INTRODUCTION TO MATLAB

BEFORE YOU BEGIN

PREREQUISITE LABS

- None

EXPECTED KNOWLEDGE

- Algebra and fundamentals of linear algebra.

EQUIPMENT

- None

MATERIALS

- None

OBJECTIVES

After completing this lab you should know how to use MATLAB to load experimental data, plot the data, plot exact functions such as sinusoids, manipulate plots to your satisfaction, and to save plots to image files that can be included in your lab report. You should also know how to use the online help to expand your knowledge of MATLAB as needed.

INTRODUCTION

MATLAB is a programming language and environment offered by The MathWorks company (http://www.mathworks.com/) that is widely used in the engineering industry. MathWorks offers many analysis, design, and systems simulation products, based on the MATLAB environment.

Among its many features, MATLAB enables you to import data, export data, manipulate data, and it has many methods of visualizing and plotting data.

STARTING MATLAB

Start MATLAB by clicking on Start on the Windows task bar and then selecting Start -> Programs -> MATLAB -> MATLAB 7.0.

MATLAB HELP

When you first start MATLAB, you are presented with the following message:

> To get started, select "MATLAB Help" from the Help menu.

The MATLAB manuals can be accessed from within MATLAB by one of the following methods:

- By typing "helpdesk" at the MATLAB prompt.
- By selecting Help ⇒ MATLAB Help from the menu bar.

When the Help window is opened, you will see the screen similar to Figure 1.

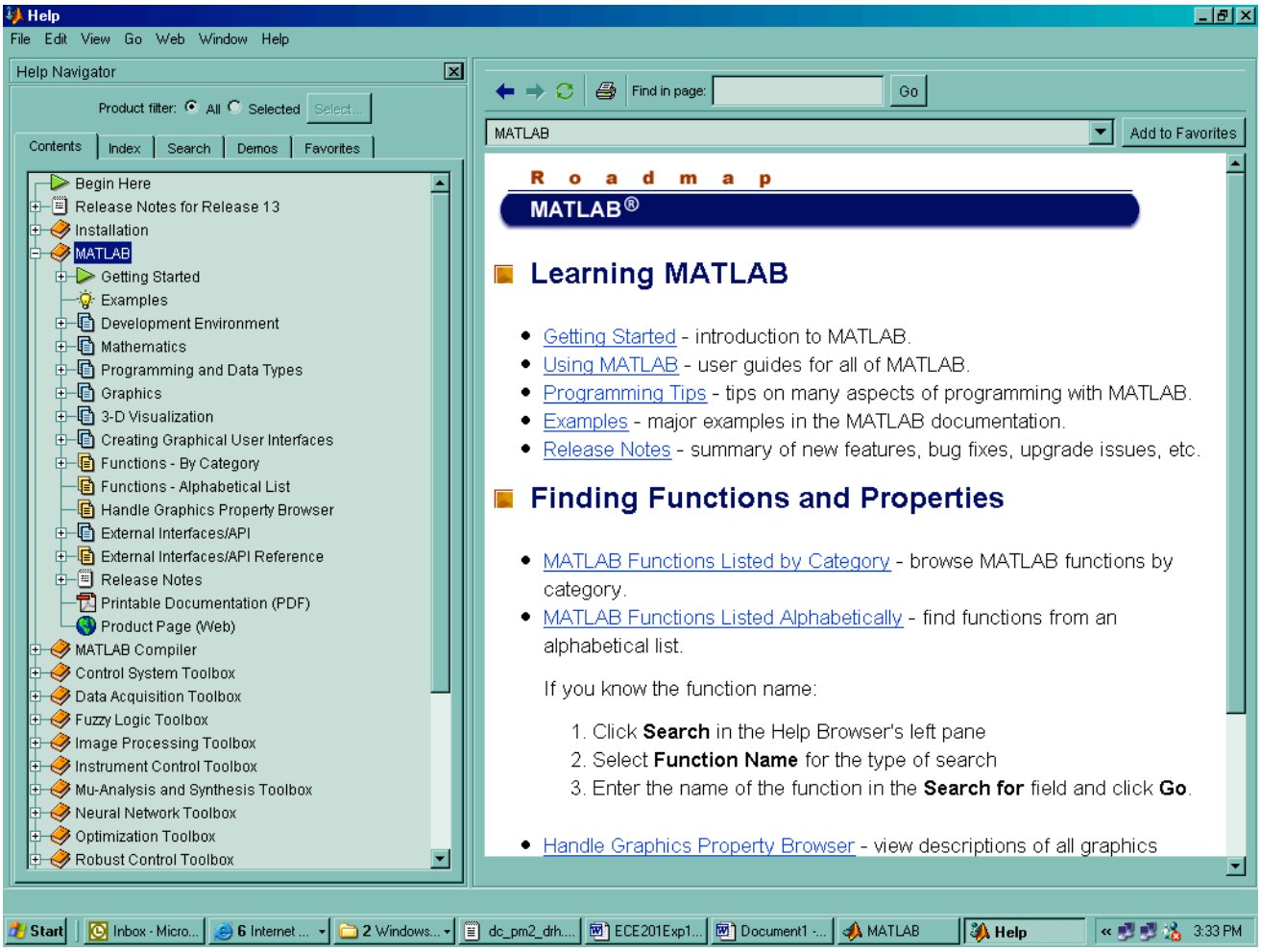
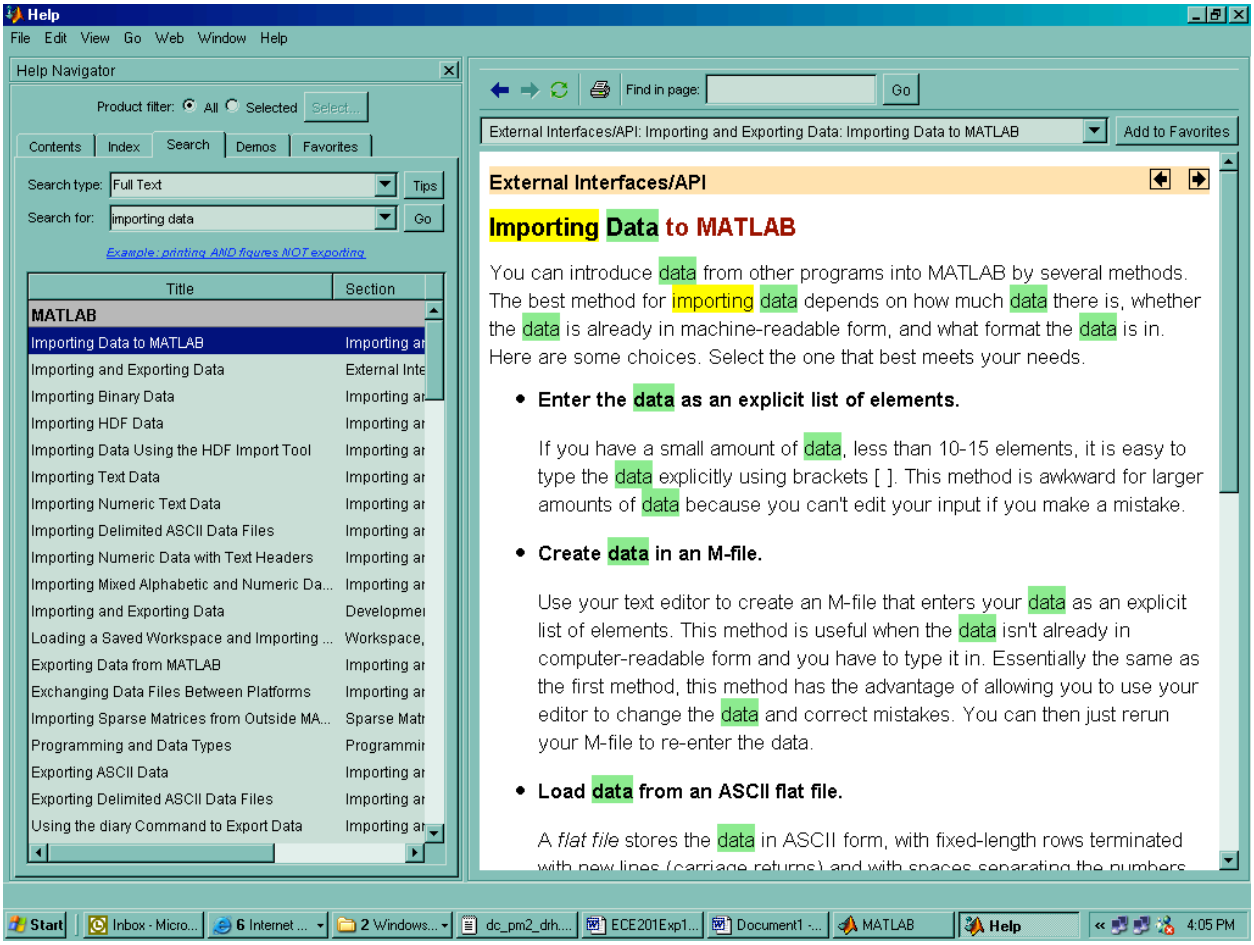


Figure 1. Example of MATLAB Help Page

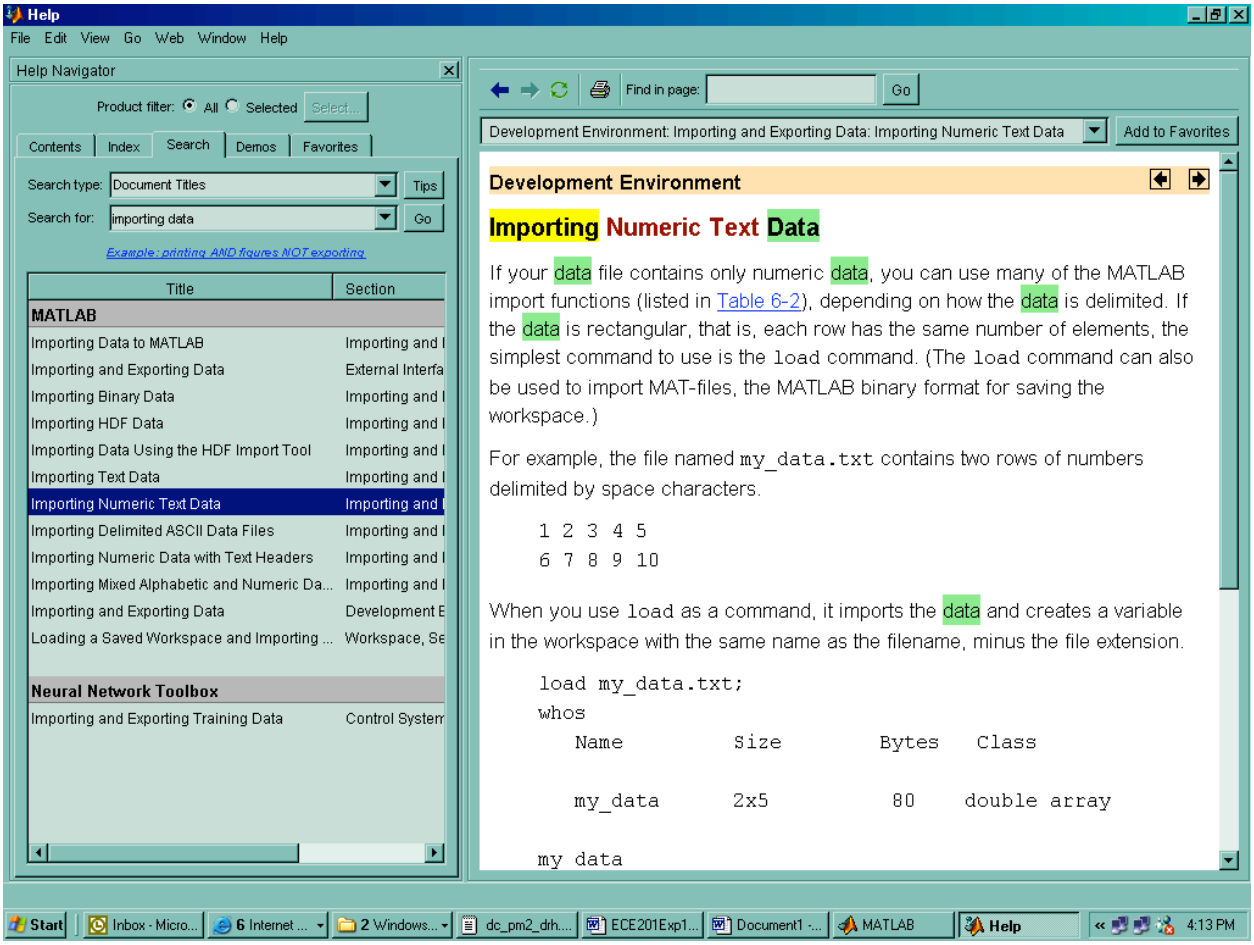
Note that the help window is divided into two panes, one allowing hierarchical browsing of help topics, and the other pane displaying the currently selected help topic. The left window has tabs that may be selected to reference topics via an "index", permit various types of text search, view a list of available demos of MathWorks products, and save links to particular help topics to a list of user "favorites".

If you need to find help on how to perform a task but do not know the name of the command you need, you can do a help search by doing a search in the Help window. In the left pane of the Help window, select the Search tab. (By default, a Full Text Search will be done, but you can limit your search to document titles, function names, etc. by changing the **Search Type** field. In the **Search For** field, type in the word (or words) you which to search for then click on "Go". A list of help topics related to importing data into MATLAB will be displayed. Click on the help topic you are interested in and that help topic will be displayed.

For example, type **importing data** in the **Search For** field, and click on “Go”; then click on **Importing data to MATLAB** to get an overview of methods of getting data into the MATLAB environment, as shown below:

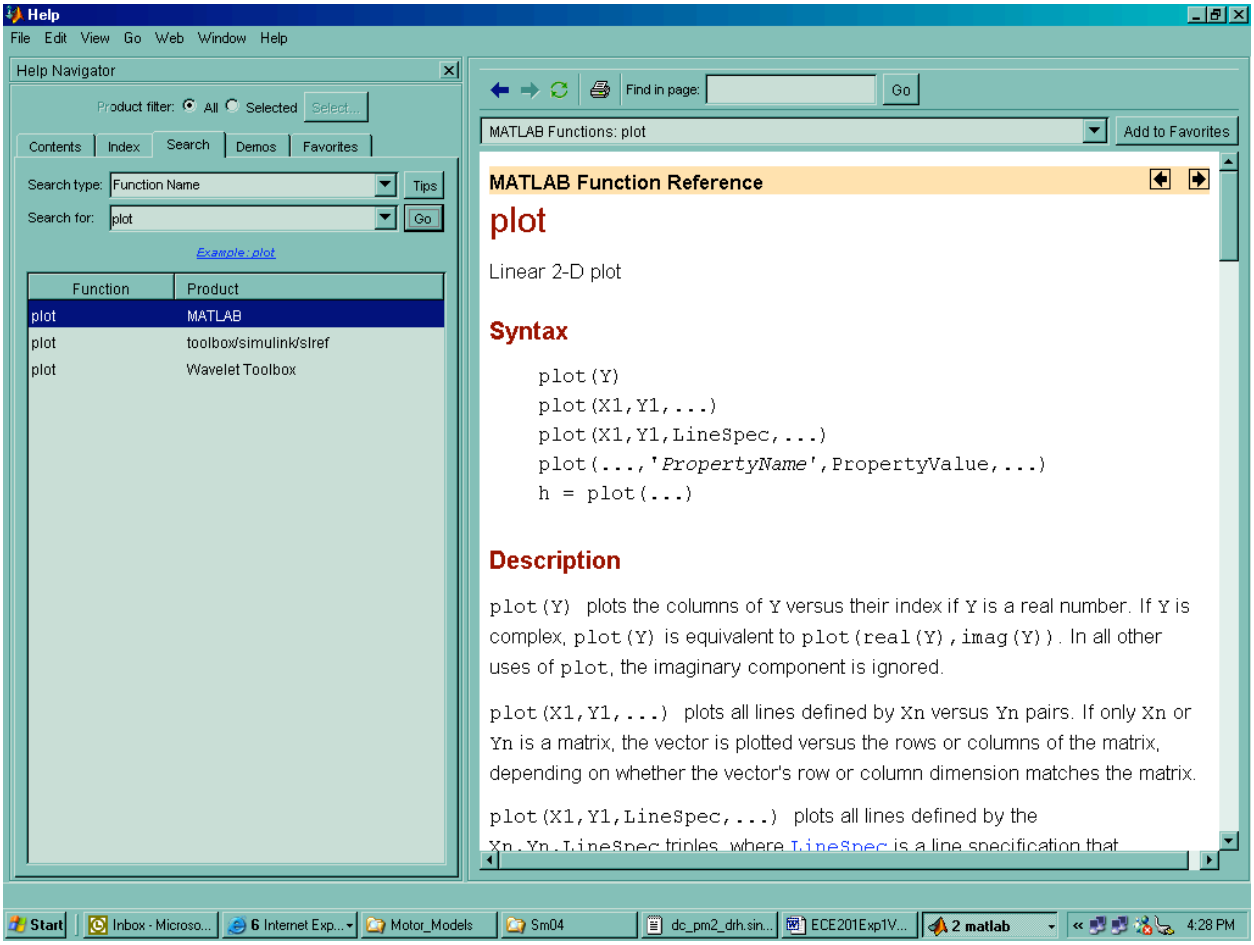


As you can see, there are many help references that have either “importing” or “data” in the text of the help topic. You can limit your search to only document titles by changing the **Search Type** field. As shown below, (after clicking “Go”, and selecting **Importing Numeric Text Data**) this returns a much shorter list of topics.



Any blue, underlined text in the help topic display pane is a hyperlink to another topic within the suite of documentation. Clicking on this hyperlink will take you to that topic. You may use the "arrow" buttons at the top of the display pane to go forward or back to previously viewed topics.

If you know the MATLAB function you need to use, you may limit your search to function names only by setting the **Search Type** field to **Function Name**. For example, to find help with the **plot** function, set the **Search Type** field to **Function Name** , and type "plot" in the **Search For** field and click on "Go". The following should appear:



You may also get much of this same information in the MATLAB command window by typing at a command prompt (“>>”)

» help plot

As a last resort, you can do a help search by using the function **lookfor**. The syntax for the **lookfor** function is:

»lookfor <keyword>

such as

»lookfor curve

MATLAB will search for and display all functions with the keyword in its description. Be warned, this can take up to 30 minutes to complete the search. Doing such searches in the Help window is much more efficient.

Much insight into what can be done in MATLAB and how to do it can be gained by running some of the available demos. You may access the demos by selecting the “demo” tab in the Help window, or by simply typing

» demo

at a MATLAB command prompt.

Browse the demos that are available, and run several, including Desktop Overview, Importing Data, Workspace, Help Browser, Basic Matrix Operations, and 2-D Plots. Don't worry about remembering everything you see demonstrated. Just get a feel for what can be done, and a flavor of the way MATLAB can be used to approach calculation and graphic display.

Answer Question 1.

REPRESENTING AND STORING NUMBERS

The most fundamental data structure used by MATLAB to store data is called a matrix. (In fact, MATLAB is short for MATrix LABoratory.) A matrix in MATLAB is a rectangular array of data, organized in rows and columns. Although MATLAB considers all stored data as matrices, it is convenient to think of three structures in MATLAB for storing data: Scalars (a 1x1 array), vectors (a 1xN row vector array or Nx1 column vector array), and matrices (a NxM array.)

SCALARS

A scalar is a single value, such as 2 or 6. Some examples of scalar declarations are:

```
» a = 5;
» b = 7.521;
```

The semicolon at the end of the statement suppresses the output to the command window. If you omitted the semicolon your command window transcript would look like this:

```
» a = 5

a =

    5
```

VECTORS

Vectors are defined as one-dimensional arrays. Vectors may be declared in MATLAB as either a row vector or a column vector. Row vectors can be declared in several ways, including:

```
» c = [1 3 5 7 9];
» d = [2,5,8,3,9];
```

The variable name is typed followed by the equal sign and then the values of the vector elements. Notice that the values must be enclosed in square brackets. Either a space or a comma must separate the values inside the brackets. A row vector can also be declared with the colon operator. For example:

```
» e = 1:1:10;
```

The syntax for variable definitions via the colon operator is

variable = start:incr:stop

where

start is the starting value

incr is the increment (or step) value
stop is the final value

In the above example, e was assigned the elements 1 through 10.

Column vectors are declared the same way except a semicolon separates the values. Since a column vector is the transpose of a row vector, a column vector can be declared by using the transpose operator (') on a row vector. The following are a couple of examples of declaring column vectors:

```
» f = [2;4;6;8;10];
» g = [8 6 5 3 7]';
```

MATRICES

A matrix is defined as two-dimensional array. Matrices are declared by combining the techniques used to declare vectors. Here are a couple examples of matrix declarations:

```
» A = [1 4 7;2 5 8;3 6 9]

A =

    1    4    7
    2    5    8
    3    6    9

» B = [[1 2 3]'[4 5 6]'[7 8 9]']

B =

    1    4    7
    2    5    8
    3    6    9
```

Conventionally, scalar and vector variable names use lower case letters and matrix variable names use upper case letters. This is the representation that will be used in this experiment and throughout the rest of the lab. Since MATLAB variable names are case sensitive, the following declarations create two separate matrices: f and F.

```
» f = [12 9 18];
» F = [12 9 18;13 6 24];
```

OPERATIONS

Array operations are performed element by element (i.e. on array values in the same (row, column) positions). For element by element operations, the matrices must be the same dimension (i.e. having the same number of rows and the same number of columns.) Array operations differ from matrix operations in that the operator is preceded by a period (.). For example, to perform an element by element multiplication of matrices A and B, the syntax would be

```
A .* B
```

To do element-by-element array multiplication, A and B must be of the same size.

This is not the same as the matrix multiplication operation

A * B

To multiply two matrices, they must be "conformable", i.e. A must have the same number of columns as B has rows. The result has the same number of rows as A and the same number of columns as B.

Table 1 and Table 2 list the element-by-element and matrix arithmetic operations for matrices. Notice the operators for addition and subtraction are the same for both element-by-element and matrix operations.

+	Addition	a + b
-	Subtraction	a – b
*	Multiplication	a * b
/	Division (Right Division) (Do "help mrdivide" to see what this does to matrices)	a / b
\	Left Matrix Division (Do "help mldivide" to see what this does to matrices)	a\b
^	Matrix Power	a ^ b

Table 1. Scalar Arithmetic Operators.

+	Addition	a + b
-	Subtraction	a – b
.*	Array Multiplication	a .* b
./	Right Array Division	a ./ b
.\	Left Array Division	a.\b
.^	Array Power	a .^ b

Table 2. Array Element-by-Element Arithmetic Operators.

BUILT IN VARIABLES

MATLAB comes with some built in constants. Some useful constants are:

pi	π
j	$\sqrt{-1}$
i	$\sqrt{-1}$

Answer Question 2 & 3.

MATLAB Built In Functions

MATLAB has many built in functions. Table 3 lists some of the functions that you will need to know in order to complete this lab. It is a good idea to start compiling your own table of commands for future reference. Available built in functions may be explored in the help system under the topic "MATLAB Functions: Functions - By Category"

max(x)	Maximum element
min(x)	Minimum element
mean(x)	Mean (average)
std(x)	Standard deviation
median(x)	Median
cos(x)	Cosine
sin(x)	Sine
exp(x)	Exponent
log(x)	Natural log
log10(x)	Log base 10

Table 3: Common MATLAB functions.

Answer Questions 4 – 8.

PLOTTING

There are many times when a visual presentation of data is useful. MATLAB has many helpful functions to display data visually. In this section, you will be introduced to the basics of MATLAB plotting.

Plot of a Single Data Set

Most of the plotting that will be done in this lab will be done with the use of the **plot** command. The syntax for the **plot** command is:

```
plot(x,y)
```

where

x is a vector representing the abscissa (x-axis)
y is a vector representing the ordinate (y-axis)

Type the following commands to plot $\sin(2\pi t)$ for $-5 \le t \le 5$:

```
» t = -5:.01:5;
» y = sin(2*pi*t);
» plot(t,y);
```

You should see a plot of a sine function that has a magnitude of 1 and a frequency of 1 rad/s

PLOTS OF MULTIPLE SINGLE DATA SETS

More than one function may be plotted on the same graph. The syntax to graph multiple plots is:

plot(x_1,y_1,x_2,y_2,…,x_n,y_n)

Type the follow commands to plot the previous sine function together with cos(2πt):

```
» z = cos(2*pi*t);
» plot(t,y,t,z);
```

COLORS, LINE STYLES, DATA MARKERS

Notice that the color for the sine function is blue and the color for the cosine function is green. By default, MATLAB assigns the first plot the color of blue, the second plot the color of green, and picks other colors for further plots. MATLAB cycles through all the available colors and, after the last color has been used, repeats the process from the beginning.

MATLAB allows you to override the default color and choose any available color. The syntax to plot in a specified color is:

plot(x,y,'c',…)

where

c is the variable for the desired color.

Notice that the color variable must be enclosed in single quotes.

Change the color of the sine and cosine plots to red and black respectfully by typing the following command:

```
» plot(t,y,'r',t,z,'k')
```

It is often useful to use different line styles or data markers to differentiate lines on a plot, or to show the actual discrete data values used to construct the plot. The plot command can take a Line Style specifier to set these line attributes.

For example, plot(x,y,'-.or')

plots y versus x using a dash-dot line (-.), places circular markers (o) at the data points, and colors both line and marker red (r). Specify the components (in any order) as a quoted string after the data arguments.

If you specify a marker, but not a line style, MATLAB plots only the markers, and does not connect them with lines.

For example, plot(x,y,'d')

plots blue diamonds at the (x,y) coordinates specified by the vectors x and y.

To see a list of all the Line Specifier properties that can be used for plotting, see help topic **MATLAB Functions: LineSpec** or type the command:

```
» help plot
```

Answer Question 9.

CHANGING AXIS LIMITS OF A PLOT

The default values for the axes of a plot can be overridden with the **axis** command. The syntax for the **axis** command is:

axis(v)
axis([xmin, xmax, ymin, ymax])

where v is a vector with the values [xmin, xmax, ymin, ymax]. xmin and xmax define the domain and ymin and ymax define the range of the plot.

Change the domain of the plot from $-5 \le t \le 5$ to $-\pi \le t \le \pi$ by typing the following:

```
» axis([-pi, pi, -1 1])
```

Even though we do not want to change the range, we still need to enter the range limits. Since the amplitude of the functions is 1, we will keep the range minimum and maximum limits at –1 and 1 respectively.

LABELING PLOTS

All plots should have a descriptive title and appropriate axis labels. The axis labels should include the type of measurement represented by the axis, as well as the unit of measurement, when applicable. Type in the following commands to label your plot:

```
» title('Sinusoid Functions')
» xlabel('Time (sec)')
» ylabel('Amplitude (volts)')
```

ADDING TEXT TO A PLOT

It is also possible to add text to the plot. Text can be added to the plot with the **text** and the **gtext** commands.

The **text** command adds the text string at the location specified by the coordinates (x,y). The syntax is:

```
text(x,y,'string')
```

where

x is the coordinate of the independent axis to place the text
y is the coordinate of the dependent axis to place the text
string is the text to add to the plot. The string must be bracketed by single quotes

The **gtext** command displays the active graph window with a set of crosshairs. The mouse is used to position the crosshairs. Pressing any mouse button causes the text to be written at the selected location. The syntax is

gtext('string')

Use the gtext command to label the different plots on the graph by typing the following commands:

```
» gtext('sin(2*pi*t)')
» gtext('cos(2*pi*t)')
```

CHANGING PLOT PROPERTIES

Occasionally you will want to change some of the specific properties of a plot such as the width of the lines used. For example, type the following sequence of commands.

```
» t = 0:0.1:15;
» y = exp(-0.5*t).*sin(2*pi*t);
» h = plot(t,y);
```

The variable **h** contains a number that uniquely identifies the line generated by the plot command. Use the **get** command to see a complete list of the properties of the line **h** that you can change.

```
» get(h)
```

You can use the **set** command to get a list of the possible options you can use for each property.

```
» set(h)
```

You can also use the set command to see the possible options for a specific property. For example, type the following command to see the possible markers that can be used to identify each point in the plot.

```
» set(h,'Marker')
```

Type the following sequence of commands and observe how each one modifies the plot.

```
» set(h,'Marker','.');
» set(h,'MarkerSize',15)
» set(h,'Color',[0 0.5 0]); % Dark Green
» set(h,'LineWidth',1.5);
```

You can also change the properties of the figure and of the axis that contains the plot. Both the figure and the axis have their own handles that uniquely identify them and their properties. Type the following commands to get the handle of the figure and list its properties.

```
» hf = gcf;
» get(hf)
```

Note that the command **gcf** returns the handle of the current active figure. Type the following commands to get the handle of the axis and list its properties.

```
» ha = gca;
» get(ha)
```

Note that the 'Children' property of the axis is the handle returned by the plot command. Note also that the 'Parent' property of the axis is the handle of the figure.

Type the following command to turn off the bounding box of the axis, to plot a tick mark for every second rather than every 5 seconds, and to put grid marks for each tick on the y-axis.

```
» set(ha,'Box','off');
» set(ha,'XTick',0:1:15);
» set(ha,'YGrid','On')
```

NOTE: Do not close this figure because you want to save it to a file and include it in your lab report, as discussed in later sections.

Table 4 summarizes the commands discussed in this section.

gcf	Returns the handle of the current figure.
gca	Returns the handle of the axis of the current figure.
set(h)	List parameters of handle h.
set(h,'PropertyName')	List the options for a parameter of handle h
set(h,'PropertyName','PropertyValue')	Changes parameter value of PropertyName to value specified by 'PropertyValue'.

Table 4. Summary of the commands used to change properties of plots.

LOGARITHMIC PLOTS

MATLAB can generate plots using logarithmic axis scales. Table 5 lists the commands for logarithmic plots. Figure 2 shows the function $f(x) = 3x^2$, $0 \le x \le 1000$, using the various combinations of linear and logarithmic axis scalings.

plot(x,y)	Linear scale for both axes.
semilogx(x,y)	Log scale for the abscissa (x-axis) only.
semilogy(x,y)	Log scale for the ordinate (y-axis) only.
loglog(x,y)	Log scale for both axes.

Table 5. Summary of 2-D plot commands.

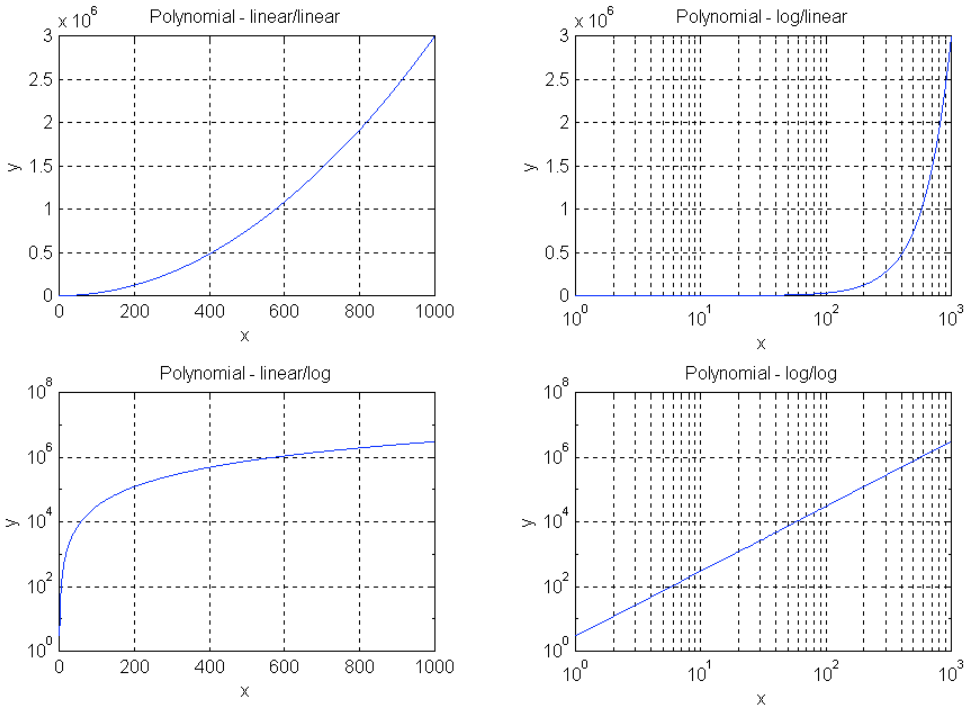


Figure 2. Example of logarithmic scales for exponential plots.

PRINTING AND SAVING PLOTS

The **print** command is used to make hard copies of your plot as well as saving the plot to a file. To list all of the options for the print command, view Help topic "MATLAB Functions: print, printopt" or type the following:

```
» help print
```

Although there are numerous options for the print command, only three are discussed in this lab.

Printing the Plot

The syntax to print a copy of the current plot at the default printer is:

```
» print
```

You may also print to a specific printer by going to the file menu in the figure window and using File ⇒ Print.

To see how the figure will look printed before you print it, in the figure window use File ⇒ Print Preview.

You can change the position and size of the figure by setting the figure's PaperPosition property. The following sequence of commands moves the figure so that the bottom left corner is 2.5 inches from the margin, the bottom of the figure is 1 inch from the bottom of the page, the width of the figure is 3 inches, and the height of the figure is 2 inches.

```
» LLC = 2.5;
» LBC = 1.0;
» WD = 4.2;
» HT = 3.1;
» set(gcf,'PaperPosition',[LLC LBC WD HT])
```

Go to Print Preview again to see how the figure is displayed on the page.

Saving the Plot as a TIFF File

If you want to save the plot as a graphic file so you can import it into other programs, type the print command with the following syntax:

print –dtiff <filename.tiff>

where

–dtiff saves the file in TIFF format.
<filename.tiff> is the name, including the path, the file is saved as

The PaperPosition width and height properties control the width and height of the Test.tiff image.

Answer Question 10.

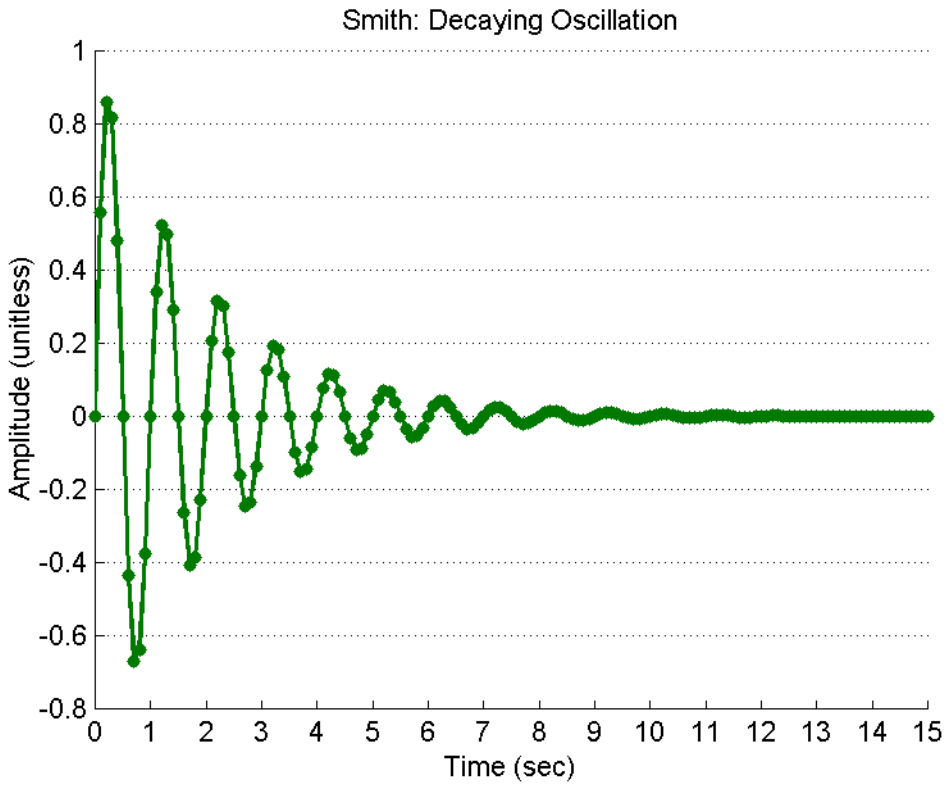


Figure 3. Example of an image created by the –dtiff option of the print command.

Saving the Plot as a JPEG File

If you want to save the file in JPEG format (for viewing through a web browser) use the **print** command with the following syntax:

print –djpeg <filename.jpg>

where

-djpeg saves the file in JPEG format
<filename.jpg> is the name, including the path, the file is saved as

CLOSING THE PLOT WINDOW

When you are finished plotting your data and all copies of your plot have been printed or saved, you need to close the plot window. Closing the plot window can be done in many ways.

1. From the menu toolbar select File⇒Close.
2. Press ALT-F, C.
3. Press ALT-F4.
4. From the command prompt, type close.

FILE ACCESS

THE MATLAB WORKSPACE

When you exit MATLAB, all variables are cleared from memory. If you want to store your variables for future use you will have to save them to a file. Variables declared in MATLAB are collectively referred to as the workspace. To display the workspace on the screen type the command **whos**. If you type whos now you should see something like

```
» whos
  Name      Size          Bytes  Class

  A         2x2              32  double array
  ans       1x1               8  double array
  b         2x1              16  double array
  x         2x1              16  double array

Grand total is 9 elements using 72 bytes
```

To clear all variables from the workspace you can use the command **clear all**. To clear a specific variable from the workspace, such as A, type **clear A**.

SAVING THE WORKSPACE

To save the workspace, from the menu toolbar, select File ⇒ Save Workspace As, and enter the name for the file to save the workspace under. To load a saved workspace select File ⇒ Load Workspace and select the file you want to load.

Workspaces can also be saved and loaded from within the MATLAB command window. The commands and syntax are

```
save filename
load filename
```

LOADING VARIABLES FROM A TEXT FILE

MATLAB can also read text files containing numbers created by the user or other programs. Create an ASCII text file that represents the matrix

```
3  7  11
2  3  9
4  5  8
```

Start by opening a text editor such as Notepad (Start ⇒ Run ⇒ Notepad). Type in the matrix of numbers just as you see it above, using spaces between the numbers. Save it as the file "Test.txt" and exit the editor. At the MATLAB command prompt type the command

```
» load Test.txt
```

When used this way, the load command stores the values in the text file in a variable name which is the same as the filename without the extension. Using the command whos, you will see a variable named Test, which is a 3 X 3 matrix.

If you want to load the contents of a file into a specific variable name, the syntax of the **load** command is:

variable = load('filename')

where

variable is the name of the variable you wish to load the values into
filename is the name of the file to load the data from

Answer Question 11.

EXITING MATLAB

When you are finished using MATLAB you can exit the program by selecting File ⇒ Exit from the menu toolbar. Another alternative way to exit is to simply use MATLAB's **exit** or **quit** command at the command prompt. When you quit the MATLAB command window, all other MATLAB windows, such as figures, will be closed and all unsaved data is erased.