Chapter 9

Filter Design, Multirate, and Correlation

Chapter 7 of this book presented the discrete-time counterparts to the continuous-time results in Chapters 1 through 6. Chapter 8 presented several applications of these discrete-time results. Chapter 9 now presents additional topics in discrete-time signal processing that have no obvious continuous-time counterparts, but are usually treated in discrete-time signal processing. These include: *spectral leakage* and the resulting need for *data windows*; *spectrograms*, which are time-varying spectra computed using a series of data windows centered at different times; discrete-time filter design, which has obvious applications in discrete-time signal processing, and one of the techniques for which requires data windows; *downsampling* and *upsampling*, which require discrete-time filters and which will be used extensively in Chapter 10 for discrete-time *wavelet transforms*; and various forms of *correlation*, which are used to compute the period of a periodic signal, the time delay of a signal, and to classify a signal as one of several possible types. Applications of correlation to three biomedical signal processing problems are also included.

9.1 Data Windows

9.1.1 Preliminaries

Review of Spectral Leakage Phenomenon

Consider the use of the DFT to compute the spectrum of a pure sinusoid from its samples in an interval of finite length. From Table 7-8, the DTFT of the pure sinusoid $\cos(\Omega_o n)$ is a pair of impulses at Ω_o and $-\Omega_o$. But in Section 8-6 we found that the DFT of $\cos(\Omega_o n)$ was nonzero at frequencies other than Ω_o and $-\Omega_o$. This was due to the DFT using only a finite number of samples of $\cos(\Omega_o n)$.

We have seen other examples of this. In Fig. 8-21, the DFT-computed spectrum of a pure sinusoid consisted of a pair of lines with a "base spread" around them. In Fig. 6-33, the spectrum of a trumpet signal (computed using the DFT) consisted of a set of spectral lines (harmonics) with "noise-like underbrush" labelled in the figure.

If the interval over which samples of the sinusoid are known is short, this problem is more serious. For example, the one-sided spectrum of the segment $\{\cos(0.2\pi n), 0 \le n \le 24\}$ is shown in Fig. 9-1.

There is a broad peak centered at $\Omega = 0.2\pi$, but there are also many smaller peaks over the entire frequency range $0 \le \Omega \le \pi$. These smaller peaks could mask another sinusoid (see Example 9-2 below).

The goal of this section is to analyze this phenomenon, which was called *spectral leakage* in Section

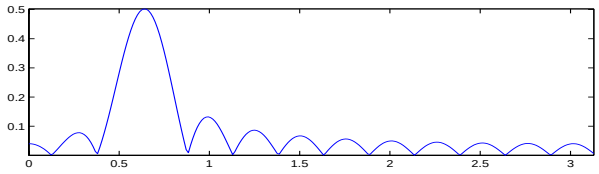


Figure 9.1: Spectrum of $\{\cos(0.2\pi n), 0 \le n \le 24\}$.

8-6.2, and determine how to reduce it.

Review of Periodic Extension of an Integral Number of Periods

Define the *segment* of length L of the pure sinusoid $\{\cos(\Omega_o n), -\infty < n < \infty\}$ as the set of L values $\{\cos(\Omega_o n), 0 \le n \le L-1\}$. We are given samples of the sinusoid only for $0 \le n \le L-1$. This is what happens in practice–we do not have samples from all past and future times n.

Section 8-6.2 showed spectral leakage occurs unless the segment $\{\cos(\Omega_o n), 0 \le n \le L-1\}$ consists of an integer number of periods of the sinusoid. Then the periodic extension of this segment is $\{\cos(\Omega_o n), -\infty < n < \infty\}$ and the L-point DFT of the segment is zero except at the two indices corresponding to Ω_o and $-\Omega_o$.

But in practice the segment does not consist of an integer number of periods of the sinusoid, and the periodic extension of the segment is not $\{\cos(\Omega_o n), -\infty < n < \infty\}$. This was illustrated in Fig. 8-22(b). We show next that the spectrum of the segment still has peaks at Ω_o and $-\Omega_o$, but the peaks seem to leak out at their bases.

Having identified the problem of spectral leakage, we now show how to alleviate it. We will show that we can virtually eliminate the base spread observed in the computed spectrum in Fig. 8-21(b), at the price of making the peak broader.

9.1.2 Spectrum of Sinusoidal Segment

Review of Rectangular Window

Define the *rectangular window* $r[n]$ of length L as

$$r[n] = \begin{cases} 1 & \text{for } 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \tag{9.1}$$

$r[n]$ is the rectangular pulse defined in Eq. (7.162) delayed by N, with $L-1 = 2N$.

The DTFT of the rectangular pulse defined in Eq. (7.162) is the discrete sinc function defined in Eq. (7.164). Three periods of the discrete sinc function are plotted in Fig. 7-21b. Seting $N = (L-1)/2$ in Eq. (7.164) and using the time delay property in Table 7-7 of the DTFT, the DTFT $\mathbf{R}(e^{j\Omega})$ of $r[n]$ is

$$\mathbf{R}(e^{j\Omega}) = \frac{\sin(\Omega L/2)}{\sin(\Omega/2)} e^{-j\Omega(L-1)/2}. \tag{9.2}$$

The magnitude $|\mathbf{R}(e^{j\Omega})|$ is plotted in Fig. 9-2. Note the resemblance to one period of Fig. 7-21b.

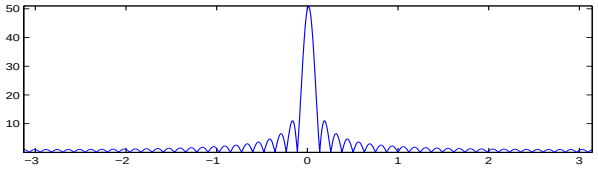


Figure 9.2: Magnitude $|\mathbf{R}(e^{j\Omega})|$ of Discrete Sinc Function.

DTFT of Sinusoidal Segment

We are given the segment $\{\cos(\Omega_o n), 0 \le n \le L-1\}$. Let $x[n]$ be defined from this segment by setting all undefined values (outside $0 \le n \le L-1$) to zero:

$$x[n] = \begin{cases} \cos(\Omega_o n) & \text{for } 0 \le n \le L-1 \\ 0 & \text{otherwise.} \end{cases} \tag{9.3}$$

$x[n]$ is known since the segment $\{\cos(\Omega_o n), 0 \le n \le L-1\}$ is known.

Multiplication of any signal by $r[n]$ sets it to zero outside the interval $0 \le n \le L-1$, and does not affect the signal inside this interval. We may express $x[n]$ as

$$\begin{aligned} x[n] &= \cos(\Omega_o n) r[n] \\ &= r[n] e^{j\Omega_o n}/2 + r[n] e^{-j\Omega_o n}/2. \end{aligned} \tag{9.4}$$

Using the frequency shift property in Table 7-7 of the DTFT, the DTFT $\mathbf{X}(e^{j\Omega})$ of $x[n]$ is

$$\mathbf{X}(e^{j\Omega}) = \mathbf{R}(e^{j(\Omega-\Omega_o)})/2 + \mathbf{R}(e^{j(\Omega+\Omega_o)})/2. \tag{9.5}$$

The spectrum of the sinusoidal segment consists of two discrete sinc functions, centered at Ω_o and $-\Omega_o$. The "base spread" is the magnitude of a discrete sinc function centered on the frequency of the sinusoid. Spectral leakage is actually due to each line in a line spectrum being replaced with a discrete sinc function.

Non-Rectangular Window

Consider the effect of multiplying the segment $\{\cos(\Omega_o n), 0 \le n \le L-1\}$ by a function $w[n] \neq r[n]$ that is also zero outside $0 \le n \le L-1$. $w[n]$ has the general form

$$w[n] \begin{cases} \neq 1 & \text{for } 0 \le n \le L-1 \\ = 0 & \text{otherwise} \end{cases}. \qquad (9.6)$$

Now define $y[n]$ as

$$\begin{aligned} y[n] &= \begin{cases} \cos(\Omega_o n) w[n] & \text{for } 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \\ &= \cos(\Omega_o n) w[n]. \end{aligned} \qquad (9.7)$$

$y[n]$ is known since the segment $\{\cos(\Omega_o n), 0 \le n \le L-1\}$ is known.

It may seem counterintuitive to alter the L samples we do have. But repeating the derivation of Eq. (9.2), the DTFT $\mathbf{Y}(e^{j\Omega})$ of $y[n]$ is

$$\mathbf{Y}(e^{j\Omega}) = \mathbf{W}(e^{j(\Omega-\Omega_o)})/2 + \mathbf{W}(e^{j(\Omega+\Omega_o)})/2 \qquad (9.8)$$

where $\mathbf{W}(e^{j\Omega})$ is the DTFT of $w[n]$. The spectrum of $y[n]$ is a pair of DTFT's $\mathbf{W}(e^{j\Omega})$ centered at Ω_o and $-\Omega_o$.

The point is that $\mathbf{W}(e^{j\Omega})$ need not be a discrete sinc function. It can be any function whose inverse DTFT $w[n]$ is zero outside the range $0 \le n \le L-1$. Some choices of $w[n]$ are tabulated in Table 9-2 below. In general, $w[n]$ should taper to zero (or near zero) at each endpoint $n = 0$ and $n = L-1$, and bulge upward to a maximum value of one at $n = (L-1)/2$ (if this is an integer).

$w[n]$ is called a *data window*. Multiplying the given data by $w[n]$ is called *windowing the data*. So "window" is both a noun and a verb. It is important to remember that the data is *multiplied* by $w[n]$, not convolved with it.

> To use the window $w[n]$ on data segment $x[n]$, compute the DFT of $w[n]x[n]$, **not** $w[n] * x[n]$!

9.1.3 Choices of Data Windows

In the remainder of this section, L is the (fixed) length of the data segment, and N is the order of the DFT. Increasing N is equivalent to discretizing the DTFT to more points N in the range $|\Omega| \le \pi$, as noted in Section 7-15.3. It is also equivalent to zero-padding the data segment of length L with zeros to create a segment of length N (see Section 7-15.2). We define the *main lobe* and *side lobes* of $\mathbf{W}(e^{j\Omega})$ in Fig. 9-3.

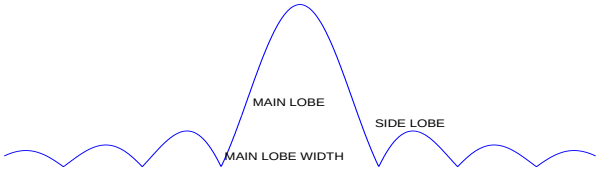


Figure 9.3: Definitions of Main Lobe and Side Lobes.

The two main measures used for data windows are:

- The ratio of the height of the first (and highest) side lobe to the height of the main lobe. This measures how well the window reduces side lobes.
- The width of the main lobe. This measures the resolution of the window–the wider the main lobe, the poorer the resolution.

There is a tradoff between these measures.

The following table summarizes features of some commonly-used windows. Note that all of these windows have their maximum value of one in the middle of their range, and taper to zero (or near zero) at their endpoints. In this table:

- "Formula" is the formula for the window $w[n]$ for $0 \le n \le L$. In this table, L is one less than the length of the window, but this makes the formulae simpler. $w[n] = 0$ outside the range $0 \le n \le L$, reflecting that the data segment is unknown outside the range $0 \le n \le L$.
- "Main" is the width of the main lobe, measured between its first zero crossings on either side of $\Omega = 0$.

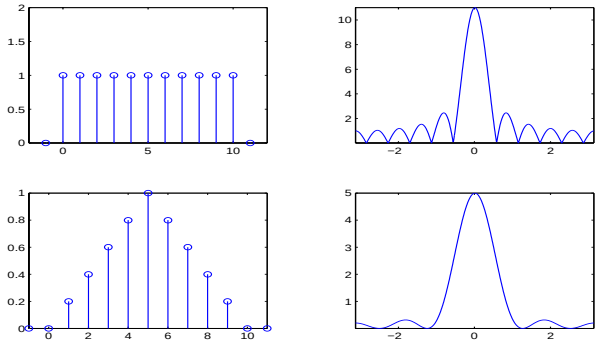
- "Side" is the ratio (in dB) $20\log_{10}\frac{1^{\text{st}}\text{ SIDE LOBE}}{\text{MAIN LOBE}}$ of the height of the first side lobe relative to the height of the main lobe.

Name	Formula	Main	Side
Rectangle	1	$4\pi/L$	–13
Bartlett	$1–2\lvert n-\frac{L}{2}\rvert/L$	$8\pi/L$	–27
Hanning	$0.50–0.50\cos(\frac{2\pi n}{L})$	$8\pi/L$	–32
Hamming	$0.54–0.46\cos(\frac{2\pi n}{L})$	$8\pi/L$	–43
Blackman	$0.42\,^{-0.50\cos(\frac{2\pi n}{L})}_{+0.08\cos(\frac{4\pi n}{L})}$	$12\pi/L$	–58

Table 9-1: Summary of Common Window Functions.

The width of the main lobe decreases as the length of the segment increases. For long segments, the main lobe appears on a plot of the spectrum of the segment as a narrow peak, or even a line. But the side lobes do not decrease in size as the length of the segment increases–there is still spectral leakage.

Four of these windows, and their DTFTs, are shown in Fig. 9-4 for $L = 10$.



MATLAB code used for this figure:

```
clear;figure
N=[-1:11];K=[-255:256];O=K/256*pi;
R=[0 ones(1,11) 0];FR=fftshift(abs(fft(R,512)));
subplot(221),stem(N,R),axis([-2 12 0 2])
subplot(222),plot(O,FR),axis tight
B=[0 [0:0.2:1] [0.8:-0.2:0]
0];FB=fftshift(abs(fft(B,512)));
subplot(223),stem(N,B),axis tight
subplot(224),plot(O,FB),axis tight
figure
```

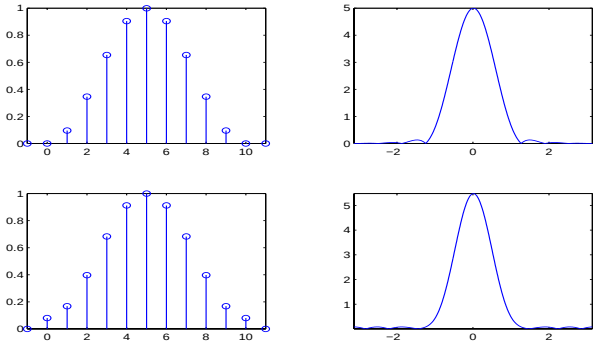


Figure 9.4: Windows of Lengths $L = 10$ and their DTFTs. (a) Rectangular (b) Bartlett (c) Hanning (d) Hamming

```
HN=[0 0.5-0.5*cos(pi/5*[0:10])
0];FHN=fftshift(abs(fft(HN,512)));
subplot(221),stem(N,HN),axis tight
subplot(222),plot(O,FHN),axis tight
HM=[0 0.54-0.46*cos(pi/5*[0:10])
0];FHM=fftshift(abs(fft(HM,512)));
subplot(223),stem(N,HM),axis tight
subplot(224),plot(O,FHM),axis tight
```

A rectangular window means no window is used. A Bartlett window is a triangle centered on the middle of the data segment. Hanning and Hamming windows are also called "raised cosines" since they are both cosines plus constants. The slightly different values used in the Hamming window do a better job of suppressing side lobes, as Table 9-1 shows. The tradeoff between resolution and sidelobe suppression is apparent. And yes, two people named Hanning and Hamming both worked on this problem!

In practice, the Hamming window is the most commonly used window, since it is an intermediate choice. Rectangular, Bartlett, and Hanning windows are analyzed in Problems 9-5 and 9-6.

Example 9-1: Using a Data Window.

Use a 256-point DFT to compute the one-sided spectrum of the segment $\{\cos(0.2\pi n), 0 \le n \le 24\}$ using: (1) a rectangular window; and (2) a Hamming window.

Solution:

The results are shown in Fig. 9-5. The second plot, using a Hamming window, has a main peak twice as wide as the first plot. But the sidelobes are now barely apparent.

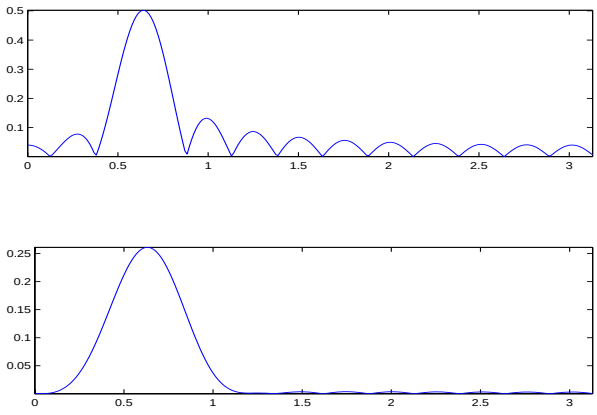


Figure 9.5: Computed Spectrum using (a) Rectangular Window (b) Hamming Window.

MATLAB code used for this figure:

```
clear;N=512;K=[1:N/2];W=2*pi*(K-1)/N;
X=cos(0.2*pi*[0:24]);
FX=abs(fft(X,N))/25;
figure,subplot(211),plot(W,FX(K)),axis
tight
H=0.54-0.46*cos(2*pi*[0:24]/24);Y=X.*H;
FY=abs(fft(Y,N))/25;
figure,subplot(211),plot(W,FY(K)),axis
tight
```

9.1.4 Significance of Data Windows

It may seem that some base spreading of spectral lines is not a major issue. But the following example shows that it is.

Example 9-2: Significance of Data Windows.
A 256-point DFT is used to compute the one-sided spectrum of a segment of length 75 of the sum of several sinusoids using: (1) a rectangular window; and (2) a Hamming window. How many sinusoids are present?

Solution:

Results are shown in Fig. 9-6. The first plot, using a rectangular window, seems to suggest that three sinusoids are present. The second plot, using a Hamming window, shows that there are actually four sinusoids present. The 4^{th} peak was lost in the side lobes when a rectangular window was used!

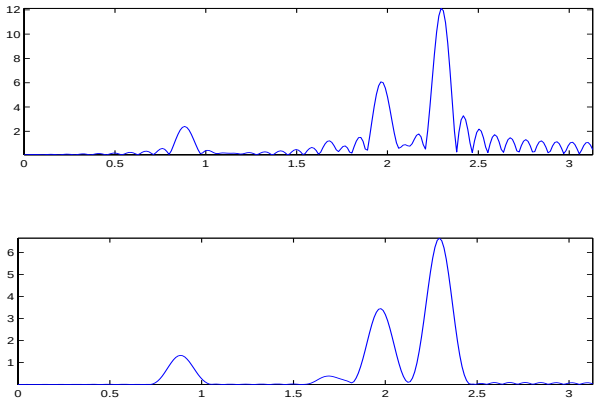


Figure 9.6: Spectrum Computed using (a) Rectangular Window. There Seem to be 3 Sinusoids. (b) Hamming Window. There are Actually 4 Sinusoids.

MATLAB code used for this figure:

```
clear;load p8.mat;L=length(X1);
N=512;K=[1:N/2];W=2*pi*(K-1)/N;
FX1=abs(fft(X1,N))/L;
figure,subplot(211),plot(W,FX1(K)),axis
tight
H=0.54-0.46*cos(2*pi*[0:L-1]/(L-1));Y1=X1.*H;
FY1=abs(fft(Y1,N))/L;
figure,subplot(211),plot(W,FY1(K)),axis
tight
```

However, data windows are not always beneficial. While they reduce side lobes, they also broaden the main lobe, and this reduces *resolution*. Resolution is the ability to detect two closely-spaced sinusoids in a short segment.

We investigate resolution in the following example.

Example 9-3: Resolving Two Closely-Spaced Sinusoids
Use an N-point DFT to compute the spectrum of a segment of length L of the sum of two sinusoids at

frequencies Ω=0.3π and Ω=0.31π using

1. $L = 125, N = 512$ and a rectangular window
2. $L = 125, N = 1024$ and a rectangular window
3. $L = 140, N = 512$ and a rectangular window
4. $L = 140, N = 512$ and a Hamming window

Solution:

Results are plotted in Fig. 9-7.

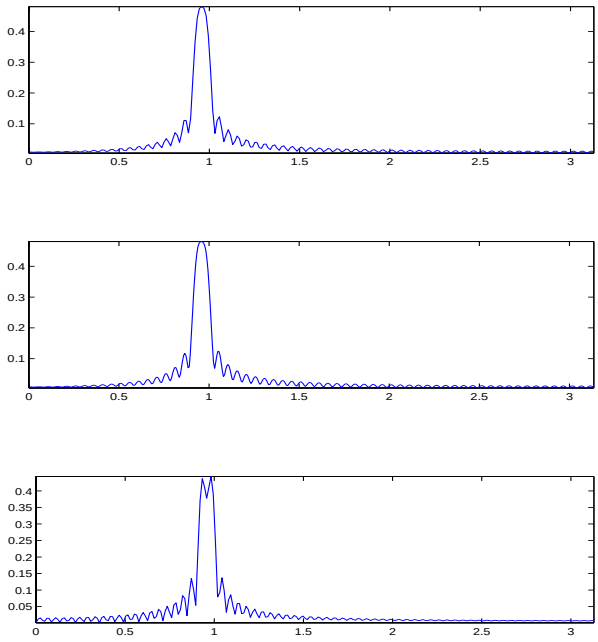
In Fig 9-7a, only a single peak is present. The two sinusoids are not resolved.

In Fig 9-7b, increasing the DFT order N means a finer discretization of the DTFT. The two sinusoids are not resolved, so this is not due to an insufficiently fine discretization.

In Fig 9-7c, increasing the data segment length L successfully resolved the two sinusoids.

In Fig 9-7d, the Hamming window produced broader main lobes, so the two sinusoids were not resolved, despite the longer data segment.

This is investigated further in problems 9-1 through 9-4.



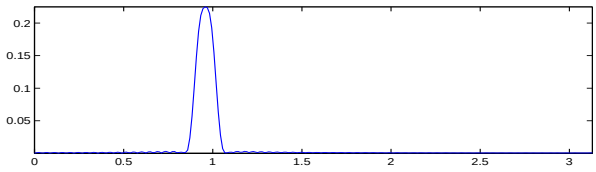


Figure 9.7: Computed Spectrum of Segment of Sum of two Sinusoids for: (a) $L = 125$ and $N = 512$; (b) $L = 125$ and $N = 1024$; (c) $L = 140$ and $N = 512$; (d) $L = 140$ and $N = 512$ with a Hamming Window.

MATLAB code used for this figure:

```
clear;
L=125;N=512;X=cos(0.3*pi*[0:L-1])+cos(0.31*pi*[0:L-1]
K=[1:N/2];W=(K-1)*2*pi/N;figure,subplot(211),plot(W,F
tight
L=125;N=1024;X=cos(0.3*pi*[0:L-1])+cos(0.31*pi*[0:L-1
K=[1:N/2];W=(K-1)*2*pi/N;figure,subplot(211),plot(W,F
tight
L=140;N=512;X=cos(0.3*pi*[0:L-1])+cos(0.31*pi*[0:L-1]
K=[1:N/2];W=(K-1)*2*pi/N;figure,subplot(211),plot(W,F
tight
L=140;N=512;X=cos(0.3*pi*[0:L-1])+cos(0.31*pi*[0:L-1]
K=[1:N/2];W=(K-1)*2*pi/N;figure,subplot(211),plot(W,F
tight
```

To summarize windows and resolution:

- Multiply the segment and the window function point-by-point. Don't convolve them!
- Increasing the DFT order N does not help to resolve two spectral peaks.
- Increasing the segment length L does help to resolve two spectral peaks.
- Using a data window does not help resolve two spectral peaks; in fact, it reduces resolution, since the main lobe is wider.
- But using a data window reduces side lobes. This makes the computed spectrum easier to interpret, especially if small peaks are present.

Exercise 9-1: Compute the coefficients of a 5-point Bartlett window.

Answer: $\{0, \frac{1}{2}, 1, \frac{1}{2}, 0\}$

Exercise 9-2: Compute the coefficients of a 5-point Hamming window.
Answer: $\{0.08, 0.54, 1.00, 0.54, 0.08\}$

9.2 Spectrograms

The problem is now to compute signal spectra that vary with time. While a time-varying spectrum is a contradiction in terms, some signals have useful interpretations as having time-varying spectra. These include:

- Musical signals, in which the note played by, say, a trumpet changes at specific times, so that its line spectrum shifts frequencies abruptly at specific times;
- Radar and sonar signals altered by *Doppler* shifts. The Doppler shift determines motion, relative to the antenna, which may vary with time.
- *Chirp signals*, which are sinusoidal signals whose frequency changes linearly with time. Bird chirps and dolphin clicks can be modelled well as chirps.

Spectrograms are a useful and simple tool for analyzing signals with time-varying spectra. Spectrograms are computed by computing spectra of segments of the signal using a series of data windows, with varying delays, plotting the computed spectra alongside each other, and then viewing the result from above as an image.

9.2.1 Definition of Spectrogram

Problem Statement

Let $x(t)$ have the *time-varying spectrum*

$$\begin{aligned} x(t) &= \sum_{k=1}^{\infty} A_{k,1} \cos\left(2\pi f_{k,1} t + \theta_{k,1}\right), T_0 < t < T_1 \\ &= \sum_{k=1}^{\infty} A_{k,2} \cos\left(2\pi f_{k,2} t + \theta_{k,2}\right), T_1 < t < T_2 \\ &= \sum_{k=1}^{\infty} A_{k,3} \cos\left(2\pi f_{k,3} t + \theta_{k,3}\right), T_2 < t < T_3 \\ &\vdots \quad \vdots \quad \vdots \end{aligned} \tag{9.9}$$

Musical signals have this form:

- The possible durations T_{i+1}–T_i of each note are known. All whole notes have the same duration; all half notes have half the duration of whole notes, etc. So there is no difficulty in segmenting the signal into different intervals;
- The frequencies $f_{k,i}$ can only take on twelve different values, all known, in an octave.

More generally, the times $\{T_i\}$ and frequencies $f_{k,i}$ are unknown.

Continuous-Time Spectrogram

The spectrogram computes the spectrum (squared magnitude of the Fourier transform) $|\mathbf{X}(\omega)|^2$ of each segment $\{x(t), T_i < t < T_{i+1}\}$ of the signal. The signal is sampled, and the spectrum of each segment computed using the DFT with a data window over that interval. Since the T_i are usually unknown, intervals $\{t : \tau - T/2 < t < \tau + T/2\}$ of equal lengths T are used, centered at times τ. In continuous time, the spectrogram $S(\omega, \tau)$ is

$$S(\omega, \tau) = |\int_{\tau - T/2}^{\tau + T/2} w(t - \tau + T/2) x(t) e^{-j\omega t} dt|^2. \tag{9.10}$$

The squared Fourier transform magnitude is used so that $S(\omega, \tau)$ has units of power. $S(\omega, \tau)$ is displayed as an image, so that peaks appear as bright pixels.

Discrete-Time Spectrogram

In discrete time, the signal is sampled to $x[n]$, a data window $w[n]$ (such as those in Section 9-2) is used, and the intervals $\{N - L/2 \le n \le N + L/2\}$ have lengths $L + 1$, where L is even, and are centered at N. In discrete time, the spectrogram $S(e^{j\Omega}, N)$ is

$$S(e^{j\Omega}, N) = |\sum_{n=N-L/2}^{N+L/2} w[n - N + L/2] x[n] e^{-j\Omega n}|^2. \tag{9.11}$$

Here, we consider only non-overlapping intervals and use a rectangular window. Choosing a non-rectangular window does not affect significantly the display of $S(e^{j\Omega}, N)$ as an image.

Matlab/Mathscript Recipe for Spectrograms

To display as an image the spectrogram, using intervals of length `L`, of a signal sampled at S samples per second and stored in Matlab/Mathscript as vector `X`, apply the following code:

```
LX=length(X);%L=interval length
  XX=reshape(X,L,length(X)/L);
FXX=abs(fft(XX)).*abs(fft(XX));
          imagesc(FXX)
```

The length of `X` must be a multiple of the interval length L, so that an integer number of intervals can be used. The horizontal axis of the image is the time of the left end (not the center) of the interval over which the spectrum is computed. The horizontal axis should be multiplied by $\frac{1}{S}$ seconds per sample (sic) to be interpreted as seconds.

The vertical axis of the image is the frequency axis for the spectrum displayed in each vertical slice of the spectrogram. The vertical axis should be multiplied by S/L Hertz (see Subsection 8-6.1 and Example 9-? below). Matlab/Mathscript displays images with the origin at the upper left corner, but since $S(e^{j\Omega}, N)$ is conjugate symmetric in the vertical axis the lowerleft corner can also be treated as the origin.

9.2.2 Motivation for Spectrograms

A motivation for using spectrograms is given in the following example

Example 9-4: Motivation for Spectrograms

We wish to interpret a music signal sampled at 8192 samples per second. The signal consists entirely of whole notes of durations 0.3662 seconds each.

Solution:

The line spectrum of the signal is plotted in Fig. 9-8. This is not much help; it only provides a histogram of the musical notes. The spectral lines heights indicate *how often* that note was played, but not *when* it was played.

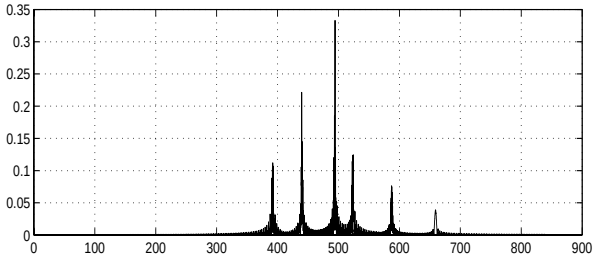


Figure 9.8: Line Spectrum of Music Signal.

The spectrogram of the signal, using 26 segments of lengths 3000 each, is depicted in Fig. 9-9. The duration of each segment is $\frac{L}{S} = \frac{3000}{8192} = 0.3662$ secs. The spectrum of each segment is one sinusoid. The vertical position of the line indicates the frequency of this sinusoid. To get frequency in Hertz, multiply the vertical index by $\frac{S}{L} = \frac{8192}{3000}$ Hertz.

The spectrogram shows that the signal is a piece of music. It is a pure tonal version of the chorus of "The Victors," the fight song of the University of Michigan. Another example of this type, in which the spectrogram is used to identify two fight songs, making it possible to eliminate one of them, is in problem 9-12.

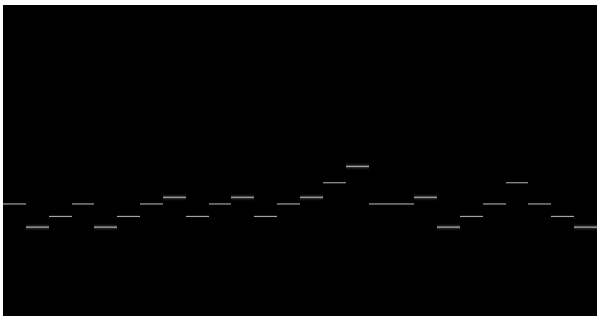


Figure 9.9: Spectrogram of Music Signal.

MATLAB code used for this figure:

```
load victorstone.mat;LX=length(X);S=8192;
FX=2/LX*abs(fft(X));F=[0:LX-1]*S/LX;
figure,subplot(211),plot(F(1:8000),FX(1:8000))
```

```
L=26;N=LX/L;XX=reshape(X',N,L);
FXX=abs(fft(XX));FXX=FXX(5*N/6:N,:);
figure,subplot(211),imagesc(FXX)
colormap(gray),axis off
```

Figure 9.10: Spectrogram with 13 Segments.

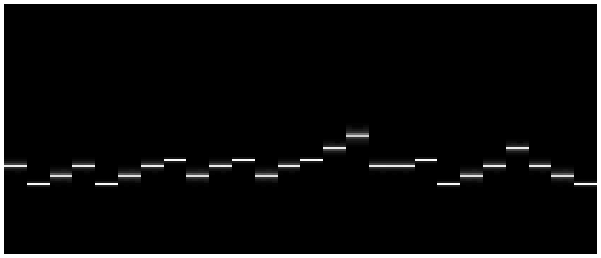


Figure 9.11: Spectrogram with 104 Segments.

9.2.3 Time vs. Frequency Resolution Tradeoff

Let the duration N of the sampled signal $x[n]$ be factored as $N = LM$. Then the spectrogram can be computed using M non-overlapping intervals of lengths L each, for any factorization $N = LM$. Using different factorizations allows a tradeoff between time and frequency resolution:

Smaller L and larger M: The shorter the intervals, the finer the resolution in time. Changes in spectra can be tracked more quickly. However, the shorter length of each interval creates coarser resolution in frequency (see Example 9-3 above). Single sinusoids will have broader peaks, and closely-spaced sinusoids may not be resolved.

Smaller M and larger L: Using longer intervals improves the frequency resolution. But now the time resolution is coarser–changes in the spectrum will not be tracked as well.

Example 9-5: Time vs. Frequency Resolution Tradeoff

Redo Example 9-4 using (a) 13 segments of lengths 6000 samples each; and (b) 104 segments of lengths 750 samples each.

Solution:

(a) The spectrogram using only 13 segments is depicted in Fig. 9-10. Each segment now contains two sinusoids, representing two notes. A musician would have to guess which note to play in each segment!

(b) The spectrogram using 104 segments is depicted in Fig. 9-11. Each note now lasts through four segments, but the line representing each sinusoid is smeared out, representing a broader peak. The frequency of each note is not indicated as sharply as in (a).

9.2.4 Chirp Signal

Definition of Chirp

A *chirp* signal is a sinusoid whose frequency increases or decreases linearly with time. Bird chirps and dolphin clicks are naturally occurring chirp signals.

For notational convenience, we will consider only chirps whose frequency increases linearly with time, starting with 0 Hz at time $t = 0$. The general form of such a chirp is

$$\begin{aligned} x(t) &= A\cos(\omega_o t^2)u(t) \\ &= A\cos((\omega_o t)t)u(t) \end{aligned} \tag{9.12}$$

This suggests that a chirp is a sinusoid whose frequency at time t is $\omega_o t$ radians/second. In fact, Example 9-6 below shows that this is incorrect–the spectrogram of a chirp is a line with slope $2\omega_o$, so the frequency at time t is actually $2\omega_o t$ radians/second.

Example 9-6: Spectrogram of a chirp

Compute the spectrogram of the chirp $\{\cos(t^2), 0 \leq t \leq 81.91\}$ seconds, using a sampling rate of 100 sam-

ples/second.

Solution:

The first 20 seconds of the chirp is shown in Fig. 9-12.

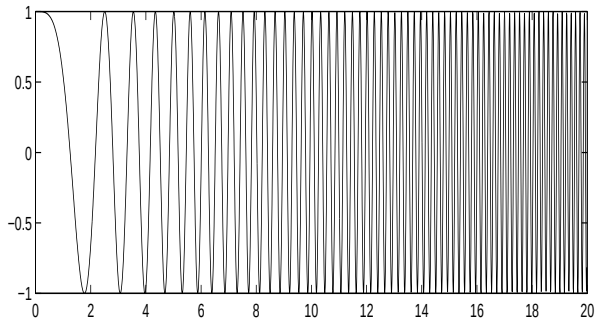


Figure 9.12: Chirp Signal $\cos(t^2)$.

The chirp clearly does look like a sinusoid whose frequency is increasing with time.

The sampling rate is 100 samples/second, so set $t = n/100$ and use $0 \leq n \leq (81.91)(100) = 8191$. The sampled signal is then

$$x[n] = \{\cos(n^2/10000), 0 \leq n \leq 8191\}. \qquad (9.13)$$

The length of the data segment is 8192=(32)(256). To compute the spectrogram, we use 32 intervals of lengths 256 each. The spectrogram is depicted in Fig. 9-13.

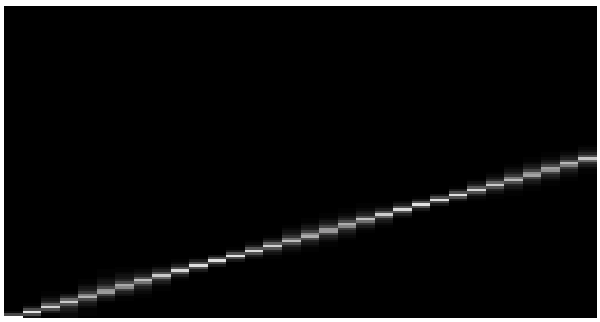


Figure 9.13: Spectrogram of Chirp.

The spectrogram indicates that the chirp is a sinusoid whose frequency increases linearly with time. The intervals make the spectrogram look like a staircase. Using more and shorter intervals would reduce the staircase effect, but would also blur each step of the staircase, since a shorter interval is used to compute each spectrum. Using a Hamming window produces no visible change.

MATLAB code used for this figure:

```
X=cos([0:8191].*[0:8191]/10000);
T=linspace(0,19.99,2000);
subplot(211),plot(T,X(1:2000))
FXX=abs(fft(reshape(X,256,32)));
FXX=FXX(129:256,:);
figure,subplot(211),imagesc(FXX)
axis off,colormap(gray)
```

Instantaneous Frequency

Putting $\cos(t^2)$ into the form of (9.12) results in $\omega_o = 1$. After 81.91 seconds, the final frequency of the chirp seems to be 81.91 radians/second, or 13 Hz. However, the rightmost step in the spectrogram is centered at vertical index 67, measured from the bottom. From Subsection 8-6.1, this corresponds to a final frequency of

$$(67-1)\frac{100\,\text{samples/second}}{256\,\text{samples}} = 26\,\text{Hz}, \qquad (9.14)$$

double what the form of (9.12) suggested. In fact, the instantaneous frequency of (9.12) is $2\omega_o t$, not $\omega_o t$, radians/second. The reason for this is that the instantaneous phase at time t is $\omega_o t^2$, and the instantaneous frequency is $\frac{d}{dt}\omega_o t^2 = 2\omega_o t$. This topic is addressed in higher-level courses.

Spectrograms are the simplest (and most commonly used) type of *time-frequency distribution*. There is much more to this field, but it belongs in a graduate-level course.

9.3 FIR Filter Design

9.3.1 Overview of Discrete-Time Filter Design

A *discrete-time filter* is an LTI system that emphasizes some desired frequency components of a signal

while reducing or eliminating others. Lowpass filtering is a common discrete-time filter task. Lowpass filtering will be needed in Section 9-6 on multirate filtering; it can also be used for denoising, although discrete-time wavelets (see Chapter 10) will prove to be more effective at denoising. Discrete-time filtering can also be used to perform tasks such as differentiation (e.g., computing speed from distance travelled) and integration of signals from their samples.

In Chapter 7, we showed how placing poles and zeros can be used to design notch and comb filters. But an attempt to design a half-band lowpass filter showed that placing poles and zeros is not the best way to design filters other than notch or comb. Other techniques are needed.

This chapter covers design procedures for discrete-time filters. There are two major types of filters:

- *FIR filters*, in which the impulse response $h[n]$ has finite duration;
- *IIR filters*, in which the impulse response $h[n]$ has infinite duration.

FIR filtering has mostly supplanted IIR filtering in practice. We will cover three common techniques for FIR filtering, and two for IIR filtering. FIR filtering belongs in this chapter because the first FIR filter design technique applies data windows to the inverse DTFT of the desired frequency response function.

9.3.2 Overview of FIR Filter Design

An FIR filter has the form

$$y[n] = b_0x[n]+b_1x[n-1]+\ldots+b_Mx[n-M]. \quad (9.15)$$

for some constant coefficients $\{b_n, 0 \le n \le M\}$ and constant order M. Note that the impulse response $h[n] = b_n$ for $0 \le n \le M$.

An FIR filter is almost the same as an MA system (see Section 7-3). The only difference is that an MA system must be causal, while an FIR filter need not be causal. A noncausal FIR filter can be implemented by delaying it to make it causal. Suppose $h[n] \neq 0$ for $\{-L \le n \le L\}$. Let $\tilde{h}[n] = h[n-L]$ be $h[n]$ delayed by L. Then $\tilde{h}[n]$ is causal, and

$$\begin{aligned} y[n] &= h[n] * x[n] = \sum_{i=-L}^{L} h[i]x[n-i] \rightarrow \\ y[n-L] &= \tilde{h}[n] * x[n] = \sum_{i=0}^{2L} \tilde{h}[i]x[n-i] \quad (9.16) \end{aligned}$$

by the time-shift property of convolution. So the noncausal FIR filter becomes a MA system of order $2L$. The output is delayed by L time samples, but this is not a serious problem. Also, in image processing, noncausal filters can be implemented as is, since the entire image is available before filtering begins.

The advantages of using an FIR filter are as follows:

- FIR filters are always stable, unlike IIR filters;
- FIR filters have a transient response that is zero after a finite time N, unlike IIR filters;
- FIR filters have no phase distortion, unlike IIR filters.

There are 3 major approaches to FIR filter design:

- *Windowing* the ideal IIR filter computed as the inverse DTFT of the desired frequency response function, usually using a Hamming window;
- *Frequency sampling*, in which the desired frequency response is attained exactly at a finite number of frequencies, usually equally spaced;
- *Minimax*, in which the iterative Parks-McClellan algorithm is used to minimize the maximum (minimax) absolute weighted error.

Before presenting these approaches, we present some common forms of the desired (ideal) filter frequency response function.

9.3.3 Forms of Desired Frequency Response Functions

Let $\mathbf{H_D}(e^{j\Omega})$ be the desired frequency response function. Recall $\mathbf{H_D}(e^{j\Omega})$ is periodic in Ω with period 2π, and conjugate symmetric: $\mathbf{H_D}(e^{-j\Omega}) = \mathbf{H_D}(e^{j\Omega})^*$. Then,

Forms of Frequency Response

For a low-pass filter with cutoff frequency Ω_o:

$$\mathbf{H_D}(e^{j\Omega}) = \begin{cases} 1 & \text{for } 0 \le |\Omega| < \Omega_o \\ 0 & \text{for } \Omega_o < |\Omega| \le \pi \end{cases} \tag{9.17}$$

For a band-pass filter with cutoff frequencies Ω_L and Ω_H:

$$\mathbf{H_D}(e^{j\Omega}) = \begin{cases} 0 & \text{for } 0 \le |\Omega| < \Omega_L \\ 1 & \text{for } \Omega_L < |\Omega| < \Omega_H \\ 0 & \text{for } \Omega_H < |\Omega| \le \pi \end{cases} \tag{9.18}$$

For an ideal differentiator (used for digital speedometers):

$$\mathbf{H_D}(e^{j\Omega}) = j\Omega \text{ for } |\Omega| < \pi \tag{9.19}$$

For a *Hilbert transform* (used in digital communications):

$$\mathbf{H_D}(e^{j\Omega}) = \begin{cases} -j & \text{for } 0 < \Omega < \pi \\ j & \text{for } -\pi < \Omega < 0 \end{cases} \tag{9.20}$$

Note that for the differentiator and Hilbert transform $\mathbf{H_D}(e^{j\Omega})$ is discontinuous at $\Omega = \pi$.

Forms of Impulse Response

From conjugate symmetry of $\mathbf{H_D}(e^{j\Omega})$, its inverse DTFT $h_D[n]$ should have the following forms:

- $\mathbf{H_D}(e^{j\Omega})$ for the low-pass and band-pass filters is real and even. So $h[n]$ should be real and even.
- $\mathbf{H_D}(e^{j\Omega})$ for the differentiator and Hilbert transform is pure imaginary and odd. So $h[n]$ should be real and odd.
- We design $h[n]$ to be real, even or odd, and noncausal. Then we delay it by half its length to make it causal, as in (9.16).

There are four types of FIR filters, depending on whether $h[n]$ is symmetric or antisymmetric, and on whether its length is odd or even. We depict these four types for FIR filters of lengths four or five (orders three or four) as

Type	Form	Restriction
I	$\{b, a, c, a, b\}$	None
II	$\{b, a, a, b\}$	$\mathbf{H}(e^{j\pi})$=0
III	$\{b, a, 0, -a, -b\}$	$\mathbf{H}(e^{j0,j\pi})$=0
IV	$\{b, a, -a, -b\}$	$\mathbf{H}(e^{j0})$=0

Table 9-2. Forms of FIR Filters.

Note that we have deliberately not specified which element of $h[n]$ is zero. Delaying $h[n]$ in time by L will only delay the output by the same L. The frequency response function $\mathbf{H}(e^{j\Omega})$ will then be multiplied by $e^{-j\Omega L}$, from the delay property of the DTFT. This is called a *linear phase* term, since its effect on the phase of $\mathbf{H}(e^{j\Omega})$ is to decrease the phase by an amount ΩL, proportional to (so a linear function of) Ω. Using time invariance, we may avoid worries about linear phase terms. It should be noted that even if $\mathbf{H}(e^{j\Omega})$ is real-valued, the phase of $\mathbf{H}(e^{j\Omega})e^{-j\Omega L}$ is not a straight line–it jumps by $\pm\pi$ at frequencies where the sign of $\mathbf{H}(e^{j\Omega})$ changes.

The restrictions follow automatically from the forms of $h[n]$, as follows:

- $h[n] = \{b, a, a, b\} \rightarrow \mathbf{H}(e^{j\pi}) = b - a + a - b = 0$.
- $h[n] = \{b, a, 0, -a, -b\} \rightarrow \mathbf{H}(e^{j\pi}) = b - a + a - b = 0$.
- $h[n]$ antisymmetric$\rightarrow \mathbf{H}(e^{j0}) = b + a - a - b = 0$.

So using a type II filter automatically satisfies the low-pass filter criterion of rejecting $\Omega = \pi$, and using a type IV filter automatically satisfies the high-pass filter criterion of rejecting $\Omega = 0$. But the savings here are only minimal. We will consider only types I and III (which have odd lengths).

Summary

Specifically, we design $h[n]$ to have the form

- $h[n] = \{h[-L], \ldots, h[-1], \underline{h[0]}, h[1], \ldots h[L]\}$
- OR $= \{-h[-L], \ldots, -h[-1], \underline{0}, h[1], \ldots h[L]\}$

and then implement not the noncausal $h[n]$ but the causal $\tilde{h}[n] = h[n - L]$, since by time-invariance we have

- $x[n] \rightarrow \boxed{h[n]} \rightarrow y[n]$ implies

- $x[n] \rightarrow \boxed{h[n-L]} \rightarrow y[n-L]$.

The causal $\tilde{h}[n]$ can be implemented as in (9.15) with $N = 2L$. The output will be delayed by L, but at the standard CD sampling rate of 44100 samples per second, L=100 is only $\frac{1}{441} \approx 2.2$ msec.

we can halve the number of multiplications required by noting that, since $h[i] = \pm h[-i]$ (depending on whether $h[n]$ is symmetric or antisymmetric)

$$\begin{aligned} y[n] &= \sum_{i=-L}^{L} h[i]x[n-i] \\ &= \sum_{i=1}^{L} h[i](x[n-i] \pm x[n+i]) \\ &+ h[0]x[n]. \end{aligned} \tag{9.21}$$

Delaying by L gives

$$\begin{aligned} y[n-L] &= \sum_{i=1}^{L} h[i](x[n-L-i] \pm x[n-L+i]) \\ &+ h[0]x[n-L]. \end{aligned} \tag{9.22}$$

This system does not require future values $\{x[n+i], i > 0\}$ of the input $x[n]$.

Saving half the multiplications (and storage elements) can be significant when implementing this FIR filter on a DSP chip.

9.3.4 FIR Filter Design by Windowing

The idea behind designing an FIR filter using windowing is to use a data window on the ideal impulse response, computed as the inverse DTFT of the desired frequency response function:

1. Compute $h_D[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathbf{H_D}(e^{j\Omega})e^{j\Omega n}d\Omega$.
2. Compute $h[n]$=$h_D[n]w[n]$ for some data window $w[n]$.
3. $w[n]$ should be advanced in time so that $w[n] \neq 0$ for $|n| \leq L$, not the usual $0 \leq n \leq 2L$.
4. Implement the causal filter $\tilde{h}[n] = h[n-L]$.

Example 9-7: Window design of FIR differentiator

Design an FIR differentiator using:

1. A five-point rectangular window.
2. A five-point Hamming window.

Solution:

The desired frequency response function of an ideal differentiator is

$$\mathbf{H_D}(e^{j\Omega}) = j\Omega, |\Omega| < \pi. \tag{9.23}$$

$h_D[n]$ is computed from $\mathbf{H_D}(e^{j\Omega})$ using the inverse DTFT:

$$\begin{aligned} h_D[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathbf{H_D}(e^{j\Omega})e^{j\Omega n}d\Omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} j\Omega e^{j\Omega n}d\Omega \\ &= \begin{cases} \frac{(-1)^n}{n} & n \neq 0 \\ 0 & n = 0 \end{cases} \\ &= \{\ldots \frac{1}{3}, -\frac{1}{2}, 1, \underline{0}, -1, \frac{1}{2}, -\frac{1}{3}\ldots\} \end{aligned} \tag{9.24}$$

The five-point rectangular $w_R[n]$ and Hamming $w_H[n]$ windows, advanced in time by two to be centered at $n = 0$, are

$$\begin{aligned} w_R[n] &= \{1, 1, \underline{1}, 1, 1\} \\ w_H[n] &= 0.54 + 0.46\cos\left(\frac{2\pi n}{5-1}\right) \\ &= \{0.08, 0.54, \underline{1}, 0.54, 0.08\} \end{aligned} \tag{9.25}$$

The two FIR differentiator impulse responses are:

$$\begin{aligned} h_R[n] &= \{-1/2, 1, \underline{0}, -1, 1/2\} \\ \tilde{h}_R[n] &= \left\{\underline{-1/2}, 1, 0, -1, 1/2\right\} \\ h_H[n] &= \{-0.04, 0.54, \underline{0}, -0.54, 0.04\} \\ \tilde{h}_H[n] &= \{\underline{-0.04}, 0.54, 0, -0.54, 0.04\} \end{aligned} \tag{9.26}$$

9.3.5 FIR Filter Design by Frequency Sampling

The idea behind designing an FIR filter using frequency sampling is to choose $h[n]$ so that the DTFT

of $h[n]$ equals the desired $\mathbf{H_D}(e^{j\Omega})$ at the $2L+1$ choice frequencies

$$\Omega = \frac{2\pi k}{2L+1}, k = -L, \ldots L.$$

and hope that the DTFT of the resulting $h[n]$ is close to $\mathbf{H_D}(e^{j\Omega})$ at all other frequencies. The choice frequencies can also be chosen to be not equally spaced, so that the DTFT of $h[n]$ is closer to $\mathbf{H_D}(e^{j\Omega})$ where the choice frequencies are closely spaced, and farther from $\mathbf{H_D}(e^{j\Omega})$ where the choice frequencies are widely spaced.

FIR Filter Design by Solving a Linear System of Equations

This seems to produce the linear system of $2L+1$ equations in $2L+1$ unknowns

$$\sum_{n=-L}^{L} h[n] e^{-j\frac{2\pi nk}{2L+1}} = \mathbf{H_D}(e^{j\frac{2\pi k}{2L+1}}) \qquad (9.27)$$

for $k = -L, \ldots L$. But the symmetry or antisymmetry of $h[n]$, and the conjugate symmetry of $\mathbf{H_D}(e^{j\Omega})$, reduces this to $L+1$ equations in $L+1$ unknowns, as the following example shows.

Example 9-8: Design of lowpass filter using frequency sampling.

Design an FIR lowpass filter of length five using frequency sampling and a Type I filter, and these specs:

- $\underline{\Omega = 0} : \mathbf{H}(e^{j0}) = 1;$
- $\underline{\Omega = \pi} : \mathbf{H}(e^{j\pi}) = 0;$
- $\underline{\Omega = \frac{\pi}{2}} : \mathbf{H}(e^{j\frac{\pi}{2}}) = \frac{3}{4}.$

Solution:

A type I filter centered at $n = 0$ has the form

$$h[n] = \{a, b, \underline{c}, b, a\}.$$

The DTFT $\mathbf{H}(e^{j\Omega})$ of $h[n]$ is

$$\begin{aligned} \mathbf{H}(e^{j\Omega}) &= ae^{j2\Omega} + be^{j\Omega} + c \\ &+ be^{-j\Omega} + ae^{-j2\Omega} \\ &= c + 2b\cos(\Omega) + 2a\cos(2\Omega). \end{aligned} \qquad (9.28)$$

Evaluating $\mathbf{H}(e^{j\Omega})$ at

$$\Omega = 0, \pm\frac{\pi}{2}, \pm\frac{2\pi}{2}$$

gives three equations in three unknowns:

- $\underline{\Omega = 0} : \mathbf{H}(e^{j0}) = 1 = c + 2b + 2a$
- $\underline{\Omega = \pi} : \mathbf{H}(e^{j\pi}) = 0 = c - 2b + 2a$
- $\underline{\Omega = \frac{\pi}{2}} : \mathbf{H}(e^{j\frac{\pi}{2}}) = \frac{3}{4} = c - 2a$

Solving these three equations in three unknowns gives

$$a = -1/16; \quad b = 1/4; \quad c = 5/8 \qquad (9.29)$$

so the FIR lowpass filter is

$$\begin{aligned} h[n] &= \left\{-\frac{1}{16}, \frac{1}{4}, \underline{\frac{5}{8}}, \frac{1}{4}, -\frac{1}{16}\right\} \\ \tilde{h}[n] &= \left\{\underline{-\frac{1}{16}}, \frac{1}{4}, \frac{5}{8}, \frac{1}{4}, -\frac{1}{16}\right\} \end{aligned} \qquad (9.30)$$

FIR Filter Design Using an Inverse DFT

By choosing the frequencies to exclude $\Omega = \pi$, and some reordering, the solution to the linear system of equations can be replaced with an inverse DFT. The following example illustrates how this can be done.

Example 9-9: Design of lowpass filter using frequency sampling, revisited.

Redo Example 9-8, now using the specs:

- $\underline{\Omega = 0} : \mathbf{H}(e^{j0}) = 1;$
- $\underline{\Omega = \frac{2\pi}{5}} : \mathbf{H}(e^{j\frac{2\pi}{5}}) = 1;$
- $\underline{\Omega = \frac{4\pi}{5}} : \mathbf{H}(e^{j\frac{4\pi}{5}}) = 0.$

Solution:

The DTFT $\mathbf{H}(e^{j\Omega})$ of $h[n]$ is still

$$\mathbf{H}(e^{j\Omega}) = c + 2b\cos(\Omega) + 2a\cos(2\Omega). \qquad (9.31)$$

Evaluating $\mathbf{H}(e^{j\Omega})$ at

$$\Omega = 0, \pm\frac{2\pi}{5}, \pm\frac{4\pi}{5}$$

gives three equations in three unknowns:

- $\underline{\Omega = 0} : \mathbf{H}(e^{j0}) = 1 = c + 2b + 2a$
- $\underline{\Omega = \frac{2\pi}{5}} : \mathbf{H}(e^{j\frac{2\pi}{5}}) = 1 = c + 2(0.309)b - 2(0.809)a$
- $\underline{\Omega = \frac{4\pi}{5}} : \mathbf{H}(e^{j\frac{4\pi}{5}}) = 0 = c - 2(0.809)b + 2(0.309)a$

Solving these three equations in three unknowns gives

$$a = -0.1236; \quad b = 0.3236; \quad c = 0.6000. \tag{9.32}$$

so the FIR lowpass filter is

$$\begin{aligned} h[n] &= \{-0.1236, 0.3236, \underline{0.6}, 0.3236, -0.1236\} \\ \tilde{h}[n] &= \{\underline{-0.1236}, 0.3236, 0.6, 0.3236, -0.1236\} \end{aligned} \tag{9.33}$$

But for this choice of frequencies, the system of equations is just a five-point DFT. So they can be solved using an inverse five-point DFT:

`real(ifft([1 1 0 0 1]))` gives $\{c, b, a, a, b\}$

as computed above. Note the ordering:

- We must compute the inverse DFT of $\{X_0, X_1, X_2, X_3 = X_2^*, X_4 = X_1^*\}$.
- The result is a single period of the periodic extension of $h[n]$: $\{h[0], h[1], h[2], h[3] = h[-2], h[4] = h[-1]\}$.
- This is *not* the same as the desired filter $\{h[-2], h[-1], h[0], h[1], h[2]\}$ but a circular shift of it by half its length.
- This can be performed using `fftshift`. So:
- $h[n] = \{-0.1236, 0.3236, \underline{0.6}, 0.3236, -0.1236\}$
- $\tilde{h}[n] = \{\underline{-0.1236}, 0.3236, 0.6, 0.3236, -0.1236\}$ in agreement with the answer obtained by solving a linear system of equations.

In Example 9-8, the design specs constituted a 4-point DFT, so an inverse DFT gives the *aliased* solution (a is doubled from its correct value)

$$a = -1/8; \quad b = 1/4; \quad c = 5/8. \tag{9.34}$$

In practice, solving the linear system of equations is easier than trying to choose frequencies so that an inverse DFT can be used. In fact, frequency sampling is not used much anymore.

9.3.6 FIR Filter Design using Minimax Criterion

The idea behind designing an FIR filter using a *minimax* criterion is to compute the $h[n]$ satisfying the criterion

$$\underset{\text{h[n]}}{\text{MIN}} \; \underset{\Omega}{\text{MAX}} \left\{ |\mathbf{E}(e^{j\Omega})| \mathbf{W}(e^{j\Omega}) \right\} \tag{9.35}$$

where the error $\mathbf{E}(e^{j\Omega})$ and weight $\mathbf{W}(e^{j\Omega})$ are

$$\begin{aligned} \mathbf{E}(e^{j\Omega}) &= \mathbf{H}_D(e^{j\Omega}) - \sum_{n=0}^{N-1} h[n] e^{-j\Omega n} \\ \mathbf{W}(e^{j\Omega}) &= \text{weight to penalize error at } \Omega \end{aligned} \tag{9.36}$$

Filters designed using the criterion (9.35) tend to have gains that oscillate ("ripple") around the desired gain, but the size of the ripple is constant. So these filters are often called *equiripple* filters.

The way to interpret the minimax criterion (9.35) is as a game being played between you, the designer, and the math of discrete-time filter performance:

- Math plays 2^{nd}, and will choose the Ω that makes the weighted error as large as possible.
- You play 1^{st}, and know that math will take the worst case (value of Ω) for your design.
- So you play (choose $h[n]$) to minimize the worst case (minimum maximum error magnitude).

This can be done using an iterative algorithm developed in 1972 by Parks and McClellan at Georgia Tech. We will skip the approximation theory behind this algorithm (Remez exchange and alternation theorems, among other things) and simply show how to implement it using the Matlab Signal Processing Toolbox command `firpm` (this command is not available in Mathscript).

Example 9-10: Equiripple design using `firpm`.

Design a bandpass filter having the specs:

$$\mathbf{H}(e^{j\Omega}) = \begin{cases} 0 & \text{for } 0.0\pi \le \Omega < 0.2\pi \\ dc & \text{for } 0.2\pi < \Omega < 0.3\pi \\ 1 & \text{for } 0.3\pi < \Omega < 0.7\pi \\ dc & \text{for } 0.7\pi < \Omega < 0.8\pi \\ 0 & \text{for } 0.8\pi < \Omega < 1.0\pi \end{cases} \tag{9.37}$$

where *dc*="don't care," and having duration 21.

Solution:
The "don't cares" are *transition bands* between the *passband* (gain=1) and the two *stopbands* (gain=0). The order=20 of the filter is one less than its length. The Matlab recipe for designing this filter is

```
F=[0.0 0.2 0.3 0.7 0.8 1.0];
G=[0.0 0.0 1.0 1.0 0.0 0.0];
H=firpm(20,F,G);freqz(H)
```

- `F` is a vector of pairs of frequencies $\frac{\Omega}{\pi}$. `F` must have even length, and include 0 and 1.
- `G` is a vector of gains $\mathbf{H}(e^{j\Omega})$ at frequencies `F`.
- `firpm` designs a filter whose gain varies linearly: between `(F(1)`π`,G(1))` and `(F(2)`π`,G(2))`; and between `(F(3)`π`,G(3))` and `(F(4)`π`,G(4))`; etc. The computed impulse response is stored in `H`.

The default design is a symmetric $h[n]$, suitable for filters. To design an antisymmetric $h[n]$, use either

- `H=firpm(20,[0 1],[1 1],'hilbert');` or
- `H=firpm(20,[0 1],[0 pi],'differentiator')`
  The latter uses weighting function $\mathbf{W}(e^{j\Omega})=\frac{1}{\Omega}$ to penalize heavily errors at low frequencies. The criterion to be minimized is $|\frac{\mathbf{H}(e^{j\Omega})-j\Omega}{j\Omega}|$.

Exercise 9-3: An FIR filter with $h[n] = \{a, b, c, d, 0, -d, -c, -b, -a\}$ is guaranteed to have which one of these four forms: **(a)** Lowpass **(b)** Bandpass **(c)** Highpass **(d)** Band-reject.

Answer: Bandpass, since both low and high frequencies are rejected: $H(e^{j0})$=a+b+c+d+0–d–c–b–a=0 and $H(e^{j\pi})$=a–b+c–d+0–(–d)+(–c)–(–b)+(–a)=0.

Exercise 9-4: An FIR filter with $h[n] = \{a, b, c, d, d, c, b, a\}$ is guaranteed to have which one of these four forms: **(a)** Lowpass **(b)** Bandpass **(c)** Highpass **(d)** Band-reject.

Answer: Lowpass, since $H(e^{j\pi})$=a–b+c–d+d–c+b–a=0 and $H(e^{j0})$=2a+2b+2c+2d $\neq$ 0.

9.4 IIR Filter Design

9.4.1 Overview of IIR Filter Design

An IIR filter has the form of an ARMA difference equation (see Section 7-3)

$$\begin{aligned} & y[n] + a_1 y[n-1] + \ldots + a_N y[n-N] \\ = \; & b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M] \end{aligned} \quad (9.38)$$

for some constant coefficients $\{a_n, 1 \leq n \leq N\}$ and $\{b_n, 0 \leq n \leq M\}$ and constant orders (N, M). An IIR filter can also be specified in terms of its tranfer function $\mathbf{H}(\mathbf{z})$ or its infinite-duration impulse response $h[n]$.

The advantages of using an IIR filter are as follows:

- IIR filters can be much more selective in frequency than FIR filters that use the same number of coefficients. For example, the ARMA (IIR) notch filters in Section 8-2 were much more selective than the MA (FIR) notch filters;
- The two design procedures to follow guarantee a stable IIR filter. However, roundoff error in DSP chips can cause a mathematically stable filter to be unstable in its implementation;
- IIR filters, unlike FIR filters, need not be delayed in time to make them causal.

The philosophy behind IIR filter design is to take a continuous-time filter, such as the Butterworth filter in Section 6-8, and transform it into a discrete-time filter. There are two major approaches to IIR filter design:

- *Impulse invariance*: the continuous-time impulse response $h_a(t)$ is sampled to a discrete-time one $h[n]$;
- *Bilinear transform*: the continuous-time transfer function $\mathbf{H_a}(\mathbf{s})$ is mapped to a discrete-time one $\mathbf{H}(\mathbf{z})$.

9.4.2 IIR Filter Design Using Impulse Invariance

The idea behind designing an IIR filter using impulse invariance is to sample the impulse response $h_a(t)$ of

a suitable continuous-time filter. Specifically,

- Select a suitable continuous-time filter, such as a Butterworth filter, with transfer function $\mathbf{H_a(s)}$.
- Compute its continuous-time impulse response $h_a(t) = \mathcal{L}^{-1}\{\mathbf{H_a(s)}\}$ using the inverse Laplace transform.
- Let $h[n] = Th_a(nT)$ for sampling interval T.

Multiplication by T is required dimensionally: $h_a(t)$ has units of $\frac{1}{\text{TIME}}$, since $\int_{-\infty}^{\infty}\delta(t)dt{=}1$ and dt has units of time. For example, the impulse response of an RC circuit is, from Eq. (2.17), $h_a(t){=}\frac{1}{RC}e^{-\frac{t}{RC}}u(t)$. Throughout this section, the subscript **a** designates a continuous-time (analog) LTI system.

Impulse Invariance Results in Stable Filters

We now show that $\mathbf{H(z)}$ designed using impulse invariance of $\mathbf{H_a(s)}$ has all of its poles inside the unit circle if $\mathbf{H_a(s)}$ has all of its poles in the left half-plane. That is, impulse invariance preserves stability.

The poles $\{p_i = -a_i + jb_i\}$ of $\mathbf{H_a(s)}$ are all in the left half-plane if $a_i > 0$. Let $\{A_i\}$ be the residues of the partial fraction expansion of $\mathbf{H_a(s)}$. Then

$$h_a(t) = \sum_{i=1}^{N} A_i e^{p_i t}u(t). \tag{9.39}$$

Applying the continuous-to-discrete mapping $h[n] = Th_a(nT)$ gives

$$h[n] = Th_a(nT) = \sum_{i=1}^{N}(A_iT)(e^{p_iT})^n u[n]. \tag{9.40}$$

The discrete-time poles are $\{e^{p_iT}\}$. They have magnitudes

$$|e^{p_iT}| = |e^{-a_iT}| \cdot |e^{jb_iT}| = e^{-a_iT} < 1. \tag{9.41}$$

The discrete-time poles lie inside the unit circle $|z| = 1$ since $a_i > 0$, and the discrete-time system is stable.

Example 9-11: IIR Filter Design Using Impulse Invariance

Given $T = 0.001$ and the first-order continuous-time lowpass filter (see Section 6-1.2)

$$\mathbf{H_a(s)} = \frac{1000}{\mathbf{s} + 1000}, \tag{9.42}$$

use impulse invariance to design an IIR filter.

Solution:

The continuous-time impulse response is

$$h_a(t) = \mathcal{L}^{-1}\{\mathbf{H_a(s)}\} = 1000e^{-1000t}u(t). \tag{9.43}$$

The discrete-time impulse response is

$$h[n] = (0.001)(1000)e^{-(1000)(0.001)n} = e^{-n}u[n]. \tag{9.44}$$

The discrete-time transfer function is

$$\mathbf{H(z)} = \mathcal{Z}\{e^{-n}u[n]\} = \frac{\mathbf{z}}{\mathbf{z} - 0.368}. \tag{9.45}$$

$\mathbf{H(z)}$ is in fact a very crude lowpass filter, since frequencies near $\Omega = \pi$ are attenuated more than frequencies near $\Omega = 0$. But it hardly qualifies as a decent lowpass filter.

Example 9-12: IIR Filter Design Using Impulse Invariance

The continuous-time filter

$$\mathbf{H_a(s)} = \frac{\mathbf{s} + 0.1}{(\mathbf{s} + 0.1)^2 + 16} \tag{9.46}$$

has a sharp resonant peak at $\omega = 4$ in its frequency response $\mathbf{H_a}(j\omega)$, since its poles $\{-0.1 \pm j4\}$ are close to the imaginary axis (see Section 6-5). Use impulse invariance to design an IIR filter that also has a sharp resonant peak.

Solution:

The continuous-time impulse response is, from Table 3-2,

$$h_a(t) = \mathcal{L}^{-1}\{\mathbf{H_a(s)}\} = e^{-0.1t}\cos(4t)u(t). \tag{9.47}$$

The discrete-time impulse response is

$$h[n] = Te^{(-0.1T)n}\cos((4T)n)u[n]. \tag{9.48}$$

The discrete-time transfer function is, from Table 7-5,

$$\mathbf{H(z)} = T\frac{\mathbf{z}^2 - \mathbf{z}e^{-0.1T}\cos(4T)}{\mathbf{z}^2 - \mathbf{z}2e^{-0.1T}\cos(4T) + e^{-0.2T}}. \tag{9.49}$$

The poles of $\mathbf{H}(\mathbf{z})$ are $\{e^{-0.1T}e^{\pm j4T}\}$. For small T, these poles will produce a sharp resonant peak in its frequency response $\mathbf{H}(e^{j\Omega})$ at $\Omega = 4T$ (see Section 8-1).

9.4.3 IIR Filter Design Using Bilinear Transformation

The idea behind designing an IIR filter using bilinear transformation is to map the given continuous-time transfer function $\mathbf{H_a}(\mathbf{s})$ to the discrete-time transfer function $\mathbf{H}(\mathbf{z})$ using the *bilinear transformation*

$$\mathbf{s} = \frac{2}{T}\frac{1-\mathbf{z}^{-1}}{1+\mathbf{z}^{-1}} = \frac{2}{T}\frac{\mathbf{z}-1}{\mathbf{z}+1}. \tag{9.50}$$

Bilinear transformation gets its name from (9.50) being the ratio of two linear functions.

The resulting discrete-time transfer function $\mathbf{H}(\mathbf{z})$ is then

$$\mathbf{H}(\mathbf{z}) = \mathbf{H_a}\left(\frac{2}{T}\frac{\mathbf{z}-1}{\mathbf{z}+1}\right). \tag{9.51}$$

The following example demonstrates IIR filter design using bilinear transformation.

Example 9-13: IIR Filter Design Using Bilinear Transformation

Repeat Example 9-11 using bilinear transformation instead of impulse invariance.

Solution:

The bilinear transformation (9.50) with $T = 0.001$ is

$$\mathbf{s} = \frac{2}{0.001}\frac{\mathbf{z}-1}{\mathbf{z}+1} = 2000\frac{\mathbf{z}-1}{\mathbf{z}+1}. \tag{9.52}$$

The discrete-time transfer function $\mathbf{H}(\mathbf{z})$ obtained by substituting (9.52) in (9.42) is

$$\begin{aligned}\mathbf{H}(\mathbf{z}) &= \mathbf{H_a}\left(2000\frac{\mathbf{z}-1}{\mathbf{z}+1}\right)\\ &= \frac{1000}{2000\frac{\mathbf{z}-1}{\mathbf{z}+1}+1000}\left[\frac{\mathbf{z}+1}{\mathbf{z}+1}\right]\\ &= \frac{\mathbf{z}+1}{2(\mathbf{z}-1)+(\mathbf{z}+1)} = \frac{\mathbf{z}+1}{3\mathbf{z}-1}.\end{aligned}$$

This is a crude lowpass filter, but the zero at –1, which corresponds to $\Omega = \pi$, means that frequency $\Omega = \pi$ will be completely rejected by this filter.

9.4.4 Bilinear Transformation Results in Stable Filters

Left Half-Plane to Interior of Unit Circle

We now show that $\mathbf{H}(\mathbf{z})$ designed using bilinear transformation of $\mathbf{H_a}(\mathbf{s})$ has all of its poles inside the unit circle if $\mathbf{H_a}(\mathbf{s})$ has all of its poles in the left half-plane. That is, bilinear transformation preserves stability.

The poles $\{p_i = -a_i + jb_i\}$ of $\mathbf{H_a}(\mathbf{s})$ are all in the left half-plane if $a_i > 0$. We now show that the bilinear transformation maps the entire left half-plane Real[$\mathbf{s}$] < 0 to the interior $|\mathbf{z}| < 1$ of the unit circle.

Solving the bilinear transformation (9.50) for $\mathbf{z}$ in terms of $\mathbf{s}$ gives

$$\mathbf{s} = \frac{2}{T}\frac{\mathbf{z}-1}{\mathbf{z}+1} \rightarrow \mathbf{z} = \frac{1+sT/2}{1-sT/2}. \tag{9.53}$$

Substituting $\mathbf{s} = -a + jb$ gives

$$\mathbf{z} = \frac{(1-aT/2)+j(bT/2)}{(1+aT/2)-j(bT/2)}. \tag{9.54}$$

The squared magnitude $|\mathbf{z}|^2$ of $\mathbf{z}$ is

$$|\mathbf{z}|^2 = \frac{(1-aT/2)^2+(bT/2)^2}{(1+aT/2)^2+(bT/2)^2}. \tag{9.55}$$

We know $-aT < aT$ since $a > 0$ and $T > 0$. Adding $1 + (aT/2)^2$ to both sides of this gives

$$(1-aT/2)^2 < (1+aT/2)^2 \tag{9.56}$$

This shows that the numerator of (9.55) is smaller than the denominator of (9.55), so $|\mathbf{z}|^2 < 1$. The left half-plane of $\mathbf{s}$ is mapped to the interior $|\mathbf{z}| < 1$ of the unit circle, and bilinear transformation, like impulse invariance, preserves stability.

Interior of Unit Circle to Left Half-Plane

We now show that bilinear transformation also maps the interior $|\mathbf{z}| < 1$ of the unit circle to the left half-plane Real[$\mathbf{s}$] < 0. This is the reverse of the result we have just shown.

Recall that for any complex number $\mathbf{z}$ we have

$$\begin{aligned}\mathbf{z}\mathbf{z}^* &= |\mathbf{z}|^2\\ \mathbf{z}-\mathbf{z}^* &= j2\cdot\text{Imag}[\mathbf{z}]\end{aligned} \tag{9.57}$$

Using these, rewrite (9.50) as

$$\begin{aligned}\mathbf{s} &= \frac{2}{T}\frac{\mathbf{z}-1}{\mathbf{z}+1}\left[\frac{\mathbf{z}^*+1}{\mathbf{z}^*+1}\right] \\ &= \frac{2}{T}\frac{\mathbf{z}\mathbf{z}^*+\mathbf{z}-\mathbf{z}^*-1}{(\mathbf{z}+1)(\mathbf{z}^*+1)} \\ &= \frac{2}{T}\frac{|\mathbf{z}|^2-1}{|\mathbf{z}+1|^2}+j\frac{4}{T}\frac{\mathcal{I}\Updownarrow[\mathbf{z}]}{|\mathbf{z}+1|^2} \qquad (9.58)\end{aligned}$$

The real part of this is

$$\text{Real}[s]=\frac{2}{T}\frac{|\mathbf{z}|^2-1}{|\mathbf{z}+1|^2}<0 \qquad (9.59)$$

if and only if $|\mathbf{z}|<1$. So bilinear transformation also maps the inside of the unit circle $|\mathbf{z}|$=1 to the left half-plane $\mathcal{R}\rceil[\mathbf{s}]<0$.

9.4.5 Frequency Warping

The significance of bilinear transformation as a filter design technique is that it maps the imaginary axis Real[$\mathbf{s}$] = 0 to the unit circle $|\mathbf{z}|=1$. This follows from the previous subsection by setting $a=0$. But it also shows that the frequency response $\mathbf{H}(e^{j\Omega})$ of the discrete-time filter has the same shape as the frequency response $\mathbf{H_a}(j\omega)$ of the continuous-time filter. The difference is that the frequency axis has been warped, to compress the interval $\{0\le\omega\le\infty\}$ to $\{0\le\Omega\le\pi\}$. We now derive this.

Let $\mathbf{z}$=$e^{j\Omega}$. The bilinear transformation becomes

$$\begin{aligned}\mathbf{s} &= \frac{2}{T}\frac{e^{j\Omega}-1}{e^{j\Omega}+1} \\ &= \frac{2}{T}\frac{e^{j\Omega/2}-e^{-j\Omega/2}}{e^{j\Omega/2}+e^{-j\Omega/2}}\left[\frac{e^{j\Omega/2}}{e^{j\Omega/2}}\right] \\ &= \frac{2}{T}\frac{2j\sin(\Omega/2)}{2\cos(\Omega/2)} \\ &= j\frac{2}{T}\tan(\Omega/2)=j\omega. \qquad (9.60)\end{aligned}$$

The resulting $\mathbf{s}$ lies on the imaginary axis. So continuous-time frequency ω maps to discrete-time frequency Ω using the *(pre)warping* formula

$$\omega=(2/T)\tan(\Omega/2) \qquad (9.61)$$

The tangent function maps $\{0\le\Omega\le\pi\}$ to $\{0\le\omega\le\infty\}$.

The discrete-time frequency response $\mathbf{H}(e^{j\Omega})$ is the continuous-time frequency response $\mathbf{H_a}(j\omega)$ *nonlinearly compressed in frequency*:

$$\mathbf{H}(e^{j\Omega})=\mathbf{H_a}(j(2/T)\tan(\Omega/2)). \qquad (9.62)$$

For bilinear-transformation-designed filters:
Discrete-time frequency response $\mathbf{H}(e^{j\Omega})$
Continuous-time frequency response $\mathbf{H_a}(j\omega)$
are related by the prewarping formula
$\mathbf{H}(e^{j\Omega})=\mathbf{H_a}(j(2/T)\tan(\Omega/2))$

The significance of the prewarping formula is shown by the following example.

Example 9-14: IIR Filter Design using Prewarping.

Repeat Example 9-12 using bilinear transformation instead of impulse invariance. However, we now wish to obtain a discrete-time filter with a sharp resonant peak at specifically $\Omega=\pi/2$.

Solution:

First, we choose T to map $\omega=4$ to $\Omega=\frac{\pi}{2}$ using the prewarping formula (9.61):

$$\begin{aligned}\omega &= (2/T)\tan(\Omega/2) \\ 4 &= (2/T)\tan(\pi/2/2)=(2/T). \qquad (9.63)\end{aligned}$$

This has the solution $T=\frac{1}{2}$.

Second, we use this value of T in the bilinear transformation (9.50):

$$\begin{aligned}\mathbf{s} &= \frac{2}{1/2}\frac{\mathbf{z}-1}{\mathbf{z}+1} \\ &= 4\frac{\mathbf{z}-1}{\mathbf{z}+1}. \qquad (9.64)\end{aligned}$$

Third, we perform the bilinear transformation as in (9.51):

$$\begin{aligned}\mathbf{H}(\mathbf{z}) &= \mathbf{H_a}\left(4\frac{\mathbf{z}-1}{\mathbf{z}+1}\right) \\ &= \frac{4\frac{\mathbf{z}-1}{\mathbf{z}+1}+0.1}{\left(4\frac{\mathbf{z}-1}{\mathbf{z}+1}+0.1\right)^2+16}\left[\frac{100(\mathbf{z}+1)^2}{100(\mathbf{z}+1)^2}\right] \\ &= \frac{\frac{1}{8}\mathbf{z}^2+0.0061\mathbf{z}-0.1189}{\mathbf{z}^2+0.0006\mathbf{z}+0.9512} \qquad (9.65)\end{aligned}$$

$\mathbf{H}(\mathbf{z})$ has the following poles and zeros:

- Poles at $\{-0.0003 \pm j0.9753\} \approx \{0.9753e^{\pm j\pi/2}\}$
- Zeros at $\{-1, 0.952\} \approx \{0.952e^{j0, j\pi}\}$

So its frequency response $\mathbf{H}(e^{j\Omega})$ has:

- Sharp peaks at $\Omega = \pm\frac{\pi}{2}$, as desired
- Dips to zero at Ω=0 and π.

The algebra can be performed in Matlab's Signal Processing Toolbox using

```
[B A]=bilinear([1 0.1],[1 0.2 16.01],2);
```

which gives the output

```
B=[0.125 0.0061 -0.1189]
A=[1.000 0.0006 0.9512]
```

which are the coefficients in $\mathbf{H}(\mathbf{z})$.
Note `bilinear` uses $\frac{1}{T} = 2$, not $T = \frac{1}{2}$ directly.

9.4.6 Results of Different Approaches to Filter Design

We conclude this section by comparing the half-band lowpass filters designed using various FIR and IIR filter design techniques.

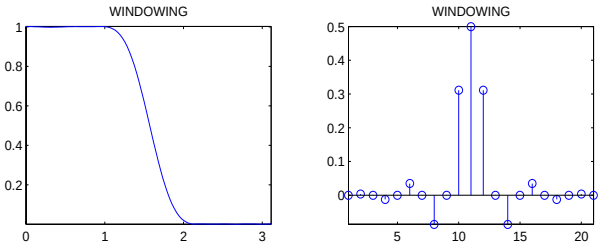
Example 9-15: Lowpass Filter Designs.
Design a half-band (cutoff frequency $\frac{\pi}{2}$) lowpass filter of length 21 using the following approaches:

1. Windowing using a Hamming window;
2. Frequency sampling using inverse DFT;
3. Minimax with transition 0.4π to 0.6π;
4. Discrete-time 10^{th}-order Butterworth filter. This is the bilinear transformation of the continuous-time Butterworth filter, and it has 21 coefficients, matching the other designs.

Solution:
The results are shown in Fig 9-14. For each design technique, the impulse response is the stem plot in the left half and the frequency response is the continuous plot in the right half. They are all quite similar. The Minimax design has a sharper transition, but also has ripples in the passband and stopband.

MATLAB code used for this figure:

```
clear;W=2*pi*[0:104]/210;
H1=0.5*sinc(0.5*[-10:10]);
H1=H1.*hamming(21)';
G1=abs(fft(H1,210));figure
subplot(221),plot(W,G1(1:105)),
axis tight,title('WINDOWING')
subplot(222),stem(H1),
axis tight,title('WINDOWING')
F=[1 1 1 1 1 .5 zeros(1,10) .5 1 1 1 1];
H2=fftshift(real(ifft(F)));
G2=abs(fft(H2,210));figure
subplot(221),plot(W,G1(1:105)),
axis tight,title('SAMPLING')
subplot(222),stem(H2),
axis tight,title('SAMPLING')
H3=firpm(20,[0 0.4 0.6 1.0],[1 1 0 0]);
G3=abs(fft(H3,210));figure
subplot(221),plot(W,G3(1:105)),
axis tight,title('MINIMAX')
subplot(222),stem(H3),
axis tight,title('MINIMAX')
[B A]=butter(10,0.5);figure
G4=abs(fft(B,210))./abs(fft(A,210));
subplot(221),plot(W,G4(1:105)),
axis tight,title('BUTTERWORTH')
subplot(222),zplane(B,A)
title('BUTTERWORTH')
```



Matlab's Signal Processing Toolbox also has a nice GUI for filter design called `fdatool`. It is pretty much self-explanatory. A screen shot is in Fig. 9-15.

Exercise 9-5: Using the analog filter $h_a(t) = \delta(t) - 3e^{-3t}u(t)$ and $T = 2$, design a digital filter using impulse invariance.

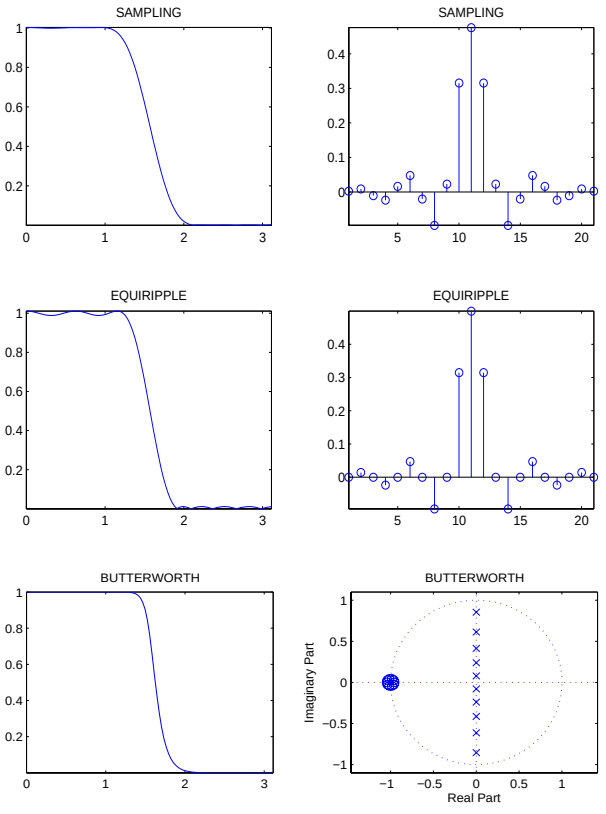


Figure 9.14: Half-Band Lowpass Filter of Length 21 Designed Using: (a) Hamming Window; (b) Frequency Sampling; (c) Minimax Criterion; (d) Butterworth Filter.

Answer: $h[n] = \delta[n] - Th_a(nT) = \delta[n] - 6e^{-6n}u[n]$. The impulse is just feedthrough.

Exercise 9-6: Using the analog filter $H_a(s) = \frac{s}{s+1}$ and $T = 2$, design a digital filter using bilinear transformation.

Answer: Setting $s = \frac{2}{2}\frac{z-1}{z+1}$ in $H_a(s)$ gives $H(z) = \frac{(z-1)/(z+1)}{1+(z-1)/(z+1)} = \frac{z-1}{(z+1)+(z-1)} = \frac{1}{2}(1 - z^{-1})$. $h[n] = \{\underline{\frac{1}{2}}, -\frac{1}{2}\}$.

Exercise 9-7: We wish to design an IIR digital lowpass filter with cutoff frequency $\omega{=}\frac{\pi}{2}$ using the bilinear transformation with $T = 0.001$. What should the analog lowpass filter cutoff frequency Ω be?

Answer: $\Omega = \frac{2}{T}\tan(\omega/2) = \frac{2}{0.001}\tan\left(\frac{\pi/2}{2}\right) = (2000)\tan(\pi/4) = 2000\frac{\text{radians}}{\text{second}}$.

Exercise 9-8: Using bilinear transformation with $T = 0.1$, the continuous-time frequency $\Omega = 20$ maps to what discrete-time frequency?

Answer: $20{=}\Omega{=}\frac{2}{0.1}\tan(\frac{\omega}{2}) \rightarrow 1{=}\tan(\frac{\omega}{2}) \rightarrow \omega{=}\pi/2$.

Exercise 9-9: Use bilinear transformation with $T = 2$ to design an IIR ideal differentiator.

Answer: $H_a(s) = s$ and $s{=}\frac{2}{2}\frac{z-1}{z+1}$. So $H(z) = \frac{z-1}{z+1} = \frac{Y(z)}{X(z)} \rightarrow y[n]{+}y[n{-}1]{=}x[n]{-}x[n{-}1]$.

9.5 Multirate Signal Processing

9.5.1 Overview of Multirate Signal Processing

The basic idea behind multirate signal processing is changing the sampling rate after the fact. We assume throughout that the original sampling rate exceeded double the maximum continuous-time frequency of the signal, so that the signal was not aliased after sampling at the original rate.

If a signal was originally sampled at 8000 samples per second, and we would like to determine what the samples of the signal would have been had we sampled it at 1000 samples per second, we can do this by *downsampling*. Note this may induce aliasing.

If a signal was originally sampled at 1000 samples per second, and we would like to determine what the samples of the signal would have been had we sampled it at 8000 samples per second, we can do this by *upsampling*, followed by (discrete-time) *interpolation*. This cannot undo any aliasing in the original signal.

If we want to change the sampling rate by a rational but non-integer factor, we use a combination of *upsampling and interpolation* and *downsampling*. The collection of these techniques is called *multirate* signal processing.

Three reasons for changing the sampling rate are:

- To replace the final discrete-time-to-continuous-time analog interpolating filter with a simpler analog filter, or with sample-and-hold interpolation;
- To design very selective (in frequency) discrete-time filters that have very sharp transitions between their pass-band and their stop-band. We can do this using a series of less-selective filters and multirate signal processing;
- To create the sounds of a musical instrument playing all musical notes from the sound of it playing a single musical note.

There are four parts to multirate signal processing:

- Upsampling (also known as *zero-stuffing*)
- Downsampling (also known as *decimating*)
- Interpolation (low-pass filtering)
- Multirate processing (combining all of these)

We analyze each of these in the next four subsections.

Since interpolation requires discrete-time lowpass filtering, we study multirate filtering after FIR and IIR filter design.

9.5.2 Upsampling

Upsampling in the Time Domain

Upsampling, also called *zero stuffing*, followed by *interpolation*, increases the sampling rate by an integer factor. "Upsampling" is also used to denote the combination of zero stuffing and interpolation. This describes the effect, but destroys the notational symmetry with downsampling. We will use upsampling to mean only zero-stuffing.

To upsample by L, simply insert $L-1$ zeros between each sample of a signal.

Example 9-16: Upsampling in the Time Domain.

Upsample by a factor of 2 the signal
$x[n]=\{\ldots \underline{3}, 1, 4, 1, 5, 9 \ldots\}$

Solution:

Inserting 2–1=1 zeros between samples gives
$y[n]=\{\ldots \underline{3}, 0, 1, 0, 4, 0, 1, 0, 5, 0, 9 \ldots\}$

The notation for upsampling by two is

$$x[n] \rightarrow \boxed{\uparrow 2} \rightarrow y[n]=\begin{cases} x[n/2] & \text{for even } n \\ 0 & \text{for odd } n \end{cases}$$

$y[0]=x[0]$, $y[2]=x[1]$, $y[4]=x[2]$, $y[6]=x[3]$,etc.

In Matlab/Mathscript code, upsampling by L is performed by

```
Z=[X;zeros(1,length(X))];Y=Z(:)';
```

Upsampling in the Frequency Domain

Let $y[n]$ be $x[n]$ upsampled by L. Then

$$y[n] = \{\ldots \underline{x[0]}, \underbrace{0 \ldots 0}_{L-1}, x[1], \underbrace{0 \ldots 0}_{L-1}, x[2] \ldots\} \quad (9.66)$$

The DTFT $\mathbf{Y}(e^{j\Omega})$ of $y[n]$ is

$$\begin{aligned} \mathbf{Y}(e^{j\Omega}) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\Omega n} \\ &= \sum_{n'=-\infty}^{\infty} x[n'] e^{-j\Omega L n'} \\ &= \mathbf{X}(e^{j\Omega L}) \end{aligned} \quad (9.67)$$

where $n' = \frac{n}{L}$ if n is a multiple of L and $n' = 0$ otherwise.

Since upsampling expands time by inserting zeros, it is not surprising that upsampling compresses the spectrum. The spectrum is now periodic with period $\frac{2\pi}{L}$. It still repeats every 2π, but now also more often.

Example 9-17: Upsampling in the Frequency Domain.

A signal has the spectrum shown in Fig. 9-16a. Sketch the spectrum of the signal after upsampling by 4.

Solution:

The solutions is shown in Fig. 9-16b. The spectrum is compressed by a factor of 4.

[Change 3 to π, 6 to 2π, etc.]

Upsampling $\rightarrow \boxed{\uparrow \mathrm{L}} \rightarrow$ compresses spectrum by L

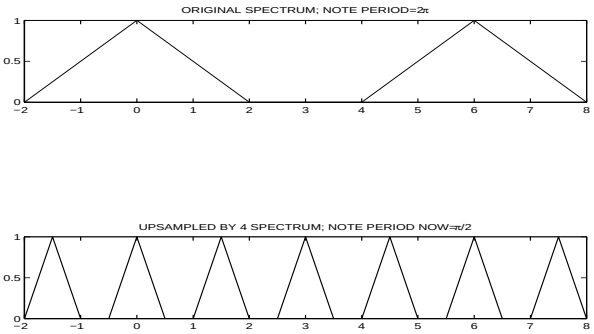


Figure 9.15: Effect of Upsampling on Spectrum. (a) Original Spectrum. (b) Upsampled Spectrum.

9.5.3 Downsampling

Downsampling in the Time Domain

Downsampling, also called *decimation*, reduces the sampling rate by an integer factor. This is very easy to do. To change the sampling rate of a signal from 1200 samples per second to 600 samples per second, we simply omit every other sample. To change the rate from 1200 to 400 samples per second, we simply omit two out of three samples, keeping only every third sample.

Example 9-18: Downsampling in the Time Domain.
Downsample by a factor of 2 the signal
$x[n]$=$\{\ldots \underline{3}, 1, 4, 1, 5, 9, 2, 6, 5 \ldots\}$

Solution:
Omitting every other sample leaves
$y[n]$=$\{\ldots \underline{3}, 4, 5, 2, 5 \ldots\}$

This is what we would have gotten had we sampled the original signal at half of the sampling rate. So we have indeed halved the sampling rate.

The notation for downsampling by two is

$x[n] \rightarrow \boxed{\downarrow 2} \rightarrow y[n] = x[2n]$ for all integers n:

$y[0]$=$x[0]$, $y[1]$=$x[2]$, $y[2]$=$x[4]$, $y[3]$=$x[6]$,etc.

We could also retain $x[n]$ only at odd times $n = 1, 3, 5 \ldots$. This would be a different *polyphase* component of $x[n]$. In the sequel, we assume that downsampling retains $x[0]$ specifically, as in this example.

Downsampling by an integer L is implemented by $y[n] = x[Ln]$. In Matlab/Mathscript code, downsampling by L is performed by `Y=X[1:L:end];`

Downsampling in the Frequency Domain

The effect of downsampling on a sinusoid is

$$\cos(\Omega_o n) \rightarrow \boxed{\downarrow \mathrm{L}} \rightarrow \cos(\Omega_o(Ln)) = \cos((\Omega_o L)n). \tag{9.68}$$

$\boxed{\rightarrow \boxed{\downarrow \mathrm{L}} \rightarrow \text{ increases frequency by a factor of } L}$

Determining the effect of downsampling on a spectrum is more complicated than determining the effect of upsampling on a spectrum, and within the derivation uses the effect of upsampling on a spectrum.

Let $y[n]$ be $x[n]$ downsampled by 2. Define the function $s[n]$ as

$$\begin{aligned} s[n] &= \frac{1}{2}\left[1 + (-1)^n\right] \\ &= \frac{1}{2}\left[1 + \cos(\pi n)\right] \\ &= \begin{cases} 1 & \text{for } n \text{ even} \\ 0 & \text{for } n \text{ odd} \end{cases} \end{aligned} \tag{9.69}$$

Then note that

$$\begin{aligned} x[n]s[n] &= \begin{cases} x[n] & \text{for } n \text{ even} \\ 0 & \text{for } n \text{ odd} \end{cases} \\ &= \{\ldots \underline{x[0]}, 0, x[2], 0, x[4], 0, x[6], 0 \end{aligned} \tag{9.70}$$

So $x[n]s[n]$ is $x[n]$ downsampled by 2, which is $y[n]$, and then upsampled by 2.

Using the modulation property of the DTFT and the result of the upsampling subsection, the DTFT of $x[n]s[n]$ is

$$\mathbf{Y}(e^{j\Omega 2}) = \frac{1}{2}\mathbf{X}(e^{j\Omega}) + \frac{1}{2}\mathbf{X}(e^{j(\Omega+\pi)}). \tag{9.71}$$

Replacing Ω with $\Omega/2$ gives

$$\begin{aligned} \mathbf{Y}(e^{j\Omega}) &= \frac{1}{2}\mathbf{X}(e^{j\Omega/2}) + \frac{1}{2}\mathbf{X}(e^{j(\Omega/2+\pi)}) \\ &= \frac{1}{2}\mathbf{X}(e^{j\Omega/2}) + \frac{1}{2}\mathbf{X}(e^{j(\Omega+2\pi)/2}) \end{aligned} \tag{9.72}$$

Since $y[n] = x[2n]$ compresses time by a factor of 2, it makes sense that $\mathbf{X}(e^{j\Omega/2})$ expands frequency by 2. The period of $\mathbf{X}(e^{j\Omega/2})$ then doubles from 2π to 4π. But $\mathbf{Y}(e^{j\Omega})$ *must* be periodic with period 2π. To ensure this, we must insert *additional* copies of $\mathbf{X}(e^{j\Omega/2})$, shifted in frequency by 2π from the original copies.

For downsampling by L, replace $s[n]$ defined in (9.69) with the more general relation

$$\begin{aligned} s[n] &= \frac{1}{L}\sum_{k=0}^{L-1} e^{j2\pi nk/L} \\ &= \begin{cases} 1 & \text{for } n \text{ a multiple of } L \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \quad (9.73)$$

A derivation similar to the $L = 2$ derivation above gives

$$\mathbf{Y}(e^{j\Omega}) = \frac{1}{L}\sum_{k=0}^{L-1} \mathbf{X}(e^{j(\Omega+2\pi k)/L}). \quad (9.74)$$

as the effect on spectrum of downsampling by L.

Example 9-19: Downsampling in the Frequency Domain.

A signal has the spectrum shown in Fig. 9-17a. Sketch the spectrum of the signal after downsampling by 4.

Solution:

The spectrum is stretched by a factor of 4. Three copies are inserted to keep the period 2π. The height is reduced by a factor of 4.

[Change 3 to π, 6 to 2π, etc.]

[Reduce the height of the spectra in the second plot to 1/4 the height of the spectra in the first plot.]

Downsampling $\rightarrow$ ↓L $\rightarrow$ expands spectra by L.
Extra copies are inserted to maintain period 2π.

9.5.4 Interpolation

Interpolating a signal $x(t)$ means assigning values to $x(t)$ at times t for which the signal is not already specified. If $x(t)$ is specified for $t = nT$, so that $\{x(nT)\}$ is known, interpolating $x(t)$ means assigning values to $\{x(t), t \neq nT\}$, based on the given values $\{x(nT)\}$.

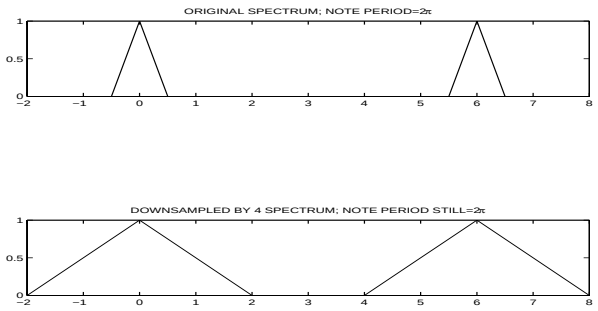


Figure 9.16: Effect of Downsampling on Spectrum. (a) Original Spectrum. (b) Downsampled Spectrum.

For example, linear interpolation means $x(t)$ varies linearly between $x(nT)$ and $x((n+1)T)$. Another example is interpolating a bandlimited signal from its samples (see Section 6-12.6). Interpolation "connects the dots" between given values $\{x(nT)\}$.

Interpolation can also occur entirely in discrete time. Let $x[n]$ be a signal that has been upsampled by L. Then $x[n]$ has the form

$$x[n] = \{\ldots x[0], \underbrace{0\ldots0}_{L-1}, x[L], \underbrace{0\ldots0}_{L-1}, x[2L], \ldots\} \quad (9.75)$$

We interpolate $x[n]$ by changing its zero values to the nonzero values that make the spectrum of $x[n]$ have maximum frequency $\frac{\pi}{L}$. The goal of this subsection is to show we can do this by filtering $x[n]$ with the brick-wall lowpass filter with frequency response

$$\mathbf{H}(e^{j\Omega}) = \begin{cases} L & \text{for } 0 \le |\Omega| < \frac{\pi}{L} \\ 0 & \text{for } \frac{\pi}{L} < |\Omega| < \pi \end{cases} \quad (9.76)$$

and impulse response

$$h[n] = L\frac{\sin((\pi/L)n)}{\pi n}. \quad (9.77)$$

In practice, we would use a lowpass filter designed using FIR or IIR filter design techniques. This would work very well, but not perfectly.

Let $y[n]$ be $x[n]$ lowpass-filtered using $h[n]$. Then

$$y[n] = \sum_{i=-\infty}^{\infty} x[i]h[n-i] = \sum x[i]L\frac{\sin((\pi/L)(n-i))}{\pi(n-i)}. \quad (9.78)$$

The goal is to show that $y[n] = x[n]$ for n a multiple of L, so that interpolating $x[n]$ by convolving it with $h[n]$ preserves the given values of $x[n]$. Convolving $x[n]$ with $h[n]$ will change the zero values of $x[n]$ to values that make $y[n]$ have maximum frequency $\frac{\pi}{L}$, since $h[n]$ is a lowpass filter.

We are interested only in values of n that are multiples of L, so let $n = n'L$. Since $x[i] = 0$ unless i is a multiple of L, let $i = i'L$ in Eq. (9.78). Then Eq. (9.78) becomes

$$\begin{aligned} y[n'L] &= \sum_{i'=-\infty}^{\infty} x[i'L] L \frac{\sin((\pi/L)(n'L - i'L))}{\pi(n'L - i'L)} \\ &= \sum_{i'=-\infty}^{\infty} x[i'L] \frac{\sin(\pi(n' - i'))}{\pi(n' - i')} \\ &= \sum_{i'=-\infty}^{\infty} x[i'L] \delta[n' - i'] \\ &= x[n'L]. \end{aligned} \tag{9.79}$$

so the given values $x[n'L]$ of $x[n]$ are preserved after filtering with $h[n]$, which thus interpolates $\{x[n'L]\}$, preserving the given values and replacing the zero values of $x[n]$ with the appropriate numbers so that $x[n]$ is lowpass.

9.5.5 Upsampling and Interpolation

Upsampling by L alone does not just reduce frequencies of sinusoids by a factor of L. It also introduces additional sinusoids. But these can be removed by a lowpass filter. The combined operation of *upsample and interpolate* reduces frequencies *without adding new ones*, since additional frequencies are filtered out by interpolation.

Example 9-20: Upsampling and Interpolation.

A signal has the spectrum shown in Fig. 9-18a. Sketch the spectrum of the signal after upsampling by 4 and interpolation.

Solution:

The spectrum is compressed by a factor of 4. But interpolation removes the three additional copies of the spectrum that have been compressed into the interval $0 \le \Omega \le 2\pi$. The signal has all its frequencies reduced by 4.

[Change 3 to π, etc.]

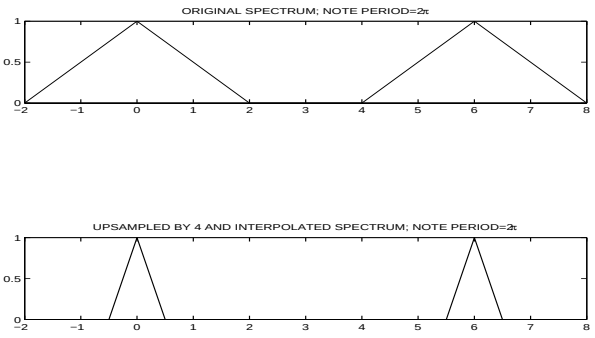


Figure 9.17: Effect of Upsampling and Interpolation on Spectrum. (a) Original Spectrum. (b) Upsampled and Interpolated Spectrum.

9.5.6 Multirate Signal Processing

The sampling rate can be multiplied by a rational number $\frac{M}{N}$ by: (1) Upsampling by M; (2) Interpolating using a lowpass filter with cutoff frequency $\frac{\pi}{M}$; (3) Downsampling by N:

$$x[n] \rightarrow \boxed{\uparrow M} \rightarrow \boxed{h[n] = \frac{\sin(\frac{\pi}{M}n)}{\pi n}} \rightarrow \boxed{\downarrow N} \rightarrow y[n] \tag{9.80}$$

Matlab/Mathscript Recipe for Multirate Signal Processing:

To multiply the sampling rate by $\frac{M}{N}$:

```
L=length(X);Z=[X;zeros(M-1,L)];Y=Z(:)';
F=fft(Y);F(L/2:L*M+2-L/2)=0;Y=ifft(F);
Y=Y(1:N:end);
```

It is important to perform the operations in this specific order, since downsampling first may lead to aliasing at the middle step. Upsampling followed by interpolation cannot lead to aliasing, but upsampling after downsampling may be too late to avoid aliasing. Upsampling first avoids aliasing, provided the final sampling rate exceeds double the maximum frequency of the original signal.

The following example illustrates this. We label spectral lines with their continuous-time frequencies f in Hertz, rather than with discrete-time frequency $\Omega{=}2\pi\frac{f}{S}$, where S is the sampling rate, for clarity.

Example 9-21: Multirate Signal Processing. Each system shown uses a sampling rate of 2400 samples per second. The input is a single sinusoid at the frequency shown. Determine the frequencies of the output sinusoids, in Hertz. Analog sampling and reconstruction are omitted. $h[n]$ is a brickwall lowpass filter with cutoff frequency $\pi/2$,

1. 600 Hertz$\rightarrow\boxed{\uparrow 3}\rightarrow$?
2. 500 Hertz$\rightarrow\boxed{\downarrow 3}\rightarrow\boxed{\uparrow 2}\rightarrow$?
3. 500 Hertz$\rightarrow\boxed{\uparrow 2}\rightarrow\boxed{h[n]}\rightarrow\boxed{\downarrow 3}\rightarrow$?

Solution:
A component at f_o implies components at $\pm\{f_o, 2400 \pm f_o, 4800 \pm f_o\}$ (see Section 6-12.8).

1. 600 Hertz$\rightarrow\boxed{\uparrow 3}\rightarrow\boxed{\{200, 600, 1000\}}$
   Input after sampling: $\pm\{600, 2400 \pm 600, \ldots\}$
   Upsampling divides by 3: $\pm\{200, 600, 1000, \ldots\}$
   Others are above the Nyquist rate of 1200 Hertz.
   Note a lowpass filter would remove two of these, leaving only $\frac{1}{3}(600){=}200$ Hertz.

2. 500$\rightarrow\boxed{\downarrow 3}\rightarrow\boxed{\uparrow 2}\rightarrow\boxed{\{450, 750\}}$
   Input after sampling: $\pm\{500, 2400 \pm 500, \ldots\}$
   Downsampling mults by 3: $\pm\{1500, 5700, \ldots\}$
   Aliases down to: $\pm\{1500$–$2400, 5700$–$4800 \ldots\}$
   $=\pm\{900, 2400 \pm 900\}$. Rest above Nyquist rate.
   Upsampling divides by 2: $\pm\{450, 750, \ldots\}$
   We wanted $\frac{3}{2}(500){=}750$! How can we do this?

3. 500 Hertz$\rightarrow\boxed{\uparrow 2}\rightarrow\boxed{h[n]}\rightarrow\boxed{\downarrow 3}\rightarrow\boxed{750}$
   Input after sampling: $\pm\{500, 2400 \pm 500, \ldots\}$
   Upsampling divides by 2: $\pm\{250, 950, \ldots\}$
   Lowpass filtering removes 950, leaving $\pm\{250\}$.
   Downsampling multiplies by 3: $\pm\{750\}$ only.

9.5.7 Oversampling by Upsampling

Overview of Oversampling by Upsampling

Ultimately, discrete-time signals must be converted to continuous-time signals, which can be physically displayed or heard. Sampling creates copies of the original signal spectrum repeated every S Hertz. These copies must be eliminated, using an *analog* lowpass filter such as the Butterworth filter presented in Section 6-8, to obtain the original signal spectrum.

If the original signal is sampled at a rate only slightly greater than the Nyquist rate, then a very selective *analog* filter is requires to reconstruct the signal from its samples. This analog filter will require many capacitors and resistors and physical space. But if the original signal is sampled at a rate many times greater than the Nyquist rate (oversampling), the analog filter can be much less selective, requiring fewer capacitors and resistors and taking up less physical space. This is illustrated in Section 6-12.12 in Example 6-20, in which oversampling at 4.5 times the Nyquist rate was necessary for a third-order Butterworth filter to reconstruct the original signal from its samples.

Alternatively, we can increase the sampling rate by upsampling and interpolating. This replaces a selective *analog* filter with a selective *discrete-time* filter, which can be implemented entirely in software and designed using the methods in the previous section. This is called *oversampling*; "16x oversampling" means the implemented sampling rate is 16 times the actual sampling rate. Making the implemented sampling rate this high means only a crude analog lowpass filter is needed; the *mechanical* lowpass filter inherent in the frequency response of an earbud may be enough.

Implementing Oversampling

Let $x[n]$ be samples of a signal $x(t)$ with maximum frequency B Hertz sampled at $S = 4B$ samples per second This is double the Nyquist rate of $2B$ samples per second. A selective analog filter, with about seven capacitors, would be necessary to recover $x(t)$ from its samples $x[n]$. The gain of this analog filter must

satisfy

$$|\mathbf{H}(j2\pi f)| = \begin{cases} 1 & \text{for } |f| < B \\ < 0.001 & \text{for } |f| > 3B \end{cases} \quad (9.81)$$

Instead, we can implement the oversampling system:

$$x[n] \rightarrow \boxed{\uparrow 10} \rightarrow \boxed{h[n]} \rightarrow \boxed{\text{ANALOG}} \rightarrow x(t) \quad (9.82)$$

where the discrete-time filter $h[n]$ satisfies ($S = 4B$)

$$|\mathbf{H}(e^{j\Omega})| = \begin{cases} 1 & \text{for } |\Omega| < 2\pi\frac{B}{10S}=\frac{\pi}{20} \\ < 0.001 & \text{for } |\Omega| > 2\pi\frac{3B}{10S}=\frac{3\pi}{20} \end{cases} \quad (9.83)$$

The analog filter must now satisfy ($10S$–B=$39B$)

$$|\mathbf{H}(j2\pi f)| = \begin{cases} 1 & \text{for } |f| < B \\ < 0.001 & \text{for } |f| > 39B \end{cases} \quad (9.84)$$

which requires only two capacitors. Using upsampling, we have obviated the need for a highly selective analog reconstruction filter.

Also note that since the upsampled $x[n]$ is 90% zeros, the computation of filtering with $h[n]$ is reduced by 90% to 10% of its usual amount.

9.5.8 Audio Signal Processing

Suppose we are given a snippet of a trumpet playing a single note. The goal is to generate snippets of the trumpet playing all of the other musical notes from the single note. We can do this using multirate filtering and the *Circle of Fifths* from music theory.

Music Notation and Frequencies

Most Western music is based on the 12-tone or chromatic musical scale. This scale consists of 7 whole notes, designated by letters A,B,C,D,E,F,G and 5 accidental notes, designated by A#,C#,D#,F#,G# and pronounced "A-sharp," etc. The frequencies of these 12 notes are in geometric progression, with a common ratio of $2^{\frac{1}{12}}$, so they span a factor of two. This is called an octave. The reason for using frequencies in geometric progression is to preserve the ratio between frequencies of notes after a key change.

The frequency of A# is between the frequencies of notes A and B. A# is also designated Bb, pronounced "B-flat," and similarly for the other accidental notes. On a piano keyboard, whole notes are played on the white keys, and accidental notes are played on the black keys. This can all be summarized as follows:

Note	A	A#	B	C	C#	D
$\frac{\text{FREQ}}{440\,\text{Hz}}$	$2^{\frac{0}{12}}$	$2^{\frac{1}{12}}$	$2^{\frac{2}{12}}$	$2^{\frac{3}{12}}$	$2^{\frac{4}{12}}$	$2^{\frac{5}{12}}$
Hertz	440	466	494	523	554	587

Note	D#	E	F	F#	G	G#
$\frac{\text{FREQ}}{440\,\text{Hz}}$	$2^{\frac{6}{12}}$	$2^{\frac{7}{12}}$	$2^{\frac{8}{12}}$	$2^{\frac{9}{12}}$	$2^{\frac{10}{12}}$	$2^{\frac{11}{12}}$
Hertz	622	659	698	740	784	830

Table 9-3. Frequencies of Musical Notes.

Musical Circle of Fifths

Note A is 440 Hertz and E is 659 Hertz, almost exactly a 3:2 ratio. The 3:2 ratio holds between different pairs of notes called "fifths." Indeed, Western music was first based on using these ratios, rather than the equally-spaced logarithms of frequencies used today. The change occurred when J.S. Bach composed "The Well-Tempered Clavier" using the 12-tone scale. It sounded better than using the frequencies based on 3:2 ratios ("The Ill-Tempered Clavier"?).

The relevance of this to the present problem is apparent from the following table:

Freq. / 440 Hz	$\frac{(\frac{3}{2})^0}{2^0}$	$\frac{(\frac{3}{2})^1}{2^0}$	$\frac{(\frac{3}{2})^2}{2^1}$	$\frac{(\frac{3}{2})^3}{2^1}$	$\frac{(\frac{3}{2})^4}{2^2}$	$\frac{(\frac{3}{2})^5}{2^2}$
Hertz	440	660	495	742	557	835
Note	A	E	B	F#	C#	G#

$\frac{(\frac{3}{2})^6}{2^3}$	$\frac{(\frac{3}{2})^7}{2^4}$	$\frac{(\frac{3}{2})^8}{2^4}$	$\frac{(\frac{3}{2})^9}{2^5}$	$\frac{(\frac{3}{2})^{10}}{2^5}$	$\frac{(\frac{3}{2})^{11}}{2^6}$	$\frac{(\frac{3}{2})^{12}}{2^7}$
626	470	705	529	793	595	446
D#	A#	F	C	G	D	A

Table 9-4. Frequencies in the Circle of Fifths.

This is called the Circle of Fifths because arranging the 12 semitones in a circle, like hours on a clock face, and taking repeated jumps of 7 "hours" each, we can attain all 12 "hours." A Fifth is the interval between 5 whole tones (such as A to E above) or 7 notes. This will sweep out all of the notes in the octave because: (1) 7 and 12 are relatively prime; and (2) $\frac{3}{2} \approx 2^{7/12}$.

Application of Multirate Signal Processing

By repeatedly upsampling by 2 and downsampling by 3, we can start with the snippet of a trumpet playing note A and multiply that frequency (440 Hertz) by powers of $\frac{3}{2}$. This sweeps out all of the 12 notes, in the order shown in the above table. Occasionally an extra upsampling by 2 is necessary to keep the frequencies within this octave. We can then obtain the 12 notes in all other octaves by upsampling and downsampling by 2 to synthesize the same note in a different octave. This is implemented in problem 9-35 by repeatedly using

- $x[n] \rightarrow \boxed{\uparrow 2} \rightarrow \boxed{h[n]} \rightarrow \boxed{\downarrow 3} \rightarrow y[n]$
- $\mathbf{H}(e^{j\Omega}) = \begin{cases} 3 & \text{for } 0 < |\Omega| < \frac{\pi}{2} \\ 0 & \text{for } \frac{\pi}{2} < |\Omega| < \pi \end{cases}$
- If the fundamental frequency>784 Hertz then upsample by 2 to reduce the frequency by half.

Ironically, this would be easier to implement in continuous time. A variable speed tape recorder or record turntable could implement multirate processing by speeding up or slowing down the tape or turntable, using a variable-speed motor. "Alvin and the Chipmunks" were recorded this way in the 1950's by speeding up the playback by a factor of two on a variable-speed tape recorder. It was then necessary to (literally!) cut and paste snippets of tape.

9.5.9 Filter Design

A selective filter has a sharp transition between its pass-band and its stop-band. We have seen in continuous time that a selective Butterworth filter requires many capacitors. In discrete time, there are no such physical requirements, but the order (length) of the filter must be large. We have seen this for notch filters, for which a more selective frequency response requires poles nearer the unit circle, and thus longer impulse and transient responses.

We can use multirate to implement a very selective filter as a cascade of less selective filters, with downsampling and upsampling. The idea is as follows:

- Use downsampling to stretch the frequency axis;
- Design the filter on the stretched axis;
- Use upsampling to compress the frequency axis.

The transition band is wider on the stretched axis.

We illustrate this idea with an example.

Example 9-22: Filter Design Using Multirate Signal Processing.

Implement the filter with specs

$$\mathbf{H_D}(e^{j\Omega}) = \begin{cases} 1 & \text{for } |\Omega| < 0.0095\pi \\ 0 & \text{for } |\Omega| > 0.01\pi \end{cases} \tag{9.85}$$

using the three filters with specs

$$\mathbf{H_1} = \begin{cases} 1 & \text{for } |\Omega| < 0.0095\pi \\ 0 & \text{for } |\Omega| > 0.1\pi \end{cases} \tag{9.86}$$

$$\mathbf{H_2} = \begin{cases} 1 & \text{for } |\Omega| < 0.095\pi \\ 0 & \text{for } |\Omega| > 0.1\pi \end{cases} \tag{9.87}$$

$$\mathbf{H_3} = \begin{cases} 1 & \text{for } |\Omega| < 0.01\pi \\ 0 & \text{for } |\Omega| > 0.1\pi \end{cases} \tag{9.88}$$

Note the transition widths:

- $\mathbf{H_D}(e^{j\Omega})$: 0.0005π. This is very sharp. This is the desired frequency response.
- $\mathbf{H_1}(e^{j\Omega})$: 0.0905π. This is very dull. This is an anti-alias filter before downsampling.
- $\mathbf{H_2}(e^{j\Omega})$: 0.0050π. This is rather sharp. $\mathbf{H_2}(e^{j\Omega})$ is $\mathbf{H_D}(e^{j\Omega})$ stretched by 10.
- $\mathbf{H_3}(e^{j\Omega})$: 0.0900π. This is very dull. This is the interpolating filter after upsampling.

We use the following system:

$$x[n] \rightarrow \boxed{\mathbf{H_1}} \rightarrow \boxed{\downarrow 10} \rightarrow \boxed{\mathbf{H_2}} \rightarrow \boxed{\uparrow 10} \rightarrow \boxed{\mathbf{H_3}} \rightarrow y[n] \tag{9.89}$$

First, we design the rather sharp filter $\mathbf{H_2}(e^{j\Omega})$. Then upsampling compresses the frequency axis, so both the pass-band-to-transition-band border 0.095π and the stop-band-to-transition-band border 0.1π are divided by 10, which are the specs for $\mathbf{H_2}(e^{j\Omega})$. We implemented $\mathbf{H_D}(e^{j\Omega})$ with less-sharp filters!

Exercise 9-10: $\cos(0.6\pi n) \rightarrow \boxed{\uparrow \mathbf{3}} \rightarrow \boxed{\downarrow \mathbf{3}} \rightarrow ?$

Answer: Upsampling inserts zeros, downsampling removes them. So the output equals the input for any input!

Exercise 9-11: $\cos(0.6\pi n) \rightarrow \boxed{\downarrow 3} \rightarrow ?$
Answer: $\cos(0.6\pi n)$, since $\omega = 0.6\pi$ becomes $\omega = 1.8\pi$ which aliases to $\omega = 0.2\pi$, since $1.8\pi \equiv -0.2\pi \equiv 0.2\pi$.

Exercise 9-12: $\cos(0.8\pi n) \rightarrow \boxed{\downarrow 4} \rightarrow ?$
Answer: $\cos(0.8\pi n)$, since $\omega = 0.8\pi$ becomes $\omega = 3.2\pi$ which aliases to $\omega = 0.8\pi$, since $3.2\pi \equiv -0.8\pi \equiv 0.8\pi$.

Exercise 9-13: $\cos(0.4\pi n) \rightarrow \boxed{\uparrow 4} \rightarrow ?$
Answer: $\omega = \{0.4\pi, (2-0.4)\pi, (2+0.4)\pi, (4-0.4)\pi\}$ become $\omega = \{0.1\pi, 0.4\pi, 0.6\pi, 0.9\pi\}$. The input was a single sinusoid, but the output is four sinusoids.

Exercise 9-14: $\cos(0.8\pi n) \rightarrow \boxed{\uparrow 4} \rightarrow ?$
Answer: $\omega = \{0.8\pi, (2-0.8)\pi, (2+0.8)\pi, (4-0.8)\pi\}$ become $\omega = \{0.2\pi, 0.3\pi, 0.7\pi, 0.8\pi\}$.
The input was a single sinusoid, but the output is four sinusoids.

9.6 Correlation

Correlation is a general term that describes three different functions of two signals $x[n]$ and $y[n]$. These functions, their applications, and their definitions, are as follows:

- *Auto*correlation, for the period of a signal. Autocorrelation $r_x[n] = x[n] * x[-n]$;
- *Cross*-correlation, for the delay of a signal. Cross-correlation $r_{xy}[n] = x[n] * y[-n]$;
- *Correlation*, for presence or absence of signals. Correlation $r_{xy}[0] = \sum_{i=0}^{N-1} x[i]y[i]$, where N is the durations of $x[n]$ and $y[n]$.

The cross-correlation of two signals is another signal, while the corrrelation of two signals is a number. Correlation can also be used to determine which one of several possible signals is present (see below).

We discuss each of these three functions in a separate subsection. We then show how to apply them to real-world signals.

9.6.1 Autocorrelation

Definition of Autocorrelation

The *autocorrelation* $r_x[n]$ of $x[n]$ is

$$r_x[n] = x[n] * x[-n] = \sum_{i=-\infty}^{\infty} x[i]x[i+n] \qquad (9.90)$$

The *lag* n has units of time but it is *not* time. Instead, lag is the delay of one signal relative to itself.

Writing out the definition of $r_x[n]$ for the lags $n = 0, \pm 1, \pm 2$ gives

$$\begin{aligned} r_x[0] &= \ldots x[-1]^2 + x[0]^2 + x[1]^2 + \ldots \\ r_x[\pm 1] &= \ldots x[-1]x[0] + x[0]x[1] + x[1]x[2] + \ldots \\ r_x[\pm 2] &= \ldots x[-1]x[1] + x[0]x[2] + x[1]x[3] \end{aligned} \qquad (9.91)$$

Autocorrelation has these properties:

- $r_x[n]$ is an even function: $r_x[n] = r_x[-n]$;
- $r_x[0] = \sum_{i=-\infty}^{\infty} x[i]^2 \geq 0$=energy of $x[n]$;
- $r_x[0] \geq r_x[n]$ for any *lag* n.
  Usually $r_x[0] >> r_x[n]$ unless $x[n]$ is periodic.

The DTFT $\mathbf{R_x}(e^{j\Omega})$ of $r_x[n]$ is related to $\mathbf{X}(e^{j\Omega})$:

$$\begin{aligned} \mathbf{R_x}(e^{j\Omega}) &= \mathbf{X}(e^{j\Omega})\mathbf{X}(e^{-j\Omega})^* \\ &= |\mathbf{X}(e^{j\Omega})|^2. \end{aligned} \qquad (9.92)$$

using the time-reversal and conjugate-symmetry properties of the DTFT. Also note that $|\mathbf{X}(e^{j\Omega})|^2$ is real and even, confirming that $r_x[n]$ is even.

If $x[n]$ has finite duration N, then $r_x[n] = 0$ except for $-(N-1) \leq n \leq N-1$ and $r_x[n]$ has duration $2N-1$. Delaying $x[n]$ by D does not affect $r_x[n]$, since the DTFT of $x[n-D]$ is $\mathbf{X}(e^{j\Omega})e^{-j\Omega D}$ by the time-delay property of the DTFT. Then

$$\begin{aligned} \mathbf{R_{x-D}}(e^{j\Omega}) &= |\mathbf{X}(e^{j\Omega})e^{-j\Omega D}|^2 \\ &= |\mathbf{X}(e^{j\Omega})|^2 \\ &= \mathbf{R_x}(e^{j\Omega}). \end{aligned} \qquad (9.93)$$

since the DTFTs of the autocorrelations are equal, the autocorrelations are equal.

Matlab/Mathscript Recipe for Computing Autocorrelation:

Compute the autocorrelation of `X` using
`N=length(X);M=nextpow2(2*N-1);`
`RX=real(ifft(abs(fft(X,M)).^2));`
`RX=fftshift(RX)` puts $r_x(0)$ in the middle.

Example 9-23: Computing Autocorrelation.
Compute the autocorrelation of $x[n] = \{3, 1, 4\}$. Note the location of $n = 0$ in $x[n]$ is irrelevant.

Solution:

$$\begin{aligned} r_x[0] &= 3^2 + 1^2 + 4^2 = 26. \\ r_x[\pm 1] &= (3)(1) + (1)(4) = 7. \\ r_x[\pm 2] &= (3)(4) = 12. \\ r_x[n] &= \{12, 7, \underline{26}, 7, 12\}. \end{aligned} \quad (9.94)$$

$r_x[n]$ is even and $r_x[0] \geq r_x[n]$, as expected.

Using Autocorrelation to Compute Period

Let $x[n]$ be a signal with unknown period N. The goal is to estimate N from the noisy observations

$$y[n] = x[n] + w[n] \quad (9.95)$$

where $w[n]$ is white Gaussian noise (presented in Section 6-9.1).

The autocorrelation $r_y[n]$ of $y[n]$ is

$$\begin{aligned} r_y[n] &= (x[n] + w[n]) * (x[-n] + w[-n]) \\ &= x[n] * x[-n] + w[n] * w[-n] \\ &+ \underbrace{x[n] * w[-n]}_{r_{xw}[n] \approx 0} + \underbrace{x[-n] * w[n]}_{r_{xw}[-n] \approx 0} \\ &\approx r_x[n] + r_w[n] \approx r_x[n] + \sigma^2 \delta[n] \end{aligned} \quad (9.96)$$

But since $x[n] = x[n + N]$,

$$\begin{aligned} r_x[N] &= \sum x[i]x[i+N] = \sum x[i]x[i] = r_x[0] \\ r_x[2N] &= \sum x[i]x[i+2N] = \sum x[i]x[i] = r_x[0] \\ &\vdots \quad \vdots \quad \vdots \\ r_x[n] &\text{ is periodic with period } N. \end{aligned} \quad (9.97)$$

So, to a good approximation,

$$r_y[n] \approx \begin{cases} \sigma^2 + \sum x[i]^2 & \text{for } n = 0 \\ \sum x[i]^2 & \text{for } n = N \\ \sum x[i]^2 & \text{for } n = 2N \\ \vdots & \vdots \\ 0 & \text{otherwise} \end{cases} \quad (9.98)$$

So the period N is found by looking for large peaks in $r_y[n]$, which will be at $n = 0, N, 2N, \ldots$. The peak at $n = 0$ is much larger, since σ^2 is added to it.

The following example demonstrates how to use autocorrelation to determine the period of a signal from noisy observations of it.

Example 9-24: Estimation of Period.
Determine the note played by a trumpet from noisy observations of the trumpet signal.

Solution:
The results are plotted in Fig. 9-19.

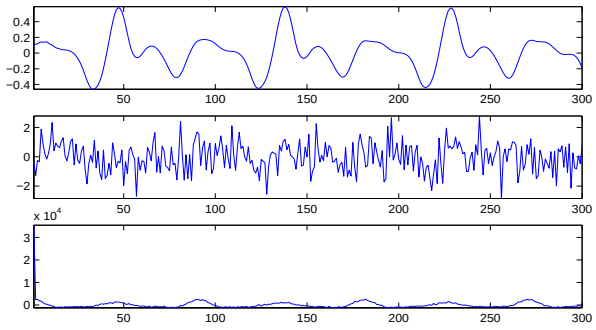


Figure 9.18: Estimating Period of Trumpet. (a) Noiseless Trumpet Signal; (b) Noisy Trumpet Signal; (c) Autocorrelation of Noisy Trumpet Signal.

Try estimating the period by examining the second plot (the noisy trumpet signal)! But the autocorrelation is clearly periodic with period 90, except the peak at $n = 0$ is larger since σ^2 is added to it.

The sampling rate is the usual CD sampling rate of 44100 samples per sec., so the period is $\frac{90}{44100} = \frac{1}{490}$ seconds and the fundamental frequency is 490 Hertz. The trumpet is playing note B (494 Hertz).

The Matlab code for this example is on the CD.

9.6.2 Cross-Correlation

Definition of Cross-correlation

The *cross-correlation* $r_{xy}[n]$ of $x[n]$ and $y[n]$ is

$$\begin{aligned} r_{xy}[n] &= x[n] * y[-n] = \sum_{i=-\infty}^{\infty} x[i]y[i-n]. \\ r_{yx}[n] &= y[n] * x[-n] = \sum_{i=-\infty}^{\infty} x[i]y[i+n] \end{aligned} \quad (9.99)$$

The *lag* n has units of time but it is *not* time. Lag is the delay of one signal relative to the other.

Writing out the definition of $r_{xy}[n]$ for $n = 0, 1, 2$ gives

$$\begin{aligned} r_{xy}[0] &= \ldots x[-1]y[-1] + x[0]y[0] + x[1]y[1] + \ldots \\ r_{xy}[1] &= \ldots x[0]y[-1] + x[1]y[0] + x[2]y[1] + \ldots \\ r_{xy}[2] &= \ldots x[0]y[-2] + x[1]y[-1] + x[2]y[0] \end{aligned} \quad (9.100)$$

Cross-correlation, unlike autocorrelation, is not an even function, although $r_{yx}[n] = r_{xy}[-n]$. The DTFT $\mathbf{R_{xy}}(e^{j\Omega})$ of $r_{xy}[n]$ is related to $\mathbf{X}(e^{j\Omega})$ and $\mathbf{Y}(e^{j\Omega})$ by

$$\mathbf{R_{xy}}(e^{j\Omega}) = \mathbf{X}(e^{j\Omega})\mathbf{Y}(e^{j\Omega})^* \quad (9.101)$$

using the time-reversal and conjugate symmetry properties of the DTFT.

Matlab/Mathscript Recipe for Computing Cross-Correlation:

Compute the cross-correlation of `X` and `Y` using:
`L=length([X Y]);M=nextpow2(L-1);`
`RXY=real(ifft(fft(X,M).*conj(fft(Y,M))));`
`RXY=fftshift(RXY)` puts $r_{xy}(0)$ in the middle.

Example 9-25: Compute Cross-correlation.
Compute the cross-correlation of $x[n] = \{3, 1, 4\}$ and $y[n] = \{2, 7, 1\}$.

Solution:
We can use $\{3, 1, 4\} * \{1, 7, 2\}$ or

$$\begin{aligned} r_{xy}[-2] &= (3)(1) = 3. \\ r_{xy}[-1] &= (3)(7) + (1)(1) = 22. \\ r_{xy}[0] &= (3)(2) + (1)(7) + (4)(1) = 17. \\ r_{xy}[1] &= (1)(2) + (4)(7) = 30. \\ r_{xy}[2] &= (4)(2) = 8. \\ r_{xy}[n] &= \{3, 22, \underline{17}, 30, 8\}. \end{aligned} \quad (9.102)$$

Using Cross-Correlation to Compute Time Delay

Let $x[n]$ be a known signal with unknown time delay N. Without loss of generality, let the energy of $x[n]$ be one. The goal is to estimate N from the noisy observations

$$y[n] = x[n-N] + w[n] \quad (9.103)$$

$w[n]$ is again white Gaussian noise. $r_{yx}[n]$ is

$$\begin{aligned} r_{yx}[n] &= (x[n-N] + w[n]) * x[-n] \\ &= \underbrace{x[n-N] * x[-n]}_{r_x[n-N]} + \underbrace{w[n] * x[-n]}_{r_{xw}[-n] \approx 0} \\ &\approx r_x[n-N] \approx \delta[n-N] \end{aligned} \quad (9.104)$$

The final approximation $r_x[n] \approx \delta[n]$ is *very* rough. But the cross-correlation of the observations with the known signal is expected to have a peak at $n = N$.

Example 9-26: Estimation of Time Delay.
A known short random pulse $x[n]$ is delayed by an unknown N. Estimate N from noisy observations.

Solution:
Results are shown in Fig. 9-20.

Try estimating the delay by examining the second plot (the noisy delayed pulse)! But the third plot shows clearly that the delay is $N = 300$. This is an extreme example; usually the peak is not that sharp. But $r_x[n]$ does tend to have a peak at $n = 0$.

9.6.3 Correlation

Definition of Correlation

The *correlation* between signals $x[n]$ and $y[n]$ of durations N each (zero-pad them to the same length if needed) is the *number*

$$r_{xy}[0] = \sum_{i=0}^{N-1} x[i]y[i]. \quad (9.105)$$

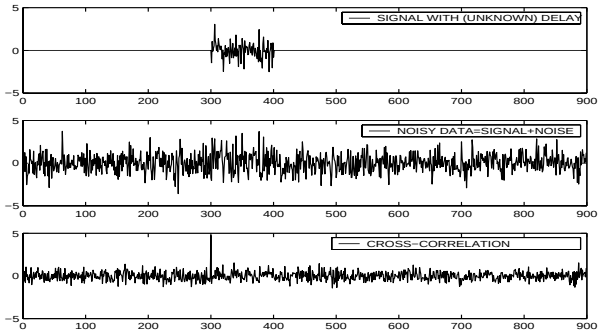


Figure 9.19: Time Delay Estimation. (a) Noiseless Delayed Pulse; (b) Noisy Delayed Pulse; (c) Cross-Correlation with Non-Delayed Pulse.

This is the inner product of the vectors of the elements of $\{x[n]\}$ and $\{y[n]\}$. In Matlab/Mathscript this is `X'*Y`. Recalling that the inner product of vectors $\vec{x}$ and $\vec{y}$ is

$$\vec{x}^T\vec{y} = ||\vec{x}|| \cdot ||\vec{y}|| \cos\theta \tag{9.106}$$

where θ is the angle between $\vec{x}$ and $\vec{y}$ leads to the concept of signal similarity, which is discussed next.

Signal Similarity

The significance of correlation is that it is a measure of *signal similarity* (how much two signals are alike). To show this, we require the *Cauchy-Schwarz inequality*

$$|r_{xy}[0]| \leq \sqrt{r_x[0]r_y[0]}. \tag{9.107}$$

The Cauchy-Schwarz inequality is derived as follows:

$$\begin{aligned} 0 &\leq \sum_{n=0}^{N-1}\left[\frac{x[n]}{\sqrt{r_x[0]}} \pm \frac{y[n]}{\sqrt{r_y[0]}}\right]^2 \\ &= \sum_{n=0}^{N-1}\frac{x[n]^2}{r_x[0]} + \sum_{n=0}^{N-1}\frac{y[n]^2}{r_y[0]} \pm 2\sum_{n=0}^{N-1}\frac{x[n]y[n]}{\sqrt{r_x[0]r_y[0]}} \\ &= 1 + 1 \pm 2\frac{r_{xy}[0]}{\sqrt{r_x[0]r_y[0]}} \end{aligned} \tag{9.108}$$

Multiplying by $\sqrt{r_x[0]r_y[0]}$ and choosing the sign so $\pm r_{xy}[0] \leq 0$ yields the Cauchy-Schwarz inequality.

We define the "angle" θ between $x[n]$ and $y[n]$ by

$$\cos\theta = r_{xy}[0]/\sqrt{r_x[0]r_y[0]} = r_{xy}[0]/\sqrt{E_xE_y} \tag{9.109}$$

since $r_x[0]$ is the energy E_x of $x[n]$.

$\cos\theta$ is called the *correlation coefficient* of $x[n]$ and $y[n]$; it is the energy-normalized version of correlation. In statistics, the correlation coefficient is defined by first subtracting off the means of $x[n]$ and $y[n]$ and then applying (9.109) to the results, but that is not necessary here. The closer θ is to zero, the more similar are $x[n]$ and $y[n]$, excluding a scale factor.

Example 9-27: Signal Similarity
Let the signals $x[n]$ and $y[n]$ be

$$\begin{aligned} x[n] &= A_1\cos(2\pi(M/N)n + \theta_1) \\ y[n] &= A_2\cos(2\pi(M/N)n + \theta_2). \end{aligned} \tag{9.110}$$

Solution:
Some straightforward algebra shows that the correlation between these two sinusoids is

$$\cos\theta = r_{xy}[0]/\sqrt{r_x[0]r_y[0]} = \cos(\theta_1 - \theta_2). \tag{9.111}$$

θ is literally the phase angle between $x[n]$ and $y[n]$!

In particular, $A_1\cos(2\pi(M/N)n)$ and $A_2\sin(2\pi(M/N)n)$ are both *orthogonal* since $\theta = 90^o$. Note that using frequency $2\pi(M/N)$ ensures an integer number M of periods in the interval $0 \leq n \leq N-1$.

Classification of Signals

Suppose we have noisy observations of exactly one (but *which* one is unknown) of L possible known signals of durations N each: $\{x_1[n] \ldots x_L[n]\}$:

$$y[n] = w[n] + \begin{cases} x_1[n] & \text{OR} \\ \vdots & \\ x_L[n] & \end{cases} \tag{9.112}$$

where $w[n]$ is zero-mean white Gaussian noise. The goal is to determine which one of $\{x_1[n] \ldots x_L[n]\}$ is present, from the noisy observations $y[n]$.

The solution is to choose the signal that minimizes

$$\sum_{n=0}^{N-1}\left[\frac{y[n]}{\sqrt{r_y[0]}}-\frac{x_i[n]}{\sqrt{r_{x_i}[0]}}\right]^2 \tag{9.113}$$

which is the "squared distance" between the energy-normalized data $y[n]$ and each of the energy-normalized signals $x_i[n]$. Using the same algebra (9.108) used to derive the Cauchy-Schwarz inequality, this is minimized by computing the correlation coefficient of $y[n]$ with each of $\{x_1[n]\dots x_L[n]\}$ and choosing the largest (closest to one):

$$\underset{i}{\text{argmax}}\left[\frac{r_{yx_i}[0]}{\sqrt{r_y[0]r_{x_i}[0]}}=\frac{\sum_{n=0}^{N-1}y[n]x_i[n]}{\sqrt{\sum_{n=0}^{N-1}y[n]^2\sum_{n=0}^{N-1}x_i[n]^2}}\right] \tag{9.114}$$

Example 9-28: Time Delay Estimation, Revisited

Formulate the time delay problem as a signal detection problem, and solve it using correlation.

Solution:

Formulate the time delay problem as

$$y[n]=w[n]+\begin{cases}x[n] & \text{OR}\\ x[n-1] & \text{OR}\\ x[n+1] & \text{OR}\\ \vdots\end{cases} \tag{9.115}$$

so that $x_i[n]$=$x[n-i]$. No energy normalization is necessary here, since all of the $x_i[n]$ have the same energy. The solution is then

$$\underset{i}{\text{argmax}}\left[\sum y[n]x_i[n]=\sum y[n]x[n-i]=r_{yx}[i]\right] \tag{9.116}$$

which agrees with the solution used before.

By combining time delay and signal detection, we can also incorporate unknown time delays N_i in each of the $x_i[n]$ in the general signal detection problem.

Example 9-29: Signal Classification.

We have noisy observations, shown in Fig. 9-21c, of one of the two signals shown in Fig. 9-21a and 9-21b. The first signal is the waveform of a clarinet, and the second signal is the waveform of a trumpet.

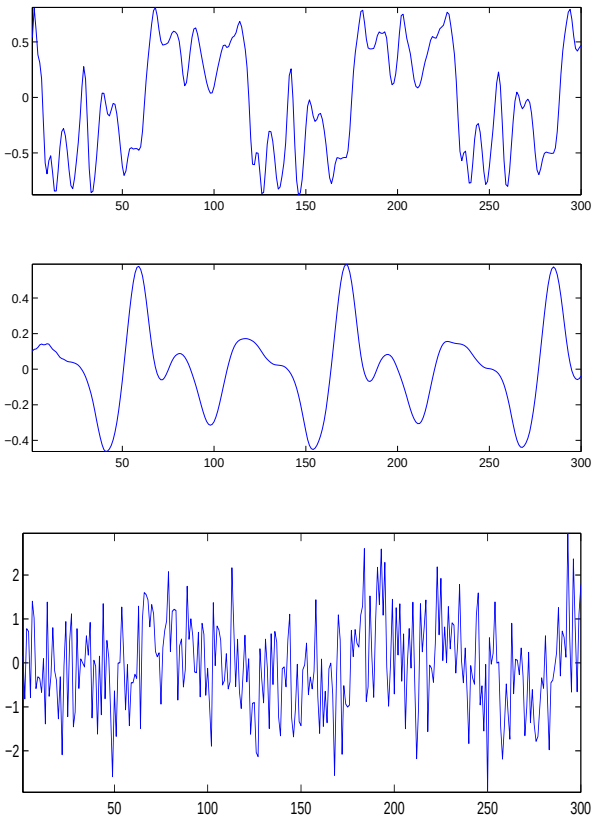


Figure 9.20: (a) Waveform of Clarinet; (b) Waveform of Trumpet; (c) Noisy Waveform of One or the Other.

It is not evident which signal is present. The goal is to determine which signal is present.

The correlation coefficients of the noisy observations with each of the two signals are, respectively, 0.441 and -0.055. So the *first* signal is the one that is present. The signal to noise ratio was -5.95.

MATLAB code used for this figure:

```
clear;load corre;N=randn(1,1000);
subplot(211),plot(X1(1:300)),axis tight
subplot(212),plot(X2(1:300)),axis tight
Y=X1+N;figure%Don't know it's X1.
subplot(211),plot(Y(1:300)),axis tight
RYX1=Y*X1'/sqrt(X1*X1')/sqrt(Y*Y');
RYX2=Y*X2'/sqrt(X2*X2')/sqrt(Y*Y');
[RYX1,RYX2,10*log10((X1*X1')/(N*N'))]
```

9.7 Biomedical Applications

We now present three biomedical applications:

- Computation of heartbeat rate from an EKG using autocorrelation;
- Classification of an EKG into one of three possible types (one normal, two abnormal);
- Computation of time delay in ultrasound.

A sampling rate of 100 samples/second is used. Each EKG signal has a duration of 10 seconds (1000 samples). The program used to generate these results is on the CD.

9.7.1 Electrocardiograms (EKG's)

An *electrocardiogram* (EKG) is a measurement of the electrical potential of the heart as a function of time. An EKG is measured using a collection of wire leads taped to the surface of the chest. Notch filters (see Section 8-2) are often used to remove 60 Hz interference from the wire leads. It is (hopefully!) periodic with a period of about one second (60 beats per minute); exercise raises this rate, but a fast heartbeat (i.e., smaller period) for an extended time is cause for concern.

An EKG waveform is useful in diagnosing heart malfunctions. Electrocardiograms are also called ECG's. We will use EKG here. EKG's and ECG's should not be confused with EEG's, which are electroencephalograms (brain electrical potentials).

Three types of EKG waveforms are shown in Fig. 9-22. Although these waveforms are synthetic, they do bear some resemblance to actual EKG waveforms. The vertical scales are in millivolts (mv).

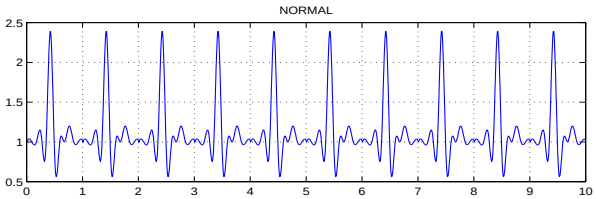


Fig. 9-22a shows the waveform is a normal EKG, with a period of one second (60 beats per minute).

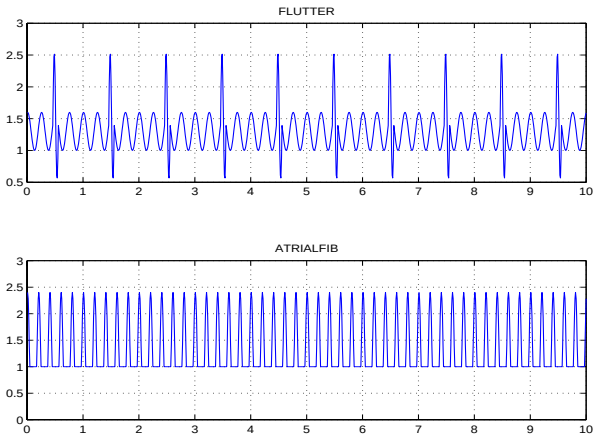


Figure 9.21: EKG's: (a) Normal; (b) Atrial Flutter; (c) Atrial Fib.

Fig. 9-22b is the waveform is an *atrial flutter*. Note there are three additional bumps in the waveform, so the heart is beating faster, at 240 beats per minute. The period of this waveform is one second, but its spectrum (see Fig. 9-24b below) is dominated by its 4^{th} harmonic of 4 Hz (with period 0.25 second).

Fig. 9-22c is the waveform of an *atrial fibrillation*, often called "atrial fib." The waveform consists entirely of bumps, at 300 beats per minute. The fundamental of its spectrum is at 5 Hz. This is a serious heart malfunction; a pacemaker is required to correct this to normal cardiac action.

9.7.2 Heart Rate by Autocorrelation

The obvious way to compute the period of a periodic signal in noise is to compute its autocorrelation.

Zero-mean white Gaussian noise with standard deviation 0.5 (see Section 6-9.1) was added to the ten-second-long segment of each waveform. Autocorrelations of the noisy signals were then computed. The noisy waveforms, and their autocorrelations, are shown in Fig. 9-23.

The autocorrelation of the noisy normal EKG has an evident peak at one second, so this is the period (60 beats per minute). There is a second peak at double the period, as expected. Despite the noise,

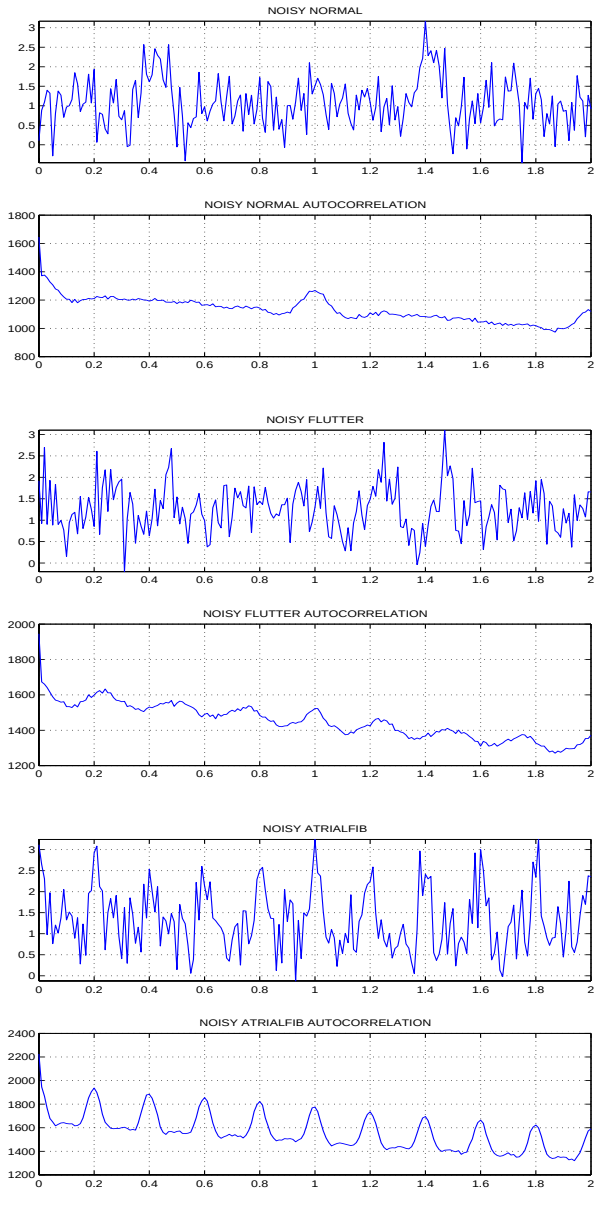


Figure 9.22: Noisy EKG's and their Autocorrelations. (a) Noisy Normal EKG and (b) its Autocorrelation. (c) Noisy Atrial Flutter EKG and (d) its Autocorrelation. (e) Noisy Atrial Fib EKG and (f) its Autocorrelation.

the peak is quite apparent.

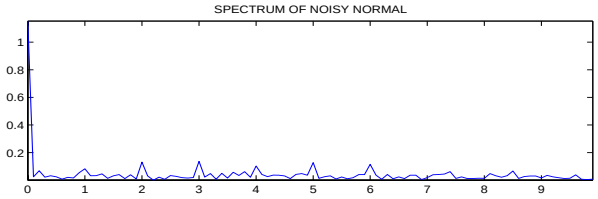
The autocorrelation of the noisy atrial flutter EKG has peaks at $\{0.25, 0.50, 0.75 \ldots\}$ seconds, confirming that the period is 0.25 second (240 beats per minute). However, the larger peak at 1 second confirms that a second periodicity is present. Althought the period of the signal is technically one second (it repeats every one second), the signal can also be regarded as the sum of two periodic signals with periods 0.25 and one second. This is not at all evident from the noisy waveform directly.

The autocorrelation of the noisy atrial fib EKG has peaks at $\{0.2, 0.4, 0.6 \ldots\}$ seconds, confirming that the period is 0.2 second (300 beats per minute). This is not at all evident from the noisy waveform directly. This patient requires immediate attention.

9.7.3 Heart Rate from Spectrum

The period (1 second) of the atrial flutter EKG is not a good indicator of the actual heart rate (240 beats per minute). This suggests computing the spectrum of an EKG signal does a better job of indicating its features. This can also handle larger amounts of noise.

Zero-mean white Gaussian noise with standard deviation 1.0 (see Section 6-9.1) was added to the ten-second-long segment of each waveform. This is *double the noise level used for autocorrelation above.* Spectra of the noisy signals were then computed. The results are shown in Fig. 9-24.



The spectrum of the normal EKG has clear peaks at 1,2,3... Hz (period one second). The harmonics seem to be significant only up to 10 Hz.

For the spectrum of the atrial flutter EKG, the fundamental at 1 Hz is barely discernable; the harmonic at 4 Hz (period 0.25 seconds) dominates the spectrum. This indicates a heart rate of 240 beats per

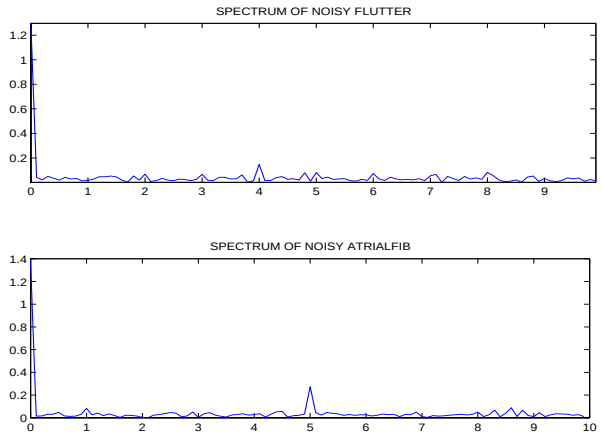


Figure 9.23: Spectrum of Noisy EKG's. (a) Normal; (b) Atrial Flutter; (c) Atrial Fib.

minute, not the normal 60 beats per minute. The spectrum does a much better job of indicating the elevated heart rate.

The spectrum of the atrial fib EKG has a peak at 5 Hz (period 0.20 seconds) that dominates the spectrum. This indicates an elevated heart rate of 300 beats per minute.

Example 9-30: Classification.

Zero-mean white Gaussian noise with standard deviation 0.5 (see Section 6-9.1) was added to the ten-second-long segment of one of the three EKG waveforms. The noisy waveform is shown in Fig. 9-25. Which one of the three EKG waveforms is this?

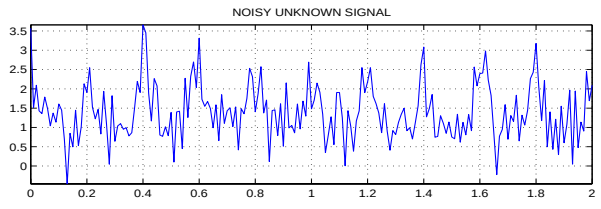


Figure 9.24: Noisy Unknown Waveform

Solution:

The corrrelation coefficients of the noisy waveform with each of the three known waveforms were computed. The results were: $\{0.8763, 0.8586, 0.9451\}$, respectively. So the unknown waveform is found (correctly) to be atrial fib. This is important in cardiac diagnosis.

9.7.4 Ultrasound: Time Delay

In ultrasound imaging, a pulse which is a short (about one μsec) segment of a decaying (about 5 MHz) sinusoid is sent from a transducer into a body part. The pulse propagates into the body and reflects off of a bone, organ, or inclusion. A few μsec later, the reflected pulse returns to the transducer. The time delay between the transmitted and received pulses is the two-way traveltime to the inclusion. Knowing the speed of ultrasound propagation in the body (about 1.54 mm/μsec), the depth of the inclusion can be computed from this time delay. A delay of 13 μsec indicates a depth of $\frac{1}{2}(1.54\frac{\text{mm}}{\mu\text{sec}})(13\mu\text{sec})$=10 mm=1 cm. However, because of scattering of the pulse off of soft body tissues, the transducer measures not only the delayed pulse, but noise.

An ultrasound pulse was modelled as a 1μsec segment of a 5 MHz sinusoid that decayed from 2 V to 1 V in the 1μsec. The pulse was delayed by an unknown time, and zero-mean white Gaussian noise with standard deviation two (see Section 6-9.1) was added to the delayed pulse. The sampling rate is 10^8 samples/second.

The noisy received signal, and its cross-correlation with the known pulse, are shown in Fig. 9-26. The amount by which the pulse is delayed is not at all evident from the noisy received signal, but is clearly seen to be 30μsec on the cross-correlation plot. This corresponds to an inclusion depth of 2.31 cm. Compare this example to Example 9-26.

We note in passing that NTS Labview is very useful for acquiring biological signals and converting them to numerical form.

Exercises: None.

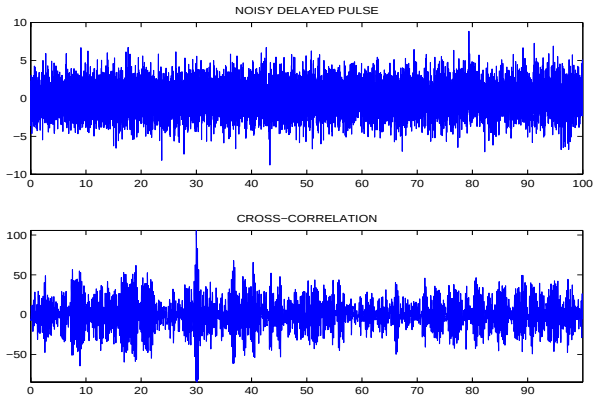


Figure 9.25: Time Delay Estimation for Ultrasound Pulse: (a) Noisy Delayed Pulse; (b) Cross-Correlation with Non-Delayed Pulse.

Chapter 9 Summary

Concepts

- Data windows are used to compute the spectrum of a signal from a short segment of the signal. Windows suppress side lobes but broaden main lobes.
- Spectrograms depict a time-varying signal spectrum by segmenting the signal using data windows, computing the spectrum of each segment, and displaying the result as an image.
- Discrete-time filters have impulse responses that have either finite durations (FIR) or infinite durations (IIR).
- FIR filters can be designed by: (1) using a data window on the inverse DTFT of the desired frequency response; (2) computing the impulse response whose frequency response matches samples of the desired frequency response; or (3) using a minimax criterion, which leads to an equiripple frequency response.
- IIR filters are designed from a given continuous-time filter, such as a Butterworth filter, by: (1) sampling the impulse response of the continuous-time filter; or (2) performing a bilinear transformation on the transfer function of the continuous-time filter.
- Downsampling a signal by L keeps only every L^{th} sample and deletes all other values. It stretches the spectrum of the signal by a factor of L, and introduces $L-1$ copies of one period of the spectrum to keep it periodic with period 2π.
- Upsampling a signal by L, also known as zero-stuffing, inserts $L-1$ zeros between samples of the signal. It compresses the spectrum of the signal by a factor of L, so that its period is $2\pi/L$.
- Interpolation of a signal lowpasses filters an upsampled signal, maintaining its given values and changing the zeros to values that make the signal be lowpass.
- Multirate signal processing uses upsampling by L, interpolation, and downsampling by M, in that order, to alter the sampling rate by a factor of L/M.
- The autocorrelation of a signal is the convolution of the signal with its time reversal. Autocorrelation is used to compute the period of a periodic signal.
- The cross-correlation of two signals is the convolution of one signal with the time reversal of the other. Cross-correlation is used to compute time delay.
- The correlation of two signals is the zeroth lag of their cross-correlation, or equivalently, their inner product. Correlation is to classify a signal as one of several possible signals.
- Autocorrelation, cross-correlation, and correlation have applications to EKG's and ultrasound imaging.

Mathematical and Physical Models

- $x_{\text{WINDOWED}}[n] = x[n]w[n]$
- Hamming window:
  $w[n] = 0.54 - 0.46\cos(2\pi n/L)$

- Spectrogram:
  $S(e^{j\Omega}, N) = |\sum_{n=N-L/2}^{N+L/2} w[n]x[n]e^{-j\Omega n}|^2$
- Bilinear transformation: $\mathbf{s} = \frac{2}{T}\frac{\mathbf{z}-1}{\mathbf{z}+1}$
- Frequency (pre)-warping: $\omega = (2/T)\tan(\Omega/2)$
- Upsampling: $x[n] \rightarrow \boxed{\uparrow L} \rightarrow y[n] \Leftrightarrow \mathbf{Y}(e^{j\Omega}) = \mathbf{X}(e^{j\Omega L})$
- Downsampling: $x[n] \rightarrow \boxed{\downarrow L} \rightarrow y[n] \Leftrightarrow y[n] = x[Ln]$
- Autocorrelation $r_x[n] = x[n] * x[-n]$
- Cross-correlation $r_{xy}[n] = x[n] * y[-n]$
- Correlation $r_{xy}[0] = \sum x[n]y[n]$

Glossary of Important Terms

Provide definitions or explain the meaning of the following terms:

Autocorrelation	Bilinear Transformation
Chirp	Correlation
Cross-Correlation	Data window
Downsampling	EKG
Equiripple Design	FIR Filter
Frequency Sampling	Frequency (Pre)-warping
Hamming Window	IIR Filter
Impulse Invariance	Interpolation
Main Lobe	Minimax Criterion
Multirate	Rectangular Window
Resolution	Side Lobe
Spectral Leakage	Spectrogram
Upsampling	Zero-Stuffing