1

MATLAB AND SIMULINK BASICS

- Arithmetic operators.
- Vector and matrix manipulation.
- Symbolic math.
- Script file (m-file) and user-defined functions.

1.1 OPERATING ON VARIABLES AND PLOTTING GRAPHS IN MATLAB

The fundamental MATLAB commands can be categorized into six groups, each of which is covered in one subsection. The first four subsections deal with the operations of different variable types and the last two subsections deal with the plotting commands that are frequently used in this book. On a PC that is installed with MATLAB, start MATLAB. A command window will appear where one can type in and execute MATLAB commands. Execute the set of commands/codes in the boxes and check the results. This self-study method is one of the fastest ways to master the basic MATLAB commands.

In a report, document what each command does. Focus on the specific actions and purposes, rather than the execution results. For commands that return an error message, document the reasons. Follow this guideline for all exercises in Section 1.1.

A sample report is available from the companion website. For information to access this website, refer to the guide at the beginning of this book.

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

1.A Operation of scalar variables.

1.	X	6.	X*Y-X*3-Y	11.	X=12e6
2.	X=12	7.	X=Y^2	12.	clc
3.	X=X+2	8.	Z=sqrt(Y)	13.	x=rand
4.	Y=X+3	9.	X=2; Y=4; Z=X+Y	14.	x=rand
5.	Y*6	10.	Z=X^Y	15.	help rand

In addition, explain why the same command executed twice in item 13 and item 14 generates different results.

1.B Operation of complex numbers.

1.	i	6.	Z=X*Y	11.	angle(Z)
2.	j	7.	real(Z)	12.	who
3.	X=1+3*j	8.	imag(Z)	13.	whos
4.	Y=-2+j ;	9.	conj(Z)	14.	clear
5.	Z=X+Y	10.	abs(Z)	15.	who

1.C Operation of vectors.

1.	X=2 : 2: 10	12.	Y=[2; 1; 4; -3]	23.	Y=rand(1,5)
2.	Y=1 : 5	13.	Z=Y'	24.	Y=rand(4)
3.	Z=X + Y	14.	Z(1)	25.	Y=[7 3 -1 2]
4.	Z=X.*Y	15.	Z(2)	26.	mean(Y)
5.	Z=X*Y	16.	Z(1:3)	27.	var(Y)
6.	Z=X./Y	17.	Z(2:4)	28.	min(Y)
7.	Z=X/Y	18.	length(Z)	29.	max(Y)
8.	2*Y	19.	X=[2 4 8 16]	30.	[a b]=min(Y)
9.	Z=0 : 10	20.	Y=log2(X)	31.	sort(Y)
10.	sum (Z)	21.	Y^2	32.	Y=[Y 5]
11.	Y=[2 1 4 –3]	22.	Y.^2	33.	Z=[Y(3:4) X(1:2)]

1.D Operation of matrices.

1.	X=[3 6 -2 -1; 0 5 2 1; 7 -1 4 8];	11.	Y(1, :)=X(2, :)	21.	Z=X.^2 +3*Y
2.	X(2,1)	12.	Y(2, :)=X(1, :)	22.	max(Z)
3.	X(2,3)	13.	Y(3, :)=[1 2 3 4]	23.	[T1 T2]=max(Z)
4.	X(1, :)	14.	Z=X – Y	24.	mean(Z)
5.	X(:, 2)	15.	Z=X*Y	25.	max(mean(Z))
6.	X(1:2,:)	16.	Z=X*Y'	26.	max(max(Z))
7.	X(: , 2:3)	17.	Z=X.* Y	27.	Z=rand(4)
8.	Y=[1 0 2; 3 2 1; 2 3 4]	18.	Z=X^2	28.	X=inv(Z)
9.	Y'	19.	Z=X.^2	29.	Y=X*Z
10.	Y=zeros(3,4);	20.	Z=2.^X	30.	size(Z)

1.E Plotting some basic functions.

```
1.  x=0:0.1:10
2.  y1=sin(x)
3.  y2=cos(x);
4.  plot(x)
5.  plot(y1)
6.  plot(x,y1) %Compare to 5
7.  grid
8.  hold on
9.  plot(x,y2)
10. y3=exp(-x)
11. plot(x,y3,'r')
12. legend('sin(x)','cos(x)','exp(-x)')
13. axis([-5 15 -3 3])
14. axis([0 10 -2 2])
15. figure
16. subplot(3,1,1)
17. plot(x,y1)
18. subplot(3,1,2)
19. plot(x,y2)
20. subplot(3,1,3)
21. plot(x,y3)
22. semilogy (x,y3)
23. help plot
24. help semilogy
```

1.F Boolean operations and plotting graphs over a limited range of the *x* axis.

```
1.  A=[0 1 2 3 4];
2.  A<3
3.  B=(A>2)
4.  C=([1 1 0 0] & [ 1 1 1 0])
5.  C=([1 1 0 0] | [ 1 1 1 0])
6.  C=~[1 0 1 0 0]
7.  C=([1 0 1 1 1]==[1 0 1 0 0])
8.  C=([1 0 1 1 1]~=[1 0 1 0 0])
9.  x=0:0.01:10;
10. y=(x<3);
11. figure
12. plot(x,y); axis([0 10 -2 2]); grid;
13. y=(1<x)&(x<4);
14. plot(x,y); axis([0
    10 -2 2]); grid;
15. y=1<x<4;
16. plot(x,y); axis([0
    10 –2 2]); grid;
```

In addition, explain the difference between items 13 and 15.

1.2 USING SYMBOLIC MATH

In the symbolic math in MATLAB, the characters (or words) such as **a**, **b**, and **temp** are treated as symbolic variables, not numeric variables. Mathematical expressions can be computed or manipulated in symbolic forms. Find out what else can be done using symbolic math in the following problems.

2.A Write down what each of the lines in the following box does and capture the execution result.

```
>>syms  a b c x t
>>y=sin(t);
>>diff(y)
>>int(y)
>>int(y, t, 0, pi)
>>z=int(xˆ2*exp(-x),x,1,3)
>>double(z)
>>limit(sin(t)/t,t,0)
>>symsum(xˆ2, x,1,4)
>>T=solve(a*xˆ2+b*x+c,x)
```

```
>>T2=solve(a*x^2+b*x+c,b)
>>a=1;b=2;c=3;
>>z=eval(T)
>>a=t;
>>z=eval(T)
```

2.B Verify the following quantities by using the symbolic math. Capture the calculation results.

$$\int_{-\infty}^{\infty} e^{-z^2} \mathrm{d}z = \sqrt{\pi}, \tag{1.1}$$

$$\sum_{r=1}^{\infty} \left(\frac{1}{3}\right)^r = \left(\frac{1}{2}\right), \tag{1.2}$$

$$\lim_{x\to\infty} \left(1 + \frac{1}{x}\right)^x = e. \tag{1.3}$$

2.C Calculate the following integral by using the symbolic math. Be sure to perform **double(c)** after symbolic integration. Also explain why executing **double ( )** is needed to obtain the solution.

$$c = \int_1^2 \sin(z) e^{-z} \mathrm{d}z. \tag{1.4}$$

1.3 CREATING AND USING A SCRIPT FILE (m-FILE)

The commands and functions we have covered so far are all executable directly in the command window. Using a "script file," which is also called an "m-file" in the earlier versions of MATLAB, users can execute various algorithms or can implement user-defined functions. In this book, the traditional term "m-file" will be used to refer to a MATLAB "script file."

3.A Follow the steps below and learn how to create and execute an m-file.

Step 1. Open a new script file editing window.

Step 2. [WWW]Shown in the box below is an m-file that plots $y = x \sin(ax)$, for the cases of $a = 0.1$, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8 over the range $0 < x < (10 + D)$, where $D =$ the last digit of your student ID number. Write this m-file and save it as **CH1_3A.m**. The m-file name must begin with a letter; files with a name that begins with a number will not execute in MATLAB. Be sure not to use a space or mathematical operator (e.g., +, −, /, *) in the file name.

NOTE: For the parts with the superscript [WWW] prefixed, the companion website provides the supporting file or the required data. For information to access this website, refer to the guide section: "ABOUT THE COMPANION WEBSITE" at the beginning of this book.

```
clear
x=0:0.1:(10+The last digit of your student ID number);
for n=1:8
  a=n/10;
  if (a==0.2)
    a=0.25;
  end
  y(n,:)=x.*sin(a*x);
end
plot(x,y)
xlabel('x')
ylabel('y=x sin(ax)')
legend('a=0,1','a=0.25','a=0.3','a=0.4','a=0.5','a=0.6','a=0.7','a=0.8')
grid on
```

3.A-1 Add a comment to explain each line in the m-file. Capture the commented m-file.

3.A-2 Execute the m-file you have created. You can either click **'run'** button in the menu bar of the m-file editor or press the F5 key on the keyboard or type the m-file name in the command window as

```
>>CH1_3A
```

Capture the result. To capture a figure, you may navigate to **'Edit/Copy Figure'** in the menu of the figure and then paste it in your report.

3.A-3 Execute the following commands in the MATLAB command window. Based on the results, document the meanings of the two variables. Do not capture the execution results.

```
>>x
>>y
```

3.B Let us write an m-file to plot sine waveforms of 10 different frequencies by properly modifying the m-file created in 3.A.

3.B-1 Consider 10 sine waveforms whose frequencies are 1, 2, 3, 4, …, 10 Hz. In your code, calculate the smallest period (the highest frequency) among these waveforms and denote it by T. Then, overlay the 10 sine waveforms in the range of $-2T < t < 2T$, the range of the time axis (x axis) in the graph. Use **legend()** to label the 10 waveforms. Use a **'for'** loop as done in the m-file in 3.A.

Capture the m-file and the execution result.

3.B-2 Use the command **mesh()** to plot the 10 sinusoids in a three-dimensional plot. Then click the axis rotation button in the menu of the figure to rotate the axis of the graph. Execute the m-file and capture the plot.

3.C The objective of this problem is to write an m-file to find the position of the maximum value in each column (or row) of a matrix and to calculate the mean of each column/row of the matrix.

3.C-1 [WWW]The m-file below, by using **rand()**, generates a 10 (row) × 9 (column) matrix **A**. The elements of the matrix are independent and identically and uniformly distributed between 0 and 3. By using **max( )** twice, the m-file finds the maximum elements of **A** and its row and column indexes.

```
%Do not append ';' at the end of the lines in this m-fie in order to see the result
of each line.
clc; clear
A=3*rand(9,10)

[B C]=max(A)
[D E]=max(B)

Max=D
Position=[C(E)  E]
```

(a) Add a comment for each line to explain what it does. Capture the m-file.
(b) Execute the m-file. Capture all the execution results displayed in the command window. You may need to scroll up or down the command window to avoid missing any part of the execution results.
(c) Is the execution result of each line what you expected to see?

3.C-2 In the m-file of 3.C-1, add the part that computes the mean (use the function **mean( )**) of each row of the matrix. Also add the function to find the row with the largest mean value. Capture the m-file and the execution result.

3.D [WWW]The m-file below plots the discontinuous function $y(t)$ given in equation (1.5) using logical operators.

$$y(t) = \begin{cases} \sin\left(2\pi \times 5t + \dfrac{\pi}{3}\right) & 1 \le t \le 2, \\ 0 & 0 \le t < 1 \text{ or } 2 < t \le 5. \end{cases} \tag{1.5}$$

```
clear;
t=0:0.01:5;
x=(1<=t)&(t<=2);
x2=sin(2*pi*5*t+pi/3);
y=x.*x2;
plot(t, y); axis([-1 6 -2 2])
```

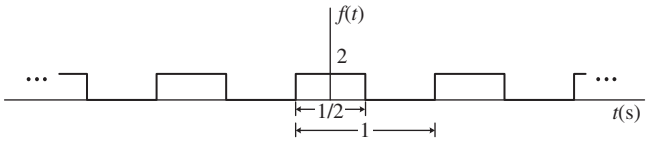

FIGURE 1.1 Periodic function $f(t)$.

3.D-1 (a) Add your explanation to each line as a comment and capture the m-file. (b) Execute this m-file and capture the result.

3.D-2 Write an m-file to plot $f(t)$ in Fig. 1.1 using **sin( )** (or **cos( )**) and Boolean operators (==, >, < , <=, >=). Capture your m-file and the execution result.

1.4 [A]USER-DEFINED MATLAB FUNCTION

Similar to many other programming languages, MATLAB also supports the use of user-defined functions to avoid repeatedly editing the main body of a code. User-defined functions are similar to the built-in MATLAB commands or functions; they follow certain syntax and are normally saved in the same folder where the main m-file is located in, but it can be saved in a different folder. Through the following problem you will learn how to write and to use user-defined MATLAB functions.

4.A Let us write a MATLAB function that converts a number in linear scale into dB scale. In MATLAB, open the script file editor and write the following m-file. Save the m-file as **lin2dB.m** (if you click "save," the default file name will be **lin2dB.m**).

```
function xdB=lin2dB(x)
xdB=10*log10(x);
```

4.A-1 Execute **lin2dB(100)** in the command window. Capture the result.

4.A-2 Execute **lin2dB([1 2 10 20 1/10])** in the command window. Capture the result and check whether or not the results are correct.

4.B Let us write a MATLAB function that plots the Gaussian probability density function.

4.B-1 Write the following m-file and save it. Add your comments explaining what each line does or means.

```
function plot_gaussian(m, v)
x=m+sqrt(v)*(-5:0.01:5);
fx=1/sqrt(2*pi*v)*exp(-(x-m).^2/(2*v));
plot(x,fx)
```

4.B-2 Execute **plot_gaussian(0, 1)** in the command window and capture the result.

4.B-3 Try a few arbitrary values for the arguments (i.e., the mean **m** and variance **v**) of **plot_gaussian()**. Capture your results.

4.C Write a user-defined function **swap(A,row0col1,c,d)** that swaps two rows (or columns) of a matrix. If **row0col1** is 0, then **swap(A, row0col1,c,d)** swaps the **c**-th row and the **d**-th row of a matrix **A** and returns the swapped matrix. If **row0col1** is 1, then **swap(A,row0col1,c,d)** swaps the **c**-th column and the **d**-th column of a matrix A and returns the swapped matrix.

4.C-1 [WWW]An incomplete version of **swap.m** is provided below. Complete all parts marked by **'?'** and add a comment for each line you are completing.

```
function e=swap(A,row0col1,c,d)
e=A;
if row0col1==0
   e(d,:)=A(c,:);
   e(?,:)=A(?,:);
end

if row0col1==1
   ??;
   ??'
End
```

4.C-2 Execute the following command lines and capture the results. Check whether or not your swap function works correctly.

```
>>x=rand(4,5)
>>y=swap(x,0,2,4)
>>z=swap(y,1,5,1)
```

1.5 DESIGNING A SIMPLE SIMULINK FILE

Complete all of the following steps but document only the results of Step 5.F-2 and Step 5.G-6 in a report.

5.A Creating a new Simulink design file.

Step 5.A-1 Start MATLAB.

Step 5.A-2 In the command window, execute **simulink** to start Simulink as shown below. You can also start Simulink from the menu bar, which might be different for different MATLAB versions.

```
>> simulink
```

Step 5.A-3 Press 'Cntrl+N' keys when the **Simulink Library browser** window is active. A Simulink design window, which we call "design window" in short hereafter, will open. Alternatively, you can use the shortcut icon in the menu bar, which may be different for different Simulink versions.

5.B Adding blocks to the Simulink design window.

In the design window, you can import and add various functional blocks from the Simulink library.

Step 5.B-1 The left side of **Simulink Library browser** window provides a list of the function blocks.

Let us add a block that generates a sine waveform in the new Simulink model. The sine waveform generator is one of the Simulink sources. A click on **Simulink/Sources** will show all the blocks in the source directory. In order to get familiar with Simulink, you might navigate to different categories such as **Math Operations** and **Logic and Bit Operations** to check out the blocks in these directories.

Step 5.B-2 Click the block **Sine Wave** in **Simulink/Sources** and drag it into the empty design window created in Step 5.A-3. This can also be done by right-clicking the block and then choose '**Add**…'.

Step 5.B-3 Browse through the **Simulink/Sinks** category and add the **Scope** block in your design window as shown in Fig. 1.2.

If you are not sure in which directory (category) your desired block is, you can search for it by entering the block name in the search input field in the menu of **Simulink Library Browser** window.

5.C Connecting the blocks.

In order to get a desired system function, we must properly connect the output of each block to the input of another block. Let us connect the output of the **Sine Wave** block to the input of the **Scope** block. This can be done by simply clicking and dragging the output port of the **Sine Wave** block to the input port of the **Scope**. One can click on the source block and then 'Cntrl+click' on the destination block.

5.D Setting block parameters and simulation time.

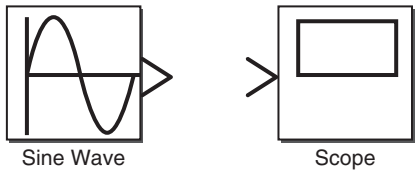

FIGURE 1.2 Adding blocks to a new design.

The Simulink blocks typically have their default parameters. Double-click the block to open the parameter setting window where a description of that block is also provided.

NOTE: The same block may have different names, parameter names, and procedures to set its parameters in different Simulink versions. If the instructions do not work for your Simulink version, you may use the completed Simulink design files uploaded on the companion website.

Step 5.D-1 Open the parameter setting window of the **Sine Wave** block. Check all the parameters and try to understand what each of these parameters means.

Step 5.D-2 In this tutorial, we consider an example to generate $\sin(4\pi t)$. Read the description of the **Sine Wave** block and properly set **Amplitude**, **Bias**, **Frequency (rad/s)**, **Phase (rad)** to generate $\sin(4\pi t)$. Note that in MATLAB pi is a reserved variable equal to π.

Set the parameter Sample time of **Sine Wave** block to 1/100. The note below provides some details about the parameter **Sample time** that is required for most of the blocks to be used later.

NOTE: All the signals generated in Simulink blocks have their own **Sample time** parameter. The Sample time parameter sets the sample time interval of the signal generated by the block. Typically, **Sample time** should be set much smaller than the inverse of the Nyquist rate. Such setting will make the sampled signal look like a continuous signal when plotted. On the contrary, too small a value for **Sample time** will increase simulation time. Note that for blocks with input port(s), **Sample time** of -1 simply copies (inherits) the **Sample time** of the input signal(s).

Step 5.D-3 Open the **Scope** display window by double-clicking the **Scope** block. Then, in the menu bar of the Scope display window, click the icon named **Configuration Properties** (or **Parameters** in some old versions) to open the **Scope** parameter setting window.

(a) The parameter **Number of ports** (**Number of axes** or simply **Axes** in some old versions) determines the number of input ports of the **Scope** block. Set it as 1, since only one **Sine Wave** block's output will be monitored.
(b) Click the **Logging** (**History** or **Data History** in some old versions) tab and unselect the check box **Limit data points to last.**

Be aware that the graphical user interface such as the menu bar and the parameter input fields might be different for different versions of Simulink.

Step 5.D-4 There is one input field in the menu bar of the design window. That input field is for a parameter **Simulation stop time**. The number typed in that field determines the execution time of the simulation, that is, the time up to which point the signal is generated and processed, not the actual time required for running the simulation. In this tutorial, we want to see 20 cycles of the output waveform of the **Sine Wave** block set in Step 5.D-2, that is, $\sin(4\pi t)$. Thus, we set the **Simulation stop time** to $20 \times (2\pi/4\pi) = 10$ seconds. Type in 10 in that input field.

5.E Saving the files.

By using **'File/Save as'** in the menu bar, save your design (currently **untitled***). In the Simulink versions before R2012a, the file extension is ***.mdl**. For R2012a and newer versions, the file extension is ***.slx** by default, but the extension ***.mdl** is still supported. You can save your design in any folder of your choice. Save your design file as a new file.

5.F Running the simulation and observing the output waveforms.

Step 5.F-1 On the left side of the **Simulation stop time** input field, there is a play button. Click it to run the simulation.

Step 5.F-2 If the simulation is complete, double-click the Scope block to open the **Scope** display window. Capture the **Scope** display window. Examine whether the waveform displayed in the **Scope** display window displays 20 cycles of the desired sine waveform, that is, $\sin(4\pi t)$.

Step 5.F-3 Change the parameters of the **Sine Wave** block to generate a different sine waveform and capture your result. Examine whether the waveform is generated as you set.

5.G Adding more blocks and observing multiple waveforms.

Before proceeding to the following steps, be sure to restore the parameters of **Sine Wave** to those set in Step 5.D-2 to generate $\sin(4\pi t)$.

Step 5.G-1 If more than one block of the same function are needed for the design, you can copy and paste the one configured by right-clicking it and selecting **copy** in the pop-up menu and then right-clicking anywhere else in the design and selecting **paste**. You can also copy and paste the block by 'Cntrl+C' and 'Cntrl+V'. Add one more Sine Wave block using copy and paste. By default, the pasted block will be named **Sine Wave1**.

Step 5.G-2 Search for the block **Add** (or **Sum**) in the **Simulink library browser** and add it to the slx (or mdl) file.

Step 5.G-3 Referring Step 5.D-3, set **Number of ports** of the **Scope** block to 3. Then, set **Layout** to **3×1** (no need in some old versions). Now the **Scope** block should display three input ports.

NOTE: Throughout this book, be sure to properly set **Layout** dimension of the **Scope** blocks to separately display the input signals as done here.

Step 5.G-4 Change the parameter **Amplitude of Sine Wave** into 2 to generate $2\sin(4\pi t)$ and set the parameters of **Sine Wave1** to generate $\sin(5.2\pi t)$.

Step 5.G-5 Connect the blocks as shown in Fig. 1.3. To connect an output of a block to the inputs of multiple destination blocks, left click and drag for connecting to the first destination block. Then, right-click and drag for connecting to the rest of the destination blocks.

Step 5.G-6 Run the simulation. Capture the **Scope** display window.

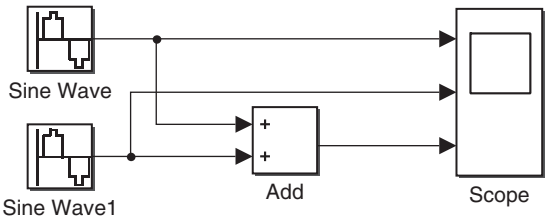

FIGURE 1.3 A test design for sine waveform generation and observation.

Step 5.G-7 You can change the viewing ranges of the x axis (time axis) and y axis in the **Scope** display window using the zoom icons in the menu bar. Locate the corresponding icons for **Zoom** (to zoom in on data in both the x and y directions), **'Zoom X-axis'**, **'Zoom Y-axis'**, and **Autoscale**. **Autoscale** displays the whole graph. Selecting any of the other three allows you to use the cursor to specify any viewing range.

1.6 CREATING A SUBSYSTEM BLOCK

In a Simulink model, right-clicking any component will pop-up a menu that allows the user to **'Create Subsystem from Selection'**, among many other functions. This feature allows us to group certain parts, for example, the frequently used parts of a design, into a single subsystem. The subsystem can be saved as a "user-defined" block to enrich the library Simulink provides. For a complex design with large number of components, creating subsystems will make the design a lot easier to read and to understand.

In this section, we design two user-defined blocks, a **Sound Source** and a **Spectrum Viewer**, that will be used frequently later in other chapters. Complete all of the following steps but document only the results of 6.C-1 and 6.C-2 in a report.

6.A Creating the Sound Source subsystem block.

Step 6.A-1 [WWW]Download **sound.mat** from companion website and save it in your work directory. Design a new Simulink model as shown in Fig. 1.4.

Set the parameters of each block as follows. Do not change other parameters not mentioned here.

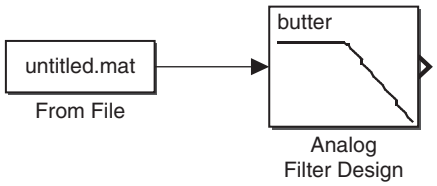

FIGURE 1.4 Design for the subsystem named **Sound Source**.

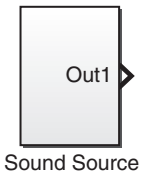

FIGURE 1.5 Creating a subsystem **Sound Source**.

1. **From File**
   - **File name = sound.mat**
2. **Analog Filter Design**
   - **Passband edge frequency[rads/s] = 2*pi*4e3**

Step 6.A-2 Select both blocks. This can be done either by pressing and holding your primary mouse button (typically the left button) while dragging the cursor to box in all components you want to select or by holding down the 'Shift' key while selecting the individual components one by one. To select all components in the design, you can simply use 'Cntrl+A'.

Then right-click one of the selected blocks to activate a pop-up menu and select **'Create Subsystem from Selection'**, or simply press 'Ctrl+G'. Change the default subsystem name, **Subsystem,** into **Sound Source** as shown in Fig. 1.5. Save the current design as **Sound_Source.mdl/slx** in a directory.

Step 6.A-3 Double-click the **Sound Source** block to see the internal design. Capture the internal design window.

6.B Creating the **Spectrum Viewer** subsystem block

Step 6.B-1 Open a new design window and design a new Simulink model as shown in Fig. 1.6. Note that the **Spectrum Analyzer** was named **Spectrum Scope** in earlier versions of Simulink. Be sure to use the **Signal Specification** block in the **Simulink/Signal Attribute** category and use the **Spectrum Analyzer** block in **DSP System Toolbox/Sinks**.

Step 6.B-2 For old Simulink versions that provide **Spectrum Scope**, instead of **Spectrum Analyzer**, set the parameters of **Spectrum Scope** as follows. Do not change any parameters not mentioned here.

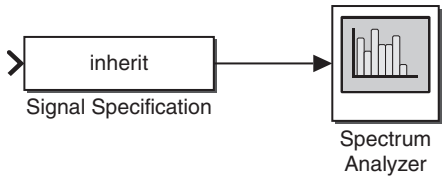

FIGURE 1.6 Design for the subsystem named **Spectrum Viewer**.

1. **Scope Properties** tab
   - **Spectrum Units** = **dBm** (only for the versions that have this parameter)
   - **Buffer input** : Select (check the box)
   - **Buffer size** = 1024
   - **Number of spectral averages** = 200
2. **Axis Properties** tab
   - **Frequency range** = **[-Fs/2** … **Fs/2]** (only for the versions that have this parameter)
   - **Minimum Y-limit** = -40
   - **Maximum Y-limit** = 25

For Simulink versions that provide **Spectrum Analyzer**, instead of **Spectrum Scope**, set the parameters of **Spectrum Analyzer** as follows.

1. Open the **Spectrum Analyzer** display window and browse '**View/Spectrum Settings**' from the menu bar or click the icon named **Spectrum Settings** on the toolbar. Then, set the parameters as shown below. Do not change any other parts not mentioned here.
   - **Main options/Type** = **Power**
   - In **Main options**, change **RBW(Hz),** which is default selection into **Window length** and set **Window length** = 1024.
   - **Windows options/Overlap (%)** = 6.25
   - **Trace options/Units** = dBm
   - **Trace options/Average** = 200
2. Browse **View/Configuration Properties**… from the menu bar or click the icon named **Configuration Properties** on the toolbar. Set the parameters as follows.
   - **Y-limits (Minimum)** = -40
   - **Y-limits (Maximum)** = 25

The details of the parameter settings above have been tested in several Simulink versions. For some other Simulink versions or future versions, you may need to investigate a bit more, but the process will be pretty similar.

Step 6.B-3 Set the parameters of the **Signal Specification** block as follows. Do not change other parameters not mentioned here.

- **Sample time** = 1/16e4

Step 6.B-4 As done in Step 6.A-2, select all and create the subsystem. Change the subsystem name from **Subsystem** into **Spectrum Viewer**. Save the current design as **Spectrum_Viewer.mdl/.slx.**

6.C Testing the subsystems created.

In this section, we observe the output spectrum of the **Sound Source** user-defined block created in 6.A using the **Spectrum Viewer** user-defined block created in 6.B.

6.C-1 Design a new mdl/.slx as shown in Fig. 1.7. To import **Sound Source** and **Spectrum Viewer** to your new design window, open **Sound_Source.mdl/.slx** and

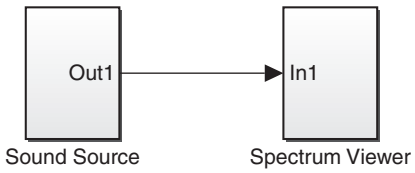

FIGURE 1.7 Design for testing the user-defined blocks **Sound Source** and **Spectrum Viewer**.

Spectrum_Viewer.mdl/.slx that you have saved as mentioned in in 6.A and 6.B and copy and paste them.

Capture the competed design window.

6.C-2 Set **Simulation stop time** to 3 seconds and run the simulation. After simulation is finished, capture the **Spectrum analyzer** display window. Follow the guidelines in the note below for capturing the window.

NOTE: Before capturing the **Spectrum Analyzer** display window, be sure to 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. Also do not **autoscale** or change the axis limits unless you are instructed to do so. Follow this guideline throughout all the problems in this book that require the **Spectrum Analyzer** display window.