EE 216 – Experiment 1
MATLAB Structure and Use

This first laboratory experiment is an introduction to the use of MATLAB. The basic computer-user interfaces, data entry techniques, operations, functions, mathematical expressions and programming techniques are reviewed. The student is asked to perform the various tasks required to use MATLAB effectively in the performance of engineering analysis and design.

Succinct information concerning the basic MATLAB concepts required is presented in Appendix A in the form of a brief tutorial. The MATLAB User's and Reference Guides should be used to extend the student's breadth and depth of understanding.

In this experiment, the student is first asked to reproduce the examples in the tutorial to gain hands-on familiarity and confidence with MATLAB operational methods and procedures. Further understanding of these methods and procedures is then promoted with additional examples of the student's own design. Throughout this experiment, the **Numeric Format** should be set to **Compact** to reduce the space required for computer output and in the Laboratory Notebook.

I. Starting MATLAB Session

1. Logon to your account. Click on the **MATLAB 6.1** icon on the desktop. (MATLAB is also available in the Start Menu) The large window is the **Command Window.**

2. Clear the **Command Window** using **Edit-Clear Command Window**. Switch to the compact numeric format under **File-Preferences-Command Window-Numeric Display**.

3. Enter your name and the date as a **comment** at the top of the **Command Window**. Comments are preceded by a **%**. Start every section of the lab with your name, the section number, and a divider:

   Example:

   ```
   % Name
   % Lab 1 Section 1
   % *****************
   ```

II. MATLAB Statements

1. Reproduce the statement examples in Section 1.1 of the tutorial.
2. Enter some data and define some variables of your own using all of the concepts presented in Section 1.1 of the tutorial.
3. Print the **Command Window** before continuing.

III. M-Files

M-files are one of the most important concepts in MATLAB. They are files of statements that can be executed by MATLAB when the file name is typed in the **Command Window**. They are created and edited in the **Editor/Debugger.** One of the benefits of using M-files is that you need not retype commands over and over in the Command Window every time you make a mistake. M-files are often referred to as **scripts**. It is very important to understand M-files, as most of the laboratory experiments will be performed using M-files.

See Section 1.2 in the Tutorial for instructions on creating M-files.

1. Create a folder for your EE 216 laboratory work on your s:/ drive. Save your m-files to this folder. To add this folder to the MATLAB path go to **File-Set Path-Add Folder**. Be sure to save before you exit. You will now need to change to this directory. Use the **pwd** command to see the current directory and use the **cd** command to change directory.

   Example: **cd s:\ee216**

Input data for a script can be supplied by using a **load** statement (the data is loaded into Matlab's Workspace). The data must be stored in a file with a **.mat** extension. Data can be saved in such a file by using the **save** statement. See the tutorial or use the **help** command for instruction on using these commands.

2. In the command window, use the **clear all** command and then define the variables **a**, **b**, **c**, and **d**. Choose any values. Use the **save** command to place these values into a file called **variables.mat**. Again use the **clear all** command. Type **whos** to make sure the workspace is clear. Now use the **load** command to reinstate your variables.

The remainder of the experiment should be done with m-files. Each section of an experiment will be written in one large m-file. Using the Debugging mode of the editor will help in writing your scripts. One can set break points, indicated by a red dot, to run the code from the beginning to a specific line. After finishing with breakpoints click the Exit Debug mode menu button. One can also run a specific segment of code by highlighting it, right clicking, and using the Evaluate Selection menu option. For more information on the Debugging mode search for Using Debugging Features in the Matlab help system.

IV. Numeric Format

1. Reproduce the examples of complex number format and data entry in Section 1.3.1 of the tutorial.

2. Create and enter some complex numbers of your own using the concepts introduced in section 1.3.1.

V. Matrix and Vector Format

1. Reproduce the examples of matrix and vector format and data entry in Sections 1.3.2 and 1.3.3 of the tutorial.

2. Use the MATLAB command **roots** to find the roots of the polynomial $2\,x^{2}\ 4x\ 10$. (Hint: Use the **help** command to find out how to use the **roots** function.) Verify your answer by using the quadratic formula to manually compute the roots.

3. Read Sections 1.3.4 and 1.3.5 of the tutorial.

VI. Character Strings

1. Reproduce the character string examples in Section 1.4 of the tutorial.

2. Enter the character string **c='We learn to use MATLAB in EE 266 Laboratory'**. Print the statement **MATLAB Laboratory** by selecting the desired words and spaces from the variable **c**.

VII. Arithmetic Operations

1. Reproduce the examples in Section 1.5 of the tutorial.

2. Find the matrix $\mathbf{a}-\mathbf{b}\mathbf{c}^{2}+2\mathbf{d}'$ when the matrices **a**, **b**, **c**, and **d** are

$$a = \begin{bmatrix} 1.5 & 3.3 \\ 6.0 & -4.5 \\ -2.5 & 0.7 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 0.5 & 0.3 \\ -0.1 & 0.2 \\ 0.4 & -0.3 \end{bmatrix}$$

$$c = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \quad , \quad d = \begin{bmatrix} 3.1 & 1.4 & -0.3 \\ -0.5 & 1.6 & 0.1 \end{bmatrix}$$

3. We define the element of the matrix **a** in row ***i*** and column ***j*** to be $\mathbf{a}_{ij}$. Use array operations to find $b_{ij} - c_{ij}\,(d_{ij})^{4}$ for all i and j for your choice of three 2x3 arrays **b**, **c,** and **d**. Manually verify the above result for $i = 1$ and $j = 2$.

VIII. Input/output in M-files, Function M-files

See Section 1.6 in the Tutorial for instructions on creating function and I/O M-files. Do not replicate the instructions, only read them.

The input-output characteristics of a script m-file make it impossible to use as a subroutine within a program. Function m-files are useful for this purpose since they can be created with input and/or output variables. No internal variable values remain in the **Workspace** after a function m-file is exited. Thus, such data must be passed back out or saved with a **save** statement within the function m-file if it is to be used again.

1. Create a function m-file, called **sumsin.m** that sums two sinusoids. The inputs should be **t**, **f1**, and **f2**. The outputs should be $s1 = \sin(2\pi \mathrm{f1} t)$, $s2 = \sin(2\pi \mathrm{f2} t)$, and $s3 = \mathrm{s1} + \mathrm{s2}$.

To test your function you may wish to call it in the **Command Window**. To do this, use the command line [**s1 s2 s3] =** ***sumsin*****(t,f1,f2)**. If there are no errors, you may assume that your function works correctly.

IX. Logical Operators

1. Reproduce the logical operator examples in Section 1.7 of the tutorial in the **Command Window**.

2. In a script M-file, use logical and array operations to produce matrix **b** from matrix **a** where

$$a = \begin{bmatrix} 1.2 & -3.2 & 24 \\ 0.6 & -0.3 & -0.5 \\ -2.3 & 1.6 & 20 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1.2 & 0 & 0 \\ 0 & -0.6 & -1.0 \\ 0 & 1.6 & 0 \end{bmatrix}$$

X. Mathematical Functions and Expressions

1. Read Section 1.8 of the tutorial and reproduce the mathematical expression examples in Section 1.8.1.

2. Try the three functions **round**, **floor**, and **ceil**, on

   $x = [-3.6, -2.5, -1.4, -1, 0, 1.4, 2.5, 3.6]$

   to observe their characteristics.

3. Find the value of *x* when

$$x = \frac{\log[2+\sin^2(3)]+e^{-0.2}}{\sqrt{2^{1.6}+3^{-0.5}}}$$

4. If t varies from –1.2 to 1.2 in steps of 0.4, find;

a. $w(t) = 3t^3 + 2t^2 - t + \sin(t)$.

$b.\ x(t) = \begin{Bmatrix} 0 & t < 0 \\ 2 & t \geq 0 \end{Bmatrix}$

XI. Flow Control Statements

1. Reproduce the flow control examples in Section 1.9 of the tutorial.

2. Use **for** statements to find the values of $x(t) + 3\cos(2\pi \text{ ft } + 0.1)$ for $t = 0, 0.1, 0.2, 0.3, 0.4$ s when *f* = 10, 15, and 20 Hz. Use one set of statements to compute the values for all three frequencies and store the results in a two-dimensional array. (Hint: use two nested for loops, and a double index.) What is the value of *x*(*t*) when *f* = 15 Hz and *t* = 0.3 s?

3. Use the **while** statement to find the largest value of positive *t* for which $e^{1.2}\cos(\omega t)$ and t ^{3} are both less than 10. Make the computation for ω = 35, 40, and 45 and find your answers to the nearest 0.01.

4. Evaluate $x(t) = e^{-|t|}$ for $-1 \leq t \leq 1$ in steps of 0.2 using **for** and **if** statements. Repeat using logical operations and 0-1 arrays. Now reduce the step size to 0.0002. Do not print the values of *x*(*t*) to the **Command Window** since there are 10,001 of them. Be sure to initialize the array first.

XII. Numeric Functions

1. Read Section 1.10 of the tutorial and reproduce the numeric function examples in Section 1.10.1.

2. Create an 11-element vector with values of $x(t) = 4\cos(2\pi t + 0.2) + 3\sin(\pi^2 t)$ at equally spaced times in the interval $0 \leq t \leq 1$. **Without printing the vector**, find the maximum element value, the minimum element value, the average of the element values and the indices of the elements for which the element magnitude is greater than 4. Check your results by printing the vector to the **Command Window** and identifying the values found.

3. Given the array

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

use MATLAB statements to find : (a) the number of rows and columns in **A**, (b) the maximum and minimum element values in **A**, (c) the maximum and minimum element values in each row of **A**, (d) the average element value in each column, (e) the average of all element values.

XIII. Plotting Functions

It is very useful to plot or graph the results of a script. MATLAB has several functions for this purpose. Some of these are listed in Section 1.11 of the tutorial.

1. Create a script that uses the sum of sinusoids function created in VIII and plot the output signals. Suggested values are t = 0:0.01:10, f1 = 0.2, f2=0.425. Plot all three sinusoids on the same axis. Label the axes, and title the plot with your initials, experiment number and a brief plot description. Create a legend or label.

2. In a new figure window, plot all three sinusoids on separate axes, but in the same window. Title the first (top) axis, and label the others appropriately. You may find that the printout will be cleaner if you only label the bottom x axis.