[y[n] \quad n \quad -4\pi \quad -2\pi \quad 2\pi \quad 4\pi \]

**Figure 4.4**

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
4.3 Interpolation

4.3.1 Interpolation

Interpolation is the process of upsampling and filtering a signal to increase its effective sampling rate. To be more specific, say that $x[m]$ is an (unaliased) $T$-sampled version of $x_c(t)$ and $v[n]$ is an $L$-upsampled version of $x[m]$. If we filter $v[n]$ with an ideal $\frac{\pi}{L}$-bandwidth lowpass filter (with DC gain $L$) to obtain $y[n]$, then $y[n]$ will be a $\frac{T}{L}$-sampled version of $x_c(t)$. This process is illustrated in Figure 4.5.

\begin{align*}
X(e^{j\omega}) &= \frac{1}{T} \sum_k X_c \left( i \frac{\omega - 2\pi k}{T} \right) \quad (4.5) \\
V(e^{j\omega}) &= \frac{1}{T} \sum_k X_c \left( i \frac{\omega L - 2\pi k}{T} \right) = \frac{1}{T} \sum_k X_c \left( i \frac{\omega - 2\pi k}{L} \right) \\
Y(e^{j\omega}) &= \frac{L}{T} \sum_{k \in \mathbb{Z}} X_c \left( i \frac{\omega - 2\pi k}{T} \right) = \frac{L}{T} \sum_l X_c \left( i \frac{\omega - 2\pi l}{L} \right)
\end{align*}

Lowpass filtering with cutoff $\frac{\pi}{L}$ and gain $L$ yields

since the spectral copies with indices other than $k = lL$ (for $l \in \mathbb{Z}$) are removed. Clearly, this process yields a $\frac{T}{L}$-shaped version of $x_c(t)$. Figure 4.6 illustrates these frequency-domain arguments for $L = 2$.

---

3This content is available online at <http://cnx.org/content/m10444/2.14/>.
4.4 Application of Interpolation - Oversampling in CD Players

4.4.1 Application of Interpolation - Oversampling in CD Players

The digital audio signal on a CD is a 44.1kHz sampled representation of a continuous signal with bandwidth 20kHz. With a standard ZOH-DAC, the analog reconstruction filter would have passband edge at 20kHz and stopband edge at 24.1kHz. (See Figure 4.7) With such a narrow transition band, this would be a difficult (and expensive) filter to build.

If digital interpolation is used prior to reconstruction, the effective sampling rate can be increased and the reconstruction filter’s transition band can be made much wider, resulting in a much simpler (and cheaper) analog filter. Figure 4.8 illustrates the case of interpolation by 4. The reconstruction filter has passband edge at 20kHz and stopband edge at 156.4kHz, resulting in a much wider transition band and therefore an easier filter design.

---

4This content is available online at <http://cnx.org/content/m11006/2.3/>. Available for free at Connexions <http://cnx.org/content/col10360/1.4>
4.5 Decimation\footnote{This content is available online at \texttt{http://cnx.org/content/m10445/2.11/}.}

Decimation is the process of filtering and downsampling a signal to decrease its effective sampling rate, as illustrated in Figure 4.9. The filtering is employed to prevent aliasing that might otherwise result from downsampling.

To be more specific, say that
\[ x_c(t) = x_l(t) + x_b(t) \]
where \( x_l(t) \) is a lowpass component bandlimited to \( \frac{1}{2MT} \) Hz and \( x_b(t) \) is a bandpass component with energy between \( \frac{1}{2MT} \) and \( \frac{1}{2T} \) Hz. If sampling \( x_c(t) \) with interval \( T \) yields an unaliased discrete representation \( x[m] \), then decimating \( x[m] \) by a factor \( M \) will yield \( y[n] \), an unaliased \( MT \)-sampled representation of lowpass component \( x_l(t) \).

We offer the following justification of the previously described decimation procedure. From the sampling theorem, we have
\[
X(e^{j\omega}) = \frac{1}{T} \sum_k X_l \left( \frac{\omega - 2\pi k}{T} \right) + \frac{1}{T} \sum_k X_b \left( \frac{\omega - 2\pi k}{T} \right)
\]
The bandpass component $X_b(i\Omega)$ is the removed by $\frac{\pi}{M}$-lowpass filtering, giving

$$V(e^{i\omega}) = \frac{1}{T} \sum_k X_l \left( i\omega - \frac{2\pi k}{T} \right)$$

Finally, downsampling yields

$$Y(e^{i\omega}) = \frac{1}{MT} \sum_{p=0}^{M-1} \sum_k X_l \left( i\omega - \frac{2\pi (pM + k)}{M} \right)$$

which is clearly a $MT$-sampled version of $x_l(t)$. A frequency-domain illustration for $M = 2$ appears in Figure 4.10.

4.6 Resampling with Rational Factor\(^6\)

Interpolation by $L$ and decimation by $M$ can be combined to change the effective sampling rate of a signal by the rational factor $\frac{L}{M}$. This process is called resampling or sample-rate conversion. Rather than cascading an anti-imaging filter for interpolation with an anti-aliasing filter for decimation, we implement one filter with the minimum of the two cutoffs $\left[ \frac{\pi}{T}, \frac{\pi}{M} \right]$ and the multiplication of the two DC gains ($L$ and 1), as illustrated in Figure 4.11.

\(^6\)This content is available online at <http://cnx.org/content/m10448/2.11/>.
4.7 Digital Filter Design for Interpolation and Decimation\footnote{This content is available online at \texttt{<http://cnx.org/content/m10870/2.6/>}.}

First we treat filter design for interpolation. Consider an input signal $x[n]$ that is $\omega_0$-bandlimited in the DTFT domain. If we upsample by factor $L$ to get $v[m]$, the desired portion of $V(e^{j\omega})$ is the spectrum in $\left[\frac{-\pi}{L}, \frac{\pi}{L}\right)$, while the undesired portion is the remainder of $[-\pi, \pi)$. Noting from Figure 4.12 that $V(e^{j\omega})$ has zero energy in the regions

$$\left[\frac{2k\pi + \omega_0}{L}, \frac{2(k + 1)\pi - \omega_0}{L}\right), \quad k \in \mathbb{Z}$$

\begin{equation}
\text{(4.7)}
\end{equation}

the anti-imaging filter can be designed with transition bands in these regions (rather than passbands or stopbands). For a given number of taps, the additional degrees of freedom offered by these transition bands allows for better responses in the passbands and stopbands. The resulting filter design specifications are shown in the bottom subplot below (Figure 4.12).
Next we treat filter design for decimation. Say that the desired spectral component of the input signal is bandlimited to \( \frac{\omega_0}{M} < \frac{\pi}{M} \) and we have decided to downsample by \( M \). The goal is to minimally distort the input spectrum over \( [\frac{-\omega_0}{M}, \frac{\pi}{M}] \), i.e., the post-decimation spectrum over \( [-\omega_0, \omega_0] \). Thus, we must not allow any aliased signals to enter \( [-\omega_0, \omega_0] \). To allow for extra degrees of freedom in the filter design, we do allow aliasing to enter the post-decimation spectrum outside of \( [-\omega_0, \omega_0] \) within \( [-\pi, \pi] \). Since the input spectral regions which alias outside of \( [-\omega_0, \omega_0] \) are given by

\[
\left\{ \frac{2k\pi + \omega_0}{L}, \frac{2(k+1)\pi - \omega_0}{L} \right\}, \quad k \in \mathbb{Z}
\]  

(as shown in Figure 4.13), we can treat these regions as transition bands in the filter design. The resulting filter design specifications are illustrated in the middle subplot (Figure 4.13).
4.8 Noble Identities

4.8.1

The Noble identities (illustrated in Figure 4.14 and Figure 4.15) describe when it is possible to reverse the order of upsampling/downsampling and filtering. We prove the Noble identities showing the equivalence of each pair of block diagrams.

The Noble identity for interpolation can be depicted as in Figure 4.14:

For the left side of the diagram, we have

\[ Y(z) = H\left( z^{-L} \right) V_1(z) \]

---

*This content is available online at [http://cnx.org/content/m10432/2.12/].

Available for free at Connexions [http://cnx.org/content/col10360/1.4].
where \( V_1(z) = X(z^L) \)

\[
Y(z) = H(z^L) X(z^L)
\]

while for the right side,

\[
Y(z) = V_2(z)
\]

where \( V_2(z) = H(z) X(z) \)

\[
Y(z) = H(z^L) X(z^L)
\]

Thus we have established the Noble identity for interpolation.

The Noble identity for decimation can be depicted as in Figure 4.15:

For the left side of the preceding diagram, we have

\[
V_1(z) = \frac{1}{M} \sum_{k=0}^{M-1} X \left( e^{-i \frac{2\pi}{M} k z^L} \right)
\]

\[
Y(z) = H(z) V_1(z) = H(z) \left( \frac{1}{M} \sum_{k=0}^{M-1} X \left( e^{-i \frac{2\pi}{M} k z^L} \right) \right)
\]

(4.9)

while for the right side,

\[
Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} V_z \left( e^{-i \frac{2\pi}{M} k z^L} \right)
\]

(4.10)

where \( V_2(z) = X(z) H(z^M) \)

\[
Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} X \left( e^{-i \frac{2\pi}{M} k z^L} \right) H \left( e^{-i \frac{2\pi}{M} k M^L z^M} \right)
\]

\[
= H(z) \frac{1}{M} \sum_{k=0}^{M-1} X \left( e^{-i \frac{2\pi}{M} k^M z^M} \right)
\]

(4.11)

Thus we have established the Noble identity for decimation. Note that the impulse response of \( H(z^L) \) is the \( L \)-upsampled impulse response of \( H(z) \).

4.9 Polyphase Interpolation

4.9.1 Polyphase Interpolation Filter

Recall the standard interpolation procedure illustrated in Figure 4.16.

\[\text{This content is available online at } \langle\text{http://cnx.org/content/m10431/2.11/}\rangle.\]
Note that this procedure is computationally inefficient because the lowpass filter operates on a sequence that is mostly composed of zeros. Through the use of the Noble identities, it is possible to rearrange the preceding block diagram so that operations on zero-valued samples are avoided.

In order to apply the Noble identity for interpolation, we must transform \( H(z) \) into its upsampled polyphase components \( H_p(z^L), p = \{0, \ldots, L - 1\} \).

\[
H(z) = \sum_n h[n] z^{-n} = \sum_{kk} \sum_{p=0}^{L-1} h[kL + p] z^{-(kL+p)}
\] (4.12)

via \( k := \lfloor \frac{n}{L} \rfloor, p := n \mod L \)

\[
H(z) = \sum_{p=0}^{L-1} \sum_{k} h_p[k] z^{-(kL)} z^{-p}
\] (4.13)

via \( h_p[k] := h[kL + p] \)

\[
H(z) = \sum_{p=0}^{L-1} H_p(z^L) z^{-p}
\] (4.14)

Above, \( \lfloor \cdot \rfloor \) denotes the floor operator and \( \cdot \mod M \) the modulo-\( M \) operator. Note that the \( p^{th} \) polyphase filter \( h_p[k] \) is constructed by downsampling the "master filter" \( h[n] \) at offset \( p \). Using the unsampled polyphase components, the Figure 4.16 diagram can be redrawn as in Figure 4.17.
Applying the Noble identity for interpolation to Figure 4.18 yields Figure 4.17. The ladder of upsamplers and delays on the right below (Figure 4.17) accomplishes a form of parallel-to-serial conversion.
4.10 Polyphase Decimation Filter

4.10.1 Polyphase Decimation

Recall the standard decimation method in Figure 4.19.

Note that this procedure is computationally inefficient because it discards the majority of the computed filter outputs. Through the use of the Noble identities, it is possible to rearrange Figure 4.19 so that filter outputs are not discarded.

---

10This content is available online at <http://cnx.org/content/m10433/2.12/>.
In order to apply the Noble identity for decimation, we must transform $H(z)$ into its upsampled polyphase components $H_p(z^M)$, $p = \{0, \ldots, M-1\}$, defined previously in the context of polyphase interpolation (Section 4.9).

\[
H(z) = \sum_{n} h[n] z^{-n} = \sum_{k} \sum_{p=0}^{M-1} h[kM + p] z^{-(kM)-p}
\]

via $k := \lfloor \frac{n}{M} \rfloor$, $p := n \mod M$

\[
H(z) = \sum_{p=0}^{M-1} \sum_{k} h_p[k] z^{-(kM)} z^{-p}
\]

via $h_p[k] := h[kM + p]$

\[
H(z) = \sum_{p=0}^{M-1} H_p(z^M) z^{-p}
\]

Using these unsampled polyphase components, the preceding block diagram can be redrawn as Figure 4.20.

![Figure 4.20](http://cnx.org/content/col10360/1.4)

Applying the Noble identity for decimation to Figure 4.20 yields Figure 4.21. The ladder of delays and downsamplers on the left below accomplishes a form of serial-to-parallel conversion.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
4.11 Computational Savings of Polyphase Interpolation/Decimation

4.11.1 Computational Savings of Polyphase Interpolation/Decimation

Assume that we design FIR LPF \( H(z) \) with \( N \) taps, requiring \( N \) multiplies per output. For standard decimation by factor \( M \), we have \( N \) multiplies per intermediate sample and \( M \) intermediate samples per output, giving \( NM \) multiplies per output.

For polyphase decimation, we have \( \frac{N}{M} \) multiplies per branch and \( M \) branches, giving a total of \( N \) multiplies per output. The assumption of \( \frac{N}{M} \) multiplies per branch follows from the fact that \( h[n] \) is downsampled by \( M \) to create each polyphase filter. Thus, we conclude that the standard implementation requires \( M \) times as many operations as its polyphase counterpart. (For decimation, we count multiplies per output, rather than per input, to avoid confusion, since only every \( M \)th input produces an output.)

From this result, it appears that the number of multiplications required by polyphase decimation is independent of the decimation rate \( M \). However, it should be remembered that the length \( N \) of the \( \frac{\pi}{MF} \)-lowpass FIR filter \( H(z) \) will typically be proportional to \( M \). This is suggested, e.g., by the Kaiser FIR-length approximation formula

\[
N \approx \frac{-10 \log (\delta_p \delta_s) - 13}{2.324 \Delta (\omega)}
\]

where \( \Delta (\omega) \) is the transition bandwidth in radians, and \( \delta_p \) and \( \delta_s \) are the passband and stopband ripple levels. Recall that, to preserve a fixed signal bandwidth, the transition bandwidth \( \Delta (\omega) \) will be linearly proportional

\[\text{11This content is available online at <http://cnx.org/content/m11008/2.2/>}.\]

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
to the cutoff $\frac{\pi}{M}$, so that $N$ will be linearly proportional to $M$. In summary, polyphase decimation by factor $M$ requires $N$ multiplies per output, where $N$ is the filter length, and where $N$ is linearly proportional to $M$.

Using similar arguments for polyphase interpolation, we could find essentially the same result. Polyphase interpolation by factor $L$ requires $N$ multiplies per input, where $N$ is the filter length, and where $N$ is linearly proportional to the interpolation factor $L$. (For interpolation we count multiplies per input, rather than per output, to avoid confusion, since $M$ outputs are generated in parallel.)

4.12 Sub-Band Processing\(^{12}\)

4.12.1 Why Filterbanks?

4.12.1.1 Sub-band Processing

There exist many applications in modern signal processing where it is advantageous to separate a signal into different frequency ranges called sub-bands. The spectrum might be partitioned in the uniform manner illustrated in Figure 4.22, where the sub-band width $\Delta_k = \frac{2\pi}{M}$ is identical for each sub-band and the band centers are uniformly spaced at intervals of $\frac{2\pi}{M}$.

![Figure 4.22](http://cnx.org/content/m10423/2.14/)

Alternatively, the sub-bands might have a logarithmic spacing like that shown in Figure 4.23.

\(^{12}\)This content is available online at [http://cnx.org/content/m10423/2.14/].
For most of our discussion, we will focus on uniformly spaced sub-bands.

The separation into sub-band components is intended to make further processing more convenient. Some of the most popular applications for sub-band decomposition are audio and video source coding (with the goal of efficient storage and/or transmission).

Figure 4.24 illustrates the use of sub-band processing in MPEG audio coding. There a psychoacoustic model is used to decide how much quantization error can be tolerated in each sub-band while remaining below the hearing threshold of a human listener. In the sub-bands that can tolerate more error, less bits are used for coding. The quantized sub-band signals can then be decoded and recombined to reconstruct (an approximate version of) the input signal. Such processing allows, on average, a 12-to-1 reduction in bit rate while still maintaining "CD quality" audio. The psychoacoustic model takes into account the spectral masking phenomenon of the human ear, which says that high energy in one spectral region will limit the ear's ability to hear details in nearby spectral regions. Therefore, when the energy in one sub-band is high, nearby sub-bands can be coded with less bits without degrading the perceived quality of the audio signal. The MPEG standard specifies 32-channels of sub-band filtering. Some psychoacoustic models also take into account "temporal masking" properties of the human ear, which say that a loud burst of sound will temporarily overload the ear for short time durations, making it possible to hide quantization noise in the time interval after a loud sound burst.

In typical applications, non-trivial signal processing takes place between the bank of analysis filters and the bank of synthesis filters, as shown in Figure 4.25. We will focus, however, on filterbank design rather than on the processing that occurs between the filterbanks.
Our goals in filter design are:

1. Good sub-band frequency separation (i.e., good "frequency selectivity").
2. Good reconstruction (i.e., \(y[n] \simeq x[n-d]\) for some integer delay \(d\)) when the sub-band processing is lossless.

The first goal is driven by the assumption that the sub-band processing works best when it is given access to cleanly separated sub-band signals, while the second goal is motivated by the idea that the sub-band filtering should not limit the reconstruction performance when the sub-band processing (e.g., the coding/decoding) is lossless or nearly lossless.

### 4.13 Discrete Wavelet Transform: Main Concepts

#### 4.13.1 Main Concepts

The **discrete wavelet transform** (DWT) is a representation of a signal \(x(t) \in L_2\) using an orthonormal basis consisting of a countably-infinite set of **wavelets**. Denoting the wavelet basis as \(\{\psi_{k,n}(t) \mid k \in \mathbb{Z} \land n \in \mathbb{Z}\}\), the DWT transform pair is

\[
x(t) = \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} d_{k,n} \psi_{k,n}(t)
\]

\[
d_{k,n} = \langle \psi_{k,n}(t), x(t) \rangle = \int_{-\infty}^{\infty} \psi_{k,n}(t)x(t) \, dt
\]

where \(\{d_{k,n}\}\) are the wavelet coefficients. Note the relationship to Fourier series and to the sampling theorem: in both cases we can perfectly describe a continuous-time signal \(x(t)\) using a countably-infinite (i.e., discrete) set of coefficients. Specifically, Fourier series enabled us to describe **periodic** signals using Fourier coefficients \(\{X[k] \mid k \in \mathbb{Z}\}\), while the sampling theorem enabled us to describe **bandlimited** signals using signal samples \(\{x[n] \mid n \in \mathbb{Z}\}\). In both cases, signals within a limited class are represented using a coefficient set with a single countable index. The DWT can describe **any** signal in \(L_2\) using a coefficient set parameterized by two countable indices: \(\{d_{k,n} \mid k \in \mathbb{Z} \land n \in \mathbb{Z}\}\).

---

13This content is available online at <http://cnx.org/content/m10436/2.12/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Wavelets are orthonormal functions in $L_2$ obtained by shifting and stretching a mother wavelet, $\psi(t) \in L_2$. For example,

$$\forall k, n, k \land n \in \mathbb{Z} : \left( \psi_{k,n}(t) = 2^{-\frac{k}{2}} \psi(2^{-k}t - n) \right)$$

(4.20)

defines a family of wavelets $\{ \psi_{k,n}(t) | k \in \mathbb{Z} \land n \in \mathbb{Z} \}$ related by power-of-two stretches. As $k$ increases, the wavelet stretches by a factor of two; as $n$ increases, the wavelet shifts right.

**NOTE:** When $\| \psi(t) \|= 1$, the normalization ensures that $\| \psi_{k,n}(t) \|= 1$ for all $k \in \mathbb{Z}, n \in \mathbb{Z}$.

Power-of-two stretching is a convenient, though somewhat arbitrary, choice. In our treatment of the discrete wavelet transform, however, we will focus on this choice. Even with power-of-two stretches, there are various possibilities for $\psi(t)$, each giving a different flavor of DWT.

Wavelets are constructed so that $\{ \psi_{k,n}(t) | n \in \mathbb{Z} \}$ (i.e., the set of all shifted wavelets at fixed scale $k$), describes a particular level of 'detail' in the signal. As $k$ becomes smaller (i.e., closer to $-\infty$), the wavelets become more "fine grained" and the level of detail increases. In this way, the DWT can give a multi-resolution description of a signal, very useful in analyzing "real-world" signals. Essentially, the DWT gives us a discrete multi-resolution description of a continuous-time signal in $L_2$.

In the modules that follow, these DWT concepts will be developed "from scratch" using Hilbert space principles. To aid the development, we make use of the so-called scaling function $\phi(t) \in L_2$, which will be used to approximate the signal up to a particular level of detail. Like with wavelets, a family of scaling functions can be constructed via shifts and power-of-two stretches

$$\forall k, n, k \land n \in \mathbb{Z} : \left( \phi_{k,n}(t) = 2^{-\frac{k}{2}} \phi(2^{-k}t - n) \right)$$

(4.21)

given mother scaling function $\phi(t)$. The relationships between wavelets and scaling functions will be elaborated upon later via theory$^{14}$ and example (Section 4.14).

**NOTE:** The inner-product expression for $d_{k,n}, (4.19)$ is written for the general complex-valued case. In our treatment of the discrete wavelet transform, however, we will assume real-valued signals and wavelets. For this reason, we omit the complex conjugations in the remainder of our DWT discussions

### 4.14 The Haar System as an Example of DWT$^{15}$

The Haar basis is perhaps the simplest example of a DWT basis, and we will frequently refer to it in our DWT development. Keep in mind, however, that the Haar basis is only an example; there are many other ways of constructing a DWT decomposition.

For the Haar case, the mother scaling function is defined by (4.22) and Figure 4.26.

$$\phi(t) = \begin{cases} 
1 & \text{if } 0 \leq t < 1 \\
0 & \text{otherwise}
\end{cases}$$

(4.22)

$^{14}$"The Scaling Equation" <http://cnx.org/content/m10476/latest/>  
$^{15}$This content is available online at <http://cnx.org/content/m10437/2.10/>. 

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
From the mother scaling function, we define a family of shifted and stretched scaling functions \( \{ \phi_{k,n}(t) \} \) according to (4.23) and Figure 4.27

\[
\phi_{k,n}(t) = \forall k, n, k \in \mathbb{Z}, n \in \mathbb{Z} : \left( 2^{-\frac{k}{2}} \phi(2^{-k} t - n) \right)
= 2^{-\frac{k}{2}} \phi \left( \frac{1}{2^k} (t - n2^k) \right)
\]  

(4.23)

which are illustrated in Figure 4.28 for various \( k \) and \( n \). (4.23) makes clear the principle that incrementing \( n \) by one shifts the pulse one place to the right. Observe from Figure 4.28 that \( \{ \phi_{k,n}(t) \mid n \in \mathbb{Z} \} \) is orthonormal for each \( k \) (i.e., along each row of figures).
4.15 Filterbanks Interpretation of the Discrete Wavelet Transform\(^{16}\)

Assume that we start with a signal \(x(t) \in L_2\). Denote the best approximation at the 0\(^{th}\) level of coarseness by \(x_0(t)\). (Recall that \(x_0(t)\) is the orthogonal projection of \(x(t)\) onto \(V_0\).) Our goal, for the moment, is to decompose \(x_0(t)\) into scaling coefficients and wavelet coefficients at higher levels. Since \(x_0(t) \in V_0\) and \(V_0 = V_1 \oplus W_1\), there exist coefficients \(\{c_0[n]\}, \{c_1[n]\}\), and \(\{d_1[n]\}\) such that

\[
x_0(t) = \sum_{n} c_0[n] \phi_{0,n}(t) = \sum_{n} c_1[n] \phi_{1,n}(t) + \sum_{n} d_1[n] \psi_{1,n}(t)
\]

\(^{16}\)This content is available online at <http://cnx.org/content/m10474/2.6/>.\)
Using the fact that \( \{ \phi_{1,n}(t) \mid n \in \mathbb{Z} \} \) is an orthonormal basis for \( V_1 \), in conjunction with the scaling equation,

\[
c_1[n] = \langle x_0(t), \phi_{1,n}(t) \rangle = \langle \sum_{m} c_0[m] \phi_{0,m}(t), \phi_{1,n}(t) \rangle = \sum_{m} c_0[m] \langle \phi_{0,m}(t), \phi_{1,n}(t) \rangle \quad (4.25)
\]

where \( \delta [t - \ell - 2n] = \langle \phi(t - m), \phi(t - \ell - 2n) \rangle \). The previous expression (4.25)) indicates that \( \{c_1[n]\} \) results from convolving \( \{c_0[m]\} \) with a time-reversed version of \( h[m] \) then downsampling by factor two (Figure 4.29).

![Figure 4.29](image)

Using the fact that \( \{ \psi_{1,n}(t) \mid n \in \mathbb{Z} \} \) is an orthonormal basis for \( W_1 \), in conjunction with the wavelet scaling equation,

\[
d_1[n] = \langle x_0(t), \psi_{1,n}(t) \rangle = \langle \sum_{m} c_0[m] \phi_{0,m}(t), \psi_{1,n}(t) \rangle = \sum_{m} c_0[m] \langle \phi_{0,m}(t), \psi_{1,n}(t) \rangle \quad (4.26)
\]

where \( \delta [t - \ell - 2n] = \langle \phi(t - m), \phi(t - \ell - 2n) \rangle \).

The previous expression (4.26) indicates that \( \{d_1[n]\} \) results from convolving \( \{c_0[m]\} \) with a time-reversed version of \( g[m] \) then downsampling by factor two (Figure 4.30).

![Figure 4.30](image)
Putting these two operations together, we arrive at what looks like the analysis portion of an FIR filterbank (Figure 4.31):

![Figure 4.31](image)

We can repeat this process at the next higher level. Since $V_1 = W_2 \oplus V_2$, there exist coefficients $\{c_2[n]\}$ and $\{d_2[n]\}$ such that

$$
\begin{align*}
x_1(t) &= \sum_{nn} c_1[n] \phi_{1,n}(t) \\
&= \sum_{nn} d_2[n] \psi_{2,n}(t) + \sum_{nn} c_2[n] \phi_{2,n}(t)
\end{align*}
$$

(4.27)

Using the same steps as before we find that

$$
\begin{align*}
c_2[n] &= \sum_{mm} c_1[m] h[m - 2n] \\
d_2[n] &= \sum_{mm} c_1[m] g[m - 2n]
\end{align*}
$$

(4.28)

(4.29)

which gives a cascaded analysis filterbank (Figure 4.32):
If we use $V_0 = W_1 \oplus W_2 \oplus W_3 \oplus \cdots \oplus W_k \oplus V_k$ to repeat this process up to the $k^{th}$ level, we get the iterated analysis filterbank (Figure 4.33).

As we might expect, signal reconstruction can be accomplished using cascaded two-channel synthesis
filterbanks. Using the same assumptions as before, we have:

\[
\begin{align*}
    c_0[m] &= <x_0(t), \phi_{0,m}(t)> \\
    &= <\sum_{n} c_1[n] \phi_{1,n}(t) + \sum_{n} d_1[n] \psi_{1,n}(t), \phi_{0,m}(t)> \\
    &= \sum_{n} c_1[n] \langle \phi_{1,n}(t), \phi_{0,m}(t) \rangle + \sum_{n} d_1[n] \langle \psi_{1,n}(t), \phi_{0,m}(t) \rangle \\
    &= \sum_{n} c_1[n] h[m-2n] + \sum_{n} d_1[n] g[m-2n]
\end{align*}
\]

where \( h[m-2n] = \langle \phi_{1,n}(t), \phi_{0,m}(t) \rangle \)

and \( g[m-2n] = \langle \psi_{1,n}(t), \phi_{0,m}(t) \rangle \)

which can be implemented using the block diagram in Figure 4.34.

\[ (4.30) \]

\[
\begin{align*}
    c_1[m] &= \sum_{n} c_2[n] h[m-2n] + \sum_{n} d_2[n] g[m-2n] \\
\end{align*}
\]

\[ (4.31) \]

The same procedure can be used to derive

\[
\begin{align*}
    c_1[m] &= \sum_{n} c_2[n] h[m-2n] + \sum_{n} d_2[n] g[m-2n]
\end{align*}
\]

from which we get the diagram in Figure 4.35.
To reconstruct from the $k^{th}$ level, we can use the iterated synthesis filterbank (Figure 4.36).

The table (Table 4.1) makes a direct comparison between wavelets and the two-channel orthogonal PR-FIR filterbanks.

<table>
<thead>
<tr>
<th>Discrete Wavelet Transform</th>
<th>2-Channel Orthogonal PR-FIR Filterbank</th>
</tr>
</thead>
</table>

continued on next page
176

CHAPTER 4. MULTIRATE SIGNAL PROCESSING

<table>
<thead>
<tr>
<th>Analysis-LPF</th>
<th>$H(z^{-1})$</th>
<th>$H_0(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Symmetry</td>
<td>$H(z) H(z^{-1}) + H_0(z) H_0(z^{-1})$</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>$H(-z) H(-z^{-1}) = 2$</td>
<td>+</td>
</tr>
<tr>
<td>Analysis HPF</td>
<td>$G(z^{-1})$</td>
<td>$H_1(z)$</td>
</tr>
<tr>
<td>Spectral Reverse</td>
<td>$\forall P, P$ is odd</td>
<td>$\forall N, N$ is even</td>
</tr>
<tr>
<td></td>
<td>$(G(z) = \pm (z^{-P} H (-z^{-1})))$</td>
<td>$(H_1(z) = \pm (z^{-(N-1)} H_0 (-z^{-1})))$</td>
</tr>
<tr>
<td>Synthesis LPF</td>
<td>$H(z)$</td>
<td>$G_0(z) = 2z^{-(N-1)} H_0(z^{-1})$</td>
</tr>
<tr>
<td>Synthesis HPF</td>
<td>$G(z)$</td>
<td>$G_1(z) = 2z^{-(N-1)} H_1(z^{-1})$</td>
</tr>
</tbody>
</table>

Table 4.1

From the table, we see that the discrete wavelet transform that we have been developing is identical to two-channel orthogonal PR-FIR filterbanks in all but a couple details.

1. Orthogonal PR-FIR filterbanks employ synthesis filters with twice the gain of the analysis filters, whereas in the DWT the gains are equal.
2. Orthogonal PR-FIR filterbanks employ causal filters of length $N$, whereas the DWT filters are not constrained to be causal.

For convenience, however, the wavelet filters $H(z)$ and $G(z)$ are usually chosen to be causal. For both to have even impulse response length $N$, we require that $P = N - 1$.

4.16 DWT Application - De-noising

Say that the DWT for a particular choice of wavelet yields an efficient representation of a particular signal class. In other words, signals in the class are well-described using a few large transform coefficients.

Now consider unstructured noise, which cannot be efficiently represented by any transform, including the DWT. Due to the orthogonality of the DWT, such noise sequences make, on average, equal contributions to all transform coefficients. Any given noise sequence is expected to yield many small-valued transform coefficients.

Together, these two ideas suggest a means of de-noising a signal. Say that we perform a DWT on a signal from our well-matched signal class that has been corrupted by additive noise. We expect that large transform coefficients are composed mostly of signal content, while small transform coefficients should be composed mostly of noise content. Hence, throwing away the transform coefficients whose magnitude is less than some small threshold should improve the signal-to-noise ratio. The de-noising procedure is illustrated in Figure 4.37.

![Figure 4.37](image)

Now we give an example of denoising a step-like waveform using the Haar DWT. In Figure 4.38, the top right subplot shows the noisy signal and the top left shows it DWT coefficients. Note the presence

---

17 This content is available online at <http://cnx.org/content/m11000/2.1/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
of a few large DWT coefficients, expected to contain mostly signal components, as well as the presence of many small-valued coefficients, expected to contain noise. (The bottom left subplot shows the DWT for the original signal before any noise was added, which confirms that all signal energy is contained within a few large coefficients.) If we throw away all DWT coefficients whose magnitude is less than 0.1, we are left with only the large coefficients (shown in the middle left plot) which correspond to the de-noised time-domain signal shown in the middle right plot. The difference between the de-noised signal and the original noiseless signal is shown in the bottom right. Non-zero error results from noise contributions to the large coefficients; there is no way of distinguishing these noise components from signal components.
Figure 4.38
Chapter 5

Statistical and Adaptive Signal Processing

5.1 Introduction to Random Signals and Processes

Before now, you have probably dealt strictly with the theory behind signals and systems, as well as look at some of the basic characteristics of signals and systems. In doing so you have developed an important foundation; however, most electrical engineers do not get to work in this type of fantasy world. In many cases the signals of interest are very complex due to the randomness of the world around them, which leaves them noisy and often corrupted. This often causes the information contained in the signal to be hidden and distorted. For this reason, it is important to understand these random signals and how to recover the necessary information.

5.1.1 Signals: Deterministic vs. Stochastic

For this study of signals and systems, we will divide signals into two groups: those that have a fixed behavior and those that change randomly. As most of you have probably already dealt with the first type, we will focus on introducing you to random signals. Also, note that we will be dealing strictly with discrete-time signals since they are the signals we deal with in DSP and most real-world computations, but these same ideas apply to continuous-time signals.

5.1.1.1 Deterministic Signals

Most introductions to signals and systems deal strictly with deterministic signals. Each value of these signals are fixed and can be determined by a mathematical expression, rule, or table. Because of this, future values of any deterministic signal can be calculated from past values. For this reason, these signals are relatively easy to analyze as they do not change, and we can make accurate assumptions about their past and future behavior.
5.1.1.2 Stochastic Signals

Unlike deterministic signals, stochastic signals, or random signals, are not so nice. Random signals cannot be characterized by a simple, well-defined mathematical equation and their future values cannot be predicted. Rather, we must use probability and statistics to analyze their behavior. Also, because of their randomness, average values (Section 5.3) from a collection of signals are usually studied rather than analyzing one individual signal.

5.1.2 Random Process

As mentioned above, in order to study random signals, we want to look at a collection of these signals rather than just one instance of that signal. This collection of signals is called a random process.

**Definition 5.1: random process**

A family or ensemble of signals that correspond to every possible outcome of a certain signal measurement. Each signal in this collection is referred to as a realization or sample function of the process.

**Example**

As an example of a random process, let us look at the Random Sinusoidal Process below. We use $f[n] = A \sin(\omega n + \phi)$ to represent the sinusoid with a given amplitude and phase. Note that the
phase and amplitude of each sinusoid is based on a random number, thus making this a random process.

A random process is usually denoted by $X(t)$ or $X[n]$, with $x(t)$ or $x[n]$ used to represent an individual signal or waveform from this process.

In many notes and books, you might see the following notation and terms used to describe different types of random processes. For a **discrete random process**, sometimes just called a **random sequence**, $t$ represents time that has a finite number of values. If $t$ can take on any value of time, we have a **continuous random process**. Often times discrete and continuous refer to the amplitude of the process, and process or sequence refer to the nature of the time variable. For this study, we often just use **random process** to refer to a general collection of discrete-time signals, as seen above in Figure 5.3 (Random Sinusoidal Process).
5.2 Stationary and Nonstationary Random Processes

5.2.1 Introduction

From the definition of a random process (Section 5.1), we know that all random processes are composed of random variables, each at its own unique point in time. Because of this, random processes have all the properties of random variables, such as mean, correlation, variances, etc. When dealing with groups of signals or sequences it will be important for us to be able to show whether or not these statistical properties hold true for the entire random process. To do this, the concept of stationary processes has been developed.

The general definition of a stationary process is:

**Definition 5.2: stationary process**

A random process where all of its statistical properties do not vary with time.

Processes whose statistical properties do change are referred to as nonstationary.

Understanding the basic idea of stationarity will help you to be able to follow the more concrete and mathematical definition to follow. Also, we will look at various levels of stationarity used to describe the various types of stationarity characteristics a random process can have.

5.2.2 Distribution and Density Functions

In order to properly define what it means to be stationary from a mathematical standpoint, one needs to be somewhat familiar with the concepts of distribution and density functions. If you can remember your statistics then feel free to skip this section!

Recall that when dealing with a single random variable, the probability distribution function is a simply tool used to identify the probability that our observed random variable will be less than or equal to a given number. More precisely, let \( X \) be our random variable, and let \( x \) be our given value; from this we can define the distribution function as

\[
F_X(x) = Pr [X \leq x]
\]  

(5.1)

This same idea can be applied to instances where we have multiple random variables as well. There may be situations where we want to look at the probability of event \( X \) and \( Y \) both occurring. For example, below is an example of a second-order joint distribution function.

\[
F_{x,y}(x,y) = Pr [X \leq x, Y \leq y]
\]  

(5.2)

While the distribution function provides us with a full view of our variable or processes probability, it is not always the most useful for calculations. Often times we will want to look at its derivative, the probability density function (pdf). We define the the pdf as

\[
f_x(x) = \frac{d}{dx} F_x(x)
\]  

(5.3)

\[
f_x(x) \, dx = Pr [x < X \leq x + dx]
\]  

(5.4)

(5.4) reveals some of the physical significance of the density function. This equations tells us the probability that our random variable falls within a given interval can be approximated by \( f_x(x) \, dx \). From the pdf, we can now use our knowledge of integrals to evaluate probabilities from the above approximation. Again we can also define a joint density function which will include multiple random variables just as was done for the distribution function. The density function is used for a variety of calculations, such as finding the expected value or proving a random variable is stationary, to name a few.

---

4This content is available online at <http://cnx.org/content/m10684/2.2/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
The above examples explain the distribution and density functions in terms of a single random variable, $X$. When we are dealing with signals and random processes, remember that we will have a set of random variables where a different random variable will occur at each time instance of the random process, $X(t_k)$. In other words, the distribution and density function will also need to take into account the choice of time.

5.2.3 Stationarity

Below we will now look at a more in depth and mathematical definition of a stationary process. As was mentioned previously, various levels of stationarity exist and we will look at the most common types.

5.2.3.1 First-Order Stationary Process

A random process is classified as first-order stationary if its first-order probability density function remains equal regardless of any shift in time to its time origin. If we let $x_{t_1}$ represent a given value at time $t_1$, then we define a first-order stationary as one that satisfies the following equation:

$$f_x (x_{t_1}) = f_x (x_{t_1} + \tau) \quad (5.5)$$

The physical significance of this equation is that our density function, $f_x (x_{t_1})$, is completely independent of $t_1$ and thus any time shift, $\tau$.

The most important result of this statement, and the identifying characteristic of any first-order stationary process, is the fact that the mean is a constant, independent of any time shift. Below we show the results for a random process, $X$, that is a discrete-time signal, $x[n]$.

$$\overline{X} = m_x [n] = E [x [n]] = \text{constant (independent of } n) \quad (5.6)$$

5.2.3.2 Second-Order and Strict-Sense Stationary Process

A random process is classified as second-order stationary if its second-order probability density function does not vary over any time shift applied to both values. In other words, for values $x_{t_1}$ and $x_{t_2}$ then we will have the following be equal for an arbitrary time shift $\tau$.

$$f_x (x_{t_1}, x_{t_2}) = f_x (x_{t_1 + \tau}, x_{t_2 + \tau}) \quad (5.7)$$

From this equation we see that the absolute time does not affect our functions, rather it only really depends on the time difference between the two variables. Looked at another way, this equation can be described as

$$Pr [X(t_1) \leq x_1, X(t_2) \leq x_2] = Pr [X(t_1 + \tau) \leq x_1, X(t_2 + \tau) \leq x_2] \quad (5.8)$$

These random processes are often referred to as strict sense stationary (SSS) when all of the distribution functions of the process are unchanged regardless of the time shift applied to them.

For a second-order stationary process, we need to look at the autocorrelation function (Section 5.5) to see its most important property. Since we have already stated that a second-order stationary process depends only on the time difference, then all of these types of processes have the following property:

$$R_{xx} (t, t + \tau) = E [X(t + \tau)] = R_{xx} (\tau) \quad (5.9)$$
5.2.3.3 Wide-Sense Stationary Process

As you begin to work with random processes, it will become evident that the strict requirements of a SSS process is more than is often necessary in order to adequately approximate our calculations on random processes. We define a final type of stationarity, referred to as **wide-sense stationary (WSS)**, to have slightly more relaxed requirements but ones that are still enough to provide us with adequate results. In order to be WSS a random process only needs to meet the following two requirements.

1. \( \bar{X} = E[x[n]] = \text{constant} \)
2. \( E[X(t+\tau)] = R_{xx}(\tau) \)

Note that a second-order (or SSS) stationary process will always be WSS; however, the reverse will not always hold true.

5.3 Random Processes: Mean and Variance\(^5\)

In order to study the characteristics of a random process (Section 5.1), let us look at some of the basic properties and operations of a random process. Below we will focus on the operations of the random signals that compose our random processes. We will denote our random process with \( X \) and a random variable from a random process or signal by \( x \).

5.3.1 Mean Value

Finding the average value of a set of random signals or random variables is probably the most fundamental concepts we use in evaluating random processes through any sort of statistical method. The **mean of a random process is the average of all realizations of that process**. In order to find this average, we must look at a random signal over a range of time (possible values) and determine our average from this set of values. The **mean**, or average, of a random process, \( x(t) \), is given by the following equation:

\[
m_x(t) = \mu_x(t) = \bar{X} = E[X] = \int_{-\infty}^{\infty} x f(x) \, dx
\]

(5.10)

This equation may seem quite cluttered at first glance, but we want to introduce you to the various notations used to represent the mean of a random signal or process. Throughout texts and other readings, remember that these will all equal the same thing. The symbol, \( \mu_x(t) \), and the \( X \) with a bar over it are often used as a short-hand to represent an average, so you might see it in certain textbooks. The other important notation used is, \( E[X] \), which represents the "expected value of \( X \)" or the mathematical expectation. This notation is very common and will appear again.

If the random variables, which make up our random process, are discrete or quantized values, such as in a binary process, then the integrals become summations over all the possible values of the random variable. In this case, our expected value becomes

\[
E[x[n]] = \sum_{n} \alpha Pr[x[n] = \alpha]
\]

(5.11)

If we have two random signals or variables, their averages can reveal how the two signals interact. If the product of the two individual averages of both signals do **not** equal the average of the product of the two signals, then the two signals are said to be **linearly independent**, also referred to as **uncorrelated**.

---

\(^5\)This content is available online at <http://cnx.org/content/m10656/2.3/>.

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
In the case where we have a random process in which only one sample can be viewed at a time, then we will often not have all the information available to calculate the mean using the density function as shown above. In this case we must estimate the mean through the time-average mean (Section 5.3.4: Time Averages), discussed later. For fields such as signal processing that deal mainly with discrete signals and values, then these are the averages most commonly used.

5.3.1 Properties of the Mean

- The expected value of a constant, \( \alpha \), is the constant:
  \[
  E[\alpha] = \alpha
  \]
  (5.12)

- Adding a constant, \( \alpha \), to each term increases the expected value by that constant:
  \[
  E[X + \alpha] = E[X] + \alpha
  \]
  (5.13)

- Multiplying the random variable by a constant, \( \alpha \), multiplies the expected value by that constant.
  \[
  E[\alpha X] = \alpha E[X]
  \]
  (5.14)

- The expected value of the sum of two or more random variables, is the sum of each individual expected value.
  \[
  E[X + Y] = E[X] + E[Y]
  \]
  (5.15)

5.3.2 Mean-Square Value

If we look at the second moment of the term (we now look at \( x^2 \) in the integral), then we will have the mean-square value of our random process. As you would expect, this is written as

\[
X^2 = E[X^2] = \int_{-\infty}^{\infty} x^2 f(x) \, dx
\]
(5.16)

This equation is also often referred to as the average power of a process or signal.

5.3.3 Variance

Now that we have an idea about the average value or values that a random process takes, we are often interested in seeing just how spread out the different random values might be. To do this, we look at the variance which is a measure of this spread. The variance, often denoted by \( \sigma^2 \), is written as follows:

\[
\sigma^2 = \text{Var}(X) = E[(X - E[X])^2] = \int_{-\infty}^{\infty} (x - \bar{X})^2 f(x) \, dx
\]
(5.17)

Using the rules for the expected value, we can rewrite this formula as the following form, which is commonly seen:

\[
\sigma^2 = \bar{X}^2 - \left(\frac{1}{N}\sum_{n=0}^{N} x_n\right)^2
\]
\[
= E[X^2] - (E[X])^2
\]
(5.18)
5.3.3.1 Standard Deviation

Another common statistical tool is the standard deviation. Once you know how to calculate the variance, the standard deviation is simply the square root of the variance, or $\sigma$.

5.3.3.2 Properties of Variance

- The variance of a constant, $\alpha$, equals zero:
  \[
  \text{Var}(\alpha) = \sigma(\alpha)^2 = 0
  \]  
  (5.19)

- Adding a constant, $\alpha$, to a random variable does not affect the variance because the mean increases by the same value:
  \[
  \text{Var}(X + \alpha) = \sigma(X + \alpha)^2 = \sigma(X)^2
  \]  
  (5.20)

- Multiplying the random variable by a constant, $\alpha$, increases the variance by the square of the constant:
  \[
  \text{Var}(\alpha X) = \sigma(\alpha X)^2 = \alpha^2 \sigma(X)^2
  \]  
  (5.21)

- The variance of the sum of two random variables only equals the sum of the variances if the variable are independent.
  \[
  \text{Var}(X + Y) = \sigma(X + Y)^2 = \sigma(X)^2 + \sigma(Y)^2
  \]  
  (5.22)

  Otherwise, if the random variable are not independent, then we must also include the covariance of the product of the variables as follows:
  \[
  \text{Var}(X + Y) = \sigma(X)^2 + 2\text{Cov}(X, Y) + \sigma(Y)^2
  \]  
  (5.23)

5.3.4 Time Averages

In the case where we cannot view the entire ensemble of the random process, we must use time averages to estimate the values of the mean and variance for the process. Generally, this will only give us acceptable results for independent and ergodic processes, meaning those processes in which each signal or member of the process seems to have the same statistical behavior as the entire process. The time averages will also only be taken over a finite interval since we will only be able to see a finite part of the sample.

5.3.4.1 Estimating the Mean

For the ergodic random process, $x(t)$, we will estimate the mean using the time averaging function defined as

\[
\bar{X} = E[X] = \frac{1}{T} \int_{0}^{T} X(t) dt
\]  
(5.24)

However, for most real-world situations we will be dealing with discrete values in our computations and signals. We will represent this mean as

\[
\bar{X} = E[X] = \frac{1}{N} \sum_{n=1}^{N} X[n]
\]  
(5.25)
5.3.4.2 Estimating the Variance

Once the mean of our random process has been estimated then we can simply use those values in the following variance equation (introduced in one of the above sections)

\[ \sigma_x^2 = \bar{X}^2 - \left( \overline{\bar{X}} \right)^2 \]  

(5.26)

5.3.5 Example

Let us now look at how some of the formulas and concepts above apply to a simple example. We will just look at a single, continuous random variable for this example, but the calculations and methods are the same for a random process. For this example, we will consider a random variable having the probability density function described below and shown in Figure 5.4 (Probability Density Function).

\[ f(x) = \begin{cases} \frac{1}{10} & \text{if } 10 \leq x \leq 20 \\ 0 & \text{otherwise} \end{cases} \]  

(5.27)

Figure 5.4: A uniform probability density function.

First, we will use (5.10) to solve for the mean value.

\[ \bar{X} = \int_{10}^{20} x \cdot \frac{1}{10} \, dx \]
\[ = \frac{1}{10} \cdot \frac{x^2}{2} \bigg|_{x=10}^{20} \]
\[ = \frac{1}{10} \cdot (200 - 50) \]
\[ = 15 \]  

(5.28)
Using (5.16) we can obtain the mean-square value for the above density function.

\[ \bar{X}^2 = \int_{10}^{20} x^2 \frac{1}{10} \, dx \]
\[ = \frac{1}{10} \left[ \frac{8000}{3} - \frac{1000}{3} \right] \]
\[ = 233.33 \quad (5.29) \]

And finally, let us solve for the variance of this function.

\[ \sigma^2 = \bar{X}^2 - \left( \bar{X} \right)^2 \]
\[ = 233.33 - 15^2 \]
\[ = 8.33 \quad (5.30) \]

**5.4 Correlation and Covariance of a Random Signal**

When we take the expected value (Section 5.3), or average, of a random process (Section 5.1.2: Random Process), we measure several important characteristics about how the process behaves in general. This proves to be a very important observation. However, suppose we have several random processes measuring different aspects of a system. The relationship between these different processes will also be an important observation. The covariance and correlation are two important tools in finding these relationships. Below we will go into more details as to what these words mean and how these tools are helpful. Note that much of the following discussions refer to just random variables, but keep in mind that these variables can represent random signals or random processes.

### 5.4.1 Covariance

To begin with, when dealing with more than one random process, it should be obvious that it would be nice to be able to have a number that could quickly give us an idea of how similar the processes are. To do this, we use the covariance, which is analogous to the variance of a single variable.

**Definition 5.3: Covariance**

A measure of how much the deviations of two or more variables or processes match.

For two processes, X and Y, if they are not closely related then the covariance will be small, and if they are similar then the covariance will be large. Let us clarify this statement by describing what we mean by "related" and "similar." Two processes are "closely related" if their distribution spreads are almost equal and they are around the same, or a very slightly different, mean.

Mathematically, covariance is often written as \( \sigma_{xy} \) and is defined as

\[ \text{cov}(X, Y) = \sigma_{xy} \]
\[ = \mathbb{E} \left[ (X - \bar{X})(Y - \bar{Y}) \right] \quad (5.31) \]

This can also be reduced and rewritten in the following two forms:

\[ \sigma_{xy} = \bar{xy} - \bar{x} \bar{y} \quad (5.32) \]

---

\(^6\)This content is available online at <http://cnx.org/content/m10673/2.3/>.
\[ \sigma_{xy} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})(y - \bar{y}) f(x, y) \, dx \, dy \quad (5.33) \]

### 5.4.1 Useful Properties

- If \( X \) and \( Y \) are independent and uncorrelated or one of them has zero mean value, then
  \[ \sigma_{xy} = 0 \]
- If \( X \) and \( Y \) are orthogonal, then
  \[ \sigma_{xy} = -(E[X]E[Y]) \]
- The covariance is symmetric
  \[ \text{cov}(X, Y) = \text{cov}(Y, X) \]
- Basic covariance identity
  \[ \text{cov}(X + Y, Z) = \text{cov}(X, Z) + \text{cov}(Y, Z) \]
- Covariance of equal variables
  \[ \text{cov}(X, X) = \text{Var}(X) \]

### 5.4.2 Correlation

For anyone who has any kind of statistical background, you should be able to see that the idea of dependence/independence among variables and signals plays an important role when dealing with random processes. Because of this, the correlation of two variables provides us with a measure of how the two variables affect one another.

**Definition 5.4: Correlation**

A measure of how much one random variable depends upon the other.

This measure of association between the variables will provide us with a clue as to how well the value of one variable can be predicted from the value of the other. The correlation is equal to the average of the product of two random variables and is defined as

\[
\text{cor}(X, Y) = E[XY] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xyf(x, y) \, dx \, dy \quad (5.34)
\]

### 5.4.2.1 Correlation Coefficient

It is often useful to express the correlation of random variables with a range of numbers, like a percentage. For a given set of variables, we use the correlation coefficient to give us the linear relationship between our variables. The correlation coefficient of two variables is defined in terms of their covariance and standard deviations (Section 5.3.3.1: Standard Deviation), denoted by \( \sigma_x \), as seen below

\[
\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \quad (5.35)
\]

where we will always have

\[-1 \leq \rho \leq 1\]

This provides us with a quick and easy way to view the correlation between our variables. If there is no relationship between the variables then the correlation coefficient will be zero and if there is a perfect positive
match it will be one. If there is a perfect inverse relationship, where one set of variables increases while the other decreases, then the correlation coefficient will be negative one. This type of correlation is often referred to more specifically as the Pearson’s Correlation Coefficient or Pearson’s Product Moment Correlation.

![Figure 5.5: Types of Correlation](http://cnx.org/content/col10360/1.4)

Figure 5.5: Types of Correlation (a) Positive Correlation (b) Negative Correlation (c) Uncorrelated (No Correlation)

NOTE: So far we have dealt with correlation simply as a number relating the relationship between any two variables. However, since our goal will be to relate random processes to each other, which deals with signals as a function of time, we will want to continue this study by looking at correlation functions (Section 5.5).

5.4.3 Example

Now let us take just a second to look at a simple example that involves calculating the covariance and correlation of two sets of random numbers. We are given the following data sets:

\[ x = \{3, 1, 6, 3, 4\} \]

\[ y = \{1, 5, 3, 4, 3\} \]

To begin with, for the covariance we will need to find the expected value (Section 5.3), or mean, of \( x \) and \( y \).

\[ \bar{x} = \frac{1}{5} (3 + 1 + 6 + 3 + 4) = 3.4 \]

\[ \bar{y} = \frac{1}{5} (1 + 5 + 3 + 4 + 3) = 3.2 \]

\[ \bar{xy} = \frac{1}{5} (3 + 5 + 18 + 12 + 12) = 10 \]
Next we will solve for the standard deviations of our two sets using the formula below (for a review click here (Section 5.3.3: Variance)).

\[
\sigma = \sqrt{E[(X - E[X])^2]}
\]

\[
\sigma_x = \sqrt{\frac{1}{5}(0.16 + 5.76 + 6.76 + 0.16 + 0.36)} = 1.625
\]

\[
\sigma_y = \sqrt{\frac{1}{6}(4.84 + 3.24 + 0.04 + 0.64 + 0.04)} = 1.327
\]

Now we can finally calculate the covariance using one of the two formulas found above. Since we calculated the three means, we will use that formula (5.32) since it will be much simpler.

\[
\sigma_{xy} = 10 - 3.4 \times 3.2 = -0.88
\]

And for our last calculation, we will solve for the correlation coefficient, \(\rho\).

\[
\rho = \frac{-0.88}{1.625 \times 1.327} = -0.408
\]

### 5.4.3.1 Matlab Code for Example

The above example can be easily calculated using Matlab. Below I have included the code to find all of the values above.

```matlab
x = [3 1 6 3 4];
y = [1 5 3 4 3];

mx = mean(x)
my = mean(y)
mxy = mean(x.*y)

% Standard Dev. from built-in Matlab Functions
std(x,1)
std(y,1)

% Standard Dev. from Equation Above (same result as std(?,1))
sqrt( 1/5 * sum((x-mx).^2))
sqrt( 1/5 * sum((y-my).^2))

cov(x,y,1)

corrcoef(x,y)
```

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
5.5 Autocorrelation of Random Processes

Before diving into a more complex statistical analysis of random signals and processes (Section 5.1), let us quickly review the idea of correlation (Section 5.4). Recall that the correlation of two signals or variables is the expected value of the product of those two variables. Since our focus will be to discover more about a random process, a collection of random signals, then imagine us dealing with two samples of a random process, where each sample is taken at a different point in time. Also recall that the key property of these random processes is that they are now functions of time; imagine them as a collection of signals. The expected value (Section 5.3) of the product of these two variables (or samples) will now depend on how quickly they change in regards to time. For example, if the two variables are taken from almost the same time period, then we should expect them to have a high correlation. We will now look at a correlation function that relates a pair of random variables from the same process to the time separations between them, where the argument to this correlation function will be the time difference. For the correlation of signals from two different random process, look at the crosscorrelation function (Section 5.6).

5.5.1 Autocorrelation Function

The first of these correlation functions we will discuss is the autocorrelation, where each of the random variables we will deal with come from the same random process.

**Definition 5.5: Autocorrelation**

the expected value of the product of a random variable or signal realization with a time-shifted version of itself

With a simple calculation and analysis of the autocorrelation function, we can discover a few important characteristics about our random process. These include:

1. How quickly our random signal or processes changes with respect to the time function
2. Whether our process has a periodic component and what the expected frequency might be

As was mentioned above, the autocorrelation function is simply the expected value of a product. Assume we have a pair of random variables from the same process, $X_1 = X(t_1)$ and $X_2 = X(t_2)$, then the autocorrelation is often written as

$$R_{xx}(t_1, t_2) = E[X_1 X_2] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f(x_1, x_2) \, dx \, dx$$

(5.36)

The above equation is valid for stationary and nonstationary random processes. For stationary processes (Section 5.2), we can generalize this expression a little further. Given a wide-sense stationary processes, it can be proven that the expected values from our random process will be independent of the origin of our time function. Therefore, we can say that our autocorrelation function will depend on the time difference and not some absolute time. For this discussion, we will let $\tau = t_2 - t_1$, and thus we generalize our autocorrelation expression as

$$R_{xx}(t, t + \tau) = R_{xx}(\tau) = E[X(t) X(t + \tau)]$$

(5.37)

for the continuous-time case. In most DSP course we will be more interested in dealing with real signal sequences, and thus we will want to look at the discrete-time case of the autocorrelation function. The formula below will prove to be more common and useful than (5.36):

$$R_{xx}[n, n + m] = \sum_{n=-\infty}^{\infty} x[n] x[n + m]$$

(5.38)

---

This content is available online at <http://cnx.org/content/m10676/2.4/>. 

Available for free at Connexions <http://cnx.org/content/col10360/1.4/>.
And again we can generalize the notation for our autocorrelation function as
\[
R_{xx}[n, n+m] = R_{xx}[m]
\]
\[= E[X[n]X[n+m]] \quad (5.39)\]

### 5.5.1.1 Properties of Autocorrelation

Below we will look at several properties of the autocorrelation function that hold for stationary random processes.

- Autocorrelation is an even function for \( \tau \)
  \[R_{xx}(\tau) = R_{xx}(-\tau)\]
- The mean-square value can be found by evaluating the autocorrelation where \( \tau = 0 \), which gives us
  \[R_{xx}(0) = \bar{X}^2\]
- The autocorrelation function will have its largest value when \( \tau = 0 \). This value can appear again, for example in a periodic function at the values of the equivalent periodic points, but will never be exceeded.
  \[R_{xx}(0) \geq |R_{xx}(\tau)|\]
- If we take the autocorrelation of a period function, then \( R_{xx}(\tau) \) will also be periodic with the same frequency.

### 5.5.1.2 Estimating the Autocorrelation with Time-Averaging

Sometimes the whole random process is not available to us. In these cases, we would still like to be able to find out some of the characteristics of the stationary random process, even if we just have part of one sample function. In order to do this we can estimate the autocorrelation from a given interval, 0 to \( T \) seconds, of the sample function.

\[
\hat{R}_{xx}(\tau) = \frac{1}{T-\tau} \int_{0}^{T-\tau} x(t)x(t+\tau) \, dt \quad (5.40)
\]

However, a lot of times we will not have sufficient information to build a complete continuous-time function of one of our random signals for the above analysis. If this is the case, we can treat the information we do know about the function as a discrete signal and use the discrete-time formula for estimating the autocorrelation.

\[
\hat{R}_{xx}[m] = \frac{1}{N-m} \sum_{n=0}^{N-m-1} x[n]x[n+m] \quad (5.41)
\]

### 5.5.2 Examples

Below we will look at a variety of examples that use the autocorrelation function. We will begin with a simple example dealing with Gaussian White Noise (GWN) and a few basic statistical properties that will prove very useful in these and future calculations.

**Example 5.1**

We will let \( x[n] \) represent our GWN. For this problem, it is important to remember the following fact about the mean of a GWN function:

\[E[x[n]] = 0\]
Along with being zero-mean, recall that GWN is always independent. With these two facts, we are now ready to do the short calculations required to find the autocorrelation.

\[ R_{xx}[n, n + m] = E[x[n]x[n + m]] \]

Since the function, \( x[n] \), is independent, then we can take the product of the individual expected values of both functions.

\[ R_{xx}[n, n + m] = E[x[n]]E[x[n + m]] \]

Now, looking at the above equation we see that we can break it up further into two conditions: one when \( m \) and \( n \) are equal and one when they are not equal. When they are equal we can combine the expected values. We are left with the following piecewise function to solve:

\[
R_{xx}[n, n + m] = \begin{cases} 
E[x[n]]E[x[n + m]] & \text{if } m \neq 0 \\
E[x^2[n]] & \text{if } m = 0
\end{cases}
\]

We can now solve the two parts of the above equation. The first equation is easy to solve as we have already stated that the expected value of \( x[n] \) will be zero. For the second part, you should recall from statistics that the expected value of the square of a function is equal to the variance. Thus we get the following results for the autocorrelation:

\[
R_{xx}[n, n + m] = \begin{cases} 
0 & \text{if } m \neq 0 \\
\sigma^2 & \text{if } m = 0
\end{cases}
\]

Or in a more concise way, we can represent the results as

\[ R_{xx}[n, n + m] = \sigma^2 \delta[m] \]

### 5.6 Crosscorrelation of Random Processes

Before diving into a more complex statistical analysis of random signals and processes (Section 5.1), let us quickly review the idea of correlation (Section 5.4). Recall that the correlation of two signals or variables is the expected value of the product of those two variables. Since our main focus is to discover more about random processes, a collection of random signals, we will deal with two random processes in this discussion, where in this case we will deal with samples from two different random processes. We will analyze the expected value (Section 5.3.1: Mean Value) of the product of these two variables and how they correlate to one another, where the argument to this correlation function will be the time difference. For the correlation of signals from the same random process, look at the autocorrelation function (Section 5.5).
5.6.1 Crosscorrelation Function

When dealing with multiple random processes, it is also important to be able to describe the relationship, if any, between the processes. For example, this may occur if more than one random signal is applied to a system. In order to do this, we use the **crosscorrelation function**, where the variables are instances from two different wide sense stationary random processes.

**Definition 5.6: Crosscorrelation**

if two processes are wide sense stationary, the expected value of the product of a random variable from one random process with a time-shifted, random variable from a different random process.

Looking at the generalized formula for the crosscorrelation, we will represent our two random processes by allowing $U = U(t)$ and $V = V(t - \tau)$. We will define the crosscorrelation function as

\[ R_{uv}(t, t - \tau) = E[UV] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} uvf(u, v) \, dv \, du \]  

(5.42)

Just as the case with the autocorrelation function, if our input and output, denoted as $U(t)$ and $V(t)$, are at least jointly wide sense stationary, then the crosscorrelation does not depend on absolute time; it is just a function of the time difference. This means we can simplify our writing of the above function as

\[ R_{uv}(\tau) = E[UV] \]  

(5.43)

or if we deal with two real signal sequences, $x[n]$ and $y[n]$, then we arrive at a more commonly seen formula for the discrete crosscorrelation function. See the formula below and notice the similarities between it and the convolution (Section 1.5) of two signals:

\[ R_{xy}(n, n - m) = R_{xy}(m) = \sum_{n=-\infty}^{\infty} x[n] y[n - m] \]  

(5.44)

5.6.1.1 Properties of Crosscorrelation

Below we will look at several properties of the crosscorrelation function that hold for two wide sense stationary (WSS) random processes.

- Crosscorrelation is **not** an even function; however, it does have a unique symmetry property:

\[ R_{xy}(-\tau) = R_{yx}(\tau) \]  

(5.45)

- The maximum value of the crosscorrelation is not always when the shift equals zero; however, we can prove the following property revealing to us what value the maximum cannot exceed.

\[ |R_{xy}(\tau)| \leq \sqrt{R_{xx}(0) R_{yy}(0)} \]  

(5.46)

- When two random processes are statistically independent then we have

\[ R_{xy}(\tau) = R_{yx}(\tau) \]  

(5.47)
5.6.2 Examples

Exercise 5.6.1
Let us begin by looking at a simple example showing the relationship between two sequences. Using (5.44), find the crosscorrelation of the sequences

\[ x[n] = \{ \ldots, 0, 0, 2, -3, 6, 1, 3, 0, 0, \ldots \} \]

\[ y[n] = \{ \ldots, 0, 0, 1, -2, 4, 1, -3, 0, 0, \ldots \} \]

for each of the following possible time shifts: \( m = \{0, 3, -1\} \).

5.7 Introduction to Adaptive Filters\(^9\)

In many applications requiring filtering, the necessary frequency response may not be known beforehand, or it may vary with time. (Example; suppression of engine harmonics in a car stereo.) In such applications, an adaptive filter which can automatically design itself and which can track system variations in time is extremely useful. Adaptive filters are used extensively in a wide variety of applications, particularly in telecommunications.

Outline of adaptive filter material

1. **Wiener Filters** - \( L_2 \) optimal (FIR) filter design in a statistical context
2. **LMS algorithm** - simplest and by-far-the-most-commonly-used adaptive filter algorithm
3. **Stability and performance of the LMS algorithm** - When and how well it works
4. **Applications of adaptive filters** - Overview of important applications
5. **Introduction to advanced adaptive filter algorithms** - Techniques for special situations or faster convergence

5.8 Discrete-Time, Causal Wiener Filter\(^{10}\)

Stochastic \( L_2 \) optimal (least squares) FIR filter design problem: Given a wide-sense stationary (WSS) input signal \( x_k \) and desired signal \( d_k \) (WSS \( \Rightarrow \) \( E[y_k] = E[y_k+d], r_{yx}(l) = E[y_k z_{k+l}], \forall k, l : (r_{yy}(0) < \infty) \))

\(^9\)This content is available online at \(<http://cnx.org/content/m11535/1.3/>\).
\(^{10}\)This content is available online at \(<http://cnx.org/content/m11825/1.1/>\).
The Wiener filter is the linear, time-invariant filter minimizing $E[\epsilon^2]$, the variance of the error. As posed, this problem seems slightly silly, since $d_k$ is already available! However, this idea is useful in a wide variety of applications.

**Example 5.2**
active suspension system design

**Example 5.3**
System identification (radar, non-destructive testing, adaptive control systems)

*NOTE:* optimal system may change with different road conditions or mass in car, so an *adaptive* system might be desirable.
Exercise 5.8.1
Usually one desires that the input signal $x_k$ be "persistently exciting," which, among other things, implies non-zero energy in all frequency bands. Why is this desirable?

5.8.1 Determining the optimal length-$N$ causal FIR Wiener filter

Note: for convenience, we will analyze only the causal, real-data case; extensions are straightforward.

$$y_k = \sum_{l=0}^{M-1} w_l x_{k-l}$$

$$\arg\min_{w_l} E[\epsilon^2] = E[(d_k - y_k)^2] = E\left[\left(d_k - \sum_{l=0}^{M-1} w_l x_{k-l}\right)^2\right] = E[d_k^2] - 2 \sum_{l=0}^{M-1} w_l E[d_k x_{k-l}] + \sum_{l=0}^{M-1} \sum_{m=0}^{M-1} (w_l w_m E[x_{k-l} x_{k-m}])$$

$$E[\epsilon^2] = r_{dd}(0) - 2 \sum_{l=0}^{M-1} w_l r_{dx}(l) + \sum_{l=0}^{M-1} \sum_{m=0}^{M-1} w_l w_m r_{xx}(l-m)$$

where

$$r_{dd}(0) = E[d_k^2]$$

$$r_{dx}(l) = E[d_k x_{k-l}]$$

$$r_{xx}(l-m) = E[x_k x_{k+l-m}]$$

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
This can be written in matrix form as

\[ E[\epsilon^2] = r_{dd}(0) - 2PW^T + W^TRW \]

where

\[
P = \begin{pmatrix}
  r_{dx}(0) \\
  r_{dx}(1) \\
  \vdots \\
  r_{dx}(M-1)
\end{pmatrix}
\]

\[
R = \begin{pmatrix}
  r_{xx}(0) & r_{xx}(1) & \ldots & \ldots & r_{xx}(M-1) \\
  r_{xx}(1) & r_{xx}(0) & \ddots & \ddots & \vdots \\
  \vdots & \ddots & \ddots & \ddots & \vdots \\
  \vdots & \ddots & \ddots & \ddots & \ddots \\
  r_{xx}(M-1) & \ldots & \ldots & r_{xx}(1) & r_{xx}(0)
\end{pmatrix}
\]

To solve for the optimum filter, compute the gradient with respect to the top weights vector \( W \)

\[
\nabla = \begin{pmatrix}
  \frac{\partial \epsilon^2}{\partial w_0} \\
  \frac{\partial \epsilon^2}{\partial w_1} \\
  \vdots \\
  \frac{\partial \epsilon^2}{\partial w_{M-1}}
\end{pmatrix}
\]

\[
\nabla = -(2P) + 2RW
\]

(recall \( \frac{d}{dW} (A^T W) = A^T \), \( \frac{d}{dW} (WMW) = 2MW \) for symmetric \( M \)) setting the gradient equal to zero

\[ W_{\text{opt}}R = P \Rightarrow W_{\text{opt}} = R^{-1}P \]

Since \( R \) is a correlation matrix, it must be non-negative definite, so this is a minimizer. For \( R \) positive definite, the minimizer is unique.

### 5.9 Practical Issues in Wiener Filter Implementation

The wiener-filter, \( W_{\text{opt}} = R^{-1}P \), is ideal for many applications. But several issues must be addressed to use it in practice.

**Exercise 5.9.1**

(Solution on p. 210.)

In practice one usually won’t know exactly the statistics of \( x_k \) and \( d_k \) (i.e. \( R \) and \( P \)) needed to compute the Weiner filter.

How do we surmount this problem?

**Exercise 5.9.2**

(Solution on p. 210.)

In many applications, the statistics of \( x_k, d_k \) vary slowly with time.

How does one develop an adaptive system which tracks these changes over time to keep the system near optimal at all times?

---

11This content is available online at <http://cnx.org/content/m11824/1.1/>. Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Exercise 5.9.3
How can \( r^k_{xx}(l) \) be computed efficiently?

Exercise 5.9.4
how does one choose \( N \)?

5.9.1 Tradeoffs

Larger \( N \) → more accurate estimates of the correlation values → better \( W_{opt} \). However, larger \( N \) leads to slower adaptation.

\textbf{Note:} The success of adaptive systems depends on \( x, d \) being roughly stationary over at least \( N \) samples, \( N > M \). That is, all adaptive filtering algorithms require that the underlying system varies slowly with respect to the sampling rate and the filter length (although they can tolerate occasional step discontinuities in the underlying system).

5.9.2 Computational Considerations

As presented here, an adaptive filter requires computing a matrix inverse at each sample. Actually, since the matrix \( R \) is Toeplitz, the linear system of equations can be solved with \( O(M^2) \) computations using Levinson's algorithm, where \( M \) is the filter length. However, in many applications this may be too expensive, especially since computing the filter output itself requires \( O(M) \) computations. There are two main approaches to resolving the computation problem

1. Take advantage of the fact that \( R^{k+1} \) is only slightly changed from \( R^k \) to reduce the computation to \( O(M) \); these algorithms are called Fast Recursive Least Squareds algorithms; all methods proposed so far have stability problems and are dangerous to use.
2. Find a different approach to solving the optimization problem that doesn’t require explicit inversion of the correlation matrix.

\textbf{Note:} Adaptive algorithms involving the correlation matrix are called \textbf{Recursive least Squares} (RLS) algorithms. Historically, they were developed after the LMS algorithm, which is the simplest and most widely used approach \( O(M) \). \( O(M^2) \) RLS algorithms are used in applications requiring very fast adaptation.

5.10 Quadratic Minimization and Gradient Descent\(^{12}\)

5.10.1 Quadratic minimization problems

The least squares optimal filter design problem is quadratic in the filter coefficients:

\[
E[\epsilon^2] = r_{dd}(0) - 2P^T W + W^T RW
\]

If \( R \) is positive definite, the error surface \( E[\epsilon^2] (w_0, w_1, \ldots, w_{M-1}) \) is a unimodal "bowl" in \( \mathbb{R}^N \).

\(^{12}\text{This content is available online at <http://cnx.org/content/m11826/1.2/>.}\)
The problem is to find the bottom of the bowl. In an adaptive filter context, the shape and bottom of the bowl may drift slowly with time; hopefully slow enough that the adaptive algorithm can track it.

For a quadratic error surface, the bottom of the bowl can be found in one step by computing $R^{-1}P$. Most modern nonlinear optimization methods (which are used, for example, to solve the $L^p$ optimal IIR filter design problem!) locally approximate a nonlinear function with a second-order (quadratic) Taylor series approximation and step to the bottom of this quadratic approximation on each iteration. However, an older and simpler approach to nonlinear optimization exists, based on gradient descent.

Contour plot of $\varepsilon$-squared

The idea is to iteratively find the minimizer by computing the gradient of the error function: $E \nabla = \partial E[\varepsilon^2] / \partial w_i$. 

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
The gradient is a vector in $\mathbb{R}^M$ pointing in the steepest uphill direction on the error surface at a given point $W^i$, with $\nabla$ having a magnitude proportional to the slope of the error surface in this steepest direction.

By updating the coefficient vector by taking a step opposite the gradient direction: $W^{i+1} = W^i - \mu \nabla^i$, we go (locally) "downhill" in the steepest direction, which seems to be a sensible way to iteratively solve a nonlinear optimization problem. The performance obviously depends on $\mu$; if $\mu$ is too large, the iterations could bounce back and forth up out of the bowl. However, if $\mu$ is too small, it could take many iterations to approach the bottom. We will determine criteria for choosing $\mu$ later.

In summary, the gradient descent algorithm for solving the Weiner filter problem is:

\begin{align*}
\text{Guess } W^0 \\
\text{do } i = 1, \infty \\
\nabla^i &= -(2P) + 2RW^i \\
W^{i+1} &= W^i - \mu \nabla^i \\
\text{repeat} \\
W_{\text{opt}} &= W^\infty
\end{align*}

The gradient descent idea is used in the LMS adaptive filter algorithm. As presented, this algorithm costs $O(M^2)$ computations per iteration and doesn't appear very attractive, but LMS only requires $O(M)$ computations and is stable, so it is very attractive when computation is an issue, even though it converges more slowly than the RLS algorithms we have discussed so far.

### 5.11 The LMS Adaptive Filter Algorithm\(^{13}\)

Recall the Weiner filter problem

![Figure 5.12](http://cnx.org/content/m11829/1.1/)

---

\(^{13}\)This content is available online at [http://cnx.org/content/m11829/1.1/].

Available for free at Connexions [http://cnx.org/content/col10360/1.4>
\{x_k\}, \{d_k\} jointly wide sense stationary

Find \( W \) minimizing \( E[\epsilon_k^2] \)

\[
\epsilon_k = d_k - y_k = d_k - \sum_{i=0}^{M-1} w_i x_{k-i} = d_k - X^k W^k
\]

\[
X^k = \begin{pmatrix}
x_k \\
x_{k-1} \\
\vdots \\
x_{k-M+1}
\end{pmatrix}
\]

\[
W^k = \begin{pmatrix}
w_0^k \\
w_1^k \\
\vdots \\
w_{M-1}^k
\end{pmatrix}
\]

The superscript denotes absolute time, and the subscript denotes time or a vector index.

the solution can be found by setting the gradient 0

\[
\nabla^k = \frac{\partial E[\epsilon_k^2]}{\partial W} = E\left[2\epsilon_k \left(-X^k\right)\right] = E\left[-2 \left(d_k - X^k W_k\right) X^k\right] = -\left(2E[d_k X^k]\right) + E\left[X^k W\right] = 2P + 2RW
\]

\[
\Rightarrow (W_{opt} = R^{-1}P)
\]

Alternatively, \( W_{opt} \) can be found iteratively using a gradient descent technique

\[
W^{k+1} = W^k - \mu \nabla^k
\]

In practice, we don’t know \( R \) and \( P \) exactly, and in an adaptive context they may be slowly varying with time.

To find the (approximate) Wiener filter, some approximations are necessary. As always, the key is to make the right approximations!

**NOTE:** Approximate \( R \) and \( P \): \( \Rightarrow \) RLS methods, as discussed last time.

**NOTE:** Approximate the gradient!

\[
\nabla^k = \frac{\partial E[\epsilon_k^2]}{\partial W}
\]

Note that \( \epsilon_k^2 \) itself is a very noisy approximation to \( E[\epsilon_k^2] \). We can get a noisy approximation to the gradient by finding the gradient of \( \epsilon_k^2 \). Widrow and Hoff first published the LMS algorithm, based on this clever idea, in 1960.

\[
\nabla^k = \frac{\partial \epsilon_k^2}{\partial W} = 2\epsilon_k \frac{\partial \left(d_k - W^k X^k\right)}{\partial W} = 2\epsilon_k \left(-X^k\right) = -\left(2\epsilon_k X^k\right)
\]

This yields the LMS adaptive filter algorithm

**Example 5.4: The LMS Adaptive Filter Algorithm**

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
1. $y_k = W_k^T X_k = \sum_{i=0}^{M-1} w_k^i x_{k-i}$
2. $\epsilon_k = d_k - y_k$
3. $W_{k+1} = W_k - \mu \nabla k = W_k - \mu (-2 \epsilon_k X_k) = W_k + 2 \mu \epsilon_k X_k (w_{k+1}^i = w_k^i + 2 \mu \epsilon_k x_{k-i})$

The LMS algorithm is often called a stochastic gradient algorithm, since $\nabla k$ is a noisy gradient. This is by far the most commonly used adaptive filtering algorithm, because

1. it was the first
2. it is very simple
3. in practice it works well (except that sometimes it converges slowly)
4. it requires relatively little computation
5. it updates the tap weights every sample, so it continually adapts the filter
6. it tracks slow changes in the signal statistics well

5.11.1 Computational Cost of LMS

<table>
<thead>
<tr>
<th>To Compute</th>
<th>$y_k$</th>
<th>$\epsilon_k$</th>
<th>$W_{k+1}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>multiplies</td>
<td>$M$</td>
<td>0</td>
<td>$M + 1$</td>
<td>$2M + 1$</td>
</tr>
<tr>
<td>adds</td>
<td>$M - 1$</td>
<td>1</td>
<td>$M$</td>
<td>$2M$</td>
</tr>
</tbody>
</table>

Table 5.1

So the LMS algorithm is $O(M)$ per sample. In fact, it is nicely balanced in that the filter computation and the adaptation require the same amount of computation.

Note that the parameter $\mu$ plays a very important role in the LMS algorithm. It can also be varied with time, but usually a constant $\mu$ ("convergence weight factor") is used, chosen after experimentation for a given application.

5.11.1.1 Tradeoffs

large $\mu$: fast convergence, fast adaptivity
small $\mu$: accurate $W$ → less misadjustment error, stability

5.12 First Order Convergence Analysis of the LMS Algorithm

5.12.1 Analysis of the LMS algorithm

It is important to analyze the LMS algorithm to determine under what conditions it is stable, whether or not it converges to the Wiener solution, to determine how quickly it converges, how much degradation is suffered due to the noisy gradient, etc. In particular, we need to know how to choose the parameter $\mu$.

5.12.1.1 Mean of W

does $W_k$, $k \to \infty$ approach the Wiener solution? (since $W_k$ is always somewhat random in the approximate gradient-based LMS algorithm, we ask whether the expected value of the filter coefficients converge to the

---

14This content is available online at <http://cnx.org/content/m11830/1.1/>. Available for free at Connexions <http://cnx.org/content/col10360/1.4>/
Wiener solution)

\[
E \left[ W^{k+1} \right] = \overline{W}^{k+1} \\
= E \left[ W^k + 2\mu \epsilon_k X^k \right] \\
= \overline{W}^k + 2\mu E \left[ d_k X^k \right] + 2\mu E \left[ - \left( (W^k \tau^X) X^k \right) \right] \\
= \overline{W}^k + 2\mu P + \left[ - 2\mu E \left[ (W^k \tau^X) X^k \right] \right] \\
= \overline{W}^k + 2\mu P - \left( 2\mu \overline{R} \overline{W}^k \right) \\
\]

(5.49)

5.12.1.1 Patently False Assumption

\(X^k\) and \(X^{k-i}\), \(X^k\) and \(d^{k-i}\), and \(d_k\) and \(d_{k-i}\) are statistically independent, \(i \neq 0\). This assumption is obviously false, since \(X^{k-1}\) is the same as \(X^k\) except for shifting down the vector elements one place and adding one new sample. We make this assumption because otherwise it becomes extremely difficult to analyze the LMS algorithm. (First good analysis not making this assumption: Macchi and Eweda [1]) Many simulations and much practical experience has shown that the results one obtains with analyses based on the patently false assumption above are quite accurate in most situations.

With the independence assumption, \(W^k\) (which depends only on previous \(X^{k-i}\), \(d^{k-i}\)) is statistically independent of \(X^k\), and we can simplify \(E \left[ (W^k \tau^X) X^k \right] \)

Now \((W^k \tau^X) X^k\) is a vector, and

\[
E \left[ (W^k \tau^X) X^k \right] = E \left[ \left( \sum_{i=0}^{M-1} w_i^k x_{k-i} x_{k-j} \right) \right] \\
= \left[ \sum_{i=0}^{M-1} E \left[ w_i^k x_{k-i} x_{k-j} \right] \right] \\
= \left[ \sum_{i=0}^{M-1} w_i^k E \left[ x_{k-i} x_{k-j} \right] \right] \\
= \left[ \sum_{i=0}^{M-1} w_i^k r_{xx} (i-j) \right] \\
= R \overline{W}^k \\
\]

(5.50)

where \(R = E \left[ X^k X^k \tau^T \right] \) is the data correlation matrix.

Putting this back into our equation

\[
\overline{W}^{k+1} = \overline{W}^k + 2\mu P - \left( 2\mu \overline{R} \overline{W}^k \right) \\
= I \overline{W}^k + 2\mu P \\
\]

(5.51)
CHAPTER 5. STATISTICAL AND ADAPTIVE SIGNAL PROCESSING

Now if $W_k \to \infty$ converges to a vector of finite magnitude ("convergence in the mean"), what does it converge to?

If $W_k$ converges, then as $k \to \infty$, $W_k^{k+1} \simeq W_k$, and

$$W^\infty = I W^\infty + 2\mu P$$

$$2\mu R W^\infty = 2\mu P$$

$$R W^\infty = P$$

or

$$W_{\text{opt}} = R^{-1} P$$

the Wiener solution!

So the LMS algorithm, if it converges, gives filter coefficients which on average are the Wiener coefficients! This is, of course, a desirable result.

5.12.1.2 First-order stability

But does $W_k$ converge, or under what conditions?

Let’s rewrite the analysis in term of $V_k$, the "mean coefficient error vector" $V_k = W_k - W_{\text{opt}}$, where $W_{\text{opt}}$ is the Wiener filter

$$W_{k+1} = W_k - 2\mu R W_k + 2\mu P$$

$$W_{k+1} - W_{\text{opt}} = W_k - W_{\text{opt}} + - \left( 2\mu R W_k \right) + 2\mu R W_{\text{opt}} - 2\mu R W_{\text{opt}} + 2\mu P$$

$$V_{k+1} = V_k - 2\mu R V_k + - (2\mu R W_{\text{opt}}) + 2\mu P$$

Now $W_{\text{opt}} = R^{-1}$, so

$$V_{k+1} = V_k - 2\mu R V_k + - (2\mu RR^{-1} P) + 2\mu P = (I - 2\mu R) V_k$$

We wish to know under what conditions $V_k \to \infty \to 0$?

5.12.1.2.1 Linear Algebra Fact

Since $R$ is positive definite, real, and symmetric, all the eigenvalues are real and positive. Also, we can write $R$ as $Q^{-1} \Lambda Q$, where $\Lambda$ is a diagonal matrix with diagonal entries $\lambda_i$ equal to the eigenvalues of $R$, and $Q$ is a unitary matrix with rows equal to the eigenvectors corresponding to the eigenvalues of $R$.

Using this fact,

$$V_{k+1} = (I - 2\mu (Q^{-1} \Lambda Q)) V_k$$

multiplying both sides through on the left by $Q$, we get

$$Q V_{k+1} = (Q - 2\mu AQ) V_k = (1 - 2\mu A) Q V_k$$

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Let $V = QV$:

$$V^{k+1} = (1 - 2\mu\Lambda)V^k$$

Note that $V$ is simply $V$ in a rotated coordinate set in $\mathbb{R}^m$, so convergence of $V$ implies convergence of $V$.

Since $1 - 2\mu\Lambda$ is diagonal, all elements of $V$ evolve independently of each other. Convergence (stability) boils down to whether all $M$ of these scalar, first-order difference equations are stable, and thus $\rightarrow (0)$.

$$\forall i, i = [1, 2, \ldots, M] : (V_i^{k+1} = (1 - 2\mu\lambda_i)V_i^k)$$

These equations converge to zero if $|1 - 2\mu\lambda_i| < 1$, or $\forall i : (|\mu\lambda_i| < 1)$ so we require $\forall i : \left(\mu < \frac{1}{\lambda_i}\right)$ so for convergence in the mean of the LMS adaptive filter, we require

$$\mu < \frac{1}{\lambda_{\text{max}}} \quad (5.52)$$

This is an elegant theoretical result, but in practice, we may not know $\lambda_{\text{max}}$, it may be time-varying, and we certainly won’t want to compute it. However, another useful mathematical fact comes to the rescue...

$$\text{tr} (R) = \sum_{i=1}^{M} r_{ii} = \sum_{i=1}^{M} \lambda_i \geq \lambda_{\text{max}}$$

Since the eigenvalues are all positive and real.

For a correlation matrix, $\forall i, i \in \{1, M\} : (r_{ii} = r(0))$. So $\text{tr} (R) = Mr(0) = ME [x_k x_k]$. We can easily estimate $r(0)$ with $O(1)$ computations/sample, so in practice we might require

$$\mu < \frac{1}{Mr(0)}$$

as a conservative bound, and perhaps adapt $\mu$ accordingly with time.

### 5.12.1.3 Rate of convergence

Each of the modes decays as

$$(1 - 2\mu\lambda_i)^k$$

**Note:** The initial rate of convergence is dominated by the fastest mode $1 - 2\mu\lambda_{\text{max}}$. This is not surprising, since a gradient descent method goes "downhill" in the steepest direction

**Note:** The final rate of convergence is dominated by the slowest mode $1 - 2\mu\lambda_{\text{min}}$. For small $\lambda_{\text{min}}$, it can take a long time for LMS to converge.

Note that the convergence behavior depends on the data (via $R$). LMS converges relatively quickly for roughly equal eigenvalues. Unequal eigenvalues slow LMS down a lot.

### 5.13 Adaptive Equalization

**Note:** Design an approximate inverse filter to cancel out as much distortion as possible.

---

15This content is available online at <http://cnx.org/content/m11907/1.1/>.
In principle, $WH \approx z^{-\Delta}$, or $W \approx z^{-\frac{\Delta}{M}}$, so that the overall response of the top path is approximately $\delta(n - \Delta)$. However, limitations on the form of $W$ (FIR) and the presence of noise cause the equalization to be imperfect.

**5.13.1 Important Application**

Channel equalization in a digital communication system.

If the channel distorts the pulse shape, the matched filter will no longer be matched, intersymbol interference may increase, and the system performance will degrade.

An adaptive filter is often inserted in front of the matched filter to compensate for the channel.
This is, of course, unrealizable, since we do not have access to the original transmitted signal, \( s_k \).

There are two common solutions to this problem:

1. Periodically broadcast a known training signal. The adaptation is switched on only when the training signal is being broadcast and thus \( s_k \) is known.

2. Decision-directed feedback: If the overall system is working well, then the output \( s_k - \Delta_0 \) should almost always equal \( s_k - \Delta_0 \). We can thus use our received digital communication signal as the desired signal, since it has been cleaned of noise (we hope) by the nonlinear threshold device!

**Decision-directed equalizer**

As long as the error rate in \( s_k \) is not too high (say 75%), this method works. Otherwise, \( d_k \) is so inaccurate that the adaptive filter can never find the Wiener solution. This method is widely used in the telephone system and other digital communication networks.

Available for free at Connexions (<http://cnx.org/content/col10360/1.4>)
CHAPTER 5. STATISTICAL AND ADAPTIVE SIGNAL PROCESSING

Solutions to Exercises in Chapter 5

Solution to Exercise 5.6.1 (p. 196)

1. For \( m = 0 \), we should begin by finding the product sequence \( s[n] = x[n]y[n] \). Doing this we get the following sequence:
   \[
   s[n] = \{\ldots, 0, 0, 2, 6, 24, 1, -9, 0, 0, \ldots\}
   \]
   and so from the sum in our crosscorrelation function we arrive at the answer of
   \[ R_{xy}(0) = 22 \]

2. For \( m = 3 \), we will approach it the same way as we did above; however, we will now shift \( y[n] \) to the right. Then we can find the product sequence \( s[n] = x[n]y[n-3] \), which yields
   \[
   s[n] = \{\ldots, 0, 0, 0, 0, 0, 1, -6, 0, 0, \ldots\}
   \]
   and from the crosscorrelation function we arrive at the answer of
   \[ R_{xy}(3) = -6 \]

3. For \( m = -1 \), we will again take the same approach; however, we will now shift \( y[n] \) to the left. Then we can find the product sequence \( s[n] = x[n]y[n+1] \), which yields
   \[
   s[n] = \{\ldots, 0, 0, -4, -12, 6, -3, 0, 0, \ldots\}
   \]
   and from the crosscorrelation function we arrive at the answer of
   \[ R_{xy}(-1) = -13 \]

Solution to Exercise 5.9.1 (p. 199)

Estimate the statistics
\[
\begin{align*}
   r_{xx}(l) &\simeq \frac{1}{N} \sum_{k=0}^{N-1} x[k]x[k+l] \\
   r_{xd}(l) &\simeq \frac{1}{N} \sum_{k=0}^{N-1} d[k]x[k-l]
\end{align*}
\]
then solve \( W_{opt} = R^{-1} = P \)

Solution to Exercise 5.9.2 (p. 199)

Use short-time windowed estimates of the correlation functions.

NOTE:
\[
\begin{align*}
   (r_{xx}(l))^k &= \frac{1}{N} \sum_{m=0}^{N-1} x[k-m]x[k-m-l] \\
   (r_{dx}(l))^k &= \frac{1}{N} \sum_{m=0}^{N-1} x[k-m-l]d[k-m]
\end{align*}
\]
and \( W_{opt}^k \simeq \left( R_k \right)^{-1} P_k \)

Solution to Exercise 5.9.3 (p. 200)

Recursively!
\[
r_{xx}^k(l) = r_{xx}^{k-1}(l) + x[k]x[k-l] - x[k-N]x[k-N-l]
\]
This is critically stable, so people usually do
\[
(1 - \alpha) \left( r_{xx}^k(l) = \alpha r_{xx}^{k-1}(l) + x[k]x[k-l] \right)
\]

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Glossary

A Autocorrelation

the expected value of the product of a random variable or signal realization with a time-shifted version of itself.

C Correlation

A measure of how much one random variable depends upon the other.

Covariance

A measure of how much the deviations of two or more variables or processes match.

Crosscorrelation

If two processes are wide sense stationary, the expected value of the product of a random variable from one random process with a time-shifted, random variable from a different random process.

D Difference equation

An equation that shows the relationship between consecutive values of a sequence and the differences among them. They are often rearranged as a recursive formula so that a system’s output can be computed from the input signal and past outputs.

Example:

\[ y[n] + 7y[n-1] + 2y[n-2] = x[n] - 4x[n-1] \]  \hspace{1cm} (3.1)

F FFT

(Fast Fourier Transform) An efficient computational algorithm for computing the DFT.\(^{16}\)

P poles

1. The value(s) for \( z \) where \( Q(z) = 0 \).
2. The complex frequencies that make the overall gain of the filter transfer function infinite.

R random process

A family or ensemble of signals that correspond to every possible outcome of a certain signal measurement. Each signal in this collection is referred to as a realization or sample function of the process.

Example: As an example of a random process, let us look at the Random Sinusoidal Process below. We use \( f[n] = A \sin(\omega n + \phi) \) to represent the sinusoid with a given amplitude and phase. Note that the phase and amplitude of each sinusoid is based on a random number, thus making this a random process.

S stationary process

a random process where all of its statistical properties do not vary with time.

\(^{16}\)“Discrete Fourier Transform (DFT)" <http://cnx.org/content/m10249/latest/>

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
**Zeros**

1. The value(s) for $z$ where $P(z) = 0$.
2. The complex frequencies that make the overall gain of the filter transfer function zero.
Bibliography

Index of Keywords and Terms

**Keywords** are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. Ex. apples, § 1.1 (1) **Terms** are referenced by the page they appear on. Ex. apples, 1

A

A/D, § 2.7(63), § 2.8(73), § 2.11(96)
adaptive, 197, 199
Aliasing, § 2.3(54), § 4.5(154), § 4.7(156)
alphabet, 10
amplitude response, § 3.6(130), 130
analog, § 2.7(63), § 2.8(73), § 3.5(126)
analog signal, § 1.1(3)
analog signals, § 3.5(126)
analysis, § 4.12(165)
average, 185
auto correlation, § 5.2(182), § 5.5(192), 192
average, § 5.3(184)
average power, 185

B

bandlimited, 177, 167
basis, § 1.6(18), 21, § 1.8(29), 30
basis matrix, § 1.8(29), 31
bilateral z-transform, 114
block diagram, § 1.2(6)
blur, § 2.12(104)

c

cascade, § 1.2(6)
causal, 125
cD, § 4.3(152)
cd players, § 4.4(153)
characteristic polynomial, 112
circular convolution, § 2.11(96), 98
coefficient vector, § 1.8(29), 31
compact disc, § 4.3(152)
complement, § 1.6(18), 23
complex, § 3.4(120)
complex exponential sequence, 9
complex exponentials, 38
computational algorithm, 211
constant-Q, § 4.12(165)
continuous frequency, § 1.10(34), § 1.11(38)
continuous random process, 181
continuous time, § 1.9(33), § 1.10(34)
Continuous Time Fourier Transform, 34
Continuous-Time Fourier Transform, 35
control theory, 123
correlation, § 1.5(11), § 2.11(96), § 2.12(104)
correlation coefficient, 189
correlation functions, 190
countably infinite, 128
covariance, 188, 189
critically sampled, § 4.12(165)
Crosscorrelation, 195
crosscorrelation function, 195
CT, § 2.6(60)

D

D/A, § 2.7(63), § 2.8(73), § 2.11(96)
de-noising, § 4.16(176), 176
deblurring, § 2.12(104), 104
de-cimination, § 4.2(150), § 4.5(154), § 4.6(155), § 4.7(156), § 4.10(162)
decimator, § 4.5(154)
decompose, § 1.8(29), 30
decomposition, § 2.12(104)
delayed, 11
density function, § 5.2(182)
design, § 3.11(138)
deterministic, § 5.1(179)
deterministic signals, 179
DFT, § 1.12(41), § 1.13(44), § 2.7(63), § 2.10(91), § 2.11(96), § 3.5(126)
difference equation, § 1.4(11), 11, 109, 109, § 3.5(126)
digital, § 2.7(63), § 2.8(73), § 3.11(138)
digital audio, § 4.3(152)
digital filter, § 3.5(126)
digital signal, § 1.1(3)
digital signal processing, § 1(1), § 2.10(91), § 3.5(126)
direct method, 111
direct sum, § 1.6(18), 23
discrete fourier transform, § 2.7(63), § 2.10(91), § 2.11(96), § 3.5(126)
Discrete Fourier Transform (DFT), 64
discrete random process, 181
INDEX

available for free at Connexions <http://cnx.org/content/col10360/1.4>
mean-square value, 185
moment, 185
mother wavelet, 168
multi-resolution, 168
multiresolution, § 4.13(167)

N narrow-band spectrogram, 81
noble identities, § 4.8(158)
nonstationary, § 5.2(182), 182
normalized linear space, 28
nyquist, § 2.6(60)

O optimal, 143
order, 110
orthogonal, § 1.6(18), 25, 28
orthogonal complement, § 1.6(18), 26
orthonormal, § 1.6(18), 25, § 1.8(29)
orthonormal basis, § 1.8(29), 29
output, § 1.2(6)
oversampling, § 4.3(152), § 4.4(153)
Overview, § 2.1(49)

P parallel, § 1.2(6)
particular solution, 111
pdf, § 5.2(182)
Pearson’s Correlation Coefficient, 190
perfect reconstruction, § 4.12(165)
periodic, 167
phase delay, 132
pole, § 3.4(120)
pole-zero cancellation, 123
poles, 120
polyphase, § 4.9(159), § 4.10(162)
polyphase decimation, § 4.11(164)
polyphase interpolation, § 4.11(164)
power series, 115
probability, § 5.2(182)
probability density function (pdf), 182
probability distribution function, 182
probability function, § 5.2(182)
Proof, § 2.2(51)

R random, § 5.1(179), § 5.3(184), § 5.5(192)
random process, § 5.1(179), 180, 180, 181,
§ 5.2(182), § 5.3(184), § 5.5(192)
random sequence, 181
random signal, § 5.1(179), § 5.3(184)
random signals, § 5.1(179), 180, § 5.3(184),
§ 5.5(192)
realization, 211
Reconstruction, § 2.2(51), § 2.4(58), § 2.5(59)
Recursive least squares, 200
resampling, § 4.6(155), 155
restoration, § 2.12(104)
ROC, § 3.2(114), 115

S sample function, 211
sample-rate conversion, § 4.6(155), 155
Sampling, § 2.1(49), § 2.2(51), § 2.3(54),
§ 2.4(58), § 2.5(59), § 2.6(60), § 2.7(63)
scale function, 168, § 4.14(168), 168
second order stationary, § 5.2(182)
second-order stationary, 183
Shannon, § 2.2(51)
shift-invariant, § 1.4(11), 11, § 3.5(126)
short time fourier transform, § 2.9(78)
signal, § 1.1(3), 3, § 1.2(6)
signals, § 1.5(11), § 1.9(33)
signals and systems, § 1.5(11)
span, § 1.6(18), 20
spectral masking, 166
spectrogram, 81
spectrograms, § 2.10(91)
SSS, § 5.2(182)
stable, 125
standard basis, § 1.6(18), 27, § 1.8(29)
stationarity, § 5.2(182)
stationary, § 5.2(182), § 5.5(192)
stationary process, 182
stationary processes, 182
stft, § 2.9(78)
stochastic, § 5.1(179)
stochastic gradient, 204
stochastic signals, 180
strict sense stationary, § 5.2(182)
strict sense stationary (SSS), 183
sub-band, § 4.12(165)
sub-bands, 165
subspace, § 1.6(18), 19
superposition, § 1.4(11)
symmetries, 45
synthesis, § 4.12(165)
System, § 2.5(59)
system theory, § 1.2(6)
systems, § 1.9(33)

T The stagecoach effect, 57
time, § 2.6(60)
time-varying behavior, 75
training signal, 200
transfer function, 110, § 3.5(126), § 3.7(134)
transform pairs, § 3.3(119)
transforms, 33
twiddle factors, 45
INDEX

U
  uncorrelated, 184
  uncountably infinite, 128
  unilateral, § 3.3(119)
  unilateral z-transform, 114
  unique, 30
  unit sample, 9
  unit step, 10
  unit-sample response, 126
  unitary, § 1.6(18), 28
  upsampler, § 4.1(149), § 4.3(152)
  upsampling, § 4.1(149), 149, § 4.3(152), § 4.7(156), § 4.8(158)

V
  variance, § 5.3(184), 185
  vector, § 1.12(41)
  vectors, § 1.6(18), 19

W
  wavelet, § 4.13(167), § 4.14(168), § 4.15(170)
  wavelets, 167, 167
  well-defined, § 1.6(18), 22
  well-matched, 176
  wide sense stationary, § 5.2(182)
  wide-band spectrogram, 81
  wide-sense stationary (WSS), 184
  window, 93
  WSS, § 5.2(182)

Z
  z transform, § 1.9(33), § 3.3(119)
  z-plane, 114, § 3.4(120)
  z-transform, § 3.2(114), 114, § 3.3(119)
  z-transforms, 119
  zero, § 3.4(120)
  zero-order hold, § 4.3(152)
  zero-pad, 127
  zeros, 120
ATTRIBUTIONS

Collection: Fundamentals of Signal Processing
Edited by: Minh N. Do
URL: http://cnx.org/content/col10360/1.4/
License: http://creativecommons.org/licenses/by/3.0/

Module: "Introduction to Fundamentals of Signal Processing"
By: Minh N. Do
URL: http://cnx.org/content/m13673/1.1/
Pages: 1-2
Copyright: Minh N. Do
License: http://creativecommons.org/licenses/by/2.0/

Module: "Signals Represent Information"
By: Don Johnson
URL: http://cnx.org/content/m0001/2.27/
Pages: 3-6
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/1.0

Module: "Introduction to Systems"
By: Don Johnson
URL: http://cnx.org/content/m0005/2.19/
Pages: 6-8
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/1.0

Module: "Discrete-Time Signals and Systems"
By: Don Johnson
URL: http://cnx.org/content/m10342/2.16/
Pages: 8-11
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/3.0/

Module: "Systems in the Time-Domain"
Used here as: "Linear Time-Invariant Systems"
By: Don Johnson
URL: http://cnx.org/content/m0508/2.7/
Page: 11
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/1.0

Module: "Discrete Time Convolution"
By: Ricardo Radaelli-Sanchez, Richard Baraniuk, Stephen Kruzick, Catherine Elder
URL: http://cnx.org/content/m10087/2.30/
Pages: 11-17
Copyright: Ricardo Radaelli-Sanchez, Richard Baraniuk, Stephen Kruzick
License: http://creativecommons.org/licenses/by/4.0/

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "Review of Linear Algebra"
By: Clayton Scott
URL: http://cnx.org/content/m11948/1.2/
Pages: 18-28
Copyright: Clayton Scott
License: http://creativecommons.org/licenses/by/1.0

Module: "Hilbert Spaces"
By: Justin Romberg
URL: http://cnx.org/content/m10840/2.7/
Pages: 28-29
Copyright: Justin Romberg
License: http://creativecommons.org/licenses/by/3.0/

Module: "Orthonormal Basis Expansions"
Used here as: "Signal Expansions"
By: Michael Haag, Justin Romberg
URL: http://cnx.org/content/m10760/2.7/
Pages: 29-33
Copyright: Michael Haag, Justin Romberg
License: http://creativecommons.org/licenses/by/3.0/

Module: "Introduction to Fourier Analysis"
By: Richard Baraniuk
URL: http://cnx.org/content/m10096/2.13/
Pages: 33-34
Copyright: Richard Baraniuk
License: http://creativecommons.org/licenses/by/3.0/

Module: "Continuous Time Fourier Transform (CTFT)"
By: Richard Baraniuk, Melissa Selik
URL: http://cnx.org/content/m10098/2.17/
Pages: 34-38
Copyright: Richard Baraniuk, Melissa Selik
License: http://creativecommons.org/licenses/by/3.0/

Module: "Discrete Time Fourier Transform (DTFT)"
By: Richard Baraniuk
URL: http://cnx.org/content/m10108/2.18/
Pages: 38-41
Copyright: Richard Baraniuk
License: http://creativecommons.org/licenses/by/3.0/

Module: "DFT as a Matrix Operation"
By: Robert Nowak
URL: http://cnx.org/content/m10962/2.5/
Pages: 41-44
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "The FFT Algorithm"
By: Robert Nowak
URL: http://cnx.org/content/m10964/2.6/
Pages: 44-47
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Module: "Introduction"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m11419/1.29/
Pages: 49-51
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/1.0

Module: "Proof"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m11423/1.27/
Pages: 51-54
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/1.0

Module: "Illustrations"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m11443/1.33/
Pages: 54-58
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/1.0

Module: "Sampling and reconstruction with Matlab"
Used here as: "Sampling and Reconstruction with Matlab"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m11549/1.9/
Page: 58
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/1.0

Module: "Systems view of sampling and reconstruction"
Used here as: "Systems View of Sampling and Reconstruction"
By: Anders Gjendemsjø
URL: http://cnx.org/content/m11465/1.20/
Pages: 59-60
Copyright: Anders Gjendemsjø
License: http://creativecommons.org/licenses/by/1.0

Module: "Sampling CT Signals: A Frequency Domain Perspective"
By: Robert Nowak
URL: http://cnx.org/content/m10994/2.2/
Pages: 60-63
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0
Module: "The DFT: Frequency Domain with a Computer Analysis"
By: Robert Nowak
URL: http://cnx.org/content/m10992/2.3/
Pages: 63-72
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Module: "Discrete-Time Processing of CT Signals"
By: Robert Nowak
URL: http://cnx.org/content/m10993/2.2/
Pages: 73-78
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Module: "Short Time Fourier Transform"
By: Ivan Selesnick
URL: http://cnx.org/content/m10570/2.4/
Pages: 78-91
Copyright: Ivan Selesnick
License: http://creativecommons.org/licenses/by/1.0

Module: "Spectrograms"
By: Don Johnson
URL: http://cnx.org/content/m0505/2.21/
Pages: 91-95
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/3.0/

Module: "Filtering with the DFT"
By: Robert Nowak
URL: http://cnx.org/content/m11022/2.3/
Pages: 96-103
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Module: "Image Restoration Basics"
By: Robert Nowak
URL: http://cnx.org/content/m10972/2.2/
Pages: 104-106
Copyright: Robert Nowak
License: http://creativecommons.org/licenses/by/1.0

Module: "Difference Equation"
By: Michael Haag
URL: http://cnx.org/content/m10595/2.7/
Pages: 109-113
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/3.0/

Module: "The Z Transform: Definition"
By: Benjamin Fite
URL: http://cnx.org/content/m10549/2.10/
Pages: 114-119
Copyright: Benjamin Fite
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "Table of Common z-Transforms"
By: Melissa Selik, Richard Baraniuk
URL: http://cnx.org/content/m10119/2.14/
Pages: 119-120
Copyright: Melissa Selik, Richard Baraniuk
License: http://creativecommons.org/licenses/by/1.0

Module: "Understanding Pole/Zero Plots on the Z-Plane"
By: Michael Haag
URL: http://cnx.org/content/m10556/2.13/
Pages: 120-126
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/3.0/

Module: "Filtering in the Frequency Domain"
By: Don Johnson
URL: http://cnx.org/content/m10257/2.18/
Pages: 126-130
Copyright: Don Johnson
License: http://creativecommons.org/licenses/by/3.0/

Module: "Linear-Phase FIR Filters"
By: Ivan Selesnick
URL: http://cnx.org/content/m10705/2.3/
Pages: 130-134
Copyright: Ivan Selesnick
License: http://creativecommons.org/licenses/by/1.0

Module: "Filter Structures"
By: Douglas L. Jones
URL: http://cnx.org/content/m11917/1.3/
Page: 134
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "Overview of Digital Filter Design"
By: Douglas L. Jones
URL: http://cnx.org/content/m12776/1.2/
Pages: 134-135
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/2.0/

Module: "Window Design Method"
By: Douglas L. Jones
URL: http://cnx.org/content/m12790/1.2/
Pages: 135-136
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/2.0/
Module: "Frequency Sampling Design Method for FIR filters"
By: Douglas L. Jones
URL: http://cnx.org/content/m12789/1.2/
Pages: 136-138
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/2.0/

Module: "Parks-McClellan FIR Filter Design"
By: Douglas L. Jones
URL: http://cnx.org/content/m12799/1.3/
Pages: 138-145
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/2.0/

Module: "FIR Filter Design using MATLAB"
By: Hyeokho Choi
URL: http://cnx.org/content/m10917/2.2/
Page: 145
Copyright: Hyeokho Choi
License: http://creativecommons.org/licenses/by/1.0

Module: "MATLAB FIR Filter Design Exercise"
By: Hyeokho Choi
URL: http://cnx.org/content/m10918/2.2/
Page: 146
Copyright: Hyeokho Choi
License: http://creativecommons.org/licenses/by/1.0

Module: "Upsampling"
By: Phil Schniter
URL: http://cnx.org/content/m10403/2.15/
Pages: 149-150
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Downsampling"
By: Phil Schniter
URL: http://cnx.org/content/m10441/2.12/
Pages: 150-151
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/3.0/

Module: "Interpolation"
By: Phil Schniter
URL: http://cnx.org/content/m10444/2.14/
Pages: 152-153
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Application of Interpolation - Oversampling in CD Players"
By: Phil Schniter
URL: http://cnx.org/content/m11006/2.3/
Pages: 153-154
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "Decimation"
By: Phil Schniter
URL: http://cnx.org/content/m10445/2.11/
Pages: 154-155
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Resampling with Rational Factor"
By: Phil Schniter
URL: http://cnx.org/content/m10448/2.11/
Pages: 155-156
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Digital Filter Design for Interpolation and Decimation"
By: Phil Schniter
URL: http://cnx.org/content/m10870/2.6/
Pages: 156-158
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Noble Identities"
By: Phil Schniter
URL: http://cnx.org/content/m10432/2.12/
Pages: 158-159
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Polyphase Interpolation"
By: Phil Schniter
URL: http://cnx.org/content/m10431/2.11/
Pages: 159-162
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Polyphase Decimation Filter"
By: Phil Schniter
URL: http://cnx.org/content/m10433/2.12/
Pages: 162-164
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Computational Savings of Polyphase Interpolation/Decimation"
By: Phil Schniter
URL: http://cnx.org/content/m11008/2.2/
Pages: 164-165
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "Sub-Band Processing"
By: Phil Schniter
URL: http://cnx.org/content/m10423/2.14/
Pages: 165-167
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Discrete Wavelet Transform: Main Concepts"
By: Phil Schniter
URL: http://cnx.org/content/m10436/2.12/
Pages: 167-168
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "The Haar System as an Example of DWT"
By: Phil Schniter
URL: http://cnx.org/content/m10437/2.10/
Pages: 168-170
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Filterbanks Interpretation of the Discrete Wavelet Transform"
By: Phil Schniter
URL: http://cnx.org/content/m10474/2.6/
Pages: 170-176
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "DWT Application - De-noising"
By: Phil Schniter
URL: http://cnx.org/content/m11000/2.1/
Pages: 176-178
Copyright: Phil Schniter
License: http://creativecommons.org/licenses/by/1.0

Module: "Introduction to Random Signals and Processes"
By: Michael Haag
URL: http://cnx.org/content/m10649/2.2/
Pages: 179-181
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Module: "Stationary and Nonstationary Random Processes"
By: Michael Haag
URL: http://cnx.org/content/m10684/2.2/
Pages: 182-184
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Module: "Random Processes: Mean and Variance"
By: Michael Haag
URL: http://cnx.org/content/m10656/2.3/
Pages: 184-188
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
Module: "Correlation and Covariance of a Random Signal"
By: Michael Haag
URL: http://cnx.org/content/m10673/2.3/
Pages: 188-191
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Module: "Autocorrelation of Random Processes"
By: Michael Haag
URL: http://cnx.org/content/m10676/2.4/
Pages: 192-194
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Module: "Crosscorrelation of Random Processes"
By: Michael Haag
URL: http://cnx.org/content/m10686/2.2/
Pages: 194-196
Copyright: Michael Haag
License: http://creativecommons.org/licenses/by/1.0

Module: "Introduction to Adaptive Filters"
By: Douglas L. Jones
URL: http://cnx.org/content/m11535/1.3/
Page: 196
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "Discrete-Time, Causal Wiener Filter"
By: Douglas L. Jones
URL: http://cnx.org/content/m11825/1.1/
Pages: 196-199
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "Practical Issues in Wiener Filter Implementation"
By: Douglas L. Jones
URL: http://cnx.org/content/m11824/1.1/
Pages: 199-200
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "Quadratic Minimization and Gradient Descent"
By: Douglas L. Jones
URL: http://cnx.org/content/m11826/1.2/
Pages: 200-202
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "The LMS Adaptive Filter Algorithm"
By: Douglas L. Jones
URL: http://cnx.org/content/m11829/1.1/
Pages: 202-204
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Available for free at Connexions <http://cnx.org/content/col10360/1.4>
ATTRIBUTIONS

Module: "First Order Convergence Analysis of the LMS Algorithm"
By: Douglas L. Jones
URL: http://cnx.org/content/m11830/1.1/
Pages: 204-207
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0

Module: "Adaptive Equalization"
By: Douglas L. Jones
URL: http://cnx.org/content/m11907/1.1/
Pages: 207-209
Copyright: Douglas L. Jones
License: http://creativecommons.org/licenses/by/1.0
**Fundamentals of Signal Processing**

Presents fundamental concepts and tools in signal processing including: linear and shift-invariant systems, vector spaces and signal expansions, Fourier transforms, sampling, spectral and time-frequency analyses, digital filtering, z-transform, random signals and processes, Wiener and adaptive filters.

**About Connexions**

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions’s modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.