Appendix A: Introduction to the MATLAB Environment

Matrix Laboratory (MATLAB) is used extensively in this book for simulations. The goal here is to help students acquire familiarity with MATLAB and build basic skills in the MATLAB environment. Hence, Appendix A serves the following objectives:

1. Learn to use the help system to study basic MATLAB commands and syntax.
2. Learn array indexing.
3. Learn basic plotting utilities.
4. Learn to write script m-files.
5. Learn to write functions in MATLAB.

A.1 BASIC COMMANDS AND SYNTAX

MATLAB has an accessible help system through the help command. By issuing the MATLAB help command following the topic or function (routine), MATLAB will return information on the topic and show how to use the function. For example, by entering **help sum** at the MATLAB prompt, we see information (listed partially here) on how to use the function **sum()**.

```
» help sum
SUM  Sum of the elements.
    For vectors, SUM(X) is the sum of the elements of X.
    For matrices, SUM(X) is a row vector with the sum over
    each column.
»
```

The following examples are given to demonstrate the usage of **sum()**:

```
» x=[ 1 2 3 1.5 -1.5 -2];          % Initialize an array
» sum(x)                           % Call MATLAB function sum
ans =
  4                                % Display the result
»
» x=[1 2 3; -1.5 1.5 2.5; 4 5 6] % Initialize 3 × 3 matrix
x =                               % Display the contents of 3 × 3 matrix
  1.0000 2.0000 3.0000
  -1.5000 1.5000 2.5000
  4.0000 5.0000 6.0000
```

```
» sum(x)                      % Call MATLAB function sum
ans =
  3.5000 8.5000 11.5000       % Display the results

»
```

MATLAB can be used like a calculator to work with numbers, variables, and expressions in the command window. The following are basic syntax examples:

```
» sin(pi/4)
ans =
  0.7071
» pi*4
ans =
  12.5664
```

In MATLAB, variable names can store values, vectors and matrices. See the following examples.

```
» x=cos(pi/8)
x =
  0.9239

» y=sqrt(x)-2^2
y =
  -3.0388

» z=[1 -2 1 2]
z =
  1 -2 1 2

» zz=[1 2 3; 4 5 6]
zz =
  1 2 3
  4 5 6
```

Complex numbers are natural in MATLAB. See the following examples.

```
» z=3+4i                     % Complex number
z =
  3.0000 + 4.0000i

» conj(z)                    % Complex conjugate of z
ans =
  3.0000 - 4.0000i

» abs(z)                     % Magnitude of z
ans =
  5
» angle(z)                   % Angle of z (radians)
ans =
  0.9273

» real(z)                    % Real part of a complex number z
ans =
  3

» imag(z)                    % Imaginary part of a complex number z
ans =
  4

» exp(j*pi/4)                % Polar form of a complex number
ans =
  0.7071 + 0.7071i
```

The following shows examples of array operations. Krauss, Shure, and Little (1994) and Stearns (2003) give detailed explanation for each operation.

```
» x=[1 2; 3 4]               % Initialize 2 × 2 matrixes
x =
  1 2
  3 4

» y=[-4 3; -2 1]
y =
  -4 3
  -2 1

» x+y                        % Add two matrixes
ans =
  -3 5
  1 5
» x*y                        % Matrix product
ans =
  -8 5
  -20 13

» x.*y                       % Array element product
ans =
  -4 6
  -6 4

» x'                         % Matrix transpose
ans =
  1 3
  2 4

» 2.^x                       % Exponentiation: matrix x contains each exponent
ans =
  2 4
  8 16
```

```
» x.^3               % Exponentiation: power of 3 for each element in matrix x
ans =
  1 8
  27 64

» y.^x               % Exponentiation: each element in y has a power
                     % specified by a corresponding element in matrix x
ans =
  -4 9
  -8 1

» x=[0 1 2 3 4 5 6]  % Initialize a row vector
x =
  0 1 2 3 4 5 6

» y=x.*x-2*x        % Use array element product to compute a quadratic function
y =
  0 -1 0 3 8 15 24

» z=[1 3]'          % Initialize a column vector
z =
  1
  3

» w=x\z             % Invert matrix x, then multiply it by the column vector z
w =
  1
  0
```

A.2 MATLAB ARRAYS AND INDEXING

Let us look at the syntax to create an array as follows:

Basic syntax: **x=begin: step: end**

An array x is created with the first element value of **begin**. The value increases by a value of **step** for the next element in the array and stops when the next stepped value is beyond the specified end value of **end**. In simulation, we may use this operation to create the sample indexes or array of time instants for digital signals. The **begin**, **step,** and **end** can be assigned to integers or floating-point numbers.

The following examples are given for illustrations:

```
» n=1:3:8           % Create a vector n=[1 4 7]
n =
  1 4 7

» m=9:-1:2          % Create a vector m=[9 8 7 6 5 4 3 2 ]
m =
  9 8 7 6 5 4 3 2
```

```
» x=2:(1/4):4        % Create x=[2 2.25 2.5 2.75 3 3.25 3.5 3.75 4]
x =
  Columns 1 through 7
  2.0000 2.2500 2.5000 2.7500 3.0000 3.2500 3.5000
  Columns 8 through 9
  3.7500 4.0000

» y=2:-0.5:-1  % Create y=[2 1.5 1 0.5 0 -0.5 -1]
y =
  2.0000 1.5000 1.0000 0.5000 0 -0.5000 -1.0000
```

Next, we examine creating a vector and extracting numbers in a vetcor:

```
» xx= [ 1 2 3 4 5 [5:-1:1] ]  % Create xx=[ 1 2 3 4 5 5 4 3 2 1]
xx =
  1 2 3 4 5 5 4 3 2 1
» xx(3:7)                     % Show elements to 7, that is, [3 4 5 5 4]
ans =
  3 4 5 5 4
» length(xx)                  % Return of the number of elements in vetcor xx
ans =
  10
» yy=xx(2:2:length(xx))      % Display even indexed numbers in array xx
yy =
  2 4 5 3 1
```

A.3 PLOT UTILITIES: SUBPLOT, PLOT, STEM, AND STAIR

The following are common MATLAB plot functions for digital signal processing (DSP) simulation:

subplot opens subplot windows for plotting.
plot produces an x–y plot. It can also create multiple plots.
stem produces discrete-time signals.
stair produces staircase (sample-and-hold) signals.

The following program contains different MATLAB plot functions:

```
t=0:0.01:0.4;  % Create time vector for time instants from 0 to 0.4 second
xx=4.5*sin(2*pi*10*t+pi/4); % Calculate a sine function with a frequency of 10 Hz
yy=4.5*cos(2*pi*5*t-pi/3); % Calculate cos function with a frequency of 5 Hz
subplot(4,1,1), plot(t,xx);grid          % Plot a sine function in window 1
subplot(4,1,2), plot(t,xx, t,yy, ' -. ' ) ;grid; % Plot sine and cos functions in window 2
subplot(4,1,3), stem(t,xx);grid        % Plot a sine function in the discrete-time form
subplot(4,1,4), stairs(t,yy);grid       % Plot a cos function in the sample-and-hold form
xlabel( ‘ Time (sec.) ’ );
```

Each plot is shown in Figure A.1. Notice that dropping the semicolon at the end of the MATLAB syntax will display values on the MATLAB prompt.

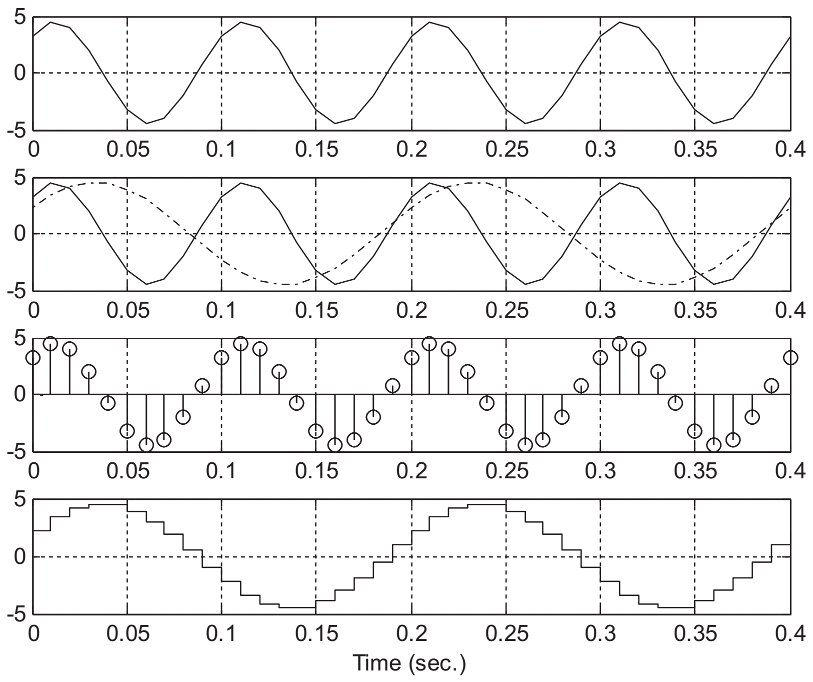

FIGURE A.1

Illustration for the MATLAB plot functions.

A.4 MATLAB SCRIPT FILES

We can create a MATLAB script file using the built-in MATLAB editor (or Windows Notepad) to write MATLAB source code. The script file is named "filename.m" and can be run by typing the file name at the MATLAB prompt and hitting the return key. The script file **test.m** is described here for illustration. Figure A.2 illustrates the plot produced by **test.m**.

At MATLAB prompt, run the program

```
>>which test   % show the folder where test.m resides
```

Go to the folder that contains test.m, and run your script from MATLAB.

```
>>test         % run the test.m
>>type test    % display the contents of test.m
```

test.m

```
t=0:0.01:1;
x=sin(2*pi*2*t);
y=0.5*cos(2*pi*5*t-pi/4);
plot(t,x), grid on
title( ' Test plots of sinusoids ' )
ylabel( ' Signal amplitudes ' );
```

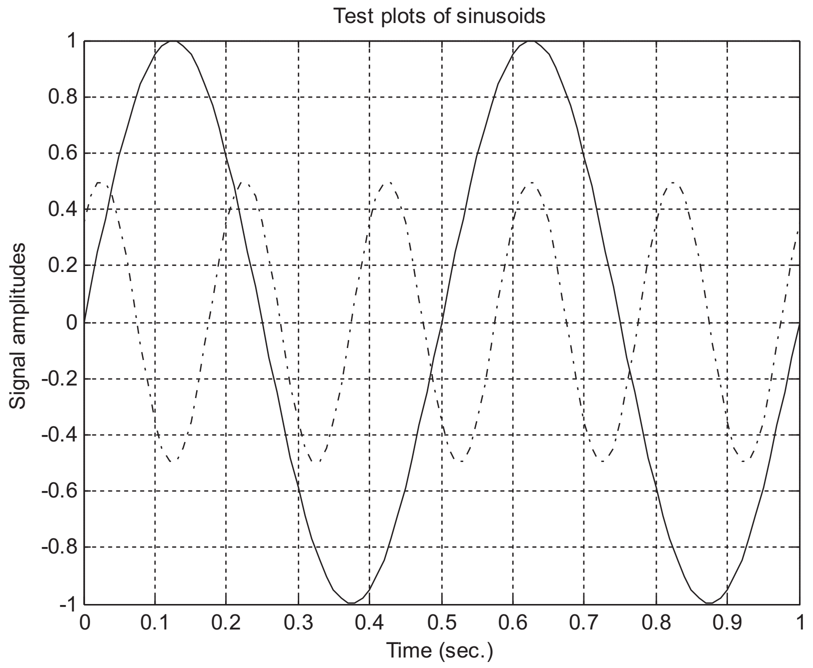

FIGURE A.2

Illustration of MATLAB script file test.m.

```
xlabel( ‘ Time (sec.) ’ ); hold on
plot(t,y, ’ -. ’ );
```

A.5 MATLAB FUNCTIONS

A MATLAB function is often used to replace the repetitive portions of MATLAB code. It is created using a MATLAB script file. However, the code begins with the keyword **function**, followed by the function declaration, comments for the help system, and program code. A function **sumsub.m** that computes the addition and subtraction of two numbers is listed here for illustration.

sumsub.m

```
function [sum, sub]=sumsub(x1,x2)
%sumsub: Function to add and subtract two numbers
% Usage:
% [sum, sub] = sumsub(x1,x2)
% x1 = the first number
% x2= the second number
% sum = x1+x2;
% sub = x1-x2
sum = x1+x2;  % Add two numbers
sub= x1-x2;   % Subtract x2 from x1
```

To use the MATLAB function, go to the folder that contains **sumsub.m**. At the MATLAB prompt, try the following:

```
» help sumsub  % Display usage information on MATLAB prompt
sumsub: Function to add and subtract two numbers
usage:
[sum, sub] = sumsub(x1,x2)
x1 = the first number
x2= the second number
sum = x1+x2;
sub = x1-x2
```

Run the function as follows:

```
» [x1, x2]=sumsub(3, 4-3i);  % Call function sumsub
» x1                         % Display the result of sum
x1 =
  7.0000 - 3.0000i
» x2                         % Display the result of subtraction
x2 =
  -1.0000 + 3.0000i
```

MATLAB functions can also be used inside an m-file. More MATLAB exercises for introduction to DSP can be explored in McClellan, Schafer, and Yoder (1998) and Stearns (2003).