6

LOW PASS FILTER AND BAND PASS FILTER DESIGN

- Design low pass filters (LPF) and band pass filters (BPF).
- Use LPF or BPF to extract desired spectral components.
- Study the frequency characteristics and impulse responses of LPF and BPF.

6.1 [T]ANALYSIS OF THE SPECTRUM OF SAMPLE AUDIO SIGNALS

1.A Execute **'help audiovideo'** in the MATLAB command window to see the audio data files provided in MATLAB. Write down the names of all the audio data files.

1.B Select any one of the audio data files and execute **'load** selected_file_name**'** (e.g., **load gong**) in the command window to load the selected audio file into the workspace. Then execute **whos** and capture the execution result. The variables **y** and **Fs** are generated in the workspace. The variable **y** is the audio sample vector and the variable **Fs** is the sampling frequency of the sampled vector **y**.

1.B-1 Type in **Fs** in the command window to see the sampling frequency. Write down the sampling frequency and calculate the sample interval.

1.B-2 Determine the length (in seconds) of the selected audio sample, that is, the size of the audio sample vector **y** and its sampling interval.

1.C [T]Execute **soundsc(y)** in the command window to play the audio signal. Load the six audio sample files provided in MATLAB one by one and play all

Problem-Based Learning in Communication Systems Using MATLAB and Simulink, First Edition.
Kwonhue Choi and Huaping Liu.

Companion website: www.wiley.com/go/choi_problembasedlearning

of them. Describe the sound you heard for each of these signals (what it is, e.g., train whistle).

1.D The MATLAB command **pwelch()** calculates and plots the power spectral density (PSD) of the input (sampled vector). Here, PSD is equivalent to the magnitude square of the Fourier transform (spectrum). The MATLAB function **pwelch(ht,[],[],[],Fs)**, where **ht** is the sampled version of a time function $h(t)$ and **Fs** is the sampling frequency, generates the PSD plot of $h(t)$, that is, $|H(\omega)|^2$ in dB-scale.

1.D-1 Execute the following in the command window to load and to plot, using **subplot**, the PSD of each of the six audio data samples provided in MATLAB in one figure. Stretch the figure vertically to clearly show all PSDs before capturing the window.

```
>> figure
>> subplot(6,1,1);load chirp;pwelch(y,[],[],[],Fs)
>> subplot(6,1,2);load gong;pwelch(y,[],[],[],Fs)
>> subplot(6,1,3);load handel;pwelch(y,[],[],[],Fs)
…
…
…
```

1.D-2 Based on the sound, which one of six audio data samples has the highest frequency (pitch)? Explain whether or not the PSD plots captured in 1.D-1 are consistent with what you heard. Do not focus on the absolute level of each PSD; instead, evaluate where the majority of the frequency components of each signal are located at in the frequency domain.

1.E The PSD plots in 1.D show the signal spectra in the positive frequency range only. This is because the magnitude spectrum of a real-valued function $h(t)$ is an even function, that is, $|H(\omega)| = |H(-\omega)|$, where $H(\omega)$ is the Fourier transform of $h(t)$. Prove that $|H(\omega)| = |H(-\omega)|$ if $h(t)$ is a real-valued signal.

1.F [WWW]The m-file below creates an audio sampled vector **y_plus_tone** and plots its PSD. The vector **y_plus_tone** is the sum of the audio data vector **y** in the audio sample file **handel** and the sampled vector of a 3.? kHz sine wave named tone, where '**?**' should be set equal to the last digit of your student ID number.

```
clear
load handel
t_step=1/Fs;
t=0:t_step:(length(y)-1)*t_step;
tone=sin(2*pi*3.?e3*t); %?=the last digit of the student ID number, e.g.,
     sin(2*pi*3.5e3*t) if the last digit is 5.
```

```
y_plus_tone=y'+tone;
figure
pwelch(y_plus_tone,[],[],[],Fs)
```

1.F-1 This m-file is incomplete and quantities to be completed are marked by **'?'**. Complete this m-file. Then execute it and capture the PSD plot.

1.F-2 Determine whether the PSD plot in F-1 is what you expect and why.

1.F-3 Execute **soundsc(y_pluse_tone)** in the command window to play the sound and describe how **y_pluse_tone** sounds. Explain the reason why it sounds so.

6.2 LOW PASS FILTER DESIGN

2.A [T]Suppose that the frequency transfer function of a linear system is $H(\omega)$.

2.A-1 If the Fourier transform of the input signal is $X(\omega)$, express the output Fourier transform $Y(\omega)$ in terms of $H(\omega)$ and $X(\omega)$.

2.A-2 The impulse response $h(t)$ of this system can be obtained from its frequency transfer function $H(\omega)$. Express $h(t)$ in terms of $H(\omega)$.

2.A-3 From the answers to 2.A-1 and 2.A-2, we can show that the output $y(t)$ given input $x(t)$ can be obtained as

$$y(t) = h(t) * x(t), \text{ where } * \text{ denotes convolution.} \tag{6.1}$$

Derive this equation. To this end, start from the answer to 2.A-1 and use the fact that the multiplication in frequency domain is equivalent to the convolution in time domain.

2.B [T]Consider a linear system with the following frequency transfer function:

$$H(\omega) = \begin{cases} 1 & \text{if } |\omega| \leq 2\pi B \text{ [rad/s]}, \\ 0 & \text{if } |\omega| > 2\pi B \text{ [rad/s]}. \end{cases} \tag{6.2}$$

2.B-1 Explain why this system is called a low pass filter [1] using the answer to 2.A-1.

2.B-2 What is the bandwidth of this LPF in Hz?

2.B-3 Determine $H(\omega)$ in the equation (6.2) as B approaches infinity. Using this result and the answer to 2.A-1, explain why $Y(\omega) = X(\omega)$ if the filter has an infinite bandwidth.

2.C In this problem, we determine the impulse response $h(t)$ of a linear time invariant system from its frequency transfer function $H(\omega)$.

2.C-1 [T]Using the answer to 2.A-2, show that the impulse response of the LPF given in equation (6.2) can be derived as

$$h(t) = 2B\ \mathrm{sinc}(2Bt), \quad \text{where } \mathrm{sinc}(x) \triangleq \frac{\sin(\pi x)}{\pi x}. \tag{6.3}$$

NOTE: In some of the existing textbooks, sinc(x) is defined as sin(x)/x. In this book, we adopted the definition in equation (6.3), which is also what the MATLAB function **sinc(x)** implements. In some literature, sinc(x) is written as Sa(x).

2.C-2 [WWW]The m-file below creates and plots the sampled and truncated version of $h(t)$ expressed in equation (6.3), assuming that $B = 200$ Hz in the range of $t = [-0.02\ 0.02]$. The sampling interval is set to 1/8192, which is equal to the sampling interval of **y_plus_tone** created in 1.F-1.

```
clear
B=200;
t=-0.02:(1/8192):0.02;
ht=2*B*sinc(2*B*t);
plot(t,ht);
grid
```

Execute this m-file and capture the resulting figure.

2.C-3 Execute the m-file above for the cases of B equaling 100 Hz and 400 Hz and capture the figure for each case. From the captured plots, summarize the relationship between the bandwidth of the LPF and the length of its impulse response. The first zero crossing point in the impulse response is a good metric to quantify the impulse response length.

2.C-4 If B increases to infinity, what kind of function do you expect $h(t)$ to converge to?

2.C-5 Determine $y(t)$ in equation (6.1) by substituting the answer in 2.C-4 into $h(t)$ in equation (6.1).

2.C-6 Is the result in 2.C-5 consistent with the answer to 2.B-3?

2.C-7 Explain why filters with the impulse responses as shown in 2.C-2 and 2.C-3 are impractical to implement?

2.D Consider an LPF with a causal impulse response expressed as

$$h(t) = \begin{cases} 2B\ \mathrm{sinc}(2B(t - t_d)), & 0 \le t \le 2t_d, \\ 0 & \text{elsewhere.} \end{cases} \tag{6.4}$$

2.D-1 [WWW] The m-file below creates and plots the sampled vector of $h(t)$ expressed in equation (6.4) for $B = 200$ Hz and $t_d = 0.02$. Execute the m-file and capture the result.

```
clear
B=200;
td=0.02
t=0:(1/8192):2*td;
ht=2*B*sinc(2*B*(t-td));
plot(t,ht);
grid
```

2.D-2 Explain why the linear system with the impulse response shown in 2.D-1 is practical as opposed to the system with the impulse responses shown in 2.C-2 or 2.C-3?

2.D-3 [WWW] Since the function $h(t)$ in equation (6.4) is different from the $h(t)$ in equation (6.3), the transfer functions corresponding to equations (6.3) and (6.4) are also different. We can obtain $H(\omega)$ of $h(t)$ expressed in equation (6.4) by taking its Fourier transform. However, it is mathematically cumbersome to calculate the Fourier transform of a truncated sinc function. In this problem, we resort to the numerical integration method to obtain $H(\omega)$, more precisely, the sampled version of $H(\omega)$ in the frequency domain. Numerical integration method was discussed in Section 2.1 of Chapter 2.

The MATLAB code fragment below generates a vector **Hw_vector**, the sampled version of $H(\omega)$. Each element of **Hw_vector** will be calculated through a separate numerical integration of **ht**, the sampled version of $h(t)$. The magnitude spectrum $|H(\omega)|$ and phase spectrum $H(\omega)$ are also plotted.

Some useful information for completing this m-file:

- Outside of the time period $0 \le t \le 0.04$, $h(t) = 0$; thus the integration boundary is $[0, 0.04]$ and the Fourier transform of $h(t)$ can be written as $\int_{-\infty}^{\infty} h(t)e^{-j\omega t}\mathrm{d}t = \int_{0}^{0.04} h(t)e^{-j\omega t}\mathrm{d}t$.
- For each value of $\omega = -20000 : 10 : 20000$ rad/s, $H(\omega)$ is calculated via a separate numerical integration.

Recall that the vector **ht** is the sampled version of $h(t)$. The line **'Hw=sum(?.*exp(-j*?*t))*t_step;'** numerically implements $\int_{0}^{0.04} h(t)e^{-j\omega t}dt$ for each specific value of ω from -20000 rad/s to 20000 rad/s in a '**for**' loop. Complete the two places marked by '**?**' and execute the completed m-file. Capture the execution result.

```
clear
B=2000; %bandwidth, currently set to 2 KHz.
td=0.02; %delay
```

```
t_step=1/8192;
t=0:t_step:0.04;

ht=2*B*sinc(2*B*(t-td)); % Equation (6.4)

Hw_vector=[];
w_vector=[];
for w=-20000:10:20000
   w_vector=[w_vector w];
   Hw=sum(?.*exp(-j*w*?))*t_step;
   Hw_vector=[Hw_vector Hw];
end

figure
subplot(3,1,1)
plot(t,ht);xlabel('t [sec]');ylabel('h(t)');grid
subplot(3,1,2)
plot(w_vector,abs(Hw_vector))
xlabel('w [rad/sec]');ylabel('|H(w)|');grid
subplot(3,1,3)
plot(w_vector,angle(Hw_vector));
xlabel('w [rad/sec]');ylabel('\angle H(w)');grid;axis([-50 50 -1 1])
```

2.D-4 From the magnitude spectrum, measure as accurate as possible the 3dB bandwidth of the LPF in Hz.

2.D-5 Measure the slope of the phase spectrum.

2.D-6 Repeat the above experiment for **td**=0.01 and **td**=0.03. Capture the execution result for each case.

2.D-7 The three plots, one captured in 2.D-3 and two captured in 2.D-6, show $h(t\text{-}t_d)$ (top subplot) and its magnitude spectrum (middle subplot) and phase spectrum (bottom subplot) for three different delay values ($t_d = 0.02$, 0.01, and 0.03), respectively. Measure the delays of each of the three top subplots and the slopes of each of the three phase spectra and record them in Table 6.1.

2.D-8 Based on the results in Table 6.1, determine the effect of the delay, that is, the center of symmetry of $h(t - t_d)$ (which equals t_d in equation (6.4)), on its phase

TABLE 6.1 Time Delay and the Slope of the Phase Spectrum.

Delay	Measured delay (center of $h(t\text{-}t_d)$)	Measured slope of the phase spectrum
$t_d = 0.01$		
$t_d = 0.02$		Replicate the answer to 2.D-5 here
$t_d = 0.03$		

spectrum $H(\omega)$. Determine the functional relationship between t_d and the slope of $H(\omega)$.

2.D-9 Based on the plots captured in 2.D-3 and 2.D-6, summarize the effect of delay of the signal $h(t)$ on its magnitude spectrum $|H(\omega)|$.

2.D-10 [A]Execute the m-file for two more cases: **B** = 250 and **B** = 1000. Measure the bandwidth from the magnitude spectrum and check whether it is equal to **B**.

2.D-11 [A]In a time invariant linear system like the LPFs we have designed, the input signal undergoes the same amount of delay as the delay of $h(t)$. This is not desirable for real-time systems. In order to reduce the output delay of the LPF, reduce **td** to 0.001 in the m-file and execute it again. Capture the resulting plot. Based on the plot, comment on the penalty one has to pay in terms of the magnitude spectrum in order to reduce the output delay.

2.D-12 [A]Explain why reducing the delay causes the problem observed in 2.D-11.

6.3 LPF OPERATION

Suppose that $c(t)$ denotes the convolution of $a(t)$ and $b(t)$, that is, $c(t) = a(t) * b(t)$. Let the sampled vectors of $a(t)$ and $b(t)$ be **at** and **bt**, respectively. Then the sampled vector of $c(t)$, **ct**, can be simply created by using the command **'ct=conv(ht ,gt);'**. We will use this approach to perform low pass filtering in the following problems.

3.A [WWW]In the following m-file, we input the audio sample vector saved in **handel.mat** to an LPF with a bandwidth of 2 kHz. The LPF outputs the vector **yt**.

```
clear
B=2000; %bandwidth, set to 2 kHz currently
td=0.02; % delay (= center of symmetry of the delayed sinc pulse ht)

t_step=1/8192;
t=0:t_step:0.04;
ht=2*B*sinc(2*B*(t-td)); % ht is the sampled vector of h(t), i.e., impulse response for
     LPF.

load handel
xt=y';
%xt is a variable declared for the sampled vector of x(t), input signal to LPF.
%In the command conv(a,b), the argument vectors a and b need to be row vectors. So,
   we need to change the column vector y (=audio sample vector in handel) into the row
   vector xt by using transpose operator ( ').

yt=conv(?,xt); % Use a theory about the relationship among input, output, and
     impulse response of linear system.
```

3.A-1 Determine the uncompleted variable in the m-file (marked by '**?**').

3.A-2 Execute the completed m-file. Then execute the following in the command window to listen to the input as well as the output sounds of the LPF. Compare the input and output sounds and summarize their differences.

```
>>soundsc(xt)
Execute the following when playing is completed.
>>soundsc(yt)
```

3.A-3 Change the bandwidth B of the LPF to 500 Hz in the m-file above and execute the completed m-file. Then listen to the sound of **yt** again. Repeat this experiment for **B** = 4000 Hz. Describe how the sound changes as the bandwidth of the LPF changes.

3.A-4 Execute the following lines in the command window and capture the execution result.

```
>> t_axis=1/8192*(1:length(yt));
>> plot(t_axis, yt)
>> axis([0 0.1 -8000 8000])
>> grid
```

3.A-5 From the graph in 3.A-4, determine the starting time, the point where the LPF output signal **yt** starts to rise to a visually noticeable level. Explain why the starting time is roughly equal to the delay of the impulse response of the LPF.

3.B The goal of this problem is to recover the audio signal interfered by a large sinusoidal signal by using a properly designed LPF.

3.B-1 [WWW]The following m-file creates a sampled audio signal **xt** by adding a sine wave to the sampled sound vector **y** in **handel.mat**. It also plots the PSD of **xt**. Execute the m-file and capture the resulting PSD.

```
clear
load handel
rand(1, XXXX); % XXXX= the last four digits of your student ID number. Be sure to
      include this line.
f=3250+500*rand;
tone=sin(2*pi*f*(1:length(y))*1/Fs);
xt=y'+ tone;
pwelch(xt,[],[],[],Fs)
clear f;
```

3.B-2 Based on the PSD plot of **xt**, estimate the frequency of the added sine wave as accurately as possible. Properly enlarge the PSD plot for accurate reading.

3.B-3 Execute the following line in the command window to play **xt**. Describe how **xt** sounds.

```
>>soundsc(xt)
```

3.B-4 [WWW]By passing **xt** through an LPF, we can remove the beeping sound (caused by interference) without significantly distorting the original audio signal **y** in **handel.mat**. In the following m-file, we will generate the output of the LPF **yt** so that it sounds almost the same as the original audio signal **y**.

Do not include the command **'clear'** in the m-file because the variable **xt** created in 3.B-1 will be needed. Read the comments for each line first and then complete the three places marked by **'?'**. Capture the completed m-file.

```
B=?; % Determine the LPF bandwidth (constant) to filter out(eliminate) the sine wave
     included in xt.
td=0.02; %delay time
t_step=1/8192;
t=0:t_step:0.04;
ht=2*B*sinc(2*B*(t-td)); % The sampled vector of h(t), i.e., the impulse response of
     LPF.
yt=?(?,xt); % Pass xt through LPF to create the output yt. The first ? is the function name.
     The second ? is the variable name.
```

3.B-5 Justify your choice of the value of **B** set in the m-file above.

3.B-6 Execute the m-file in 3.B-4 and then execute the following line in the command window to play **yt**. Describe how **yt** sounds. Has the beeping sound been removed without distorting the sound of the original signal? If there is still a beeping sound or the original sound is noticeably distorted, go back to 3.B-4 and properly change B until you get the desired sound.

```
>>soundsc(yt)
```

6.4 [A]BAND PASS FILTER DESIGN

4.A Execute the following lines in the command window to generate the impulse response of the LPF with a bandwidth of 300 Hz and plot its PSD. Capture the PSD plot.

```
>>t=0: (1/8192):0.04;
>>B=300;td=0.02;
>>ht=2*B*sinc(2*B*(t-td));
>>pwelch(ht,[],[],[], 8192);
```

4.B [T]Denote the Fourier transform of $x(t)$ by $X(\omega)$. Then prove that the Fourier transform of $x(t)\cos(\omega_0 t)$ is $\frac{1}{2}[X(\omega+\omega_0)+X(\omega-\omega_0)]$.

4.C The following problems provide an intuitive approach on how to design a BPF [1].

4.C-1 After executing the lines of code in 4.A, continue to execute the following lines. The first line creates a 3 kHz cosine waveform vector **cos3000** of the same length as that of **ht** created in 4.A. The second line multiplies **cos3000** by **ht** to create **ht_times_cos**, which is the sampled vector of a frequency up-converted signal $h(t)\cos(2\pi \times 3000t)$. The third line plots the PSD of **ht_times_cos**. Capture the resulting PSD plot.

```
>>cos3000=cos(2*pi*3000*t);
>>ht_times_cos=ht.*cos3000;
>>pwelch(ht_times_cos,[],[],[], 8192)
```

4.C-2 Using the frequency-shift formula derived in 4.B, validate the PSD result generated in 4.C-1. Note that **pwelch(a,[],[],[],b)** shows only the positive frequency components.

4.C-3 Consider an arbitrary signal $x(t)$ whose spectrum spreads over 0 to 4.5 kHz and denote its sampled vector by **xt**. Describe the difference of the PSD shape of **conv(xt,ht_times_cos)** in comparison with the PSD of **xt**. In your description, use the PSD shape of **ht_times_cos** and the frequency-domain view of convolution in the time domain (see Section 5.4 of Chapter 5).

4.D Fig. 6.1 shows the frequency transfer function of an ideal band pass filter that passes only the spectral components of the input signal in the frequency range of $[B_L\ B_H]$.

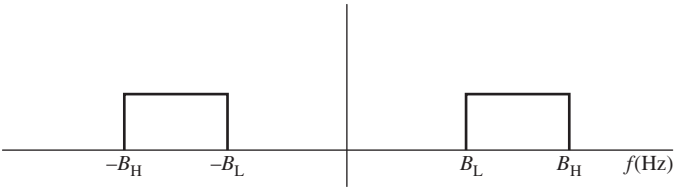

FIGURE 6.1 Frequency response of an ideal band pass filter.

4.D-1 Now, we denote the sampled impulse response of the BPF shown in Fig. 6.1 by **ht**. Based on the discussions in **4.A**–**4.C**, describe the steps to create **ht** in MATLAB.

4.D-2 [WWW]The following m-file performs the BPF operation to extract only the beeping sound from **xt** created in 3.B-1. Determine the appropriate values for the two places marked by **'?'** and justify your answer.

```
B=?; %B is used to generate ht below.
td=0.02;
t_step=1/8192;
t=0:t_step:0.04;
ht=2*B*sinc(2*B*(t-td)).*cos(2*pi*?*t); % Sample vector impulse response h(t) for BPF
yt = conv(xt,ht);
```

4.D-3 Execute the m-file above and then execute **soundsc(yt)** in the command window to play the BPF output **yt**. Describe how **yt** sounds. In case **xt** created in 3.B-1 has been cleared, execute the m-file in 3.B-1 first before executing the m-file above.

REFERENCE

[1] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 3rd ed., Philadelphia: Saunders, 1991.

7

SAMPLING AND RECONSTRUCTION

- Study how the sampling changes the signal spectrum.
- Reconstruct a signal from its sampled version using low pass filtering.
- Implement frequency up-conversion using sampling and a band pass filter.

7.1 CUSTOMIZING THE ANALOG FILTER DESIGN BLOCK TO DESIGN AN LPF

1.A Start Simulink and open a new mdl/slx file and add the blocks as shown in Fig. 7.1. These blocks can be easily identified by searching for them in the Simulink library using block names.

Set the internal variables of the **Sine Wave** block and **Analog Filter Design** block as follows.

1. **Sine Wave**
   - **Sample time** = 1/3e4
2. **Analog Filter Design**
   - **Design method** = **Chevyshev II**
   - **Filter type** = **Low pass**
   - **Filter order** = **32**
   - **Stop band edge frequency** = **2*pi*1000**
   - **Stop attenuation in dB** = 40

Problem-Based Learning in Communication Systems Using MATLAB and Simulink, First Edition.
Kwonhue Choi and Huaping Liu.

Companion website: www.wiley.com/go/choi_problembasedlearning

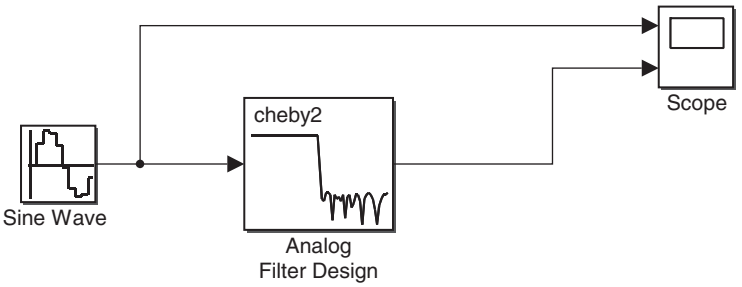

FIGURE 7.1 Test system for **Analog Filter Design** block.

1.A-1 The **Analog Filter Design** block is customized as an LPF with the above parameter setting. Determine the bandwidth of this LPF in Hz.

1.A-2 Set **Simulation Stop Time** to 0.05. Then set the parameter **Frequency (rad/s)** of the **Sine Wave** block as specified in Table 7.1. Run the simulation for each of these cases and measure the corresponding amplitudes of the LPF output and record them in the table.

1.A-3 Based on the simulation result, determine whether the LPF is designed correctly and why?

7.2 STORING AND PLAYING SOUND DATA

2.A Open **Sound_Source.mdl** (or **Sound_Source.slx**) designed in 6.A of Chapter 1. If you do not have this file, go through 6.A of Chapter 1.

Add the **To file** block from the Simulink library to the design and connect it to the **Sound Source** subsystem as shown in Fig. 7.2. Save the design as a new file and do not overwrite **Sound_Source.mdl/slx,** since it will be needed for other projects later.

Capture your design window.

TABLE 7.1 Test Inputs to LPF and the Output Amplitude.

Frequency (rad/s) of the **Sine Wave** block	Amplitude of the LPF output
2*pi*(400 + XX) XX = Last two digits of your student ID number	
2*pi*900	
2*pi*950	
2*pi*1000	
2*pi*1050	
2*pi*1500	
2*pi*3000	

FIGURE 7.2 Test system for the subsystem **Sound Source**.

2.B Set the block parameters of the design in 2.A through the following steps.

2.B-1 [WWW]Download **sound_CH7.mat** from the companion website to your MATLAB work folder. Your MATLAB work folder path is specified in the menu bar of the MATLAB main window.

Execute the following in the command window to verify whether downloading is successful.

```
>> ls *.mat
```

2.B-2 Set the parameter **File name** of the **From File** block in the subsystem **Sound Source** to **sound_CH7.mat**. Capture your parameter setting window.

2.B-3 Set the parameters of the **To File** block as follows. Capture your parameter setting window.

- **Save format** = **Array** (not required for old versions)
- **File name** = **signal.mat**
- **Sample time** = 1/8192

2.C Set **Simulation stop time** to 20 seconds and run the simulation. After the simulation is finished, execute the following in the command window. Describe the sound you heard.

```
>> clear; load signal.mat; soundsc(ans(2,:))
```

7.3 SAMPLING AND SIGNAL RECONSTRUCTION SYSTEMS

Fig. 7.3 is a simple block diagram of a sampling [1, 2] and reconstruction system to restore the original signal from its sampled version by using an LPF.

3.A In Fig. 7.3, $x(t)$ (=output of the **Sound Source** block) is an audio signal whose bandwidth B equals 4 kHz. For simplicity, we assume that the Fourier transform of $x(t)$, $X(\omega)$, has a triangular shape with a one-sided bandwidth of 4 kHz, as shown in Fig. 7.4.

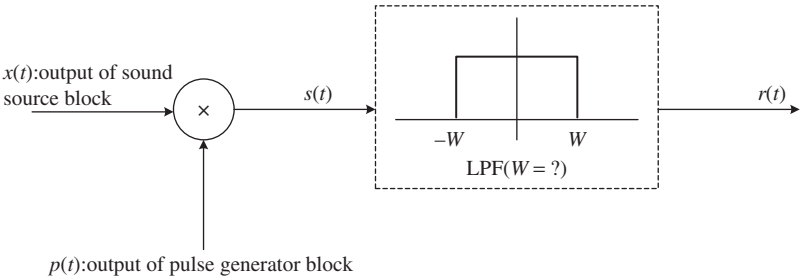

FIGURE 7.3 Sampling and signal reconstruction system.

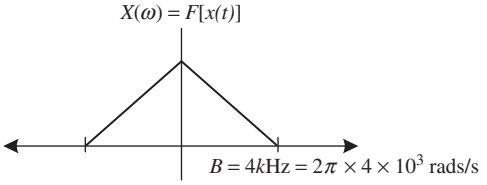

FIGURE 7.4 Spectrum of the sound signal $x(t)$.

3.A-1 [T]What is the minimum sampling frequency F_s of the periodic sampling pulse signal $p(t)$ in Fig. 7.3 so that $x(t)$ can be reconstructed from its sampled version without distortion? Justify your answer.

3.A-2 [T]Suppose that the sampling pulse signal $p(t)$ is as shown in Fig. 7.5, with a sampling frequency $F_s = 2B = 8$ kHz, pulse amplitude 1, and pulse width equaling 1/10 of the period.

The Fourier transform of the sampled signal $s(t)$ $(=x(t)p(t))$ can be derived considering the following facts:

1. Since $p(t)$ is a periodic signal, it exhibits a line spectrum, and the magnitude of nth spectral line is $2\pi P_n$, where P_n denotes the Fourier series coefficient of $p(t)$ (refer to Section 4.2 of Chapter 4).

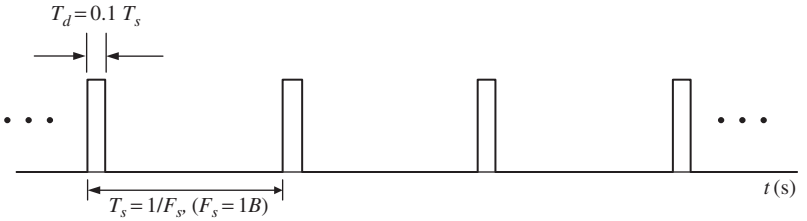

FIGURE 7.5 Sampling signal $p(t)$.

2. The Fourier transform of the product of two functions in the time domain corresponds to the convolution of the Fourier transforms of the two respective signals in the frequency domain (refer to Section 5.4 of Chapter 5).

Now for the sampling signal $p(t)$ in Fig. 7.5, determine the two quantities marked by '**?**' in the following equation:

$$S(\omega) = F[s(t)] = \sum_{n=-\infty}^{\infty} ? \times X(\omega - n \times ?), \tag{7.1}$$

where $F[s(t)]$ denotes the Fourier transform of $s(t)$. From this equation, sketch $S(\omega)$.

3.B In this problem, we design the system shown in Fig. 7.3 in Simulink. We first sample the signal $s(t)$. This is done by sampling the output of the **Sound Source** block. Then we reconstruct $x(t)$ from its sampled version using an LPF. We use the **Pulse generator** block, **Product** block, and **Analog Filter Design** block to realize these signal processing steps.

First, modify the mdl/slx file in Section 7.2 as shown in Fig. 7.6:

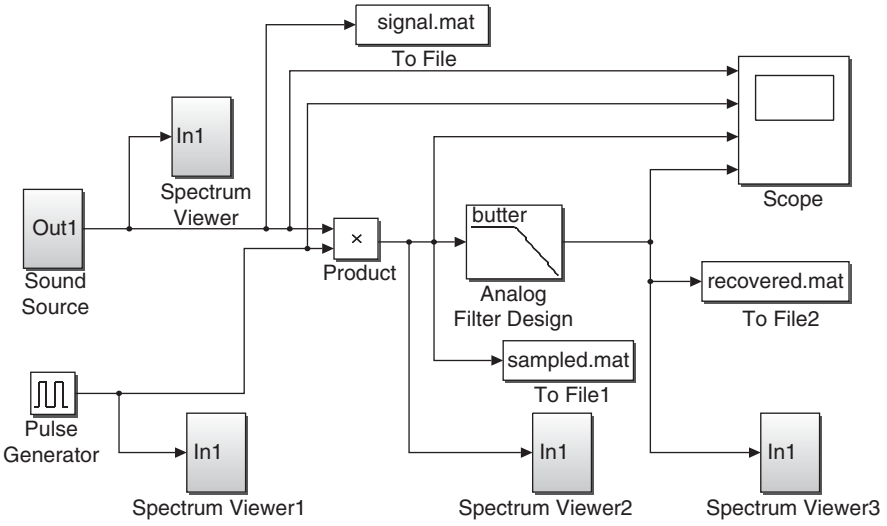

FIGURE 7.6 Simulink design of a sampling and signal reconstruction system.

Then, set the parameters of each block as follows. Do not change the parameters not mentioned here.

1. **Product**
   - **Sample time** = 1/(16e4)
2. **Pulse Generator**
   - **Period** = 1/(8e3)
   - **Pulse Width (% of Period)** = 10

3. **To File1**
   - **Save format** = **Array** (not required in old versions)
   - **File name** = **sampled.mat**
   - **Sample time** = 1/8192
4. **To File 2**
   - **Save format** = **Array** (not required in old versions)
   - **File name** = **recovered.mat**
   - **Sample time** = 1/8192
5. **Spectrum Viewer**, **Spectrum Viewer1**, **Spectrum Viewer2**, … : Use the subsystem block created in Section 1.6.B of Chapter 1. In case you do not have it, go through Section 1.6.B of Chapter 1 to create it. Make sure that the numbers in the names of these blocks occur in numerical order as shown in Fig. 7.6.

3.B-1 Capture the design window of your completed mdl/slx file.

3.B-2 Set **Simulation stop time** to 5e-4 seconds and run the simulation. Upon completing the simulation, properly enlarge the sampling signal $p(t)$ (=output of **Pulse generator**) in the **Scope** display window to measure its frequency and pulse width. Record the measured values together with the captured waveform. Is the captured waveform the same as shown in Fig. 7.5? NOTE: Prior to capturing the waveform, right-click the figure to set **Y-min** to -0.5 and **Y-max** to 1.5.

3.B-3 Set **Simulation stop time** to 1e-2 seconds and run the simulation. Upon completing the simulation, right-click to **Autoscale** $x(t)$ and $s(t)$ in the **Scope** display window. Then zoom into the range of [0.006, 0.01] along the x axis. Capture the **Scope** display window.

3.B-4 From the captured window in B-3, determine whether the sampled signal $s(t)$ is correctly created. That is, is it equal to $x(t)\ p(t)$?

3.B-5 Set the **Simulation stop time** to 4 seconds and run the simulation. Executing the following line of code in the command window will play which of the four signals $\{x(t), p(t), s(t), r(t)\}$ defined in Fig. 7.3?

```
>>load sampled.mat; soundsc(ans(2,:))
```

3.B-6 Execute the command above and describe the sound you hear. Does the signal sound right and why?

3.B-7. Capture the internal **Spectrum Analyzer** (**Spectrum Scope** in some old MATLAB versions) display windows of **Spectrum Viewer**, **Spectrum Viewer1**, and **Spectrum Viewer2** (except **Spectrum Viewer3**). Before capturing the **Spectrum Analyzer** display windows, decrease the height of the window to get a width:height ratio of about 7:1 for the graph portion as shown in Fig 4.4 in Chapter 4. Do not **Autoscale**. Follow this guideline throughout all the problems in this book that require the **Spectrum Analyzer** display window.

The line spectrum should appear in the **Spectrum Viewer1/Spectrum Analyzer** display window.

(a) Explain the reasons that cause the line spectrum.
(b) Determine the frequency interval of the line spectrum.
(c) Based on the signals chosen for the system, determine the minimum frequency interval between the spectrum lines and compare it with the observed value in (b).

3.B-8 Is the spectrum in the **Spectrum Viewer2/Spectrum Analyzer** display window consistent with your sketch in 3.A-2? The spectrum will change as time goes by. In making this assessment, focus on (1) the overall spectral shape such as spectral envelope and (2) the frequency interval between the spectrum lines.

3.B-9 Determine how the frequency interval between the spectrum lines changes if the parameter **Period** of the **Pulse Generator** block is changed to 1/(32e3).

3.B-10 Change the parameter **Period** of the **Pulse Generator** block to 1/(32e3) and run the simulation again. Observe the spectrum of $s(t)$ in the **Spectrum Viewer2/Spectrum Analyzer** display window. Upon completing the simulation, capture the **Spectrum Viewer2/Spectrum Analyzer** display window. Is the captured spectrum consistent with the answer to 3.B-9?

3.B-11 Restore the parameter **Period** of the **Pulse Generator** block back to 1/(8e3). If we reduce **Pulse Width (% of Period)** to 1, will the spectrum in the **Spectrum Viewer2/Spectrum Analyzer** display window be different from the one captured in 3.B-7? Assess the difference in terms of the spectral envelope that connects the peaks of the replicas (called "harmonics"). Which one has a wider envelope? Provide a mathematical justification for your assessment.

3.B-12 Change the **Pulse Width (% of Period)** of the **Pulse Generator** block to 1 and run the simulation. Capture the **Spectrum Viewer2/Spectrum Analyzer** display window. Is the result consistent with your answer to 3.B-11?

3.C The aim of this problem is to reconstruct the original signal $x(t)$ from the sampled signal $s(t)$. Set **Period** = 1/(8e3), **Pulse Width (% of Period)**=10 of **Pulse Generator**.

3.C-1 If the goal is to transmit or to store the information contained in the original signal $x(t)$, it is more convenient to use $s(t)$, instead of $x(t)$, as long as there is no information loss by using $s(t)$. Summarize the advantages of using $s(t)$ in terms signal processing and storage requirements.

3.C-2 In 3.B-6, we have checked that $s(t)$ sounds completely differently from $x(t)$. Hence we first reconstruct the original signal $x(t)$ from $s(t)$.

In the **Scope** display window captured in B-3, $s(t)$ is equal to $x(t)$ only for 10% of time and is 0 during the remaining 90% of time. In other words, $s(t)$ carries only 10% of the waveform of $x(t)$. If we further reduce the pulse width of the sampling pulse $p(t)$ for the advantages discussed in 3.C-1, the portion of time that $s(t) = x(t)$ will be

reduced accordingly. Recovering the original signal from the sampled signal requires recovering the lost part of the waveform. Intuitively, from the captured waveform of $s(t)$ in 3.B-3, would it be possible to recover the lost part, say, 90% of the waveform of $x(t)$?

3.C-3 Based on $S(\omega)$ obtained in A-2 (or the display window of **Spectrum Viewer2/Spectrum Analyzer** captured in 3.B-7), intuitively $X(\omega)$ could certainly be recovered from $S(\omega)$. Therefore $x(t)$ could also be recovered from $s(t)$. Design a scheme that employs an LPF with a bandwidth $B = 4$ kHz to recover $x(t)$.

3.C-4 In the mdl/slx design simulated in Section 3.B, determine a proper value for the parameter **Passband edge frequency** of the **Analog Filter Design** block so that it generates the reconstructed signal. Note that the unit is rad/s.

3.C-5 Set **Passband edge frequency** of the **Analog Filter Design** block as obtained in 3.C-4. Set **Simulation stop time** to 5e-2 seconds and run the simulation. After the simulation is completed, **Autoscale** (right click) all the waveforms in the **Scope** display window and then capture the display window.

3.C-6 Check whether or not the shape of the restored signal $r(t)$, that is, the output of the **Analog Filter Design** block, is same as $x(t)$. Focus on the shapes not the signal magnitude.

3.C-7 If the design is correct, then $r(t)$ should be nearly identical to $x(t)$, except a scaling factor of 1/10. Analytically explain this. You may refer to the answer to 3.A-2.

3.C-8 Set **Simulation stop time** to 4 seconds and run the simulation. You may set it larger if the computing power of your PC can handle it. If we run the following line of code in the command window after the simulation, which one of the four signals $\{x(t), p(t), s(t), r(t)\}$ defined in Fig. 7.3 will be played?

```
>>load recovered.mat;soundsc(ans(2,:))
```

3.C-9 Execute the command above. Compare the played sound with the original sound (**signal.mat**) you heard in 2.C.

3.D In this problem, we simulate a system that approximates the ideal impulse sampling (sampling pulse width equals 0).

3.D-1 To approximate the impulse sampling, set **Pulse Width (% of Period)** of **Pulse Generator** to 1 in the mdl/slx file. With this setting, what percentage of the waveform of $x(t)$ is directly carried through to $s(t)$?

3.D-2 Run the simulation for 4 seconds. Upon completing the simulation, execute the following in the command window to play $r(t)$ (= **Analog Filter Design** output signal). Does it sound the same as the original signal $x(t)$?

```
>>load recovered.mat;soundsc(ans(2,:))
```

3.E The goal of this problem is to investigate the relationship between the sampling frequency and the feasibility to reconstruct the original signal from its sampled version.

3.E-1 Another method to reduce the nonzero portion of the sampled signal $s(t)$ is to increase the Period of **Pulse Generator**. Suppose that the pulse width is fixed. Should the parameter **Period** be increased or decreased in order to reduce the nonzero portion of the sampled signal $s(t)$?

3.E-2 [T] At this point, the sampling signal $p(t)$; that is, the output of **Pulse Generator** is shown in Fig. 7.5, where the sampling frequency F_s is set to twice of the bandwidth of the original signal $x(t)$. If the sampling period is doubled, that is, 1/4e3, which is equivalent to reducing the sampling frequency F_s to B, then the answers to the two quantities marked by **'?'** in equation (7.1) should be modified accordingly. Determine the second quantity (for this problem, the first quantity is not of our concern). Modify the sketch of $S(\omega)$ completed in 3.A-2 according to the new quantity.

3.E-3 If the parameter **Period** of the **Pulse Generator** block is doubled to 1/4e3, **Pulse Width (% of Period)** should be decreased from 10% to 5% in order to make the actual pulse width remain unchanged.

Set **Period**= 1/(4e3) and **Pulse Width (% of Period)**= 5 and run the simulation for 4 seconds. Observe the spectrum of $s(t)$ in the display window of **Spectrum Viewer2/Spectrum Analyzer** while the simulation is in progress. After the simulation is completed, run the simulation again and capture the display window of **Spectrum Viewer2/Spectrum Analyzer**.

3.E-4 Is the spectrum captured in 3.E-3 consistent with your sketch in 3.E-2? The magnitude of the instantaneous spectrum might change. Thus, in comparing the results in 3.E-3 and 3.E-2, focus only on their overall shapes.

3.E-5 What is the main difference between the spectra captured in 3.E-3 and in the display window of **Spectrum Viewer2/Spectrum Analyzer** of 3.B-7. From this observation, explain why it is impossible to reconstruct $x(t)$ from $s(t)$ if $F_s = B =$ 4 kHz.

NOTE: A phenomenon whereby the spectrum replicas in the sampled signal overlap with one another due to an insufficient sampling frequency is called "aliasing."

3.E-6 Run the simulation for 4 seconds. After the simulation is complete, execute the following to play the recovered signal $r(t)$, the output of the **Analog Filter Design** block. Judged from the sound of the recovered signal, is the original signal recovered from its sampled version?

```
>>load recovered.mat;soundsc(ans(2,:))
```

3.E-7 Repeat the simulation for each of the following sampling periods: 1/(2e3), 1/(5e3), 1/(6.4e3), and 1/(10e3). Describe how the spectrum of $s(t)$, $S(\omega)$, displayed by **Spectrum Viewer2/Spectrum Analyzer** differs with the difference sampling frequencies. After each simulation, play the recovered signal $r(t)$ and describe what it sounds like.

3.E-8 From the simulation results so far, can you develop a condition on the minimum sampling frequency F_s with which $x(t)$ can be recovered from $s(t)$ without distortion?

3.E-9 Quantitatively justify your conclusion made in 3.E-8.

7.4 FREQUENCY UP-CONVERSION WITHOUT RESORTING TO MIXING WITH A SINUSOID

Consider the scenario that in Fig. 7.3, the LPF is replaced by a BPF with a bandwidth 2B centered at the sampling frequency F_S. For the following problems, we assume that the sampling signal $p(t)$ is the one shown in Fig. 7.5 and denote the BPF output by $z(t)$.

4.A Based on equation (7.1) and sketch of $S(\omega)$ completed in 3.A-2 or the display window of **Spectrum Viewer2/Spectrum Analyzer** captured in 3.B-7, it is straightforward to express $Z(\omega)$, the output spectrum of the BPF, by using two Fourier series coefficients of $p(t)$ and the $X(\omega)$. Write the expression of $Z(\omega)$ assuming that the BPF is ideal and the delay is negligible.

4.B [T]The time-domain output of the BPF $z(t)$ can be written as

$$z(t) = x(t) \times A\cos(\omega t + \theta). \tag{7.2}$$

Derive A, ω, and θ.

4.C We verify the answer to 4.B through simulation. Revisit the mdl/slx file designed in Section 7.3 and make sure that **Period** of the **Pulse Generator** block equals 1/(8e3), **Pulse Width (% of Period)** of the **Pulse Generator** block equals 10, so that the sampling signal $p(t)$ is the same as shown in Fig. 7.5.

4.C-1 To implement a BPF by **Analog Filter Design**, set the parameter **Filter type** = **BPF**. The two new parameters **Lower pass band edge frequency** and **Upper pass band edge frequency**, which are defined as shown in Fig. 7.7, must be set as well. Set these two parameters to implement the BPF that has a center frequency of 8 kHz (which equals the sampling frequency) and a bandwidth W (which equals $2B$ = 8 kHz). Capture your parameter setting window. Note that the unit is rad/s.

4.C-2 Modify the mdl/slx file as shown in Fig. 7.8. Set the parameters of the **Sine Wave** block properly so that the output of **Product 1** is equal to the right-hand side of equation (7.2), with the parameters A, ω, and θ derived in 4.B. Capture your parameter

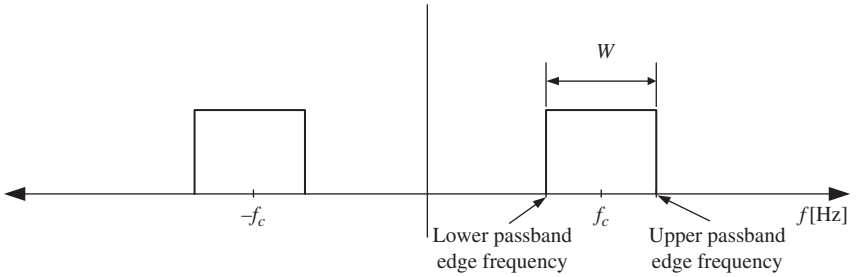

FIGURE 7.7 Definitions of the parameters of the **Analog Filter Design** block for the BPF design.

setting window of the **Sine Wave** block. Attention should be paid to the phase setting of the **Sine Wave** block so that it outputs $A\cos(\omega t + \theta)$, not $A\sin(\omega t + \theta)$.

4.C-3 Run the simulation for 0.02 seconds. Upon completing the simulation, **Autoscale** all the waveforms in the **Scope** display window and capture the window.

4.C-4 In order to check whether the answer to 4.B is correct, we can compare the waveforms in the captured window. (a) Which waveforms should we compare?

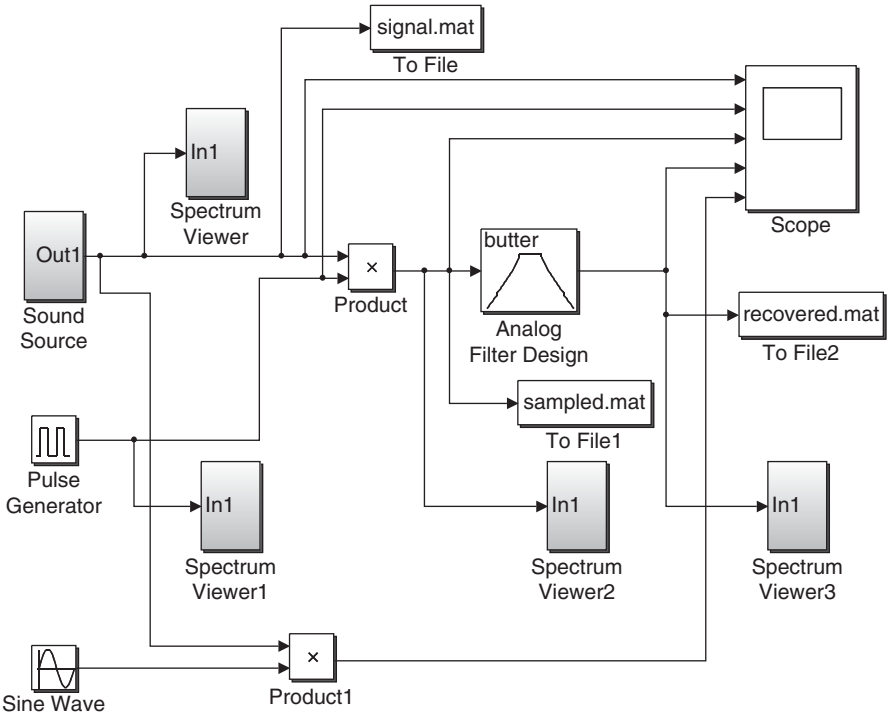

FIGURE 7.8 Design of frequency up-conversion through sampling and filtering.

(b) Is the answer to 4.B correct from the comparison? Note that the BPF introduces a certain amount of delay.

4.C-5 Run the simulation for 0.1 seconds. Capture the display window of the **Spectrum Viewer3/Spectrum Analyzer**.

4.C-6 Does the result in 4.C-5 match the answer to the problem in 4.A?

4.C-7 The signal can be up-converted to another center frequency above 8 kHz without increasing the sampling frequency F_S. Develop the method and determine the possible center frequencies.

4.D From the problems completed so far, generalize the process to up-convert any arbitrary signal to a desired center frequency ω_0 without using a mixer with a sinusoid.

REFERENCES

[1] A. V. Oppenheim, *Applications of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1978.

[2] E. J. Baghdady, *Lectures on Communication Systems Theory*, New York: McGraw-Hill, 1960.

8

CORRELATION AND SPECTRAL DENSITY

- Calculate the correlation function of two time functions using numerical integration.
- Locate a pulse in severe noise using correlation.
- Estimate the shape and parameters of unknown periodic signals in severe noise using correlation.
- Investigate the relationship between the correlation function and the spectral density.

8.1 GENERATION OF PULSE SIGNALS

1.A [WWW] The following m-file generates a vector **psint**, the sampled version of a truncated 50 Hz sine waveform with a length of 0.05 seconds. The reference time vector **t** equals −5 to 5 seconds with a step size of 0.001 seconds. The truncation vector **tmp** is generated by Boolean operation introduced in Section 5.1 of Chapter 5.

```
clear
t_step=0.001;
t=-5:t_step:5;
tmp=(0<t) & (t<0.05);
figure(1)
plot(t,tmp); title('tmp');axis([-5 5 -2 2]);
```

Problem-Based Learning in Communication Systems Using MATLAB and Simulink, First Edition.
Kwonhue Choi and Huaping Liu.

Companion website: www.wiley.com/go/choi_problembasedlearning

```
tone=sin(2*pi*50*t);
psint=tone.*tmp;
figure(2)
plot(t,psint); title('sine pulse');axis([-5 5 -2 2]);grid on;
```

1.A-1 Add a comment to each line to explain what it does.

1.A-2 Execute the m-file above. Use x axis zoom-in button in the menu bar (or use **axis()** properly) to enlarge the $[-0.1\ 0.1]$ portion of the two resulting figures. Capture the figures and comment on whether or not the waveform in the second figure is a truncated 50 Hz sine waveform with a length of 0.05 seconds.

1.B Add the following lines at the end of the m-file in 1.A to create the sampled vector **pt** of the time-limited square-wave pulse signal $p(t)$ in range of $0 \leq t \leq 0.05$ with a frequency of 50 Hz.

```
% Add the following to the m-file in 1.A.
pt=sign(psint);
figure(3)
plot(t,pt); title('pt');axis([-5 5 -2 2]);grid on;
```

1.B-1 Execute the m-file and capture the waveform of $p(t)$.

1.B-2 From the graph captured in 1.B-1, calculate the energy of $p(t)$ whose sampled vector is **pt**.

8.2 CORRELATION FUNCTION

2.A The process to shift the elements of a vector using **circshift()** was introduced in Section 1.F of Chapter 5. Execute the following lines of code in the command window. Record the results and justify the results for each case.

```
>> temp=rand(1,12)
>> temp2=circshift(temp',1)'
>> temp2=circshift(temp',4)'
>> temp2=circshift(temp',-5)'
```

2.B [WWW]The vector **pt** created in the m-file in 1.B is a sampling vector of the 50-Hz pulse signal $p(t)$ with a length of 0.05 seconds and a sampling interval of 0.001 seconds.

Suppose that the pulse signal $p(t)$ is transmitted and its delayed version $p(t - t_d)$ is received in additive white Gaussian noise $n(t)$. The delay t_d is due to the distance

between the transmitter and the receiver. Thus the received signal $b(t)$ is expressed as $b(t) = p(t - t_d) + n(t)$. Let us denote the sampled vector of the received signal $b(t)$ by **bt**. In the following problems, we generate the sampled vector of the received signal $b(t)$.

2.B-1 [WWW]Add the following lines to the end of m-file created in 1.B. Add a comment to each of the lines in bold to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

```
% Add the following to the m-file in 1.B.
randn(1,XXX); %XXX=the last three digits of your student ID number. This is irrelevant to the contents but be sure to add.
td=2+2*rand;
delayed_pt = circshift( pt', round(td/t_step) )' ;
nt=randn(1,length(delayed_pt));
bt= delayed_pt+nt;
figure(4)
plot(t, bt);axis([-5 5 -5 5]);
save mydelay.mat td
clear td delayed_pt
```

2.B-2 Note that **rand()** randomly generates a real valued number that is uniformly distributed between 0 and 1. From the line in the m-file in 2.B-1 that generates the delay time **td**, calculate the possible range of **td**. Do not use **load mydelay** to answer this question.

2.B-3 Execute the m-file and capture the waveform of $b(t)$ $(= p(t - t_d) + n(t))$. Estimate the location of the pulse, that is, estimate **td** from the captured waveform. You might need to zoom into any portion of the waveform for a close examination.

2.C The autocorrelation of a signal $f(t)$ denoted by $r_f(\tau)$ is given as [1–3]

$$r_f(\tau) = \int_{-\infty}^{\infty} f^*(t) f(t + \tau)\, dt. \tag{8.1}$$

In the following problem, we calculate the autocorrelation function using numerical integration. Numerical integration was discussed in Section 2.1 of Chapter 2.

2.C-1 The following code fragment determines the autocorrelation function $r_n(\tau)$ of the noise signal $n(t)$ for a given $\tau = 1.5$.

(a) Determine the two quantities marked by **'?'**.
(b) Explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

```
% Execute the m-file in 2.B-1 first and execute the following code fragment in the command window.
>>tau=1.5;
>>shift_samples=round(tau/t_step);
>>nt_tau=circshift(nt', -shift_samples)';
>>rn_tau=sum(conj(?).*?)*t_step % Refer to the numerical integration discussed in Section 1 of Chapter 2.
```

2.C-2 Execute the commands above in the command window and show the execution result of $r_n(1.5)$.

2.C-3 Repeat the process in 2.C-1 to find $r_n(\tau)$ for $\tau = 0, -1.5$, and 2.72.

2.D [WWW]In the code fragment below, the delay τ increases from -4 to 4 with a step size of 0.001. The autocorrelation $r_n(\tau)$ is calculated for each value of τ. Finally, $r_n(\tau)$ is plotted as a function of τ.

```
% Add the following to the m-file in 2.B-1.
tau_vector=[];
rn_vector=[];
for tau=-4:0.001:4
   tau_vector=[tau_vector tau];
   shift_samples=round(?/t_step);
   nt_tau=circshift(nt', -shift_samples)';
   rn_tau=sum(conj(?).*?)*t_step;
   rn_vector=[rn_vector rn_tau];
end
figure(5)
plot(tau_vector, rn_vector)
```

2.D-1 Complete the three places marked by **'?'** in the code fragment and add it to the end of the m-file in 2.B-1. Capture the completed m-file.

2.D-2 Execute the m-file and capture the noise autocorrelation function $r_n(\tau)$ displayed in **Figure 5.**

2.D-3 Summarize the characteristics of the white Gaussian noise on the basis of its autocorrelation function graph in 2.D-2. Discuss whether or not the graphical result is what you expected.

2.D-4 (a) [T]From equation (8.1), prove that the autocorrelation at $\tau = 0$ is equal to the signal energy. (b) From (a) and the autocorrelation function captured in 2.D-2, determine the energy of $n(t)$.

NOTE: With the ideal model (unrealistic), the background noise has infinite power and thus infinite energy. However, $n(t)$ in the m-file of 2.D corresponds to the sampled

noise after a filter of finite bandwidth. Thus, within the time interval where the noise is truncated, it has a finite energy.

2.E The m-file completed in 2.D can be modified to plot the autocorrelation function of any pulse signal $p(t)$ whose sampled vector is represented by **pt**.

2.E-1 [T]Write the expression for $r_p(\tau)$. You may modify equation (8.1).

2.E-2 [WWW]From the equation obtained in 2.E-1, properly modify the related lines of the m-file in 2.D to plot $r_p(\tau)$, instead of $r_n(\tau)$, and identify the modified lines. Let **rp_vector** denote the vector for the sampled $r_p(\tau)$.

2.E-3 Execute the modified m-file in 2.E-2 and capture the graph of $r_p(\tau)$. Prior to capturing the graph, zoom into the range of [−0.2 0.2] along the x axis using **axis()** or the magnifying button in the menu bar in order to clearly see the shape of $r_p(\tau)$.

2.E-4 Find the energy of $p(t)$ from its autocorrelation function captured in 2.E-3. Is it consistent with the answer to 1.B-2?

2.E-5 [T]Prove that $r_f(\tau)$, the autocorrelation function of $f(t)$, given in equation (8.1) satisfies the Hermitian symmetry property, that is,

$$r_f(-\tau) = r_f^*(\tau). \tag{8.2}$$

2.E-6 Check whether the graphs of $r_n(\tau)$ and $r_p(\tau)$ captured in 2.D-2 and 2.E-3, respectively, match the results given by equation (8.2).

2.F The cross-correlation function of two signals $f(t)$ and $g(t)$ denoted by $r_{fg}(\tau)$ is calculated as [1–3]

$$r_{fg}(\tau) = \int_{-\infty}^{\infty} f^*(t)\, g(t+\tau)\, dt. \tag{8.3}$$

2.F-1 [WWW]The code fragment below calculates $r_{pn}(\tau)$, the cross-correlation of $p(t)$ and $n(t)$, by using numerical integration. Recall that **pt** and **nt** are the sampled vectors of $p(t)$ and $n(t)$, respectively. The variable **rpn_tau** represents $r_{pn}(\tau)$ for a given value of τ(=**tau**).

Determine what should be placed at **'?'** in the line **'rpn_tau=sum(conj(?).*?)*t_step;'** to implement the cross-correlation equation expressed in equation (8.3). After completing this line, add the code fragment to the end of m-file in 2.B-1 and capture the complete m-file.

```
% Add the following code fragment to the m-file in 2.B-1.
tau_vector=[];
rpn_vector=[];
for tau=-4:0.001:4
   tau_vector=[tau_vector tau];
   shift_samples=round(tau/t_step);
```

```
    nt_tau=circshift(nt', -shift_samples)';
    rpn_tau=sum(conj(?).*?)*t_step;
    rpn_vector=[rpn_vector rpn_tau];
end
figure(6)
plot(tau_vector, rpn_vector); axis([-4 4 -0.05 0.05]);
```

2.F-2 Execute the completed m-file above and capture the waveform of $r_{pn}(\tau)$ displayed in MATLAB **Figure 6**.

2.F-3 From the plots generated in 2.D-2, 2.E-3, and 2.F-2, determine the maximum values of $r_n(\tau)$, $r_p(\tau)$, and $r_{pn}(\tau)$.

2.F-4 Explain the results in 2.F-3. Why is the peak value of $r_{pn}(\tau)$ smaller than the peak values of $r_n(\tau)$ and $r_p(\tau)$?

2.G The cross-correlation $r_{pb}(\tau)$ between the pulse $p(t)$ and the received signal $b(t)$ $(= p(t - t_d) + n(t))$ can be written as

$$r_{pb}(\tau) = r_p\left(\tau - t_d\right) + r_{pn}(\tau). \tag{8.4}$$

2.G-1 [T]Derive this equation.

2.G-2 From equation (8.4) and the shapes of $r_p(\tau)$ and $r_{pn}(\tau)$ captured in 2.E-3 and 2.F-2, we can more or less predict the approximate shape of $r_{pb}(\tau)$. Since the delay t_d is an unknown variable at this point, give t_d an arbitrary value to determine the shape of $r_{pb}(\tau)$.

(a) Develop your process to determine the shape of $r_{pb}(\tau)$.
(b) Where along the τ axis does the peak of $r_{pb}(\tau)$ occur?

2.G-3 From the answer in 2.G-2, develop a process to use $r_{pb}(\tau)$ to locate the pulse in the received signal, which consists of the pulse and a noise signal that is independent of the pulse. In other words, find t_d from $r_{pb}(\tau)$.

2.G-4 [WWW]The following code fragment generates **rpb_vector**, the sampled vector of $r_{pb}(\tau)$, using the numerical integration technique and generates its graph. The variable **bt_tau** represents the sampled vector of $b(t + \tau)$ for a given value of τ(=**tau**). Using a similar code structure as the m-files in 2.F-1, complete the places marked by **'?'** and then add this code fragment to the end of the m-file completed in 2.B-1. Next, execute this m-file and capture the waveform of $r_{pb}(\tau)$ displayed in MATLAB **Figure 7**.

```
% Add the following code fragment to the m-file in 2.B-1
tau_vector=[];
rpb_vector=[];
for tau=-4:0.001:4
    tau_vector=[tau_vector tau];
    shift_samples=round(tau/t_step);
```

```
    bt_tau= circshift(bt', -shift_samples)';
    rpb_tau=sum(conj(?).*?)*t_step;
    rpb_vector=[rpb_vector rpb_tau];
end
figure(7)
plot(tau_vector, rpb_vector); axis([-4 4 -0.07 0.07]);grid on;
```

2.G-5 From the graph of $r_{pb}(\tau)$ captured in 2.G-4, obtain an estimate of the pulse delay t_d. Zoom into the desired portion of the figure will help increase the estimation accuracy.

2.G-6 Execute the following in the command window to obtain the actual value of t_d, that is, **td** in the m-file. Check whether the estimate in 2.G-5 is correct. There might be a slight mismatch between these two values because of approximation in numerical calculation or variation of the instantaneous noise.

```
>>load mydelay.mat
>>td
```

2.G-7 Each time the m-file is executed, **td** and the sampled noise vector are updated. Run the simulation multiple times to generate multiple **td** values. In each simulation, compare the estimated delay with the actual value to verify that the estimation method is correct. Capture your comparison results.

2.G-8 [A]Change the line '**pt=sign(psint);**' in the m-file into '**pt=psint;**' to replace the rectangular pulse by a sinusoidal signal for $p(t)$. Execute the m-file and estimate td. Does the estimate depend on the pulse type?

2.H The pulse signal $p(t)$ is deterministic and is, in general, independent of the background Gaussian noise $n(t)$. Under this condition, $r_b(\tau)$, the autocorrelation of the received signal $b(t)$ $(= p(t - t_d) + n(t))$ can be expressed as

$$r_b(\tau) = r_p(\tau) + r_n(\tau). \tag{8.5}$$

2.H-1 [T]Derive equation (8.5).

2.H-2 Revisit the m-file in 2.B-1 and change the line '**nt=randn(1,length (delayed_pt));**' into '**nt=0.075*randn(1,length(delayed_pt));**'. Also execute the following two lines of code in the command window to calculate the energy of the revised noise sample vector **nt**.

```
>> nt=0.075*randn(1,length( delayed_pt));
>> sum(nt.^2)*t_step
```

2.H-3 [WWW] If you have finished 2.G-8, change the line **'pt=psint;'** in the revised m-file in 2.H-2 back to **'pt=sign(psint);'**. Then add the following code fragment to the end of the revised m-file. Execute the m-file and capture the graph of $r_b(\tau)$.

```
tau_vector=[];
rb_vector=[];
for tau=-4:0.001:4
   tau_vector=[tau_vector tau];
   shift_samples=round(tau/t_step);
   bt_tau=circshift(bt', -shift_samples)';
   rb_tau=sum(conj(bt).*bt_tau)*t_step;
   rb_vector=[rb_vector rb_tau];
end
figure(5)
plot(tau_vector,rb_vector);
grid on;
axis([-0.2 0.2 1.5*min(rb_vector) 1.5*max(rb_vector)]);
```

2.H-4 Assess whether the plot captured in 2.H-3 is consistent with equation (8.5). Identify first which problems and their answers are relevant for this assessment.

2.I [T] The correlation function of any power signal $f(t)$ is defined as [1–3]

$$R_f(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f^*(t) f(t+\tau)\, dt. \tag{8.6}$$

A periodic signal is a power signal. For periodic power signals with period P, $R_f(\tau)$ can be expressed as

$$R_f(\tau) = \frac{1}{P} \int_{-\frac{P}{2}}^{\frac{P}{2}} f^*(t) f(t+\tau)\, dt. \tag{8.7}$$

2.I-1 [T] Show that equation (8.6) is equivalent to equation (8.7) if $f(t)$ is periodic.

2.I-2 [T] Fig. 1.1 in the Chapter 1 illustrates a special case of a rectangular periodic function. Using equation (8.7), derive the correlation function of a periodic rectangular function $f(t)$ with period P, pulse width W ($W < P/2$), and height A.

2.I-3 [T] Sketch by hand $R_f(\tau)$ of the function given in 2.I-2 versus τ.

2.J Suppose that an unknown periodic signal $f(t)$ is received together with an additive noise signal $n(t)$, that is, $r(t) = f(t) + n(t)$. In this problem, we identify the periodic signal $f(t)$ and determine its parameters from the received signal $r(t)$.

2.J-1 [WWW] Download **rt_sampled.mat** from the companion website and execute the following lines of code. Record the name and the length of a vector contained in **rt_sampled.mat**.

```
>> clear
>> load rt_sampled.mat
>> whos
```

2.J-2 The vector **rt** contained in the data file **rt_sampled.mat** is the sampled vector of the received signal $r(t)$ with a sampling interval of 0.001 seconds and its length is 20 seconds. Execute the following to draw the waveform of $r(t)$ and capture it.

```
>> plot(0:0.001:20, rt)
```

2.J-3 Closely observe the waveform of r(t) by zooming into the various portions of the figure. Based on your observation, describe the shape of the period signal $f(t)$ distorted by noise and estimate its parameters such as period or amplitude.

2.J-4 [T] The autocorrelation function of r(t) is the sum of the autocorrelation functions of $f(t)$ and n(t) expressed as

$$R_r(\tau) = R_f(\tau) + R_n(\tau). \tag{8.8}$$

Prove equation (8.8).

2.J-5 [WWW] The following m-file calculates $R_r(\tau)$, the autocorrelation function of the received signal $r(t)$, via numerical integration and plots it. Since the signal period is unknown and the received signal is given as a vector that represents only 20 seconds of the continuous time signal in the m-file, we use equation (8.6) and set T at 20 seconds, rather than infinity, in the numerical integration.

Add a comment to each of the lines in bold to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

```
clear
load rt_sampled.mat

t_step=0.001;
Rr_vector=[];
tau_vector=[];
for tau=-4:0.01:4
   tau_vector=[tau_vector tau];
   shift_samples=round(tau/t_step);
```

```
   rt_tau=circshift(rt', -shift_samples)';
   Rr_tau=1/20*sum(conj(rt).*rt_tau)*t_step;
   Rr_vector=[Rr_vector Rr_tau];
end
figure
plot(tau_vector, Rr_vector);grid on;
```

2.J-6 Execute the m-file above and capture the graph of $R_r(\tau)$.

2.J-7 Using the figure of $R_r(\tau)$ captured in 2.J-6 and equation (8.8), sketch $R_f(\tau)$ and justify your sketch.

2.J-8 Note that the sketch of $R_f(\tau)$ in 2.J-7 should be similar to the sketch in 2.I-3. Compare these two and estimate the parameters P, W, and A. With the estimated parameters, sketch the estimated periodic signal f(t).

2.J-9 Develop a method to systematically determine the shape and estimate the parameters of an unknown periodic signal distorted by an additive noise.

8.3 ENERGY SPECTRAL DENSITY

3.A In this subsection we calculate the energy spectral density (ESD) [4] using numerical integration.

3.A-1 [T]The ESD of an energy signal $f(t)$ is defined as $|F(\omega)|^2$, where $F(\omega)$ is the Fourier transform of $f(t)$. (a) Establish relationship between $r_f(\tau)$, the autocorrelation function of $f(t)$, and its ESD $|F(\omega)|^2$; that is, calculate $|F(\omega)|^2$ from $r_f(\tau)$ or vice versa. (b) Prove this relationship.

3.A-2 [WWW]Revisit the m-file in 2.E-2. If you have completed 2.G-8, change the line '**pt=psint;**' back to '**pt=sign(psint);**'. Recall that **pt** is the sampled vector of a pulse signal $p(t)$, whose graph has been captured in 1.B-1, and **rp_vector** is the numerically calculated vector of the autocorrelation function $r_p(\tau)$.

The code fragment below calculates and plots $|P(\omega)|^2$, the ESD of the pulse signal p(t) using the relationship established in A-1. The variable **ESD_vector** denotes the numerically calculated vector of the ESD of p(t). The frequency f increases from 0 Hz to 500 Hz with a 3-Hz step size, and the ESD at each frequency is via numerical integration and concatenated to the vector **ESD_vector**. Add a comment to the line in bold to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

```
% Add the following code fragment to the m-file in 2.E-2.
tau=-4:0.001:4;
f_vector=[]; ESD_vector=[];
for f=0:3:500
```

```
    f_vector=[f_vector f];
    ESD_f=sum(rp_vector.*exp(-j*2*pi*f*tau))*0.001;
    ESD_vector=[ESD_vector ESD_f];
end
figure(8)
plot(f_vector,ESD_vector) grid on
xlabel('frequency [Hz]')
```

3.A-3 Add the code fragment above to the end of the m-file in 2.E-2 and execute the completed m-file. Capture the ESD graph of $p(t)$.

3.B From the captured ESD graph, (a) find the frequency where the ESD reaches the maximum. (b) Is the answer to (a) consistent with the waveform of $p(t)$ captured in 1.B-1? Justify your answer.

3.C [WWW] The following code fragment calculates $P(\omega)$, the Fourier transform of $p(t)$, via numerical integration and then calculates the ESD $|P(\omega)|^2$ directly from $P(\omega)$, rather than using the autocorrelation.

```
% Add the following code fragment to the m-file in 2.E-2.
f_vector=[]; P_vector=[];
for f=0:3:500
    f_vector=[f_vector f];
    P_f=sum(pt.*exp(-j*2*pi*f*t))*0.001;
    P_vector=[P_vector P_f];
end
figure(9)
plot(f_vector,abs(P_vector).^2)
grid on
xlabel('frequency [Hz]')
```

3.C-1 Add a comment to the line in bold to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

3.C-2 Add the code fragment above to the end of the m-file in 2.E-2 and execute the m-file. Capture the resulting **Figure 9** window.

3.C-3 (1) Are the graphs in 3.C-2 and 3.A-3 the same? (2) This result validates the answer to which problem in this chapter?

3.C-4 Develop two different methods to calculate the ESD of an energy signal, one uses the autocorrelation function and one does not.

REFERENCES

[1] A. C. Aitken, *Statistical Mathematics*, 8th ed., Edinburgh: Oliver & Boyd, 1957.

[2] S. Dowdy and S. Wearden, *Statistics for Research*, New York: Wiley, 1983.

[3] G. U. Yule and M. G. Kendall, *An Introduction to the Theory of Statistics*, 14th ed., Glasgow: Charles Griffin, 1950.

[4] J. Y. Stein, *Digital Signal Processing: A Computer Science Perspective*, Hoboken, NJ: Wiley, 2000.