Attia, John Okyere. “Matlab Fundamentals.”
Electronics and Circuit Analysis using MATLAB.
Ed. John Okyere Attia
Boca Raton: CRC Press LLC, 1999

CHAPTER ONE

MATLAB FUNDAMENTALS

MATLAB is a numeric computation software for engineering and scientific calculations. The name MATLAB stands for MATRIX LABORATORY. MATLAB is primarily a tool for matrix computations. It was developed by John Little and Cleve Moler of MathWorks, Inc. MATLAB was originally written to provide easy access to the matrix computation software packages LINPACK and EISPACK.

MATLAB is a high-level language whose basic data type is a matrix that does not require dimensioning. There is no compilation and linking as is done in high-level languages, such as C or FORTRAN. Computer solutions in MATLAB seem to be much quicker than those of a high-level language such as C or FORTRAN. All computations are performed in complex-valued double precision arithmetic to guarantee high accuracy.

MATLAB has a rich set of plotting capabilities. The graphics are integrated in MATLAB. Since MATLAB is also a programming environment, a user can extend the functional capabilities of MATLAB by writing new modules.

MATLAB has a large collection of toolboxes in a variety of domains. Some examples of MATLAB toolboxes are control system, signal processing, neural network, image processing, and system identification. The toolboxes consist of functions that can be used to perform computations in a specific domain.

1.1 MATLAB BASIC OPERATIONS

When MATLAB is invoked, the command window will display the prompt >>. MATLAB is then ready for entering data or executing commands. To quit MATLAB, type the command

exit or **quit**

MATLAB has on-line help. To see the list of MATLAB's help facility, type

help

The help command followed by a function name is used to obtain information on a specific MATLAB function. For example, to obtain information on the use of fast Fourier transform function, **fft**, one can type the command

help fft

The basic data object in MATLAB is a rectangular numerical matrix with real or complex elements. Scalars are thought of as a 1-by-1 matrix. Vectors are considered as matrices with a row or column. MATLAB has no dimension statement or type declarations. Storage of data and variables is allocated automatically once the data and variables are used.

MATLAB statements are normally of the form:

variable = expression

Expressions typed by the user are interpreted and immediately evaluated by the MATLAB system. If a MATLAB statement ends with a semicolon, MATLAB evaluates the statement but suppresses the display of the results. MATLAB is also capable of executing a number of commands that are stored in a file. This will be discussed in Section 1.6. A matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

may be entered as follows:

```
A = [1 2 3;  2 3 4; 3 4 5];
```

Note that the matrix entries must be surrounded by brackets [ ] with row elements separated by blanks or by commas. The end of each row, with the exception of the last row, is indicated by a semicolon. A matrix A can also be entered across three input lines as

```
A = [ 1 2 3
      2 3 4
      3  4 5];
```

In this case, the carriage returns replace the semicolons. A row vector B with four elements

```
B = [ 6 9 12 15 18 ]
```

can be entered in MATLAB as

```
B = [6 9 12 15 18];
```

or

```
B = [6 , 9,12,15,18]
```

For readability, it is better to use spaces rather than commas between the elements. The row vector B can be turned into a column vector by **transposition**, which is obtained by typing

```
C = B'
```

The above results in

```
C =
     6
     9
     12
     15
     18
```

Other ways of entering the column vector C are

```
C =  [6
      9
      12
      15
      18]
```

or

```
C  = [6; 9; 12; 15; 18]
```

MATLAB is case sensitive in naming variables, commands and functions. Thus b and B are not the same variable. If you do not want MATLAB to be case sensitive, you can use the command

casesen off

To obtain the size of a specific variable, type **size ( )**. For example, to find the size of matrix A, you can execute the following command:

```
size(A)
```

The result will be a row vector with two entries. The first is the number of rows in A, the second the number of columns in A.

To find the list of variables that have been used in a MATLAB session, type the command

whos

There will be a display of variable names and dimensions. Table 1.1 shows the display of the variables that have been used so far in this book:

Table 1.1
Display of an output of whos command

Name	Size	Elements	Byte	Density	Complex
A	3 by 3	9	72	Full	No
B	1 by 5	5	40	Full	No
C	5 by 1	5	40	Full	No
ans	1 by 2	2	16	Full	No

The grand total is 21 elements using 168 bytes.

Table 1.2 shows additional MATLAB commands to get one started on MATLAB. Detailed descriptions and usages of the commands can be obtained from the MATLAB help facility or from MATLAB manuals.

Table 1.2
Some Basic MATLAB Commands

Command	Description
%	Comments. Everything appearing after % command is not executed.
demo	Access on-line demo programs
length	Length of a matrix
clear	Clears the variables or functions from workspace
clc	Clears the command window during a work session
clg	Clears graphic window
diary	Saves a session in a disk, possibly for printing at a later date

1.2 MATRIX OPERATIONS

The basic matrix operations are addition(+), subtraction(-), multiplication (*), and conjugate transpose(‘) of matrices. In addition to the above basic operations, MATLAB has two forms of matrix division: the left inverse operator \ or the right inverse operator /.

Matrices of the same dimension may be subtracted or added. Thus if E and F are entered in MATLAB as

```
E = [7 2 3; 4 3 6; 8 1 5];

F = [1 4 2; 6 7 5; 1 9 1];
```

and

```
G = E - F
H = E + F
```

then, matrices G and H will appear on the screen as

```
G =
    6   -2    1
   -2   -4    1
    7   -8    4

H =
    8    6    5
   10   10   11
    9   10    6
```

A scalar (1-by-1 matrix) may be added to or subtracted from a matrix. In this particular case, the scalar is added to or subtracted from all the elements of another matrix. For example,

```
J = H + 1
```

gives

```
J =
    9    7    6
   11   11   12
   10   11    7
```

Matrix multiplication is defined provided the inner dimensions of the two operands are the same. Thus, if X is an n-by-m matrix and Y is i-by-j matrix,

X*Y is defined provided m is equal to i. Since E and F are 3-by-3 matrices, the product

```
Q = E*F
```

results as

```
Q =
    22   69   27
    28   91   29
    19   84   26
```

Any matrix can be multiplied by a scalar. For example,

```
2*Q
```

gives

```
ans =
    44   138   54
    56   182   58
    38   168   52
```

Note that if a variable name and the "=" sign are omitted, a variable name **ans** is automatically created.

Matrix division can either be the left division operator \ or the right division operator /. The right division a/b, for instance, is algebraically equivalent to $\frac{a}{b}$ while the left division a\b is algebraically equivalent to $\frac{b}{a}$.

If $Z * I = V$ and Z is non-singular, the left division, $Z\backslash V$ is equivalent to MATLAB expression

$$I = inv(Z) * V$$

where **inv** is the MATLAB function for obtaining the inverse of a matrix. The right division denoted by V/Z is equivalent to the MATLAB expression

$$I = V * inv(Z)$$

There are MATLAB functions that can be used to produce special matrices. Examples are given in Table 1.3.

Table 1.3
Some Utility Matrices

Function	Description
ones(n,m)	Produces n-by-m matrix with all the elements being unity
eye(n)	gives n-by-n identity matrix
zeros(n,m)	Produces n-by-m matrix of zeros
diag(A)	Produce a vector consisting of diagonal of a square matrix A

1.3 ARRAY OPERATIONS

Array operations refer to element-by-element arithmetic operations. Preceding the linear algebraic matrix operations, * / \ ‘ , by a period (**.**) indicates an array or element-by-element operation. Thus, the operators **.* , .\ , ./, .^** , represent element-by-element multiplication, left division, right division, and raising to the power, respectively. For addition and subtraction, the array and matrix operations are the same. Thus, + and .+ can be regarded as an array or matrix addition.

If A1 and B1 are matrices of the same dimensions, then A1.*B1 denotes an array whose elements are products of the corresponding elements of A1 and B1. Thus, if

```
A1 = [2 7 6
      8 9 10];

B1 = [6 4 3
      2 3 4];
```

then

```
C1 = A1.*B1
```

results in

```
C1 =
     12   28   18
     16   27   40
```

An array operation for left and right division also involves element-by-element operation. The expressions A1./B1 and A1.\B1 give the quotient of element-by-element division of matrices A1 and B1. The statement

```
D1 = A1./B1
```

gives the result

```
D1 =
    0.3333    1.7500    2.0000
    4.0000    3.0000    2.5000
```

and the statement

```
E1 = A1.\B1
```

gives

```
E1 =
    3.0000    0.5714    0.5000
    0.2500    0.3333    0.4000
```

The array operation of raising to the power is denoted by .^. The general statement will be of the form:

```
q = r1.^s1
```

If r1 and s1 are matrices of the same dimensions, then the result q is also a matrix of the same dimensions. For example, if

```
r1 = [ 7 3 5];

s1 = [ 2 4 3];
```

then

```
q1 = r1.^s1
```

gives the result

```
q1 =
    49    81   125
```

One of the operands can be scalar. For example,

```
q2 = r1.^2

q3 = (2).^s1
```

will give

```
q2 =
     49     9    25
```

and

```
q3 =
     4    16     8
```

Note that when one of the operands is scalar, the resulting matrix will have the same dimensions as the matrix operand.

1.4 COMPLEX NUMBERS

MATLAB allows operations involving complex numbers. Complex numbers are entered using function i or j. For example, a number $z = 2 + j2$ may be entered in MATLAB as

```
z = 2+2*i
```

or

```
z = 2+2*j
```

Also, a complex number za

$$za = 2\sqrt{2}\exp[(\pi / 4)j]$$

can be entered in MATLAB as

```
za =  2*sqrt(2)*exp((pi/4)*j)
```

It should be noted that when complex numbers are entered as matrix elements within brackets, one should avoid any blank spaces. For example, $y = 3 + j4$ is represented in MATLAB as

```
y = 3+4*j
```

If spaces exist around the + sign, such as

```
u= 3 + 4*j
```

MATLAB considers it as two separate numbers, and y will not be equal to u.

If w is a complex matrix given as

$$w = \begin{bmatrix} 1 + j1 & 2 - j2 \\ 3 + j2 & 4 + j3 \end{bmatrix}$$

then we can represent it in MATLAB as

```
w = [1+j  2-2*j;  3+2*j  4+3*j]
```

which will produce the result

```
w =
     1.0000 + 1.0000i   2.0000 - 2.0000i
     3.0000 + 2.0000i   4.0000 + 3.0000i
```

If the entries in a matrix are complex, then the "prime" (') operator produces the conjugate transpose. Thus,

```
wp = w'
```

will produce

```
wp =
      1.0000 - 1.0000i   3.0000 - 2.0000i
      2.0000 + 2.0000i   4.0000 - 3.0000i
```

For the unconjugate transpose of a complex matrix, we can use the point transpose (**.'**) command. For example,

```
wt = w.'
```

will yield

```
wt =
      1.0000 + 1.0000i   3.0000 + 2.0000i
      2.0000 - 2.0000i   4.0000 + 3.0000i
```

1.5 THE COLON SYMBOL (:)

The colon symbol (**:**) is one of the most important operators in MATLAB. It can be used (1) to create vectors and matrices, (2) to specify sub-matrices and vectors, and (3) to perform iterations. The statement

```
t1 = 1:6
```

will generate a row vector containing the numbers from 1 to 6 with unit increment. MATLAB produces the result

```
t1 =
     1     2     3     4     5     6
```

Non-unity, positive or negative increments, may be specified. For example, the statement

```
t2 = 3:-0.5:1
```

will result in

```
t2 =
    3.0000    2.5000    2.0000    1.5000    1.0000
```

The statement

```
t3 = [(0:2:10);(5:-0.2:4)]
```

will result in a 2-by-4 matrix

```
t3 =
         0    2.0000    4.0000    6.0000    8.0000   10.0000
    5.0000    4.8000    4.6000    4.4000    4.2000    4.0000
```

Other MATLAB functions for generating vectors are linspace and logspace. **Linspace** generates linearly evenly spaced vectors, while **logspace** generates

logarithmically evenly spaced vectors. The usage of these functions is of the form:

```
linspace(i_value, f_value, np)

logspace(i_value, f_value, np)
```

where

i_value is the initial value

f_value is the final value

np is the total number of elements in the vector.

For example,

```
t4 = linspace(2, 6, 8)
```

will generate the vector

```
t4 =
    Columns 1 through 7

     2.0000    2.5714    3.1429    3.7143    4.2857    4.8571
     5.4286

    Column 8

     6.0000
```

Individual elements in a matrix can be referenced with subscripts inside parentheses. For example, t2(4) is the fourth element of vector t2. Also, for matrix t3, t3(2,3) denotes the entry in the second row and third column. Using the colon as one of the subscripts denotes all of the corresponding row or column. For example, t3(:,4) is the fourth column of matrix t3. Thus, the statement

```
t5 = t3(:,4)
```

will give

```
t5 =
    6.0000
    4.4000
```

Also, the statement t3(2,:) is the second row of matrix t3. That is the statement

```
t6 = t3(2,:)
```

will result in

```
t6 =
    5.0000    4.8000    4.6000    4.4000    4.2000    4.0000
```

If the colon exists as the only subscript, such as t3(:), the latter denotes the elements of matrix t3 strung out in a long column vector. Thus, the statement

```
t7 = t3(:)
```

will result in

```
t7 =
         0
    5.0000
    2.0000
    4.8000
    4.0000
    4.6000
    6.0000
    4.4000
    8.0000
    4.2000
   10.0000
    4.0000
```

Example 1.1

The voltage, v, across a resistance is given as (Ohm's Law), $v = Ri$, where i is the current and R the resistance. The power dissipated in resistor R is given by the expression

$$P = Ri^2$$

If R = 10 Ohms and the current is increased from 0 to 10 A with increments of 2A, write a MATLAB program to generate a table of current, voltage and power dissipation.

Solution:

MATLAB Script

```
diary  ex1_1.dat
% diary causes output to be written into file ex1_1.dat
% Voltage and power calculation
R=10;       % Resistance value
i=(0:2:10); % Generate current values
v=i.*R;      % array multiplication to obtain voltage
p=(i.^2)*R;  % power calculation
sol=[i v p]  % current, voltage and power values are printed
diary
% the last diary command turns off the diary state
```

MATLAB produces the following result:

```
sol =
    Columns 1 through 6

    0     2     4     6     8    10

    Columns 7 through 12

    0    20    40    60    80   100

    Columns 13 through 18

    0    40   160   360   640  1000
```

Columns 1 through 6 constitute the current values, columns 7 through 12 are the voltages, and columns 13 through 18 are the power dissipation values.

1.6 M-FILES

Normally, when single line commands are entered, MATLAB processes the commands immediately and displays the results. MATLAB is also capable of processing a sequence of commands that are stored in files with extension m. MATLAB files with extension m are called m-files. The latter are ASCII text files, and they are created with a text editor or word processor. To list m-files in the current directory on your disk, you can use the MATLAB command **what**. The MATLAB command, **type**, can be used to show the contents of a specified file. M-files can either be script files or function files. Both script and function files contain a sequence of commands. However, function files take arguments and return values.

1.6.1 Script files

Script files are especially useful for analysis and design problems that require long sequences of MATLAB commands. With script file written using a text editor or word processor, the file can be invoked by entering the name of the m-file, without the extension. Statements in a script file operate globally on the workspace data. Normally, when m-files are executing, the commands are not displayed on screen. The MATLAB echo command can be used to view m-files while they are executing. To illustrate the use of script file, a script file will be written to simplify the following complex valued expression z.

Example 1.2

Simplify the complex number z and express it both in rectangular and polar form.

$$z = \frac{(3 + j4)(5 + j2)(2\angle 60^0)}{(3 + j6)(1 + j2)}$$

Solution:

The following program shows the script file that was used to evaluate the complex number, z, and express the result in polar notation and rectangular form.

MATLAB Script

```
diary ex1_2.dat
```

```
% Evaluation of Z
% the complex numbers are entered
Z1 = 3+4*j;
Z2 = 5+2*j;
theta = (60/180)*pi;   % angle in radians
Z3 = 2*exp(j*theta);
Z4 = 3+6*j;
Z5 = 1+2*j;
% Z_rect is complex number Z in rectangular form
disp('Z in rectangular form is');  % displays text inside brackets
Z_rect = Z1*Z2*Z3/(Z4+Z5);
Z_rect
Z_mag = abs (Z_rect);   % magnitude of Z
Z_angle = angle(Z_rect)*(180/pi);  % Angle in degrees
disp('complex number Z in polar form, mag, phase'); % displays text
%inside brackets
Z_polar = [Z_mag, Z_angle]
diary
```

The program is named ex1_2.m. It is included in the disk that accompanies this book. Execute it by typing ex1_2 in the MATLAB command window. Observe the result, which should be

```
Z in rectangular form is

Z_rect =
          1.9108 + 5.7095i

complex number Z in polar form (magnitude and phase) is

Z_polar =
            6.0208   71.4966
```

1.6.2 Function Files

Function files are m-files that are used to create new MATLAB functions. Variables defined and manipulated inside a function file are local to the function, and they do not operate globally on the workspace. However, arguments may be passed into and out of a function file.

The general form of a function file is

```
function variable(s) = function_name (arguments)
% help text in the  usage of the function
%
.
.
end
```

To illustrate the usage of function files and rules for writing m-file function, let us study the following two examples.

Example 1.3

Write a function file to solve the equivalent resistance of series connected resistors, R1, R2, R3, …, Rn.

Solution:

MATLAB Script

```
function req = equiv_sr(r)
% equiv_sr is a function program for obtaining
%           the equivalent resistance of series
%           connected resistors
% usage:  req = equiv_sr(r)
%             r is an input vector of length n
%             req is an output, the equivalent resistance(scalar)
%
n = length(r);    % number of resistors
req = sum (r);    % sum up all resistors
end
```

The above MATLAB script can be found in the function file equiv_sr.m, which is available on the disk that accompanies this book.

Suppose we want to find the equivalent resistance of the series connected resistors 10, 20, 15, 16 and 5 ohms. The following statements can be typed in the MATLAB command window to reference the function equiv_sr

```
a = [10 20 15 16 5];
Rseries = equiv_sr(a)
diary
```

The result obtained from MATLAB is

```
Rseries =
          66
```

Example 1.4

Write a MATLAB function to obtain the roots of the quadratic equation

$$ax^2 + bx + c = 0$$

***Solution*:**

MATLAB Script

```
function rt = rt_quad(coef)
%
% rt_quad  is a function for obtaining the roots of
%          of a quadratic equation
% usage: rt = rt_quad(coef)
%          coef is the coefficients a,b,c of the quadratic
%               equation ax*x + bx + c =0
%          rt are the roots, vector of length 2
% coefficient a, b, c are obtained from vector coef
a = coef(1);  b = coef(2);  c = coef(3);
int = b^2 - 4*a*c;
if int > 0
  srint = sqrt(int);
  x1= (-b + srint)/(2*a);
  x2= (-b - srint)/(2*a);
elseif int == 0
  x1= -b/(2*a);
  x2= x1;
elseif int < 0
  srint = sqrt(-int);
  p1 = -b/(2*a);
  p2 = srint/(2*a);
  x1 = p1+p2*j;
  x2 = p1-p2*j;
end
rt =[x1;
    x2];
end
```

The above MATLAB script can be found in the function file rt_quad.m, which is available on the disk that accompanies this book.

We can use m-file function, rt_quad, to find the roots of the following quadratic equations:

(a) $x^2 + 3x + 2 = 0$

(b) $x^2 + 2x + 1 = 0$

(c) $x^2 - 2x + 3 = 0$

The following statements, that can be found in the m-file ex1_4.m, can be used to obtain the roots:

```
diary ex1_4.dat
ca = [1 3 2];
ra = rt_quad(ca)
cb = [1 2 1];
rb = rt_quad(cb)
cc = [1 -2 3];
rc = rt_quad(cc)
diary
```

Type into the MATLAB command window the statement ex1_4 and observe the results. The following results will be obtained:

```
ra =
      -1
       -2

rb =
      -1
      -1

rc=
     1.0000 + 1.4142i
     1.0000  - 1.4142i
```

The following is a summary of the rules for writing MATLAB m-file functions:

(1) The word, function, appears as the first word in a function file. This is followed by an output argument, an equal sign and the function name. The

arguments to the function follow the function name and are enclosed within parentheses.

(2) The information that follows the function, beginning with the % sign, shows how the function is used and what arguments are passed. This information is displayed if help is requested for the function name.

(3) MATLAB can accept multiple input arguments and multiple output arguments can be returned.

(4) If a function is going to return more than one value, all the values should be returned as a vector in the function statement. For example,

```
function [mean, variance] = data_in(x)
```

will return the mean and variance of a vector x. The mean and variance are computed with the function.

(5) If a function has multiple input arguments, the function statement must list the input arguments. For example,

```
function [mean, variance] = data(x,n)
```

will return mean and variance of a vector x of length n.

(6) The last statement in the function file should be an "end" statement.

SELECTED BIBLIOGRAPHY

1. MathWorks, Inc., *MATLAB, High-Performance Numeric Computation Software*, 1995.

2. Biran, A. and Breiner, M., *MATLAB for Engineers*, Addison-Wesley, 1995.

3. Etter, D.M., *Engineering Problem Solving with MATLAB*, 2nd Edition, Prentice Hall, 1997.

EXERCISES

1.1 The voltage across a discharging capacitor is

$$v(t) = 10(1 - e^{-0.2t})$$

Generate a table of voltage, $v(t)$, versus time, *t*, for t = 0 to 50 seconds with increment of 5 s.

1.2 Use MATLAB to evaluate the complex number

$$z = \frac{(3 + j6)(6 + j4)}{(2 + j1)j2} + 7 + j10$$

1.3 Write a function-file to obtain the dot product and the vector product of two vectors a and b. Use the function to evaluate the dot and vector products of vectors x and y, where x = (1 5 6) and y = (2 3 8).

1.4 Write a function-file that can be used to calculate the equivalent resistance of n parallel connected resistors. In general, the equivalent resistance of resistors $R_1, R_2, R_3, \ldots, R_n$ is given by

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \ \ldots \ + \frac{1}{R_n}$$

1.5 The voltage V is given as $V = RI,$ where R and I are resistance matrix and I current vector. Evaluate V given that

$$R = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 6 \\ 3 & 6 & 7 \end{bmatrix} \quad \text{and} \quad I = \begin{bmatrix} 1 \\ 2 \\ 6 \end{bmatrix}$$

1.6 Use MATLAB to simplify the expression

$$y = 0.5 + j6 + 3.5e^{j0.6} + (3 + j6)e^{j0.3\pi}$$

1.7 Write a function file to evaluate n factorial (i.e. n!); where

$$n! = n(n-1)(n-2)..(2)(1)$$

Use the function to compute $x = \dfrac{7!}{3!4!}$

1.8 For a triangle with sides of length a, b, and c, the area A is given as

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where

$$s = (a+b+c)/2$$

Write a function to compute the area given the sides of a triangle. Use the function to compute the area of triangles with the lengths: (a) 56, 27 and 43 (b) 5, 12 and 13.