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.

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 6 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	9.	plot(x,y2)	17.	plot(x,y1)
2.	y1=sin(x)	10.	y3=exp(-x)	18.	subplot(3,1,2)
3.	y2=cos(x);	11.	plot(x,y3,'r')	19.	plot(x,y2)
4.	plot(x)	12.	legend('sin(x)','cos(x)','exp(-x)')	20.	subplot(3,1,3)
5.	plot(y1)	13.	axis([-5 15 -3 3])	21.	plot(x,y3)
6.	plot(x,y1) %Compare to 5	14.	axis([0 10 -2 2])	22.	semilogy (x,y3)
7.	grid	15.	figure	23.	help plot
8.	hold on	16.	subplot(3,1,1)	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];	7.	C=([1 0 1 1 1]==[1 0 1 0 0])	13.	y=(1<x)&(x<4);
2.	A<3	8.	C=([1 0 1 1 1]~=[1 0 1 0 0])	14.	plot(x,y); axis([0 10 -2 2]); grid;
3.	B=(A>2)	9.	x=0:0.01:10;	15.	y=1<x<4;
4.	C=([1 1 0 0] & [1 1 1 0])	10.	y=(x<3);	16.	plot(x,y); axis([0 10 –2 2]); grid;
5.	C=([1 1 0 0] | [1 1 1 0])	11.	figure		
6.	C=~[1 0 1 0 0]	12.	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(x^2*exp(-x),x,1,3)
>>double(z)
>>limit(sin(t)/t,t,0)
>>symsum(x^2, x,1,4)
>>T=solve(a*x^2+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.

$$\int_{-\infty}^{\infty} e^{-t^2} dz = \sqrt{\pi}, \tag{1.1}$$

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

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

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.

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

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])
```


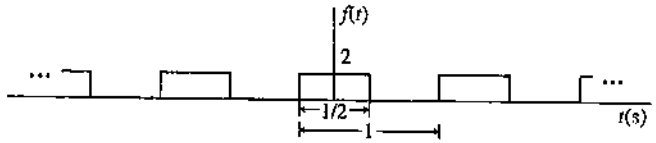

FIGURE 1.1 Periodic function $f(t)$.

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.

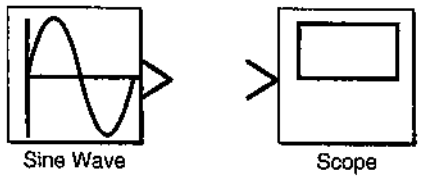

FIGURE 1.2 Adding blocks to a new design.

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.

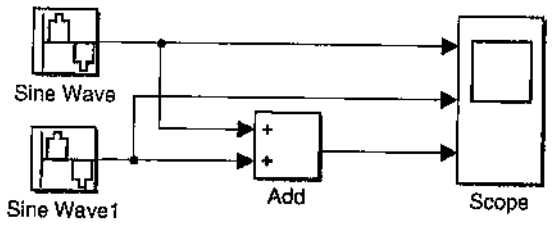

FIGURE 1.3 A test design for sine waveform generation and observation.

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.

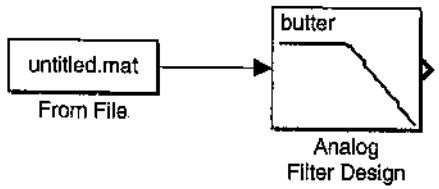

FIGURE 1.4 Design for the subsystem named **Sound Source**.

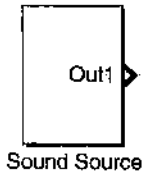

FIGURE 1.5 Creating a subsystem **Sound Source**.

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.

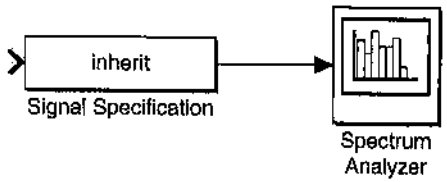

FIGURE 1.6 Design for the subsystem named **Spectrum Viewer**.

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

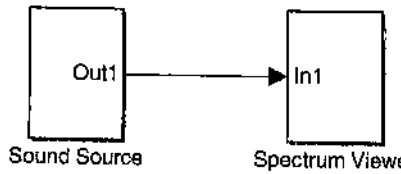

FIGURE 1.7 Design for testing the user-defined blocks **Sound Source** and **Spectrum Viewer**.

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.

[4] S. Haykin and M. Moher, *Introduction to Analog and Digital Communication*, 2nd ed., Hoboken, NJ: Wiley, 2006.

[5] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2004.

[6] G. D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, 1973, pp. 268–278.

25

FADING, DIVERSITY, AND COMBINING

- Derive and simulate the bit error rate (BER) in the Rayleigh fading environment and compare it with the BER in the additive white Gaussian noise environment (AWGN).
- Understand the concept of diversity and the diversity combining.
- Compare the performances of various diversity-combining methods.

25.1 RAYLEIGH FADING CHANNEL MODEL AND THE AVERAGE BER

Consider a two-dimensional modulation such as quadrature amplitude modulation (QAM) and MPSK, and denote s as the coordinate of the modulated symbol in the complex plane. Then, the real and imaginary parts of s correspond to the in-phase and quadrature components, respectively. The symbol energy E_s is equal to $\mathrm{E}[|s|^2]$.

With this signal model, the receiver's matched filter output in a frequency nonselective fading channel can be expressed as

$$r = hs + n, \tag{25.1}$$

where h is the signal scaling coefficient due to fading [1–3], which is often called the "fading coefficient," and n is the AWGN term, which follows the complex Gaussian distribution with zero mean and variance $N_0/2$.

1.A For Rayleigh fading, the fading coefficient h follows a complex Gaussian distribution, expressed as $h = z + jy$, where z and y are real-valued independent

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

Gaussian variables with zero mean and variance 1/2 [3–5]. The command line below generates a sample of h. Complete the quantities marked by '?'.

```
>>h=?*randn + ?*?*?
```

1.B The fading coefficient can be expressed in polar coordinate form as

$$h = |h|e^{j\angle h}. \tag{25.2}$$

1.B-1 [T]Express the fading magnitude square, that is, $|h|^2$, as a function of z and y.

1.B-2 [T]From the result in 1.B-1 and the fact that z and y are independent Gaussian random variables with zero mean and variance 1/2, determine $\mathrm{E}[|h|^2]$.

1.C To create the decision variable D, the receiver derotates the received signal r by using the phase of h as

$$\begin{aligned} D &= re^{-j\angle h} \\ &= (hs + n)e^{-j\angle h} \\ &= |h|e^{j\angle h}e^{-j\angle h}s + ne^{-j\angle h} \\ &= |h|s + ne^{-j\angle h}. \end{aligned} \tag{25.3}$$

As a result, the signal term in the decision variable, $|h|s$, has the same phase as that of the original symbol s.

1.C-1 The noise element $ne^{-j\angle h}$ of the decision variable D is still a complex Gaussian random variable, which has the same distribution as n. Therefore, rotating the phase due to fading does not affect the statistical characteristics of the noise in the decision variable D. On the contrary, fading affects the signal term by a scaling factor $|h|$. This scaling factor is the magnitude of a complex Gaussian random variable, which has the Rayleigh distribution [4,5] given as

$$f_{|h|}(x) = \begin{cases} 2x\exp(-x^2) & x \geq 0 \\ 0 & x < 0. \end{cases} \tag{25.4}$$

Execute the following lines of command to plot equation (25.4). Capture the result.

```
>>x=0:0.01:5;
>>f=2*x.*exp(-x.^2);
>>figure
>>plot(x,f)
```

1.C-2 Define $c \triangleq |h|^2$. Then c is the magnitude square of a complex Gaussian variable and follows the exponential distribution expressed as [4,5]

$$f_c(x) = \exp(-x), \quad x \geq 0. \tag{25.5}$$

Execute the following lines of commands to plot equation (25.5). Capture the result.

```
>>x=0:0.01:5;
>>f_c=exp(-x);
>>figure
>>plot(x,f_c)
```

1.C-3 [T]We can find the expected value of c as $\mathrm{E}[c] = \int_0^\infty (?) \times f_c(x)dx$. Determine the quantity marked by '?'.

1.C-4 (a) [T]Calculate the average of c using the completed equation in C-3. (b) Is the calculated result equal to the answer in 1.B-2? Also use symbolic math (discussed in Section 1.2 of Chapter 1) to verify the result in (a).

1.C-5 [T]In the absence of fading (i.e., $h = 1$), the energy (mean square) of the signal term in the decision variable D, $E[||h|s|^2]$, equals E_s. For Rayleigh fading for which h follows the distribution given by equation (25.4), calculate the energy of the signal term, $E[||h|s|^2]$, and prove that it is still equal to E_s.

1.C-6 [T]The result in 1.C-5 shows that the average symbol energies of the received signals over a Rayleigh fading channel and a Gaussian channel are same. This is based on the assumption that the power of Rayleigh fading channel coefficients is normalized to unity. Provide a discussion on the relative performances (a conjecture on what you expect to see) of the same signaling scheme over a Rayleigh fading channel and over an AWGN channel.

1.C-7 [A]Prove that the noise term $ne^{-j\angle h}$ in the decision variable D given in equation (25.3) is a complex Gaussian random variable and it has the same distribution as n.

1.D In a Rayleigh fading channel, $|h|$ in equation (25.3) is a random variable following the distribution in equation (25.4). The instantaneous symbol energy received over the fading channel is $|h|^2E_s$. Let $c = |h|^2$. Therefore, for BPSK, the instantaneous BER is given as $Q(\sqrt{2cE_b/N_0})$, where $E_b = E$. The average BER can be obtained by taking the expected value of the instantaneous BER over c as

$$\begin{aligned} BER_{\text{fading}} &= \mathrm{E}_c\left[Q\left(\sqrt{\frac{2cE_b}{N_0}}\right)\right] \\ &= \mathrm{E}_c\left[0.5\left\{1 - \mathrm{erf}\left(\sqrt{\frac{cE_b}{N_0}}\right)\right\}\right]. \end{aligned} \tag{25.6}$$

1.D-1 [WWW]The following m-file calculates equation (25.6) using symbolic math. Add a comment to each line to explain what that line does. Capture the completed m-file with the comments.

```
clear
syms c EbN0
instantBER=0.5*(1-erf(sqrt(c*EbN0)));
BER_fading=int(instantBER*exp(-c),c,0,inf);
pretty((BER_fading))
```

1.D-2 The closed-form expression for equation (25.6) is given as

$$P_e = \frac{1}{2}\left(1 - \sqrt{\frac{E_b/N_0}{1+E_b/N_0}}\right). \tag{25.7}$$

Execute the m-file in 1.D-1 and capture the result. Is the symbolic math result the same as equation (25.7)?

1.D-3 Modify the second line of the m-file in D-1 into **'syms c EbN0 positive'**. Here the argument **positive** is not a symbolic variable but a MATLAB argument to specify that the symbolic variables **c** and **EbN0** are positive (and real, of course). Execute the modified m-file and capture the results. Is the symbolic math result the same as equation (25.7) now?

NOTE: If the type of the symbolic variables is not specified, then symbolic math will try to derive a general solution assuming that the symbolic variables are complex. In such a case, it often fails to find the solution.

25.2 BER SIMULATION IN THE RAYLEIGH FADING ENVIRONMENT

2.A [WWW] The following m-file simulates the BER of BPSK over a Rayleigh fading channel.

```
clear
EbN0dB_vector=0:2:20;

Eb=1;

for snr_i=1:length(EbN0dB_vector)
   EbN0dB=EbN0dB_vector(snr_i);
   EbN0=10.^(EbN0dB/10);
   N0=Eb/EbN0;
   sym_cnt=0;
   err_cnt=0;
   while err_cnt<500
      s=sqrt(Eb)*sign(rand-0.5);
      h=sqrt(1/2)*(randn+j*randn);
```



```
      n=sqrt(N0/2)*(randn+j*randn);

      r=?*s + ?;

      D=r*exp(-j*angle(h));
      s_hat=sign(D);
      if ?
         err_cnt=err_cnt+1;
      end
      sym_cnt=sym_cnt+1;
   end
   BER(snr_i)=err_cnt/sym_cnt
end
figure
semilogy(EbN0dB_vector, BER)
xlabel('E_b/N_0 [dB]')
ylabel('BER')
grid
```

2.A-1 (a) Explain why the three parts in bold in the m-file are set as they are. (b) Complete the places marked by **'?'**, and add a comment to each line of the m-file. For lines that involve the operator **'='**, use the comment to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly. Capture the completed m-file.

2.A-2 Explain why the line **'h=sqrt(1/2)*(randn+j*randn);'** should not be placed before the **'while'** loop.

2.A-3 If the line **'h=sqrt(1/2)*(randn+j*randn);'** is placed before the **'while'** loop, what will happen to the resulting BER.

2.B Execute the completed m-file in 2.A and capture the resulting BER graph.

2.C We should find that the BER values in 2.B are all equal to 1. This is because the line **'s_hat=sign(D);'** is incorrect. It should be corrected as **'s_hat = sign(real(D));'**. Explain why it is necessary to take the real part of the decision variable **D** although the signal term of **D** (**sqrt(Eb)** or **-sqrt(Eb)**) is already real-valued.

2.D Modify the line **'s_hat=sign(D);'** into **'s_hat=sign(real(D));'** and execute the m-file. Capture the BER result.

2.E The theoretical BER of BPSK signaling over a Rayleigh fading channel was given in equation (25.7). Execute the following commands in the command window to overlay the theoretical BER on top of the simulated BER curve obtained in 2.D. Capture the resulting figure.

```
>>EbN0_vector=10.^(EbN0dB_vector/10);
>>BER_theory=0.5*(1-sqrt((EbN0_vector)./(1+EbN0_vector)));
>>hold on;
>>semilogy(EbN0dB_vector, BER_theory,'r')
>>legend('Rayleigh fading, Simulation', 'Rayleigh fading, Theory')
```

2.F Do the simulated and theoretical BER values match each other?

2.G In this section we compare the BER performances of BPSK over Rayleigh fading and Gaussian channels.

2.G-1 Rewrite the BER expression of BPSK in an AWGN channel as a function of E_b/N_0.

2.G-2 Execute the commands below to overlay the BPSK BER curve in an AWGN channel on top of the BER curve obtained in 2.D or 2.E. Capture the result.

```
>>BER_AWGN=0.5*erfc(sqrt(EbN0_vector));
>>hold on;
>>semilogy(EbN0dB_vector, BER_AWGN,'g')
>>legend('Rayleigh fading, Sim', 'Rayleigh fading, Theory', 'AWGN')
>>axis([0 20 1e-6 1])
```

2.H Summarize the characteristics of the BER curves of BPSK over Rayleigh fading and Gaussian channels in terms of the changing rate of BER and performance gap as E_b/N_0 increases and decreases.

2.I Are the observations made in 2.H from simulation consistent with what was discussed in 1.C-6?

2.J [A]Next we modify the m-file in 2.D to simulate the BER performance of QPSK over a Rayleigh fading channel. The modified m-file is given below.

2.J-1 The parts in bold are the modified parts from the previous m-file for BPSK BER simulation. Add a comment to each of these lines to justify the modifications made. Capture the completed m-file.

```
clear
EbN0dB_vector=0:2:20;
Eb=1;
for snr_i=1:length(EbN0dB_vector)
   EbN0dB=EbN0dB_vector(snr_i);
   EbN0=10.^(EbN0dB/10);
   N0=Eb/EbN0;
   sym_cnt=0;
   err_cnt=0;
   while err_cnt<500
      s=sqrt(Eb)*sign(rand-0.5)+j*sqrt(Eb)*sign(rand-0.5);
      h=sqrt(1/2)*(randn+j*randn);
      n=sqrt(N0/2)*(randn+j*randn);
      r=h*s+n;
      D=r*exp(-j*angle(h));
      s_hat=sign(real(D));
      s_hat2=sign(imag(D));
      if sign(real(s)) ~= s_hat
         err_cnt=err_cnt+1;
      end
      if sign(imag(s)) ~= s_hat2
         err_cnt=err_cnt+1;
      end
      sym_cnt=sym_cnt+2;
   end
   BER(snr_i)=err_cnt/sym_cnt
end
figure
semilogy(EbN0dB_vector, BER)
xlabel('E_b/N_0 [dB]')
ylabel('BER')
grid
```


2.J-2 Execute the m-file in 2.J-1 and capture the simulated BER plot.

2.J-3 Compare the simulated BER curve of QPSK with the BER curve of BPSK obtained in 2.E. They should be identical (ignore the small simulation errors). Justify why they should be the same.

25.3 DIVERSITY

In 1.D we defined the variable c, which determines the instantaneous symbol energy in a Rayleigh fading channel. From the distribution of c, we can easily explain why the BER performance of BPSK in a Rayleigh fading channel is significantly worse than that in an AWGN channel (no fading). The instantaneous symbol energy cE_b could change from 0 to infinity due to fading, and the instantaneous BER given as $Q(\sqrt{2cE_b/N_0})$ almost exponentially decreases as c increases. If c becomes larger than 1, then the BER is lower than the BER over an AWGN channel. However, the error rate will be dominated by the bit errors when c is smaller than 1. For a mathematical proof, refer to Jensen's inequality [4, 5], which is covered in most of the existing textbooks.

If there is a method that will drastically reduce the probability of encountering a small instantaneous symbol energy, then the BER performance will improve significantly. One such method is the diversity technique [5]. Diversity here refers to the mechanism that multiple independently faded copies of the same signal are available for use in the detection process. Common diversity methods include spatial diversity (refer to Chapter 28), temporal/time diversity, frequency diversity, and multipath diversity. Spatial diversity could be achieved by transmitting the same data through multiple transmit antennas or receiving the same data through multiple receive antennas. For such schemes to be effective, the transmit antennas or the receive antennas must operate independently or at least have sufficiently low correlations. Multipath diversity is unique [5]: if multiple received signal paths are resolvable, then this case may be considered a form of time diversity; however, the resolvable paths are closely related to the fact that the channel is frequency selective, that is, the different frequency components of the desired signal are faded differently (refer to Chapter 27).

In this section we consider the scenario that L independently received copies, $r(1), r(2), \ldots, r(L)$, all carrying the same transmitted signal s, are available for use in the detection process. These copies could be exploited to achieve a maximum diversity order of L. The received signal copies are expressed as

$$r(1) = h(1)s + n(1),\ r(2) = h(2)s + n(2), \ldots,\ r(L) = h(L)s + n(L), \quad (25.8)$$

where $h(1), h(2), \ldots, h(L)$ are assumed to be independent and identically distributed (i.i.d.), all having the same distribution as h created in 1.A; the noise terms $n(1), n(2), \ldots, n(L)$ are also i.i.d. with the same distribution as n in equation (25.1), and the transmitted signal s is a BPSK signal, taking on the values of $\sqrt{E_s/L}$ or $-\sqrt{E_s/L}$ with equal probability.

3.A [T]Assuming a noiseless and the nonfading condition ($h(1) = \ldots = h(L) = 1$ and $n(1) = \ldots = n(L) = 0$), prove that the total received symbol energy from all L branches, that is, $|r(1)|^2 + |r(2)|^2 + \ldots + |r(L)|^2$, equals E_s.

25.4 COMBINING METHODS

This section investigates techniques for combining the L branches of the received signals to form the decision variable [5,6]. Three commonly used combining methods are as follows:

1. Selection diversity combining (SDC)
2. Equal gain combining (EGC)
3. Maximum ratio combining (MRC)

The fading coefficients $h(1), h(2), \ldots, h(L)$ are assumed to be known in the receiver. In practice, most communications channels can be classified as "slow" fading channels, for which the fading coefficients will be nearly constant over many symbol periods. This allows the receiver to estimate the fading coefficients.

4.A Selection diversity combining.

In SDC, the branch with the largest fading coefficient is selected for detection and the rest are not used. The decision variable D is generated as

$$\begin{aligned} &\text{Step 1. Set } k_{\text{best}} = \arg\max_k |h(k)|, \\ &\text{Step 2. Set } D = r(k_{\text{best}})e^{-j\angle h(k_{\text{best}})}. \end{aligned} \quad (25.9)$$

4.A-1 [WWW]The following m-file simulates the BER of BPSK over a Rayleigh fading channel with three SDC branches, that is, $L = 3$. Complete the places marked by '?'. Add a comment to each of the lines in bold to explain what the line does. Especially for the lines with '=', explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

Capture the completed m-file.

```
clear
EbN0dB_vector=0:3:15;
Eb=1;
L=3;
for snr_i=1:length(EbN0dB_vector)
  EbN0dB=EbN0dB_vector(snr_i);
  EbN0=10.^(EbN0dB/10);
  N0=Eb/EbN0;
  sym_cnt=0;
  err_cnt=0;
  while err_cnt<100 % If you increase err_cnt (currently 100), the accuracy increases
and time also increase.
    b=sign(rand-0.5); %BPSK symbol {1,-1}
    s=sqrt(Eb/L)*b;
    for k=1:L
      h(k)=sqrt(1/2)*(randn+j*randn);
      n(k)=sqrt(N0/2)*(randn+j*randn);
      r(k)=?*s+?;
    end

    [T1 T2]=max(?); % Refer to (25.9). To see how to use max( ), execute '>>help
max' in the command window.
    D=r(?)*exp(-j*angle(h(?))); %Refer to (25.9).

    b_hat=sign(real(D));
    If b_hat~=b;
      err_cnt=err_cnt+1;
    end
    sym_cnt=sym_cnt+1;
```

```
    end
    BER(snr_i)=err_cnt/sym_cnt
end
figure
semilogy(EbN0dB_vector, BER)
xlabel('E_b/N_0 [dB]')
ylabel('BER')
grid
```

4.A-2 Execute the completed m-file and capture the simulated BER.

4.A-3 Execute the m-file separately for each of two other cases: $L = 1$ and 5. Then plot the three BER curves in a single figure. Methods to overlay curves from different figures in a single plot were discussed in Section 4.C-1 of Chapter 24. Finally, execute **legend('L=1','L=3','L=5')** in the command window and capture the resulting plot. Save this figure in .fig format and name it **Ch25_4A_3.fig**, as it will be needed later in this chapter.

4.A-4 (a) Analyze how the slope of the BER curves changes as the diversity order L increases, and intuitively explain the reason that causes such characteristics. (b) Explain why the BER becomes slightly worse in the low-SNR region as the diversity order L increases.

4.B Equal gain combining.

In EGC, $r(1), r(2), \ldots, r(L)$ are combined with the same weight regardless of the magnitudes of the L instantaneous fading coefficients. For coherent combining, the phase rotation due to fading at each branch is compensated first and the decision variable D is written as

$$D = \sum_{k=1}^{L} r(k)e^{-j\angle h(k)}. \tag{25.10}$$

4.B-1 In the m-file completed in 4.A-1, modify the right-hand side of the line **'D=r(?)*exp(-j*angle(h(?)));'** into **'D=sum(r.*exp(-j*angle(h)));'** and complete this line properly to implement the right-hand side of equation (25.10). Capture the modified m-file.

4.B-2 In addition, modify the line **'EbN0dB_vector=0:3:15'** into **'EbN0dB_vector=0:3:12'**. Execute the modified m-file for each of the following cases: $L = 1$, $L = 3$, and $L = 5$. Then overlay the three BER curves in a single figure. After this, execute **legend('L=1', 'L=3', 'L=5')** in the command window and capture the figure. Save this figure in .fig format and name it **Ch25_4B_2.fig**, as it will be needed later in this chapter.

4.B-3 Analyze how the BER and the slope of the BER curves change as the diversity order L changes.

4.B-4 Compare the BER results with EGC in 4.B-2 and with SDC in 4.A-3. This can be done best by overlaying all six curves in a single figure. Focus on the relative BER values and the slopes of the BER curves of the same diversity order obtained by using the two combining techniques.

4.C Maximum ratio combining.

Maximum ratio combining is similar to EGC in the sense that all branches are combined in both schemes. Unlike EGC, MRC employs different combining weights that are proportional to the fading magnitude of each branch. The decision variable is expressed as

$$D = \sum_{k=1}^{L} |h(k)|r(k)e^{-j\angle h(k)}. \tag{25.11}$$

4.C-1 [T]Show that equation (25.11) can be simplified as

$$D = \sum_{k=1}^{L} h^*(k)r(k). \tag{25.12}$$

4.C-2 In the m-file completed in 4.A-1, modify the right-hand side of the line **'D=r(?)*exp(-j*angle(h(?)));'** into **'D=sum(conj(?).*?);'**, and complete this line properly to implement the right-hand side of equation (25.12). Capture the completed line.

4.C-3 In addition, modify the line **'EbN0dB_vector=0:3:15'** into **'EbN0dB_vector=0:3:12'** as done in 4.B-2. Execute the modified m-file for each of the following cases: $L = 1$, $L = 3$, and $L = 5$. Then overlay the three BER curves in a single figure. After this, execute **legend('L=1', 'L=3', 'L=5')** in the command window and capture the resulting figure. Save this figure in .fig format and name it **Ch25_4C_3.fig**, which will be needed later in this chapter.

4.C-4 Analyze how the BER and the slope of the BER curves change as the diversity order L changes.

4.C-5 Compare the BER results with MRC in 4.C-3 and with EGC in 4.B-2. As in 4.B-4, this can be done best by overlaying all six curves in a single figure. Focus on the relative BER values and the slopes of the BER curves of the same diversity order obtained by using the two combining techniques.

4.C-6 [A,T]Prove that MRC maximizes the SNR among all three combining methods using the Cauchy Schwarz inequality [7].

4.D Now we compare the performances all three combining methods.

4.D-1 Open the figure files (.fig files) saved in 4.A-3, 4.B-2, and 4.C-3 if you have closed these figures. Overlay all of the nine BER curves in one figure. Change the

line color of the BER curves for SDC to blue, for EGC to green, and for MRC to red. Add a legend for all nine curves. Then capture the completed figure.

(a) For the same diversity order L, which combining scheme performs the best?
(b) For $L = 5$, analyze the BER gaps among the three schemes as SNR changes and summarize the observations.

4.D-2 Based on the decision variables in equations (25.9), (25.10), and (25.12), which combining method has the lowest implementation complexity and which one is most difficult to implement? Why?

4.E In this problem we investigate the diversity gain.

4.E-1 Summarize the common trend in the change of BER values with the three methods as L increases and explain the reason that causes this trend.

4.E-2 Revisit the m-file completed in 4.C for MRC. Set **L=5** and modify the '**for**' loop inside the '**while**' loop as shown below. The distribution of **h(k)** after this modification remains the same as that in the original version. Explain why.

```
hk=sqrt(1/2)*(randn+j*randn);
for k=1:L
   h(k)=hk;
   n(k)=sqrt(N0/2)*(randn+j*randn);
   r(k)=?*s+?;
end
```

4.E-3 Execute the modified m-file and capture the BER graph.

4.E-4 (a) Compare the BER curve in 4.E-3 with the BER curve obtained from the original m-file. (b) Observe that these two BER curves are significantly different. Justify it; that is, explain why the modified m-file cannot achieve any diversity gain although the distribution of **h(k)** remains the same as before.

4.E-5 Based on the results in 4.E-3 and 4.E-4, determine the conditions on the statistical properties of the fading coefficients **h(1)**, **h(2)**, ..., **h(L)**, so that the diversity gain is maximized and the BER is minimized accordingly.

REFERENCES

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.

[2] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge, UK: Cambridge University Press, 2005.

[3] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems Part I: Characterization," *IEEE Communications Magazine*, Vol. 35, No. 7, 1997, pp. 90–100.

[4] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill, 1965.

[5] J. G. Proakis, *Digital Communications*, 5th ed., New York: McGraw-Hill, 2008.

[6] D. G. Brennan, "Linear Diversity Combining Techniques," *Proceedings IRE.*, Vol. 47, 1959, pp. 1075–1102.

[7] G. Strang, *Linear Algebra and Its Applications*, 4th ed., Belmont, CA: Brooks Cole, 2005.

28

MIMO SYSTEM—PART I: SPACE TIME CODE

- Investigate the structure of Alamouti code, one of the most commonly used space time codes.
- Implement Alamouti code and verify its performance through the simulation.
- Analyze the rates and the diversity orders of various space time block codes.

28.1 SYSTEM MODEL

1.A [WWW] Received signal model for the single transmit antenna system.

The m-file below simulates the received signal in a slow and frequency flat Rayleigh fading channel. For the last five lines in bold, explain what the variable on the left-hand side represents and justify how the right-hand side expression is formulated accordingly.

```
clear
EsN0= ; %Es/N0; set it to a desired value
s= ;
% Create the complex symbols according to the modulation method, for example
% in case of QPSK,  s=sign(rand-0.5)+j*sign(rand-0.5), and in case of BPSK,
s=sign(rand-0.5);
```

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


```
Es=abs(s).^2;
N0=Es/EsN0;
h=sqrt(1/2)*(randn+j*randn);
n=sqrt(N0/2)*(randn+j*randn);
r=h*s+n;
```

NOTE: If a BER simulation includes creating the received waveform and calculating the decision variable from the received waveform as done in Chapters 21, 22, 23, 26, and 27, then we call it "waveform level BER simulation." On the contrary, if a BER simulation starts with the decision variable model and skips the process of generating the received signal waveform as done in Chapters 24, 25, and this chapter, then we call it "decision variable level simulation."

1.B [T] System with two transmit antennas.

The complex Gaussian variable **h** in the m-file in 1.A is the fading gain. As discussed in Section 25.1 of Chapter 25, **abs(h)** has the Rayleigh distribution. Also the multiplicative distortion—the desired signal is multiplied by **h** at the receiver—as implemented in the last line of the m-file above degrades the BER performance significantly. The discussion in Chapter 25 has shown that spatial diversity is a very effective way to mitigate the effects of fading. However, implementing multiple antennas at both sides of a communications link is not necessarily feasible for many applications. If one side of the link must be simple and have only one antenna (e.g., the mobile side), but the other side can have multiple antennas, how to realize spatial diversity at the one-antenna terminal?

Let us consider a system where the transmitter has two antennas, one named "antenna A" and the other "antenna B," and the receiver has only one antenna. The goal is to design a scheme to provide spatial diversity at the receiver. Let **hA** and **hB** be two independent complex Gaussian random variables that denote the Rayleigh fading coefficients from antenna A and antenna B to the receive antenna, respectively.

There are a number of ways to use the two transmit antennas. Next we consider two simple methods.

1.B-1 The first scheme is described in Table 28.1, where T1, T2, ... denote the transmission time slots and **s1**, **s2**, **s3**, ... are the data symbols to be transmitted. The transmission delay from the transmitter to the receiver is neglected without affecting the description of the transmission and reception processes.

TABLE 28.1 Space Time Symbol Mapping Method I.

	Time				
Space	T1	T2	T3	T4	...
Antenna A	**s1**	**s2**	**s3**	**s4**	...
Antenna B	**s1**	**s2**	**s3**	**s4**	...

At T1, the received signal is the sum of **hA*s1** and **hB*s1**.

(a) The m-file below simulates the received signal for the method described in Table 28.1. Justify the use of the scaling factor 2 in the fourth line **'Es=2*abs(s1)^2;'** that calculates the transmitted energy per symbol.

```
clear
EsN0= ; %Es/N0, set to a desired value
s1= ;  % Create any symbol randomly according to the modulation method
Es=2*abs(s1)^2; N0=Es/EsN0;

hA=sqrt(1/2)*(randn+j*randn);
hB=sqrt(1/2)*(randn+j*randn);
n=sqrt(N0/2)*(randn+j*randn);
sA=s1; % antenna A transmit signal
sB=s1; % antenna B transmit signal
r=hA*sA+hB*sB+n;
```

(b) The BER performance cannot be improved by using the method in Table 28.1 compared with the single antenna system discussed in 1.A. Prove that the diversity order of this scheme is still 1 even though both transmit antennas are used in the transmission process.

HINT: You may rewrite the received signal as **'r=(hA+hB)/sqrt(2)*s +n'**, and then determine the distribution of the effective fading coefficient **'(hA+hB)/sqrt(2)'**.

1.B-2 The diversity gain can be realized only if multiple independent observations of the same transmitted signals are available at the receiver, like the receive diversity case described by equation (25.8) in Chapter 25. The second scheme is described in Table 28.2.

(a) The code fragment below generates the received signal for the method described in Table 28.2. Since the same transmitted data symbol is received independently twice over two adjacent time slots, one from antenna A and one from antenna B, these two observations of the same data symbol should be combined before detection. Complete the last line of the code fragment to form the decision variable by employing MRC (which was discussed in Chapter 25).

TABLE 28.2 Space Time Symbol Mapping Method II.

	Time				
Space	T1	T2	T3	T4	...
Antenna A	**s1**	No signal	**s2**	No signal	...
Antenna B	No signal	**s1**	No signal	**s2**	...


```
.
.
.
hA=sqrt(1/2)*(randn+j*randn);
hB=sqrt(1/2)*(randn+j*randn);
n1=sqrt(N0/2)*(randn+j*randn); %received noise  at T1
n2=sqrt(N0/2)*(randn+j*randn); %received noise  at T2
r1=hA*s1+n1; % received signal at T1
r2=hB*s1+n2; % received signal at T2
z=conj(?)*r1+conj(?)*r2; % decision variable after MRC.
```

(b) Substitute the expressions of **r1** and **r2** into the expression of **z** and show that the signal scaling factor in the decision variable **z** is **abs(hA)^2 +abs(hB)^2**.

(c) Also show that the noise term in **z** is **conj(hA)*n1+conj(hB)*n2** and that it has a zero mean and instantaneous variance equal to **(abs(hA)^2+abs(hB)^2)*N0**.

(d) From (b) and (c), show that the instantaneous E_b/N_0 with the decision variable **z** obtained from MRC is **(abs(hA)^2+abs(hB)^2)*EbN0**.

(e) From the instantaneous E_b/N_0, show that the method in Table 28.2 achieves a diversity order of 2.

(f) However, the method in Table 28.2 has a major disadvantage compared with the single transmit antenna system discussed in 1.A in terms of bandwidth efficiency. Explain why.

28.2 ALAMOUTI CODE

Alamouti code [1] is the simplest and most widely used space time block code (STBC) [1–5]. This method transmits the data symbol pairs as shown in Table 28.3. For proper operation of Alamouti code, the fading coefficients **hA** and **hB** of the two transmission antennas should remain the same over at least two consecutive time slots.

2.A Let **s1** and **n1** denote, respectively, the signal and the noise terms received at the first time slot T1, and let **s2** and **n2** denote, respectively, the signal and the noise terms received at the second time slots T2. Then, **r1** and **r2** can be created as shown below.

TABLE 28.3 Space Time Symbol Mapping of Alamouti Code.

	Time						
Space	T1	T2	T3	T4	T5	T6	...
Antenna A	**s1**	**conj(s2)**	**s3**	**conj(s4)**	**s5**	**conj(s6)**	...
Antenna B	**s2**	**-conj(s1)**	**s4**	**-conj(s3)**	**s6**	**-conj(s5)**	...

```
s1= ;
s2= ;
.
.
.
hA=sqrt(1/2)*(randn+j*randn);
hB=sqrt(1/2)*(randn+j*randn);
n1=sqrt(N0/2)*(randn+j*randn); %Received noise at T1.
n2=sqrt(N0/2)*(randn+j*randn); %Received noise at T2.
r1=hA*s1+hB*s2+n1; % received signal at T1
r2= ? ; % received signal at T2
```

Complete the last line to create **r2** and capture the competed line.

2.B [A]The receiver is assumed to have perfect channel state information (CSI) (the channel gains **hA** and **hB**). The data symbol pair **s1** and **s2** are estimated by using **r1**, **r2**, **hA**, and **hB**. Next we decode Alamouti code using MLD on the basis of exhaustive search implemented through the following steps:

Step 1. Create a data symbol pair (**s1**, **s2**) and the fading gains (**hA**, **hB**) from the two transmit antennas to the receive antenna.

Step 2. Perform Alamouti coding on the data symbol pair (**s1**, **s2**) and create the received signals **r1** and **r2**.

Step 3. Prepare all possible data symbol pairs (candidate pairs) of (**s1**, **s2**) for exhaustive MLD. For example, for binary phase shift keying (BPSK), the pairs are {(1,1), (1,-1), (-1,1), (-1,1)}.

Step 4. For each candidate pair, perform Step 4-1 and Step 4-2.

Step 4-1. Perform Alamouti coding on the current candidate data symbol pair and construct the virtual received signal without noise.

Step 4-2. Calculate the Euclidean distance between the received signal (**r1**, **r2**) in Step 2 and the virtual received signal constructed in Step 4-1.

Step 5. Among all candidate data symbol pairs, select the candidate pair that has the smallest Euclidean distance with the estimated data symbol pair.

2.B-1 Review the MLD concept and justify why Step 4 and Step 5 above implement the MLD process.

2.B-2 [WWW]The m-file below simulates the BER of a BPSK system with Alamouti STBC.

(a) Identify the code lines that correspond to each of the steps described above.

Complete the places marked by '?'. For each of the lines in bold, add a comment to explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly. Capture the completed m-file.


```
clear
EbN0dB_vector=[0 3 6 9 12 15]; % Set EbN0 values in dB.
Eb=2; %Total bit energy for two time slots T1 and T2. Refer to Table 28.3.

for snri=1:length(EbN0dB_vector)
  EbN0dB=EbN0dB_vector(snri);
  EbN0=10^(EbN0dB/10);
  N0=Eb/EbN0;
  errcnt=0; symcnt=0;
  while errcnt<500 %Decrease errcnt limit(=500) if the simulation takes long time but
the result will be less accurate.

    s1=sign(rand-0.5);
    s2=sign(rand-0.5);

    %%%Creation of fading coefficients and noise %%%%%%%%%%%
    hA=sqrt(1/2)*(randn+j*randn);
    hB=sqrt(1/2)*(randn+j*randn);
    n1=sqrt(N0/2)*(randn+j*randn);  %Received noise at T1.
    n2=sqrt(N0/2)*(randn+j*randn);  %Received noise at T2.
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%Generating Alamouti coded received signal%%%%%%%%%
    r1=hA*s1+hB*s2+n1;  %received signal at T1.
    r2=?;          %received signal at T2. Refer to Table 28.3
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%ML decoding %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    s1s2pairCandidate=[[1,1];[1,-1];[-1,1];[-1,-1]];
    N_set=4; %Number of candidates
    for k_set=1:N_set
      s1candidate=s1s2pairCandidate(k_set,1);
      s2candidate=s1s2pairCandidate(k_set,?);
      r1candidate=hA*s1candidate+hB*s2candidate;
    %Create a virtual r1 assuming that (s1, s2)=(s1candidate, s2candidate) and there
is no noise.
      r2candidate=? ;

      distance(k_set)=sum(abs([r1, r2]-[r1candidate,r2candidate]).^2);
    end
    [A B]= min( ? );
    s1_hat=s1s2pairCandidate(B,1);
    s2_hat=s1s2pairCandidate(B,2);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    if(s1_hat~=s1)
      errcnt=errcnt+1;
    end
    if(s2_hat~=s2)
      errcnt=errcnt+1;
    end
    symcnt=symcnt + 2;
  end

  BER(snri)=errcnt/symcnt;
  save alamouti_ML_BER.mat EbN0dB_vector BER
end
figure
semilogy( EbN0dB_vector, BER);
title('Alamouti coded, BPSK, Rayleigh fading');
xlabel('Eb/N0 [dB]');ylabel('BER');
grid on
```

2.B-3 Explain why the part **'creation of the fading coefficients and the noise'** should not be placed outside of the **'while'** loop.

2.B-4 In an actual system, the streams of symbol pairs are continuously transmitted as illustrated in Table 28.3. However, in the m-file of 2.B-2, a loop is used to repeatedly transmit one symbol pair (**s1**, **s2**) at a time. Justify why we do not need to explicitly simulate other symbol pairs (**s3**, **s4**), (**s5**, **s6**), and so on.

2.B-5 Execute the completed m-file and capture the simulated BER plot.

2.B-6 Recall that the m-file completed in Section 4.C of Chapter 25 simulates the BER of MRC in Rayleigh fading channels. Open that m-file and set **L = 2**. Execute the m-file. Overlay the simulated BER of MRC with **L** = 2 on the BER graph in 2.B-5. (The process to copy curves displayed in different figures generated in MATLAB and overlay them in one figure was discussed in 4.C-1 of Chapter 24.)

(a) Capture the resulting BER graph.
(b) Compare the two BER curves and check whether or not the BER of Alamouti code with MLD is equal to that of two-branch (**L**=2) receive diversity with MRC.

28.3 SIMPLE DETECTION OF ALAMOUTI CODE

3.A One of the excellent features of Alamouti code is that detection based on a decision variable obtained by linearly combining the received signal pair (**r1**, **r2**) achieves the MLD performance. Let **z1** and **z2** denote the decision variables for

s1 and **s2**, respectively. They can be created by linearly combining **r1** and **r2** [1].

```
z1=conj(hA)*r1 - hB* conj(r2);
z2=conj(hB)*r1 + hA* conj(r2);
```

3.A-1 [T]Substitute the expressions for **r1** and **r2** completed in the m-file in 2.A into the expressions of **z1** and **z2** above.

(a) Rearrange the two expressions so that **s2** does not appear in the expression **z1** and **s1** does not appear in the expression **z2**.
(b) Show that the scaling terms for **s1** in **z1** and for **s2** in **z2** are both equal to **'(abs(hA)^2+abs(hB)^2)'**.
(c) Show that the noise terms in **z1** and **z2** have a zero mean and variance equal to **'(abs(hA)^2+abs(hB)^2)*N0'**.
(d) From (b) and (c), show that the instantaneous E_b/N_0 values in **z1** and **z2** are **'(abs(hA)^2+abs(hB)^2)*EbN0'**.
(e) Based on the instantaneous E_b/N_0 for **z1** and **z2**, show that the linear decoding process given above achieves the same performance as MRC with a diversity order **L** = 2.
(f) In 2.B-6(b), we showed that Alamouti code with MLD achieves the same performance as MRC with **L** = 2. From the results so far, does detection based on the decision variables obtained by linearly combining the received signals achieve the MLD performance? Justify your answer.

3.A-2 [T]Compare the instantaneous E_b/N_0 values of Alamouti code and the method described in Table 28.2, which were derived in 1.B-2(b)–1.B-2(e).

(a) Based on this comparison, assess the relative error performances of these two schemes.
(b) An Alamouti code is bandwidth more efficient than the scheme described in Table 28.2. Derive the relative bandwidth efficiency of the two schemes.

3.B [WWW]The m-file below simulates the BER with the simple linear combining method introduced in 3.A.

```
clear
EbN0dB_vector=[0 3 6 9 12 15]; % Set EbN0 values in dB.
Eb=2; %Total bit energy for two time slots T1 and T2. Refer to Table 28.3.

for snri=1:length(EbN0dB_vector)
  EbN0dB=EbN0dB_vector(snri);
  EbN0=10^(EbN0dB/10);
  N0=Eb/EbN0;
  errcnt=0; symcnt=0;
```

```
   while errcnt<500 %Decrease errcnt limit(=500) if the simulation takes long time but
the result will be less accurate.
      s1=sign(rand-0.5);
      s2=sign(rand-0.5);

      %%%Generating fading coefficient and noise %%%%%%%%%%%%
      hA=sqrt(1/2)*(randn+j*randn);
      hB=sqrt(1/2)*(randn+j*randn);
      n1=sqrt(N0/2)*(randn+j*randn);  %Received noise at T1.
      n2=sqrt(N0/2)*(randn+j*randn);  %Received noise at T2.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %%%%%Generating Alamouti coded received signal%%%%%%%%%%
      r1=hA*s1+hB*s2+n1;  %received signal at T1.
      r2=?;    %received signal at T2. Refer to Table 28.3
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %%%%%%Simple linear combining for decoding!!!!%%%%%%%%
      z1= ?; % Decision variable for s1
      z2= ?; % Decision variable for s2
      s1_hat=sign(real(z1));
      s2_hat=sign(real(z2));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      if(s1_hat~=s1)
         errcnt=errcnt+1;
      end
      if(s2_hat~=s2)
         errcnt=errcnt+1;
      end
      symcnt=symcnt + 2;
   end

   BER(snri)=errcnt/symcnt;
   save alamouti_BER.mat EbN0dB_vector BER
end
figure
semilogy( EbN0dB_vector, BER);
title('Alamouti coded, BPSK, Rayleigh fading');
xlabel('Eb/N0 [dB]');ylabel('BER');
grid on
```

3.B-1 Complete the places marked by '**?**' and capture the completed m-file.

3.B-2 Compare the computational complexities of the exhaustive MLD implemented in the m-file of 2.B-2 and the simple linear combining method.

3.B-3 Execute the m-file and capture the simulated BER graph. Do not close the figure since it will be needed in 3.B-4.

3.B-4 Execute the m-file completed in 2.B-2 to generate the file **alamouti_ML_BER.mat** in the MATLAB work folder. After this, execute the three lines of code below to plot the BER of exhaustive MLD together with the BER of the simple linear combining scheme.

(a) Capture the resulting BER graph with the two BER curves.
(b) From the two BER curves, summarize the BER performances of MLD and the simple linear combining schemes for Alamouti code.

```
>>load alamouti_ML_BER.mat
>>hold on
>>semilogy( EbN0dB_vector, BER,'r')
```

3.B-5 Modify the m-file in 3.B-1 to extend the modulation method from BPSK to QPSK. (a) Capture the modified part. (b) Execute the modified m-file and capture the simulated BER graph.

3.B-6 [T]Other than the space time mapping given in Table 28.3, there are other mapping schemes that also maintain the Alamouti code properties.

(a) Design one or more of such mapping schemes and record them using the format shown in Table 28.4. Do not consider trivial variations to the mapping in Table 28.3 such as interchange the positions of **s1** and **s2** in time or in space.
(b) For each of the schemes designed in (a), derive its corresponding linear combining rule for the decision variables **z1** and **z2** and compare them with those for Alamouti code.

TABLE 28.4 Different Version of Alamouti Code.

	Time	
Space	**T1**	T2
Antenna A	?	?
Antenna B	?	?

28.4 [A]VARIOUS STBCs, THEIR DIVERSITY ORDERS, AND THEIR RATES

4.A [T]Alamouti code is a two-dimensional (space time) code. The received signal vector can be written in vector-matrix form as [1]

$$\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} h_A \\ h_B \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}. \tag{28.1}$$

4.A-1 Complete the space time code matrix in equation (28.1) so that the expressions of **r1** and **r2** are the same as those implemented in the last two lines of the code fragment in 2.A.

4.A-2 Do the columns of the space time code matrix correspond to different time slots or different space elements (antennas)?

4.A-3 Define the rate R of the space time code as [2–5]

$$R = \frac{\text{Number of different symbols composing the space time code matrix}}{\text{Number of time slots used for transmitting one space time code}\\ (= \text{Number of rows in the space time code matrix})}. \tag{28.2}$$

For the single transmit-antenna system, this rate equals 1. Determine the R of Alamouti code.

4.B [T]Now consider an STBC with the following received signal vector:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & s_3 \\ -s_2^* & s_1^* & 0 \\ s_3^* & 0 & -s_1^* \\ 0 & s_3^* & -s_2^* \end{bmatrix} \begin{bmatrix} h_A \\ h_B \\ h_C \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}. \tag{28.3}$$

4.B-1 Determine the number of transmission antennas and the number of time slots in a block.

4.B-2 Determine the rate R of this code.

4.B-3 Determine the diversity order of this code.

4.C [A,WWW]Modify the m-file in 2.B-2 to simulate the BER of the STBC given in equation (28.3) by using exhaustive MLD.

4.C-1 Capture the modified m-file and the simulated BER plot.

4.C-2 Recall that the m-file completed in Section 4.C of Chapter 25 simulates the BER of MRC in Rayleigh fading channels. Open that m-file and set **L** = 3. Execute the m-file. Overlay the simulated BER curve of MRC with **L** = 3 on the BER graph generated in 4.C-1.

(a) Capture the resulting BER graph with both BER curves.

(b) Compare the two BER curves, one for the code expressed in equation (28.3) with MLD and one for MRC with L = 3, and summarize the relative BER performance of these two systems. Then check whether your answer to 4.B-3 is correct.

REFERENCES

[1] S. M. Alamouti, "A Simple Transmit Diversity Technique for Wireless Communications," *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 8, 1998, pp. 1451–1458.

[2] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space–Time codes for High Data Rate Wireless Communication: Performance Analysis and Code Construction," *IEEE Transactions on Information Theory*, Vol. 44, No. 2, 1998, pp. 744–765.

[3] V. Tarokh, A. Naguib, N. Seshadri, and A. R. Calderbank, "Space-Time Codes for High Data Rate Wireless Communication: Performance Criteria in the Presence of Channel Estimation Errors, Mobility and Multiple Paths," *IEEE Transactions Communications*, Vol. 47, 1999, pp. 199–207.

[4] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space–Time Block Coding for Wireless Communications: Performance Results," *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 3, 1999, pp. 451–460.

[5] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space–Time Block Codes from Orthogonal Designs," *IEEE Transactions on Information Theory*, Vol. 45, No. 5, 1999, pp. 744–765.

29

MIMO SYSTEM—PART II: SPATIAL MULTIPLEXING

- Implement detection processes for spatial multiplexing MIMO systems.
- Compare the performances of MIMO system with various detection methods.
- Investigate the performances of MIMO system with different system parameters.

29.1 MIMO FOR SPATIAL MULTIPLEXING

1.A System model.

MIMO systems may be broadly classified into two categories: spatial diversity (SD) systems and spatial multiplexing (SM) systems [1–3]. SD MIMO systems achieve diversity. Some simple SD MIMO systems were introduced in Chapter 28. SM MIMO systems exploit the multiple transmit and receive antennas to transmit multiple data streams simultaneously in the same frequency band to increase the spectral efficiency. Some of the existing literature defines only SM MIMO systems as MIMO systems.

In this chapter we study SM MIMO systems. Since the multiple streams of data are transmitted simultaneously in the same frequency band, there exists mutual interference among these data streams.

Let N_T and N_R denote the number of transmit (TX) and receive (RX) antennas, respectively. The signals received by the N_R RX antennas, written as an $N_R \times 1$ column vector $\mathbf{r}$, can be expressed as

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{n}, \tag{29.1}$$

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

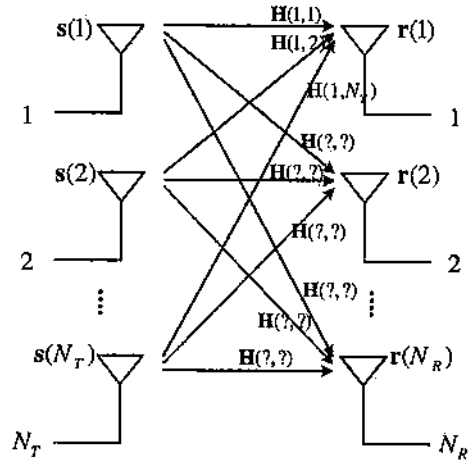

FIGURE 29.1 Fading coefficient diagram for SM MIMO.

where $\mathbf{s}$ is the $N_T \times 1$ data symbol vector whose kth element, $\mathbf{s}(k)$, denotes the data symbol transmitted from the kth TX antenna, $\mathbf{n}$ is an $N_R \times 1$ noise vector whose elements are complex Gaussian random variables, and $\mathbf{H}$ is the $N_R \times N_T$ fading coefficient (or channel gain) matrix. The model of the elements of $\mathbf{H}$ will be discussed next.

1.A-1 [T]From equation (29.1), the received signal at the kth RX antenna, $\mathbf{r}(k)$, can be expressed as

$$\mathbf{r}(k) = \mathbf{H}(?,?)\mathbf{s}(1) + (?,?)\mathbf{s}(?) + \cdots + (?,?)\mathbf{s}(N_T) + \mathbf{n}(?). \tag{29.2}$$

Complete equation (29.2).

1.A-2 [T]From the model given by equation (29.1), the (n,m)th element of the channel matrix $\mathbf{H}$, $\mathbf{H}(n,m)$, is the fading coefficient for the link from the ?-th TX antenna to the ?-th RX. Complete the two quantities marked by '**?**'.

1.A-3 [T]Fig. 29.1 shows the fading coefficient diagram from the TX antennas to the RX antennas. Complete all the places marked by '**?**' on the basis of equation (29.2).

29.2 MLD BASED ON EXHAUSTIVE SEARCH FOR SM MIMO

In this section, for convenience, whenever it does not cause a confusion, the variables/expressions we use in MATLAB and in the texts such as the number of RX antennas (N_R ,**Nr**), the received signal vector (**r**, **r**), the noise vector (**n**,**n**), and so forth, will be used interchangeably.

2.A [WWW]The m-file **ml_Nt4_bpsk.m** below is a user-defined MATLAB function that performs MLD (exhaustive search based) of an SM MIMO BPSK system with $N_T = 4$. The number of RX antennas N_R could be any nonnegative integer. This function takes the $N_R \times 1$ received signal vector **r** and the $N_R \times 4$ complex channel coefficient matrix **H** as its inputs and outputs **s_hat**, the estimate of the $N_T \times 1$ transmitted symbol vector s.

```
function s_hat=ml_Nt4_bpsk(r,H)

s_candidate_set=[];
dist_set=[];

for b1_candidate=[-1 1]
   for b2_candidate=[-1 1]
      for b3_candidate=[-1 1]
         for b4_candidate=[-1 1]
            s_candidate=[b1_candidate b2_candidate b3_candidate b4_candidate]';
            r_candidate=H*s_candidate;
            dist= sum(abs(? - r_candidate).^2);

            s_candidate_set=[s_candidate_set s_candidate];
            %Add s_candidate as a new column of s_candidate_set
            dist_set=[dist_set dist];
            %Add dist as an new element of dist_set
         end
      end
   end
end

[A B]=min(dist_set);
s_hat=s_candidate_set(:,B);
```

2.A-1 Determine the variable names that should replace '?' left in the code and justify it.

2.A-2 For each of the lines that contain '=', add a comment to explain what the variable on the left-hand side represents and justifies how the right-hand side expression is formulated accordingly. Capture the completed m-file with the comments.

2.B Extension to the different number of TX antennas.

2.B-1 [WWW]We can extend the m-file in 2.A to the case of $N_T = 3$ as shown below. Complete the m-file and save it as **ml_Nt3_bpsk.m**. Compare **ml_Nt3_bpsk.m** with **ml_Nt4_bpsk.m** to find all modified or deleted lines and explain why these lines should be modified or deleted.


```
function s_hat=ml_Nt3_bpsk(r,H)

s_candidate_set=[];
dist_set=[];

for b1_candidate=[-1 1]
   for b2_candidate=[-1 1]
      for b3_candidate=[-1 1]
            s_candidate=[b1_candidate b2_candidate b3_candidate]';
            r_candidate=H*s_candidate;
            dist= sum(abs(?-r_candidate).^2);

            s_candidate_set=[s_candidate_set s_candidate];
            %Add s_candidate as a new column of s_candidate_set
            dist_set=[dist_set dist];
            %Add dist as an new element of dist_set
      end
   end
end

[A B]=min(dist_set);
s_hat=s_candidate_set(:,B);
```

2.B-2 Similarly, modify the m-file for the case of $N_T = 2$. Save the modified m-file as **ml_Nt2_bpsk.m**. Capture **ml_Nt2_bpsk.m**.

2.C [WWW]Modify **ml_Nt4_bpsk()** as follows in order to extend to the case of quadrature phase shift keying (QPSK). Save the modified m-file as **ml_Nt4_qpsk()**. Explain why the modified lines should be modified so.

```
function s_hat=ml_Nt4_qpsk(r,H)
...
for b1_candidate=[-1-j, -1+j, 1-j, 1+j]
   for b2_candidate=[-1-j, -1+j, 1-j, 1+j]
      for b3_candidate=[-1-j, -1+j, 1-j, 1+j]
         for b4_candidate=[-1-j, -1+j, 1-j, 1+j]
            ...
         end
      end
   end
end
...
```

2.D Complexity of maximum likelihood detection (MLD).

2.D-1 In the nested **'for'** loops of **ml_Nt4_bpsk.m** in 2.A, count the total number of iterations needed to generate a new candidate (hypothesis) of the received signal and to calculate the distance between the candidate and the received signal.

2.D-2 In the nested **'for'** loops of **ml_Nt4_qpsk.m** in 2.C, count the total number of iterations needed to generate a new candidate (hypothesis) of the received signal and to calculate the distance between the candidate and the received signal.

2.D-3 Generalize the results in 2.D-1 and 2.D-3 into an $N_R \times N_T$ MIMO system that employs a modulation with an alphabet size M.

2.D-4 Summarize the problems using MLD based on exhaustive search.

29.3 ZERO FORCING DETECTION

For systems with $N_R \geq N_T$, zero forcing (ZF) detection is a method to multiply the received signal $\mathbf{r}$ by the pseudo-inverse of the channel matrix $\mathbf{H}$ to generate the decision variables in a vector $\mathbf{z}$ for the N_T simultaneously transmitted symbols:

$$\begin{aligned}\mathbf{z} &= (\mathbf{H^T H})^{-1}\mathbf{H^T} \times \mathbf{r} \\ &= (\mathbf{H^T H})^{-1}\mathbf{H^T} \times (\mathbf{Hs} + \mathbf{n}) \\ &= \mathbf{s} + (\mathbf{H^T H})^{-1}\mathbf{H^T n},\end{aligned} \tag{29.3}$$

where $(\cdot)^T$ denotes transpose and $(\mathbf{H^T H})^{-1}\mathbf{H^T}$ is the pseudo-inverse of $\mathbf{H}$; if $\mathbf{H}$ is a full-rank square matrix, then it equals the inverse of $\mathbf{H}$. Note that the signal term in $\mathbf{z}$ is equal to $\mathbf{s}$, which shows that ZF detection completely eliminates the mutual interference among the simultaneously transmitted symbols in forming the decision variables.

3.A [WWW]The user-defined MATLAB function **zf_bpsk( )** below implements ZF detection for BPSK MIMO systems. It takes the received signal vector r and channel matrix **H** as its inputs. The function **pinv(·)** computes the pseudo-inverse of a matrix. Complete the second line and save the m-file as **zf_bpsk.m**. Capture the completed m-file.

```
function s_hat=zf_bpsk(r,H)
z=pinv(?)*r;
s_hat=sign(real(z));
```

3.B Explain why **real()** and **sign()** are used in the last line.

3.C [WWW]Modify **zf_bpsk.m** for QPSK simulation as shown below. Complete the second and third lines and save the m-file as **zf_qpsk.m**. Capture the completed m-file.


```
function s_hat=zf_qpsk(r,H)
z=pinv(?)*r;
s_hat=sign(real(z))+ j * ??;
```

29.4 NOISE ENHANCEMENT OF ZF DETECTION

4.A [WWW]In the ZF detection expressed by equation (29.3), the data symbol vector s is recovered exactly for any invertible channel matrix **H**. However, the noise term after ZF becomes pinv(**H**)**n**. Therefore the performance of ZF detection will be determined by the variance of the elements of the vector pinv(**H**)**n**.

The variance of the elements of pinv(**H**)**n** depends on the condition of **H**. Let us compare the variances of the elements of pinv(**H**)**n** for the following two examples of **H**, expressed as Ha and Hb.

```
>> Ha=[2, 1; 1, 2]
Ha =
   2   1
   1   2

>> Hb=[1, 1; 1, -1]
Hb =
   1   1
   1  -1
```

4.A-1 Since the channel matrices **Ha** and **Hb** are 2 by 2 matrices, the received vector **n** is a 2(=**Nr**) × 1 vector and the noise vector after ZF is also a 2 by 1 vector. Substituting **Ha** given above into **pinv(H)*n**, we can express the two elements of **pinv(H)*n** as the linear combination of **n(1)** and **n(2)**. For example, because **pinv(Ha)** = $(\mathbf{Ha})^{-1} = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}$, the first element of the vector **pinv(Ha)*n** is expressed as **(2/3)*n(1) +(-1/3)*n(2)**. Similarly, express the second element of the vector **pinv(Ha)*n**, the first and second elements of the vector **pinv(Hb)*n**, as the linear combination of **n(1)** and **n(2)**.

4.A-2 The two elements **n(1)** and **n(2)** are i.i.d. complex Gaussian random variables. Let **v_n** denote the variance of **n(1)** and **n(2)**. Based on the expressions obtained in 4.A-1 and using the formula in equation (15.6), show that the variance of elements of **pinv(Ha)*n** is equal to **5/9*v_n** and the variance of elements of **pinv(Ha)*n** is equal to **1/2*v_n**.

4.A-3 According to 4.A-2, for ZF detection, does channel **Ha** or **Hb** result in a lower BER? Justify the answer.

4.A-4 According to 4.A-3, for ZF detection, the channel **Ha** results in a higher BER compared with the channel **Hb** although the elements of **Ha** are greater than the elements of **Hb**. This phenomenon is called "noise enhancement." Explain what kind of channel matrices cause a high noise enhancement.

4.A-5 When the channel matrix is ill-conditioned (like **Ha**), ZF will result in a high noise. Give another example of an ill-conditioned channel matrix that causes an even higher noise enhancement than the channel matrix **Ha**, that is, a channel matrix whose elements' magnitudes are greater than those of **Ha** but will result in a worse BER performance if ZF is used. Mathematically justify your example.

4.B [WWW]The m-file below simulates the BERs of ZF detection and MLD assuming that the channel matrix **H** is equal to the **Ha** given in 4.A.

```
clear
Nr=2;
Ha=[2, 1;1, 2];
Hb=[1, 1;1, -1];

EsN0dB=3;
EsN0=10^(EsN0dB/10);
Es=Nr*1;
N0=Es/EsN0;
ErrCntZF=0; ErrCntML=0;SymCnt=0;
while ErrCntML<500
%Change 500 to a smaller value to speed up simulation. However, the simulation error will increase instead.
  s=sign(rand(2,1)-0.5);
  H=Ha; % Change to H=Hb in case of simulating the BER of the channel Hb
  n=sqrt(N0/2)*(randn(2,1)+j*randn(2,1));
  r=H*s+n;

  s_hat_zf=zf_bpsk(r,H); % zf_bpsk() should be in the same folder.
  ErrCntZF=ErrCntZF+sum(s~=s_hat_zf);

  s_hat_ml=ml_Nt2_bpsk(r,H); % ml_Nt2_bpsk() should be in the same folder.
  ErrCntML=ErrCntML+sum(s~=s_hat_ml);

  SymCnt=SymCnt+2;
end
ber_zf=ErrCntZF/SymCnt
ber_ml=ErrCntML/SymCnt
```


4.B-1 For each of the lines in bold, explain what the variable on the left-hand side represents and justify how the right-hand side expression is properly formulated accordingly.

4.B-2 (a) Execute the m-file and capture the BER results.
(b) Change the line **'H=Ha'** in the m-file into **'H=Hb'**. Execute the m-file and capture the results.

4.C If the columns of the channel matrix are orthogonal, then ZF achieves the same performance as MLD. (a) Do the results in 4.B-2 confirm this fact? (b) [A]Explain why ZF achieves the same performance as MLD if the columns of the channel matrix are mutually orthogonal.

4.D Based on the results obtained in 4.B-2, compare the BERs of ZF detection for the two example channel matrices **Ha** and **Hb**. Are the BER results consistent with the noise variance analysis result in 4.A-2 and 4.A-3?

4.E In the m-file, set the channel matrix **H** to the one selected in 4.A-5 and execute the modified m-file. Capture the results and determine whether your answer to 4.A-5 is correct.

29.5 SUCCESSIVE INTERFERENCE CANCELLATION DETECTION

The successive interference cancellation (SIC) detection scheme for SM MIMO typically involves the following steps [1,2].

Step 1: Multiply the received signal vector by a linear detection matrix to create the decision variable vector **z**. For example, if the linear ZF scheme is applied, then the linear detection matrix would be **pinv(H)**.

```
z=pinv(H)*r;
```

Step 2: From **z**, detect only the N_T-th symbol (the corresponding MATLAB variable is **s(Nt)**) and store the detection result.

```
s_hat(Nt) =sign(real(z(Nt))); % for BPSK
s_hat(Nt) =sign(real(z(Nt)))+j*sign(imag(z(Nt))); % for QPSK
```

Step 3: Assuming a correct decision in Step 2, that is, assuming that **s_hat(Nt)** is equal to **s(Nt)**,

Step 3-1. Reconstruct the signal portion for **s(Nt)** in the received signal vector **r**. Refer to the note below for this step.

```
H(:,Nt)*s_hat(Nt);;
```

NOTE: The desired signal in the received vector **r** given by equation (29.1) can be decomposed as

$$\mathbf{r} = \mathbf{H}(:,1) \times \mathbf{s}(1) + \mathbf{H}(:,2) \times \mathbf{s}(2) + \cdots + \mathbf{H}\left(:,N_T\right) \times \mathbf{s}(N_T), \qquad (29.4)$$

where $\mathbf{H}(:,k)$ denotes the kth column vector of **H**. From this decomposition, we note that the contribution of $\mathbf{s}(k)$ in r is $\mathbf{H}(:,k) \times \mathbf{s}(k)$.

Step 3-2. Since the signal portion that carries **s(Nt)** in the received signal vector **r** acts as the interference to the remaining **Nt**-1 symbols, we cancel this interference from the received signal. Hence we replace the received vector r as follows.

```
r=r-H(:,Nt)*s_hat(Nt);
```

Step 4: At this point, to the other **Nt**-1 data streams the updated **r** in Step 3-2 is equivalent to the received signal as if the **Nt**-th TX antenna did not transmit any signal, effectively reducing the system to an **Nr** × (**Nt**-1) MIMO system. Now we replace **H** and **Nt** as follows.

```
H=H(:, 1:(Nt-1));
Nt=Nt-1;
```

Step 5: Repeat Steps 1–4 with r updated in Step 3-2, and **H** and **Nt** updated in Step 4 until all **Nt** symbols are detected.

Step 6: Return **s_hat** whose elements are successively stored in Step 2.

5.A [WWW]The m-file **sic_zf_bpsk.m** below is a user-defined MATLAB function that performs SIC for the received signal vector **r** and the channel matrix **H** assuming BPSK modulation.

```
function s_hat=sic_zf_bpsk(r,H)

[Nr Nt]=size(H);
s_hat=zeros(Nt,1); %To initialize s_hat.

while Nt>0
   z=pinv(H)*r;
   s_hat(Nt)=sign(real(z(Nt)));
   r=r-H(:,Nt)*s_hat(Nt);   % or r = r-H*[0 0 0....s_hat(Nt)]';
   H=H(:,1:(Nt-1));
   Nt=Nt-1;
end
```


5.A-1 Add a detailed explanation to each line as a comment. Refer to the steps explained above.

5.A-2 [WWW]The function below extends the function **sic_zf_bpsk.m** so that it works with QPSK modulation, for which the data symbols candidates are {-1-j, -1+j, 1-j, 1+j}. For this extension, only the line that computes **s_hat** (the 6th line) needs to be changed. Complete this line and save the m-file as **sic_zf_qpsk()**. Capture the completed m-file.

```
function s_hat=sic_zf_qpsk(r,H)

[Nr Nt]=size(H);
s_hat=zeros(Nt,1);

while Nt>0
   z=pinv(H)*r;
   s_hat(Nt)=sign(real(z(Nt))) + j* ? ;
   r=r- H(:,Nt)*s_hat(Nt);  %or r=r-H*[zeros(Nt-1, 1); s_hat(Nt)];
   H=H(:,1:(Nt-1));
   Nt=Nt-1;
end
```

5.B Ordered SIC.

5.B-1 Note that inside the **'while'** loop in the m-file, we substitute **r=H*s+n** into **z=pinv(H)*r** to generate the ZF decision variable vector **z**, resulting in **z=s+pinv(H)*n**, where **pinv(H)*n** is the noise vector. Let **Hinv** denote the matrix **pinv(H)** and **v_n** denote the variance of elements of the received noise vector **n**. Each element of the noise vector after ZF, **Hinv*n**, is a linear combination of the elements of **n**, where the weights depend on the channel matrix. Specifically, the **k**-th element of **Hinv*n** is a linear combination of the elements of **n** with weights equal to **Hinv(k,:)**, the **k**-th row of **Hinv**. Analyzing it a bit further, we can show that the variance of the **k**-th element of **Hinv*n** is equal to **sum(Hinv(k,:).^2) *v_n**. Prove this relationship.

5.B-2 According to 5.B-1, after ZF, the instantaneous variances of the effective noise to the Nt simultaneously transmitted symbols, **s(k)**, ⋯, **s(Nt)**, are **sum(Hinv(k,:).^2)*v_n**, **k**=1, ⋯, **Nt**, respectively. Apparently, the instantaneous noise variances for the **Nt** data symbols are different, depending on the instantaneous channel matrix. Therefore it is possible to improve the performance of the SIC scheme described above by switching the antenna indexes between the symbol with the smallest noise component and the original symbol at the **Nt**-th antenna, allowing the detection to start with the data symbol that has the smallest noise component. Such a scheme is called an ordered successive interference cancellation (OSIC) scheme [1–3].

The code fragment below implements this reordering process. Explain in detail how the variable **B** is set to the index of the element with the minimum variance among all the elements of the noise vector **pinv(H)*n**.

```
sym_index=1:Nt;
Hinv=pinv(H);
T=sum(abs(Hinv').^2);%sum(A) is the column vector which takes the sum of each row
of the matrix A as its elements.
[A B]=min(T);

C=H(:,Nt); H(:,Nt)=H(:,B); H(:,B)=C;
D=sym_index(Nt); sym_index(Nt)=sym_index(B);sym_index(B)=D;
```

5.B-3 The last line **'D=sym_index(Nt); sym_index(Nt)=sym_index(B);sym_index (B)=D;'** is for storing the original symbol index in the rearranged symbol order. Let us assume that **Nt** = 4 and the second symbol in **z** has the minimum noise variance before we execute the code lines in 5.B-2. Determine **sym_index** after executing these lines.

5.B-4 Equation (29.4) shows that if we rearrange the order of the symbols in vector **s** on the basis of the variances of the noise component associated with each symbol, then the columns of the channel matrix **H** must be rearranged accordingly. The line **'C=H(:,Nt); H(:,Nt)=H(:,B); H(:,B)=C;'** accomplishes this. Explain how this line works in detail.

5.B-5 [WWW] The m-file **osic_zf_bpsk.m** below is a user-defined MATLAB function that implements OSIC given the received signal vector **r** assuming BPSK modulation and the channel matrix **H**. Compare this m-file with **sic_zf_bpsk.m** completed in 5.A for nonordered SIC. For the two lines in bold, explain what they do and in what way they differ from the two corresponding lines in **sic_zf_bpsk.m** completed in 5.A.

```
function s_hat=osic_zf_bpsk(r,H)

[Nr Nt]=size(H);
sym_index=1:Nt;

while Nt>0
  Hinv=pinv(H);
  T=sum(abs(Hinv').^2);
  [A B]=min(T);

  C=H(:,Nt); H(:,Nt)=H(:,B); H(:,B)=C;
  D=sym_index(Nt); sym_index(Nt)=sym_index(B);sym_index(B)=D;

  z=pinv(H)*r;
  s_hat(sym_index(Nt))=sign(real(z(Nt)));
```



```
  r=r-H(:,Nt)*s_hat(sym_index(Nt));
  H=H(:,1:(Nt-1));
  Nt=Nt-1;
end
```

5.B-6 [WWW] The function below extends the function **sic_zf_bpsk.m** so that it works with QPSK modulation, for which the data symbols candidates are {-1-j, -1+j, 1-j, 1+j}. Only the line that computes **s_hat** needs to be changed. Complete this line and save the modified m-file as **osic_zf_qpsk()**. Capture the completed m-file.

```
function s_hat=osic_zf_qpsk(r,H)
...
while Nt>0
  ...
  s_hat(sym_index(Nt))=sign(real(z(Nt)))+j*??;
  ...
End
```

5.B-7 We will see in the next section that OSIC outperforms SIC. Justify why identifying the most reliable symbol and canceling it first at each SIC iteration improves the performance of SIC.

29.6 BER SIMULATION OF ZF, SIC, OSIC, AND ML DETECTION SCHEMES

6.A [WWW] The m-file below simulates the BER performances of ZF, SIC, OSIC, and ML detection methods for an SM system with **Nr** = 4, **Nt** = 4, and BPSK modulation. Examine the two scaling factors in bold in the two lines that generate **H** and **n**, and explain why they should be set as they are.

```
clear
EbN0dB_vector=0:5:30; %Lower the limit of EbN0dB(currently 30) to speed up the sim-
ulation.
Eb=1; %Because we set the symbol by s=sign(rand(Nt,1)-0.5) below.
Nr=4;Nt=4;

for snr_i=1:length(EbN0dB_vector)
  EbN0dB=EbN0dB_vector(snr_i);
  EbN0=10^(EbN0dB/10);
  N0=Eb/EbN0;

  Nerrs_zf=0;  Nerrs_sic=0;  Nerrs_osic=0;  Nerrs_ml=0;
  Nbits=0;
```

```
    Nerrs_stop=100;%Decrease Nerrs_stop if the simulation takes long time but the
    simulation error will increase.
    while Nerrs_osic < Nerrs_stop

      s=sign(rand(Nt,1)-0.5);

      H=sqrt(0.5/Nr)*(randn(Nr,Nt)+j*randn(Nr,Nt));
      n=sqrt(N0/2)*(randn(Nr,1)+j*randn(Nr,1));

      r=H*s+n;

      shat_zf=zf_bpsk(r,H); shat_sic=sic_zf_bpsk(r,H);
      shat_osic=osic_zf_bpsk(r,H); shat_ml=ml_Nt4_bpsk(r,H);

      Nerrs_zf=Nerrs_zf+sum(shat_zf~=s);
      Nerrs_sic=Nerrs_sic+sum(shat_sic~=s);
      Nerrs_osic=Nerrs_osic+sum(shat_osic~=s);
      Nerrs_ml=Nerrs_ml+sum(shat_ml~=s);

      Nbits=Nbits+Nt;
    end

    BER_vector(snr_i,:)=[Nerrs_zf Nerrs_sic Nerrs_osic  Nerrs_ml]/Nbits

end
figure
semilogy(EbN0dB_vector, BER_vector)
legend('ZF', 'SIC','OSIC', 'ML')
xlabel('E_b/N_0');ylabel('BER');grid
```

6.B Execute the m-file. (a) Capture the simulated BER graph. (b) Order the detection schemes according to their BER performances from highest to lowest.

6.C [WWW]Modify the m-file in 6.A as shown below to simulate the case with **Nr** = 2 and **Nt** = 2.

```
...
Nr=2;Nt=2;
...
for snr_i=1:length(EbN0dB_vector)
  ...
  while Nerrs_osic < Nerrs_stop
    ...
    shat_ml=ml_Nt2_bpsk(r,H);
    ...
```


TABLE 29.1 BER Comparison of Various Detection Schemes.

Detection scheme	Does the BER increase or decrease as Nr and Nt increase? (Yes/No)	Justification
ZF		
SIC		
OSIC		
ML		

6.C-1 Execute the modified m-file and capture the simulated BER graph.

6.C-2 Answer the following questions:

(a) Is the order of the detection schemes according to their performances with **Nr** = 2 and **Nt** = 2 the same as that in 6.B(b), that is, the case with **Nr** = 4 and **Nt** = 4?

(b) Let us assume that the order of the four schemes according to their error performance from highest to lowest is DS1, DS1, DS3, and DS4. At BER = 10e-3, measure (from the figure) the relative performance gap in dB between DS1 and DS2, and between DS2 and DS3, and between DS3 and DS4 for the case of (**Nr**, **Nt**) = (2,2). Also do the same for the case of (**Nr**, **Nt**) = (4,4). How do the gaps change (increase or decrease) as the number of antennas changes from (2,2) to (4,4)?

6.C-3 As the number of antennas increases from (2,2) to (4,4), does the BER of each detection scheme increase or decrease? Justify your answer in Table 29.1.

6.D [WWW]The m-file in 6.A has been modified to simulate the QPSK system with **Nr** = 4 and **Nt** = 4. The modified parts are highlighted in bold below.

```
clear
...
...
for snr_i=1:length(EbN0dB_vector)
...
  while Nerrs_osic < Nerrs_stop
    s=sign(rand(Nt,1)-0.5)+j*sign(rand(Nt,1)-0.5);
    ...
    shat_zf=zf_qpsk(r,H); shat_sic=sic_zf_qpsk(r,H);
    shat_osic=osic_zf_qpsk(r,H); shat_ml=ml_Nt4_qpsk(r,H);

    Nerrs_zf=Nerrs_zf+sum(real(shat_zf)~=real(s))+sum(imag(shat_zf)~=
    imag(s));
```

```
    Nerrs_sic=Nerrs_sic+sum(real(shat_sic)~=real(s))+sum(imag(shat_sic)~=
     imag(s));
     Nerrs_osic=Nerrs_osic+sum(real(shat_osic)~=real(s))+sum(imag(shat_
     osic)~=imag(s));
     Nerrs_ml=Nerrs_ml+sum(real(shat_ml)~=real(s))+sum(imag(shat_ml)~=
     imag(s));
     ...
     Nbits=Nbits+2*Nt;
...
```

6.D-1 Justify why the line **'Nbits=Nbits+Nt'** should be changed into **'Nbits=Nbits+2*Nt'**.

6.D-2 Execute the modified m-file and capture the simulated BER curves.

6.D-3 Compared with the BPSK system, have the relative BER performances of the four detection schemes changed?

29.7 RELATIONSHIP AMONG THE NUMBER OF ANTENNAS, DIVERSITY, AND DATA RATE

[WWW]The m-file below simulates the BER performance of MLD for each of the following cases of (**Nr, Nt**): (2, 2), (3, 2), and (2, 3).

```
clear
EbN0dB_vector=0:5:15;
Eb=1;

for snr_i=1:length(EbN0dB_vector)
  EbN0dB=EbN0dB_vector(snr_i); EbN0=10^(EbN0dB/10); N0=Eb/EbN0;

  Nerrs_2by2=0; Nerrs_2by3=0; Nerrs_3by2=0;
  NsymsNt2=0; NsymsNt3=0;
  Nerrs_stop=200;
  while Nerrs_2by3 < Nerrs_stop
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Nr=2;Nt=2;
    s=sign(rand(Nt,1)-0.5);
    H=sqrt(1/Nr)*(randn(Nr,Nt)/sqrt(2)+j*randn(Nr,Nt)/sqrt(2));
    n=sqrt(N0/2)*(randn(Nr,1)+j*randn(Nr,1));
    r=H*s+n;
    shat=ml_Nt2_bpsk(r,H);
    Nerrs_2by2=Nerrs_2by2+sum(shat~=s);
```



```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Nr=3;Nt=2;
    s=sign(rand(Nt,1)-0.5);
    H=sqrt(1/Nr)*(randn(Nr,Nt)/sqrt(2)+j*randn(Nr,Nt)/sqrt(2));
    n=sqrt(N0/2)*(randn(Nr,1)+j*randn(Nr,1));
    r=H*s+n;
    shat=ml_Nt2_bpsk(r,H);
    Nerrs_3by2=Nerrs_3by2+sum(shat~=s);

    NsymsNt2=NsymsNt2+2;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Nr=2;Nt=3;
    s=sign(rand(Nt,1)-0.5);
    H=sqrt(1/Nr)*(randn(Nr,Nt)/sqrt(2)+j*randn(Nr,Nt)/sqrt(2));
    n=sqrt(N0/2)*(randn(Nr,1)+j*randn(Nr,1));
    r=H*s+n;
    shat=ml_Nt3_bpsk(r,H);
    Nerrs_2by3=Nerrs_2by3+sum(shat~=s);

    NsymsNt3=NsymsNt3+3;
  end

  BER_vector(snr_i,:)=[Nerrs_2by2/NsymsNt2 Nerrs_3by2/NsymsNt2 Nerrs_2by3/
  NsymsNt3]
end
figure
semilogy(EbN0dB_vector(1:snr_i), BER_vector)
legend('Nr=2, Nt=2', 'Nr=3, Nt=2','Nr=2, Nt=3')
xlabel('E_b/N_0');ylabel('BER');grid
```

7.A Execute the m-file above and capture the simulated BER curves.

7.B Compare the slope of the BER curves for the cases of (**Nr, Nt**) = (2, 2) and (3, 2).

(a) For which case does the BER curve have a larger slope?
(b) The relationship between the slope of the BER curve and diversity order was investigated in Chapter 25. Justify why the slope of the BER curve increases as Nr increases.

7.C Compare the BER curves for the cases of (**Nr**, Nt) = (2, 2) and (2, 3).

(a) Which case has a worse BER performance?
(b) Justify why the BER performance is worse for the case of Nt = 3.

7.D Although the BER performance decreases as Nt increases, a larger Nt is beneficial to the system in a different perspective. Summarize the benefits.

7.E Find scenarios where a system with Nr > Nt is preferable to a system with Nr < Nt. Also find scenarios where the latter is preferable to the former.

REFERENCES

[1] G. J. Foschini, "Layered Space–Time Architecture for Wireless Communication in a Fading Environment When Using Multiple Antennas," *Bell Labs Technical Journal*, Vol. 1, No. 2, 1996, pp. 41–59.

[2] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection Algorithm and Initial Laboratory Results Using V-BLAST Space–Time Communication Architecture," *Electronics Letters*, Vol. 35, No. 1, 1999, pp.14–16. .

[3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge, UK: Cambridge University Press, 2005.

30

NEAR-ULTRASONIC WIRELESS ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING MODEM DESIGN

- Transmit an image file through a near-ultrasonic (NUS) wireless channel.
- Investigate transmit and receive algorithms for NUS orthogonal frequency division multiplexing systems.
- Observe and analyze the waveforms and spectra of NUS systems at major processing stages.

30.1 IMAGE FILE TRANSMISSION OVER A NEAR-ULTRASONIC WIRELESS CHANNEL

In this section we transmit an image file over a near-ultrasonic (NUS) wireless channel. The image file is orthogonal frequency division multiplexing (OFDM) modulated and transmitted from the speaker of a phone over an NUS wireless channel. The microphone in a PC samples the received signal and demodulates it to restore the image data.

1.A Prepare a handheld audio device (e.g., a smartphone) that is capable of playing .wav files through its internal speaker. Also prepare a laptop or desktop PC with an internal microphone (MIC) that has installed MATLAB on it.

Step 1. [WWW]Copy the following three files into the MATLAB work folder in your laptop or desktop PC.

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