EGR 182 Lab — Introductory Mathematics for Engineering Applications

Lab 1: MATLAB Introduction

California Baptist University
Fall Semester 2019
Number of Lab Periods: 1
Report Due: Beginning of next lab meeting

1 Objectives

Lab 1 is designed to introduce you to the MATLAB language/tool in a learn-by-doing manner, but also to be an easy-to-use reference for later review of the basics. Future lab assignments will build upon this first lesson.

MATLAB is a matrix based numerical processing package that is used at a large number of universities for instructional and research purposes. In addition, MATLAB is used at corporations to solve complex engineering problems. The name MATLAB stands for MATrix LABoratory, which means that all operations in MATLAB are implicitly matrix operations. (Scalar operations involve 1×1 matrix operations).

The objectives of this laboratory exercise are as follows.
1. Become familiar with the MATLAB desktop.
2. Start using various MATLAB functions and operations to solve basic mathematical problems applicable to engineering.

2 Optional Videos

As a supplement to the prelab videos (which you should have already watched), the following *optional* videos provide an additional quick overview of MATLAB (and the practice steps below are adapted from them).

Getting Started With MATLAB
http://www.mathworks.com/videos/getting-started-with-matlab-68985.html

Working in the Development Environment
http://www.mathworks.com/videos/working-in-the-development-environment-69021.html

3 The MATLAB Desktop

Starting/Quitting and Halting Execution in MATLAB

MATLAB is available under Windows. To start MATLAB, double click on the MATLAB icon. To quit MATLAB, either click on the "×" in the upper right corner of the MATLAB desktop or by typing exit or quit in the MATLAB command window. Finally, one can "kill" a process running in the MATLAB command window by depressing the `Ctrl + C` keys simultaneously.

The MATLAB Desktop

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. The first time MATLAB starts, you can see the following desktop tools — Command Window, Launch Pad, Workspace, Command History, Current Directory. Play with each one to find out what its function is. For more information on these tools, go to the help menu item `Help|Using the Desktop`.

MATLAB Help

MATLAB has extensive online help available. The fastest way to access help is to click on the question mark ("?") inside the blue ball, in the upper left menu area of the MATLAB desktop. A window will open which allows you to search or browse for topics. For example, in the search command line, upper left, try typing in the word "plot" and see what happens.

4 Start Using MATLAB–As A Calculator

Entering Variables

MATLAB is an array-based language: variables can be vectors, matrices, or n-dimensional arrays. **Type the following lines**, which have the prompt >>, into the Command Window, hit the Enter key, and notice the answers. **Read all of the comments and notes**, but do not type the comments, which begin with the percent sign %.

Some of the examples below were inspired by the following on-line tutorial:
http://users.ece.gatech.edu/bonnie/book/TUTORIAL/tutorial.pdf

```
>> a=1; %scalar variable, a
>> b=2; %using the semicolon prevents the result being echoed
```

MATLAB uses these primary arithmetic operators:

+	addition
–	subtraction
*	multiplication
/	division
ˆ	power operator
'	transpose

```
>> c = a + b    % removing the semicolon allows the answer to be echoed
c=
     3
>> a % to know the value of a variable, simply type it+Enter
a=
     1
>> 2 ˆ 2   % using the power operator
ans=
     4
```

Some predefined variables:

`i`	sqrt(−1)
`j`	sqrt(−1)
`pi`	3.1416...

```
>> y =  2 *(1 +  4 *j )  %complex number example, i and j interchangeably=
       2. 0000 + 8 . 0000 i
```

Some predefined functions :

`abs`	magnitude of a number (absolute value for real numbers)
`angle`	angle of a complex number, in radians
`cos(cosd)`	cosine function, assumes argument is in radians(degrees) sin(sind) sine function, assumes argument is in radians (degrees)
`exp`	exponential function, based on natural logarithm

```
>> c=abs(y)
c=
       8.2462

>> c=angle(y)
c=
       1.3258

>> d=cos(a)  % a=1 radian, as above
d=
       0. 5403

>> cosd(45)  % cosine of 45 degrees
ans =
       0. 7071

>> d=exp(1)  % exponential function, e raised to the 1
d =
       2. 7183
```

Vectors and Matrices

```
>> t = [ 1 2 3 4 5 ]      % row vector variable, t, with spaces between elements
t =
       1 2 3 4 5

>> t = 1: 5  % alternate row vector entry (from 1 to 5, in steps of 1)
t =
       1 2 3 4 5
```

```
>> q = [ 1 ; 2 ; 3 ; 4 ; 5 ]    % column vector, q, with semicolons between elements
q=
      1
      2
      3
      4
      5

>> x = 0: 0. 1: 1  % create row vector, from 0 to 1, in steps of 0.1
x=
      0     0 .1   0 . 2  0 .3 . . .   1 .0

>> M = [ 1 2 4 ; 3 6 8 ]  % Matrices are defined by rows, separated by semicolons
M =
      1     2     4
      3     6     8

>> whos      % whos reports all variables entered thus far
            Name      Size      Bytes     Class      Attributes

            M         2 x 3     48        double
            a         1 x 1     8         double
            ans       1 x 1     8         double
            b         1 x 1     8         double
            c         1 x 1     8         double
            d         1 x 1     8         double
            q         5 x 1     40        double
            t         1 x 5     40        double
            x         1 x 11    88        double
            y         1 x 1     16        double     complex

% scalar variables are treated as 1x1 matrices
>> clear  all      % removes all variables from Workspace
>> clc % clears Command Window
```

Vector Operations

```
>> t = 1: 5  % vector t
t =
      1 2 3 4 5

>> y = 2 * t % vector y = vector times 2
y =
      2 4 6 8 1 0
```

Simple Plotting

```
>> t = 0 : 0 . 0 1 : 1 ;  %create vector t : 101 elements, from 0 to 1, in steps of
0.01

>> %uses built-inconstant,pi
>> y=sin(2*pi*t);  %vector y = sine of 2 times pi times t; 101 elements
>> plot(t,y) % creates figure window with plot / graph of sine (2 x pi x t)
```

Matrix Operations

```
>> a = [ 1 2 3 ; 4 5 6 ; 7 8 1 0 ]     % create 3 x 3 matrix, a
a=
      1      2      3
      4      5      6
      7      8      1 0

>>size(a)    %built−in function, size, tells dimension of variable      a
ans =
      3      3      %3 rows, 3 columns
>> b = a’    %b = conjugate transpose of
b=
      1      4      7
      2      5      8
      3      6      1 0
>> b = a.’   % b = transpose of a
```

(you should use the transpose and not the conjugate transpose to avoid errors when using matrices with complex numbers – especially electrical engineers should be very wary of the conjugate transpose!!)

```
>> %matrix multiplication: sums rows of a multiplied by the columns of b
>> % e.g., c (1, 1) = 14 = 1 x 1 + 2 x 2 + 3 x 3 (using the first row of a and the
first column of b)
>> c = a * b %matrix multiplication: sums rows of a x columns of b
c =
      14     32     53
      32     77     128
      53     128    213

>> c = a .* b       %element−by−element matrix multiplication, uses dot−notation
c =
      1      8      2 1
      8      2 5    4 8
      21     48     100

>> %use built−in function, inv, to get inverse of a
>> % confirm result is identity matrix
>> inv(a)*a
ans =
      1      0      0
      0      1      0
      0      0      1
```

Select elements or sections of an array by indexing

```
>> a
a =
      1      2      3
      4      5      6
      7      8      10

>> a(2,3)    % get value at row 2, column 3
ans =
      6
```

Set values

```
>> a(1:2,:) = 0    % set rows 1 – 2 = 0
a =    % : here means all columns
       0     0     0
       0     0     0
       7     8     1 0
```

Some special matrices

```
>> M=zeros(3,3)    % n x m matrix of zeros; also works for vectors
M =
       0     0     0
       0     0     0
       0     0     0

>> V = ones(1,5)   % a matrix (vector) of ones : 1 row, 5 columns
V =
       1     1     1     1     1
```

5 Introduction to MATLAB Programming (and m files)

When using MATLAB as a calculator, you simply type variables and equations into the command line. However, the power of any programming language is that several, even hundreds, of lines of code (commands) can be typed into a file, and saved. That file is then "run" or "executed" to perform the calculation. In MATLAB, these are called script files (or m files, because they must end with ".m"). A script is a program that can perform a more complicated (and longer) calculation than you would normally type into the command line. M files can easily be shared with colleagues or later edited if you need to modify your program.

Looping/Conditional Statements and Relational Operators in MATLAB

Like any programming language, MATLAB also has statements that allow for looping (`for` and `while`) as well as conditional execution (`if`). Refer to the MATLAB help for the precise syntax of the statements as well as examples utilizing the statements (i.e., `help for`, `help while` and `help if`). An example of a `for`-statement is as follows.

```
>>for k=1:1:101
      x(k) = (k-1)*0.02;
      y(k) = x(k)*x(k)-2*x(k)+2;
>>end
```

The above for-statement creates three vectors, each with 101 elements: `k`, the index vector, contains numbers 1-101; `x` ranges from 0 to 20 in steps of 0.2; `y` holds the calculations of the given equation, $y = x^2 - 2x + 2$, for the values of x from 1-101.

Table 1: Common MATLAB Relational Operations.

MATLAB Relational Operator	Description of MATLAB Relational Operator
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	Equal
~=	not equal
&	And
|	Or
~	Not

As part of using looping/conditional statements, relational operators are used. The most commonly used relational operators are summarized in Table 1. Note that the "equal" relational operator (==) is *not* used for assigning values to a variable. To assign a value of 2 to the variable b, one would enter `b = 2` in the MATLAB command window.

Plotting in MATLAB (and running an m-file)

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. One of the most commonly used commands is `plot`. The following is an example of how to plot Figure 1 ($y = x^2 - 2x + 2$).

```
>> for k=1:1:101
      x(k) = (k-1)*0.02;
      y(k) = x(k)*x(k)-2*x(k)+2;
>> end
>> plot(x,y)
>> xlabel('x = 0:2')
>> ylabel('y=x*x-2*x+2')
>> title('Plot of a quadratic function')
```

In order to add plots to an existing graph, the hold command should be used. To put multiple plots in one figure, the subplot command is useful. More features regarding plotting, axis control, editing plots, etc., can be found in the MATLAB help facility.

Do This: A file (lab1_plotFig1.m) containing the above program has been stored on Blackboard within the Lab 01 assignment. Copy the file onto your own USB flash drive and then run it by doing these steps:

1) Browse, from within MATLAB (top, center of Desktop), to the file you copied,
2) File name appears in left panel, Current Folder,
3) double click on the file name, which should open it in the MATLAB editor,
4) Click on the green arrow, top of editor window to run the program. A figure should open that looks like Figure 1 below.

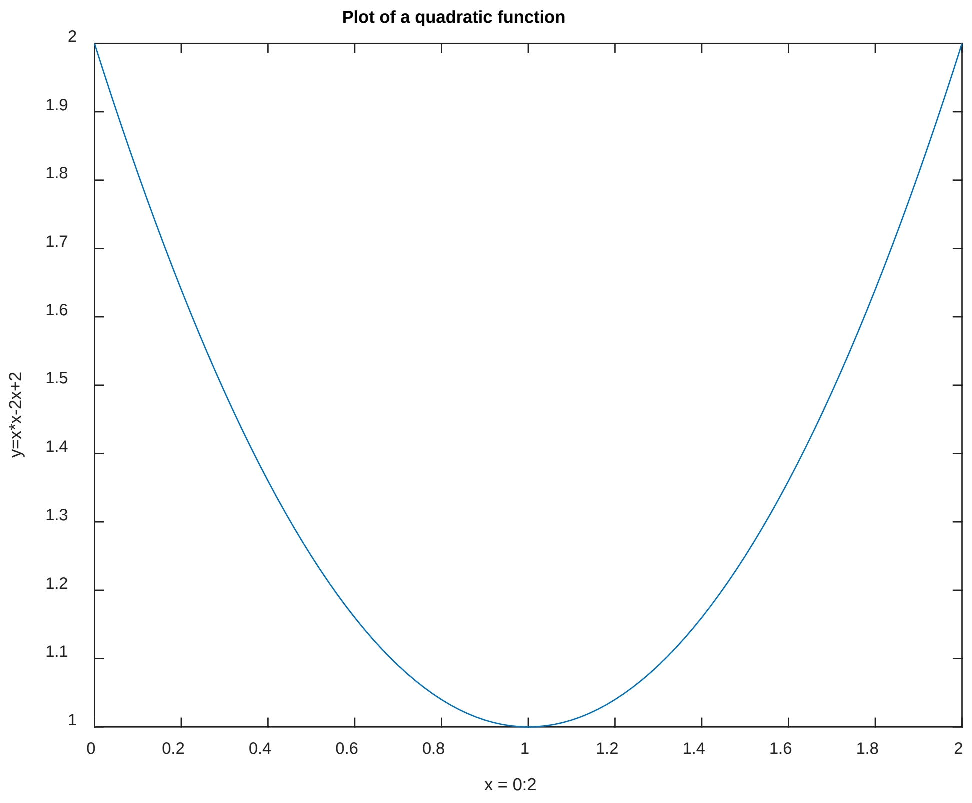

Figure 1. An Example

6 Using MATLAB to Visually Solve Systems of Linear Equations

We can use the MATLAB plotting capability to help us visually solve systems of linear equations. The following is an example.

Consider a system of two linear equations:

$$2x - y = 1 \qquad (1)$$
$$-4x + 4y = 2 \qquad (2)$$

We can rewrite the above equations as follows

$$y = 2x - 1 \qquad (3)$$
$$y = x + 0.5 \qquad (4)$$

and then use the following MATLAB commands, which create four vectors, in order to plot the two lines on the same plot (see Figure 2).

```
for k =1:1:3001                         %  Index vector, k, 3001 elements
    x(k) = (k - 1) * 0.001;             %  vector x, 3001 elements, steps of 0.001
    y1(k) = 2 * x(k) -1;                %  y1 holds value of ”2x-1” for each x value
    y2(k) = x(k) + 0.5;                 %  y2 holds value of ”x + 0.5” for each x
end
plot(x,y1,’b’)                          %  plot y1 as function of x, using a blue line
hold on                                 %  don’t overwrite previous plots
plot(x,y2,’r’)                          %  plot y2 vs x, using a red line
xlabel(’x=0:3’)
ylabel(’y1=2*x-1,y2=x+0.5’)
title(’Plot of the two lines’)
```

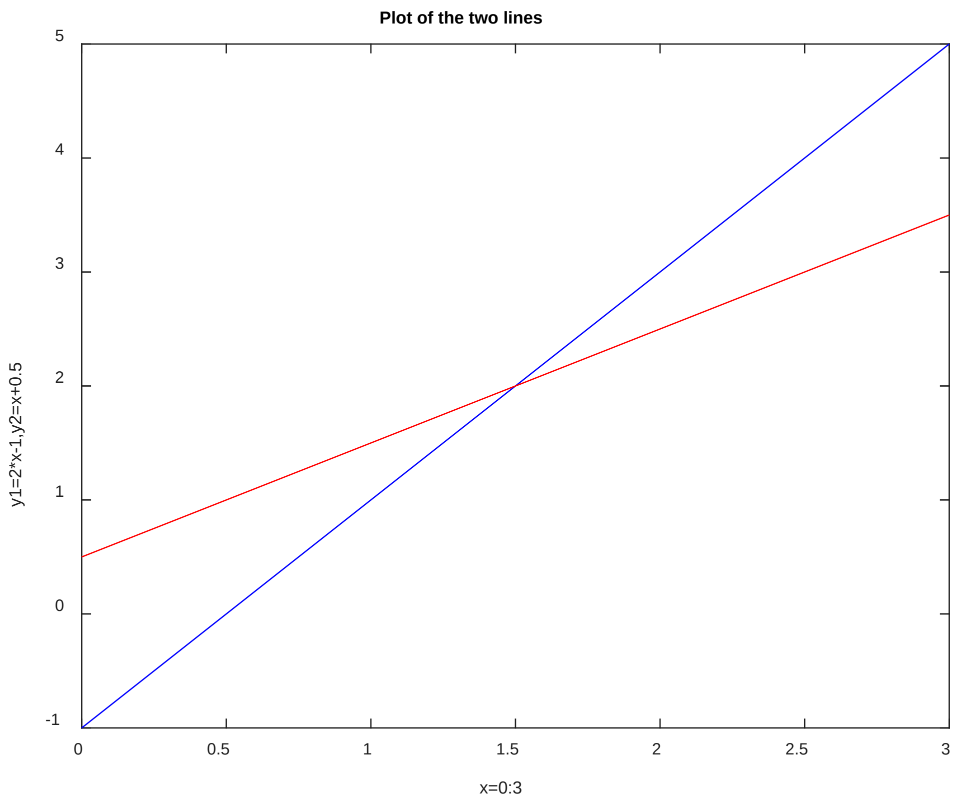

Figure 2. Plot of the two line equations

Once we obtain the plot, we can zoom in to locate the solution to the system of linear equations (see Figure 3). The above program is saved on Blackboard as lab1_plotFig2.m

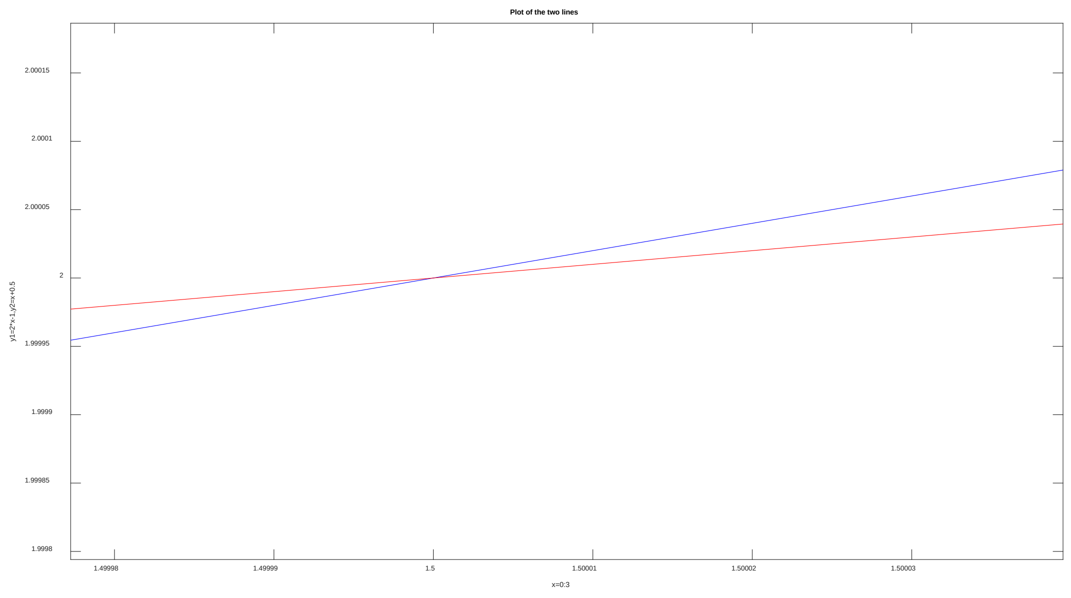

Figure 3. Plot of two line equations ZOOMED in to see intersection

7 Exercises

1. Use the COMMAND WINDOW to evaluate the following expression using MATLAB

$$10\pi + 7^3 + 2^{-2}$$

   Be sure to save the results from the COMMAND WINDOW for your lab report.

For exercises 2 – 4 create an “m” file with a separate section (use the characters %% to separate each section) for each exercise (2 – 4) and the publish feature to generate a file for your lab report. Use comments to clearly identify the code for each exercise in your output.

2. Determine the product AB where

$$A = \begin{bmatrix} 3.2 & 8 & -1 & 0 \\ 3 & 7.2 & 4.5 & -2.3 \\ -6.2 & 3.5 & 2 & -3 \\ -1.4 & -2.2 & 0 & 4.5 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 2.2 & 7 \\ 5 & 1 \\ 6 & 3.8 \\ -2 & 0.5 \end{bmatrix}$$

3. Solve Ax=b where A is the same as in step 2 and b is defined as

$$b = \begin{bmatrix} 1 \\ 0 \\ -2 \\ 3 \end{bmatrix}$$

4. **(a)** Use the programming features of MATLAB to plot equations (5) and (6) (use the program shown earlier as an example) and then zoom in to determine the numerical values of the solution (zoom in far enough to show the solution to at least two decimal points):

$$2x + y = 1 \tag{5}$$
$$-x + y = 2 \tag{6}$$

   Be sure to save **both** the original and zoomed in plot (labeled and titled).

(b) Verify that the answer from part (a) is correct by solving the system of equations using the MATLAB inverse function (i.e., `x = inv(A) * b`, where A = the matrix of coefficients, and b is the column vector of right-side values).

8 Requirements—What You Must Do

1. Read and work through the Lab 1 assignment ("MATLAB Introduction").
   - (a) Engineers must learn to read carefully!
   - (b) Especially focus on doing the examples in Section 4 (pages 2-6a)—this is vital in learning this new MATLAB language. Read and think as you type and observe the results. Discuss them with your partner.
   - (c) Read Section 5 carefully and do everything where it says "**Do This:**"
   - (d) Read Section 6 carefully and note the file that's available on Blackboard for you to use.
   - (e) Now do Section 7 ("Exercises")—**<u>20 points</u>**
     - i. In the command window, select and print what you typed and the results for Exercise 1.
     - ii. Create an "m" file containing the code for exercises 2-4 in separate sections. Be sure to save your "m" file and the output. Also be sure to create the plot AND the zoomed-in plot of the graph showing the intersection (solution).
     - iii. **Always save your MATLAB files on your own USB Flash Drive–not the lab PC**
2. Create and submit your lab report—**<u>20 points</u>**
   - (a) Two students are to work in a group to do the lab exercises. However, each student is required to compose and submit his/her own lab report. **Submit a lab report that follows the format instructions given in the "Lab Report Requirements" document, available on Blackboard.**
   - (b) Copy and edit the Sample Lab Report from Blackboard
   - (c) Put in your name and your lab partners
   - (d) Fill in the missing parts as directed
   - (e) Attach your MATLAB m-file(s) and print-outs (i.e., command line output, plot, AND zoomed-in plot) to the end of the report. If any of the MATLAB files are not included in the lab report then you will miss the points associated with completing that MATLAB exercise from Section 7.
   - (f) Submit the report by the due date mentioned above (i.e., at the beginning of the next lab period). The total points for this lab are 40 which is composed of 20 points for the lab report (i.e., format and content) and 20 points for MATLAB output in the appendix that verify you completed all of the exercises in Section 7.