Lab Exercise 1: MATLAB

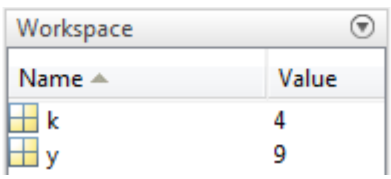
Pre-lab Assignment: No

Goals:
To introduce MATLAB functions of: creating variables, plotting, for loops, symbolic variables, and functions.

Principles:
MATLAB is a programming environment that can be used for a variety of mathematical undertakings, including doing numerical computation, analyzing data, creating algorithms and more.

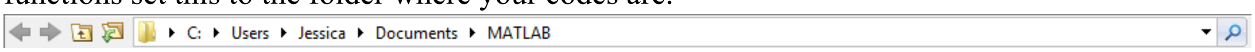
MATLAB windows:

- Workspace: holds the variables that you are using. Once you type in k = 7, the k will be stored into the workspace until you clear all the variables. If you have not cleared the workspace, you can call the k value in any function, e = k+3. If you input k = 4, the new value of k that is stored it 4.

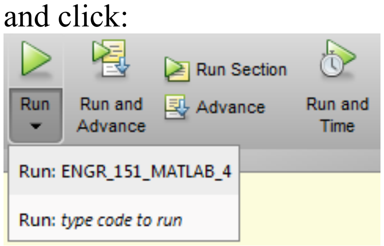


- Command window:

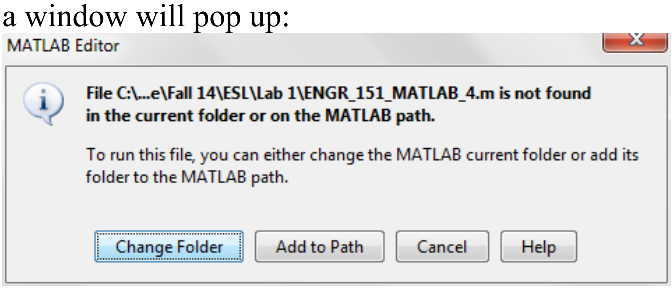
- Working directory: the folder that MATLAB will look in to find called functions. To use functions set this to the folder where your codes are.



If you want to change the working directory to the folder your file is in, open your file and click:



a window will pop up:



Then change folder.

Scripts: Runs lines of code.

Functions: has a specific thing that is does. Can perform simple or complex operations

Anything after a percent sign (%) is a comment. To create a comment in the script, simply use a percent sign and the commented text will change in color. There are various comments in the script guiding you through the process.

Procedure:
1. Save the ENGR_151_MATLAB file provided.
2. Open MATLAB
3. Open ENGR_151_MATLAB_2.m (this is called a script)
4. Follow the commands in the script

Discussion Items for Report:
1. Why does #1 need a "." but #2 does not?
2. A simple for loop that adds every other number starting at 1 to 11 (so add 1+3+5+7+9+11 to get 36)
3. A function file that is y = sin(x) calculate the integral from 0 to 2π using the trapezoid function. Why isn't the value you get exactly zero?

References:

MathWorks

Lab Exercise #2: Spreadsheet Tools

Pre-lab assignment: No

Goals:
To introduce spreadsheet functions necessary for completing course lab reports. Examples and functions will be given using Microsoft Excel, however any spreadsheet program can be used. (To download a free productivity suite similar to Microsoft Office (and fully compatible), visit www.openoffice.org.)

Items Needed:
In order to complete this assignment, students will need:

- Access to Microsoft Excel or similar spreadsheet program
- The assignment template titled *Engr151-Lab1.xls* (For versions of Excel older than 2007)
- The assignment template titled *Engr151-Lab1.xlsx* (For versions of Excel 2007 and newer)

Introduction:
The lab experiments for this course will require each student to submit a lab report. For most reports, there will be a data set that needs to be analyzed, organized and used for further calculations. Programs such as Excel are excellent for performing these functions. *This assignment assumes each student was present during the Excel instructional lab session during the first week since this document does not contain full instructions for completing the tasks.* The tasks below will help each student learn the common tools needed to take advantage of Excel's capabilities and to use it for completing lab reports.

I. Analyzing Data
Excel contains several functions to assist in the analysis of data. For example, if a large set of numbers is given, Excel can quickly determine the sum, average, standard deviation, median, etc. In Excel 2007 (and later versions), these functions can be reviewed by clicking on the *Formulas* toolbar and the *Insert Function* icon.

Problem #1

In the given Excel spreadsheet, find the area labeled *Problem #1*. For the given data, determine the following: Sum, Average, and Median.

II. Writing Equations
One of the most powerful tools of Excel is the ability to write equations and plug multiple values of the variables into those equations. When inserting an equation into a cell, always start with the = symbol.

Problem #2

In the given Excel spreadsheet, find *Problem #2*. The column of data represents the independent (x) variable in the equation $y = 4x -17x2$. In the space provided, input the equation $y = 4x -17x2$ for all values of x. When writing equations, there are often terms that might be variables that usually remain constant, but may require changing at times. For example, in the equation $y = 4x -17x2+b$, the value of b might be something that is usually a constant, but might change occasionally. If a lot of equations exist, it would be very tedious to change each value of b, if necessary. To avoid this, *absolute cells* can be utilized.

Problem #3

Similar to problem #2, values of *x* are given, but an *absolute cell* is included for the value of *b*. Insert the equation $y = 4x - 17x2 + b$, making sure to input the cell for *b* as absolute. To do so, place the $ sign in front of the cell name. So, the cell containing the first equation of *y* should look like this:
= 4*G6 – 17*G6^2 + G3
After completing this equation once, copy it down. Practice changing the value of *b*. When finished, set the value of *b* at 14.8.
It is important to note that all angles must be converted into radians before using the values in equations that contain functions such as *sin* or *cos*. Multiplying values by π/180 or using the Radians function can do this.

Problem #4

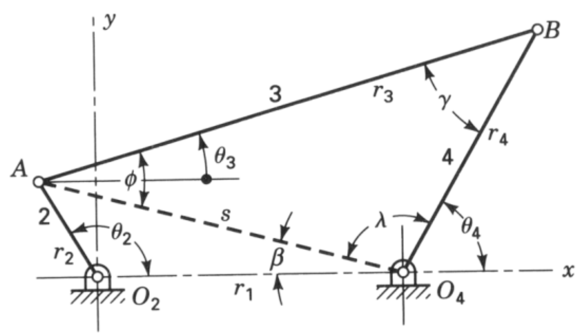
Several angular values (*θ)* are given. Convert the values into radians and then use the values to complete the equation $y = 1.2\sin(1.4\theta) - \cos(2.23\theta)$
Complete the problem below for more practice writing equations.

Problem #5

The force exerted by a spring is given by $F = -kx2$, where *F* is the force, in Newton's; *x* is the displacement of the spring from the equilibrium position, in centimeters; and *k* is a spring constant. Prepare a column of *x* values that varies from 0 to 15 cm increments of 1 cm. Include two absolute cells for *k* that represent two springs ($k1$ = 0.1 N/cm and $k2$ = 0.05 N/cm). Complete the task by creating two columns of corresponding values of *F* for the springs.
From this point on, create your own spreadsheet file (or use blank space in the current file) to complete the remaining problems.

Problem #6

The mechanism below is a "four-bar linkage system" and is used in different types of machines.



A motor is contacted at the end of member 2 at point O2 and runs continuously. The other members, connected by smooth pins, oscillate in a manner dependent on their length. The angles between members are:

$$\beta = \cos^{-1}\frac{r_1^2 + s^2 - r_2^2}{2r_1 s}$$

$$\psi = \cos^{-1}\frac{r_3^2 + s^2 - r_4^2}{2r_3 s}$$

$$\lambda = \cos^{-1}\frac{r_4^2 + s^2 - r_3^2}{2r_4 s}$$

(Not shown in figure)
Where,

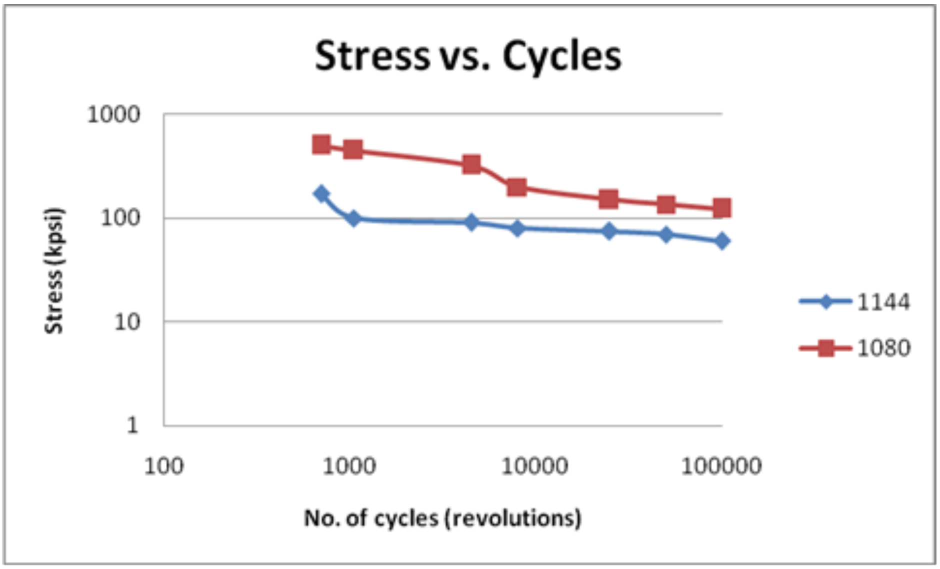
$$s = \left[r_1^2 + r_2^2 - 2r_1 r_2 \cos\theta_2\right]^{1/2}$$

In the equations above, β, ψ and λ will be in radians. Remember when entering the formula for *s* in Excel, θ2 must be in radians. Calculate the values of Beta (β), Psi (ψ) and Lambda (λ) for 1 complete revolution of θ2 in 10° increments. Let r1, r2, r3, and r4 be "absolute cells". (r1 = 10, r2 = 4, r3 = 10, r4 = 12)
The format below is suggested for entering the equation

Problem #6 and 7					
r_1	10				
r_2	4				
r_3	10				
r_4	12				
θ_2	θ_2	s	β	ψ	λ
Degrees	Radians			Radians	
0	0.0000	6.000	0.000	1.638	0.982
10	0.1745	6.100	0.114	1.626	0.983
20	0.3491	-	-	-	-
-	-	-	-	-	-
-	-				

III. Graphing Data

The ability to create graphs from large sets of data is a very useful feature of spreadsheets. Before discussing the details, it is important to discuss the format first. In order to create consistent, professional-looking graphs, all students must prepare graphs in a particular manner, as will be described. The image below is an example:



Graphing requirements
1. The format for chart title is: *Name of y-axis* vs *Name of x-axis*. Do not include units.
2. Label both axes and include units.
3. Include a legend if more than one curve is present on the graph.
4. Use different shapes for the curve markers (square and diamond in the graph above). Using different colors is not enough since most printing will be done in black and white.
5. Generally speaking, there should be no connecting lines or curves between the curve markers if data has been obtained experimentally. This implies a relationship between the data points that may or may not be valid. (The example graph above does show connecting lines for illustration purposes.) The decision to add a connecting curve will be up to each student, but it needs to be mentioned that this is generally not standard practice.
6. Some assignments may ask for a curve-fit (referred to as a "trend line" in Excel). This is different than connecting the data points. A trend line applies a best-fit curve that fits a particular mathematical function that may be linear, exponential, polynomial, etc. The best-fit curve may or may not pass through the actual data points.

Problem #7

Use your results from Problem #6 to plot the values of β, ψ and λ **on a single graph.** Use the following titles/labels:
Title: *Linkage Angle vs. Driver Angle*
x-axis label: *Driver Angle (Degrees)*
y-axis label: *Linkage Angle (Radians)*
Legend names: *Beta, Psi, Lambda* (don't need to use symbols)

Problem #8

Plot the data below on a graph. Include a trend line using a fourth order polynomial. Add the equation to the graph. (Chart title, axis labels and legend not required)

X	Y
1	8
5	500
10	40,200
20	-7,500,000

50	120,007,525
75	75,371,913
100	301,050,050

Printing Instructions

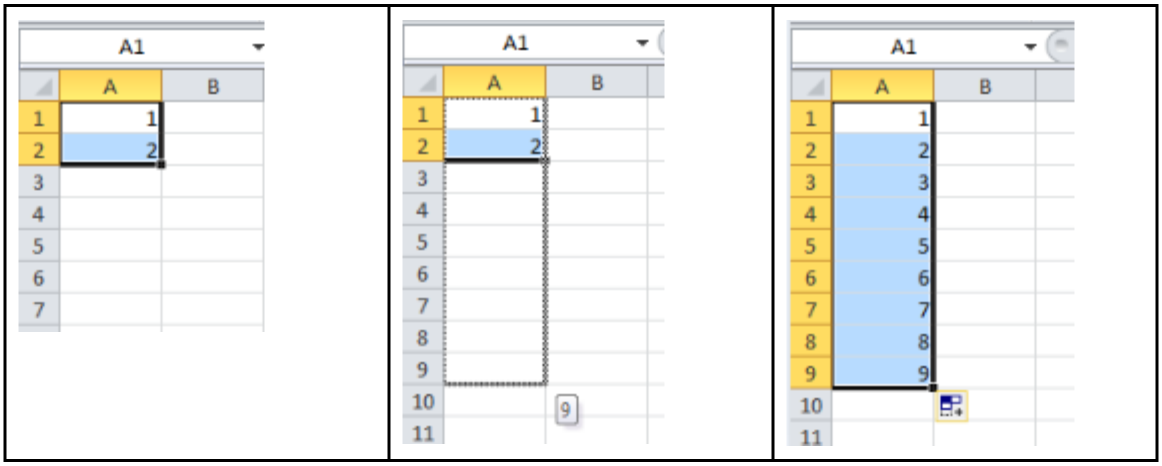
Turn in this assignment combined with the following problems (use landscape printing when necessary).

- Print 1, 2, and 3 on a single page.
- Print 4 and 5 on a single page.
- Print 6 and 7 on a single page (include data and graphs).
- Print 8 on a single page (include data and graph).

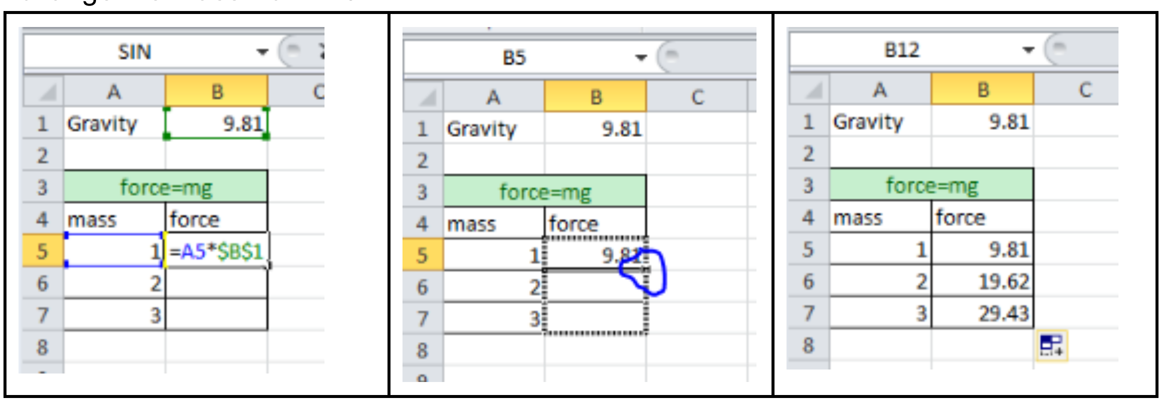
Excel Reference Sheet

Shortcuts:

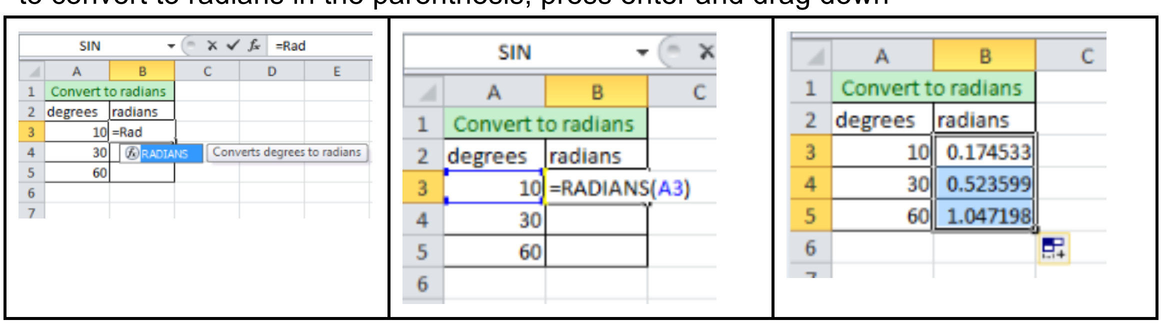
1. To number boxes simply highlight first few numbered boxes and drag down until you reach desired number.



2. To make a constant add a $ symbol in front of the letter and number identifying the box that contains the constant. Again you can drag down the clicking on the corner of the box (circled in blue) and dragging it down. Excel will keep the 9.81 as a constant but change the mass from 1-3.

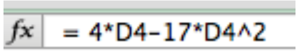


3. To convert from degrees to radians simply add the "= Radians(#)" enter the box you wish to convert to radians in the parenthesis, press enter and drag down

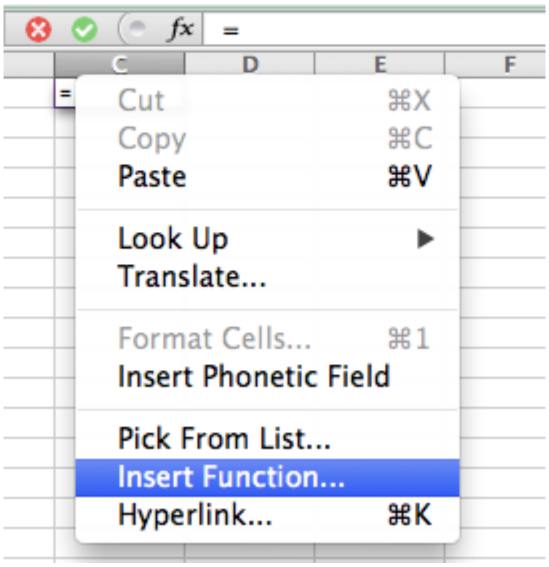


4. To start a function you must always place an "=" sign before it (Refer to image in number 5 for example).

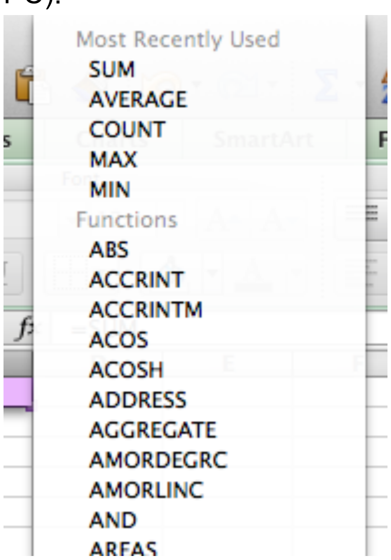
5. Every time you want to multiply any number, make sure that there is a * before whatever you are multiplying.



6. If you are having a hard time with writing a certain function. Most of them will be located in Excel itself. You will simply do the equal sign as you always do when starting to write a function and right click the cell. Once you do this you will get a drop down menu and you will see “Insert Function” (Note: this is on a Mac. On a PC it might be worded or look differently).

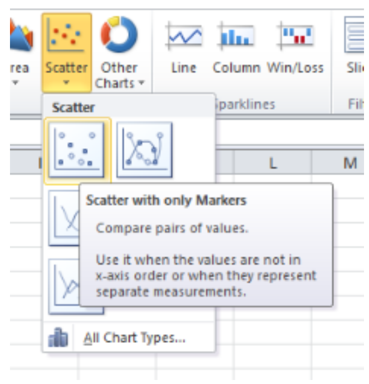


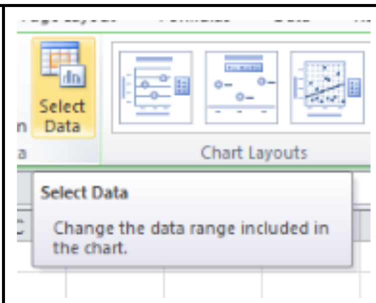
Once you click on this you will see a drop down menu of a vast amount of functions that you will be able to choose from (Again: This is on a Mac. It may look different on a PC).

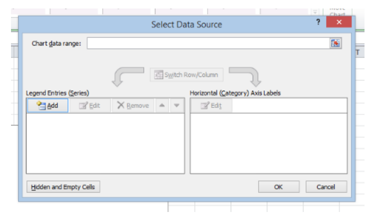


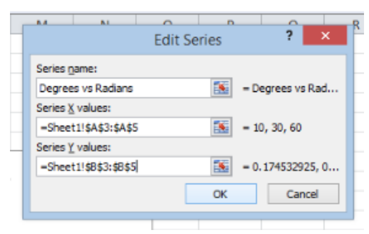
7. Making graphs for PC
   7.1. select scatter plot option > select data > delete default data in the “Select Data Source” > add > Give the series a title > Highlight and drag over your x-axis data

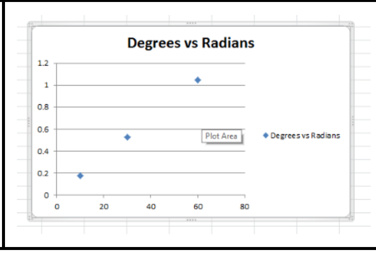
> highlight and drag over your y-axis data. Hint: delete any default data already listed under “series X values” and Series Y values”





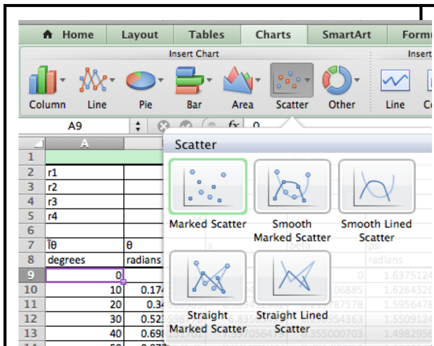


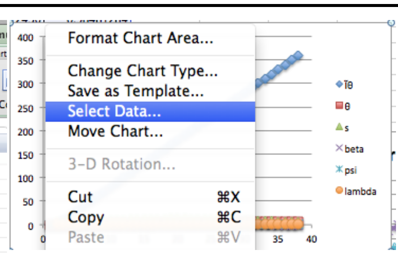


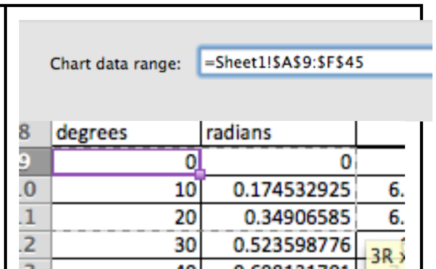


8. Making graphs for Macs

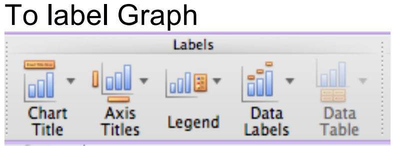
For square 3 you are selecting you’re date. You will choose your first cell with data and then continue to choose all of the them. It is demonstrated in this box for the first few cells but you will need to do so for all the data you will be using in your graph.







To label Graph



MATLAB Cheat Sheet

Basic Commands

`%`	Indicates rest of line is commented out.
`;`	If used at end of command it suppresses output. If used within matrix definitions it indicates the end of a row.
`save filename`	Saves all variables currently in workspace to file `filename.mat`.
`save filename x y z`	Saves x, y, and z to file `filename.mat`.
`save -append filename x`	Appends file `filename.mat` by adding x.
`load filename`	Loads variables from file `filename.mat` to workspace.
`!`	Indicates that following command is meant for the operating system.
`...`	Indicates that command continues on next line.
`help function/command`	Displays information about the function/command.
`clear`	Deletes all variables from current workspace.
`clear all`	Basically same as clear.
`clear x y`	Deletes x and y from current workspace.
`home`	Moves cursor to top of command window.
`clc`	Homes cursor and clears command window.
`close`	Closes current figure window.
`close all`	Closes all open figure windows.
`close(H)`	Closes figure with handle H.
`global x y`	Defines x and y as having global scope.
`keyboard`	When placed in an M-file, stops execution of the file and gives control to the user's keyboard. Type `return` to return control to the M-file or `dbquit` to terminate program.
`A=xlsread('data',...` `'sheet1','a3:b7')`	Sets A to be a 5-by-2 matrix of the data contained in cells A3 through B7 of sheet `sheet1` of excel file `data.xls`
`Succes=xlswrite(...` `'results',A,'sheet1','c7')`	Writes contents of A to sheet `sheet1` of excel file `results.xls` starting at cell C7. If successful `success=` 1.
`path`	Display the current search path for `.m` files
`addpath c:\my_functions`	Adds directory `c:\my_functions` to top of current search path.
`rmpath c:\my_functions`	Removes directory `c:\my_functions` from current search path.
`disp('random statement')`	Prints `random statement` in the command window.
`disp(x)`	Prints only the value of x on command window.
`disp(['x=',num2str(x,5)])`	Displays `x=` and first 5 digits of x on command window. Only works when x is scalar or row vector.
`fprintf(...` `'The %g is %4.2f.\n', x,sqrt(x))`	Displays `The 3 is 1.73.` on command window.
`format short`	Displays numeric values in floating point format with 4 digits after the decimal point.
`format long`	Displays numeric values in floating point format with 15 digits after the decimal point.

Plotting Commands

`figure(H)`	Makes H the current figure. If H does not exist is creates H.

1

	Note that H must be a positive integer.
`plot(x,y)`	Cartesian plot of x versus y.
`plot(y)`	Plots columns of y versus their index.
`plot(x,y,'s')`	Plots x versus y according to rules outlined by s.
`semilogx(x,y)`	Plots $\log(x)$ versus y.
`semilogy(x,y)`	Plots x versus $\log(y)$.
`loglog(x,y)`	Plots $\log(x)$ versus $\log(y)$.
`grid`	Adds grid to current figure.
`title('text')`	Adds title `text` to current figure.
`xlabel('text')`	Adds x-axis label `text` to current figure.
`ylabel('text')`	Adds y-axis label `text` to current figure.
`hold on`	Holds current figure as is so subsequent plotting commands add to existing graph.
`hold off`	Restores hold to default where plots are overwritten by new plots.

Creating Matrices/Special Matrices

`A=[1 2;3 4]`	Defines A as a 2-by-2 matrix where the first row contains the numbers 1, 2 and the second row contains the number 3, 4.
`B=[1:1:10]`	Defines B as a vector of length 10 that contains the numbers 1 through 10.
`A=zeros(n)`	Defines A as an n-by-n matrix of zeros.
`A=zeros(m,n)`	Defines A as an m-by-n matrix of zeros.
`A=ones(n)`	Defines A as an n-by-n matrix of ones.
`A=ones(n,m)`	Defines A as an m-by-n matrix of ones.
`A=eye(n)`	Defines A as an n-by-n identity matrix.
`A=repmat(x,m,n)`	Defines A as an m-by-n matrix in which each element is x.
`linspace(x1, x2, n)`	Generates n points between $x1$ and $x2$.

Matrix Operations

`A*B`	Matrix multiplication. Number of columns of A must equal number of rows of B.
`A^n`	A must be a square matrix. If n is an integer and $n > 1$ than `A^n` is A multiplied with itself n times. Otherwise, `A^n` is the solution to $A^n v_i = l_i v_i$ where l_i is an eigenvalue of A and v_i is the corresponding eigenvector.
`A/B`	This is equivalent to `A*inv(B)` but computed more efficiently.
`A\B`	This is equivalent to `inv(A)*B` but computed more efficiently.
`A.*B,A./B,` `A.\B,A.^n`	Element-by-element operations.
`A'`	Returns the transpose of A.
`inv(A)`	Returns the inverse of A.
`length(A)`	Returns the larger of the number of rows and columns of A.
`size(A)`	Returns of vector that contains the dimensions of A.
`size(A,1)`	Returns the number of rows in A.
`reshape(A,m,n)`	Reshapes A into an m-by-n matrix.

`kron(A,B)`	Computes the Kronecker tensor product of A with B.
`A = [A X]`	Concatenates the m-by-n matrix A by adding the m-by-k matrix X as additional columns.
`A = [A; Y]`	Concatenates the m-by-n matrix A by adding the k-by-n vector Y as additional rows.

Data Analysis Commands

`rand(m,n)`	Generates an m-by-n matrix of uniformly distributed random numbers.
`randn(m,n)`	Generates an m-by-n matrix of normally distributed random numbers.
`max(x)`	If x is a vector it returns the largest element of x. If x is a matrix it returns a row vector of the largest element in each column of x.
`min(x)`	Same as `max` but returns the smallest element of x.
`mean(x)`	If x is a vector it returns the mean of the elements of x. If x is a matrix it returns a row vector of the means for each column of x.
`sum(x)`	If x is a vector it returns the sum of the elements of x. If x is a matrix it returns a row vector of the sums for each column of x.
`prod(x)`	Same as `sum` but returns the product of the elements of x.
`std(x)`	If x is a vector it returns the standard deviation of the elements of x. If x is a matrix it returns a row vector of the standard deviations for each column of x.
`var(x)`	Same as `std` but returns the variance of the elements of x.

Conditionals and Loops

```
for i=1:10
  procedure
end
```

Iterates over `procedure` incrementing i from 1 to 10 by 1.

```
while(criteria)
  procedure
end
```

Iterates over `procedure` as long as `criteria` is true.

```
if(criteria 1)
  procedure 1
elseif(criteria 2)
  procedure 2
else
  procedure 3
end
```

If `criteria 1` is true do `procedure 1`, else if `criteria 2` is true do `procedure 2`, else do `procedure 3`.