Matlab Tip Sheet

COMMAND WINDOW

The Command Window is where MATLAB code is entered and run and where most output is displayed. Directly, it can be used to do all sorts of calculations.

BASIC OPERATIONS

diary diaryName *%MATLAB Code* *diary off*		%To save your work in the Command Window, you must place your MATLAB code between these two commands. diaryName is the name of the file where your work will be saved.
;		%Placed at the end of a command, the semi-colon suppresses output. Useful when output is large vectors or matrices.
%Comment		%Allows you to comment your code
%Clear		%Clears the variables from the workspace
clc		%Clears the command window
1234 + 4567	*ans =* *5801*	%You can use MATLAB as you would a regular calculator
*5*6*9/3*	*ans =* *90*	
ans	*ans =* *90*	%Recalls the answer from the previous line if it has been assigned
ans/9	*ans =* *10*	
(1+2)(1+2)*	*ans =* *9*	

BASIC OPERATION

2/3	*ans =* *0.6667*	%Fractions will be automatically evaluated to floating point numbers
pi	*ans =* *3.1416*	%pi
r =10	*ans =* *10*	%Assigns 10 to the variable ' *r* '
Area= pi r^2*	*area =* *314.1593*	%Area is now computed with the variable ' *r* '
r = 5	*r =* *5*	%Assigns 5 to the variable ' *r* '
Area	*area =* *314.1593*	%Area is not computed with the variable ' *r* '
Area= pi r^2*	*area =* *78.5398*	%Area is now computed with the new variable ' *r* '
Clear area		%Clears data from the 'Area' variable

VECTORS

v1 = [0 1 2 3 4]	*V 1 =* *0 1 2 3 4*	%Creates a row vector, v1, with the elements [1 2 3 4 5]
v2 = [2 3 4 5 6]	*V 2 =* *2 3 4 5 6*	%Creates a row vector, v2, with the elements [2 3 4 5 6]
v3 = [5; 6; 7; 8; 9]	*V 3 =* *5* *6* *7* *8* *9*	%Creates a column vector, v3, with the elements [5 6 7 8 9]
v1 + v2	*ans =* *2 4 6 8 10*	%Adds vector v1 to v2

VECTORS

*2*v1*	*ans =* *0 2 4 6 8*	%Scalar multiplication of vector
*v1*v3*	*ans =* *80*	%Vector multiplication
v1'	*ans =* *0* *1* *2* *3* *4*	%Finds the transpose of vector v1
*v1*v2'*	*ans =* *50*	%Vector multiplication (row vector × column vector) %Multiplication of v1 and v2 transpose
*v1.*v2*	*ans =* *0 3 8 15 24*	%Multiplies the first element of v1 by first element of v2,the second element of v1 by second element of v2, etc. %Dot operator used
v1(3)	*ans =* *2*	%Third element of vector v1
v1(2:4)	*ans =* *1 2 3*	%Elements 2 to 4 of vector v1
m = [1:10]	*m =* *1 2 3 4 5* *6 7 8 9 10*	%Creates a vector with elements [1 2 3 4 5 6 7 8 9 10]
n=[0:2:10]	*n =* *0 2 4 6 8* *10*	%Creates a vector from 0 to 10 with values incrementing by 2

VECTORS

p=[0:1:100];		%Creates a vector from 0 to 100 with values incrementing by 1; the semi-colon supresses the output

MATRICES

A= [1 2 3; 3 2 1; 1 3 2]	*A =* *1 2 3* *3 2 1* *1 3 2*	%Defines matrix A
B= [4 5 6; 6 5 4; 4 6 5]	*B =* *4 5 6* *6 5 4* *4 6 5*	%Defines matrix B
A+B	*ans =* *5 7 9* *9 7 5* *5 9 7*	%Matrix addition
*2*A*	*ans =* *2 4 6* *6 4 2* *2 6 4*	%Scalar multiplication
*A*B*	*ans =* *28 33 29* *28 31 31* *30 32 28*	%Matrix multiplication
*A.*B*	*ans =* *4 10 18* *18 10 4* *4 18 10*	%Element by element multiplication using the dot operator
inv(A)	*ans =* *0.0833 0.4167 -0.3333* *-0.4167 -0.0833 0.6667* *0.5833 -0.0833 -0.3333*	%Finds the inverse of A

MATRICES

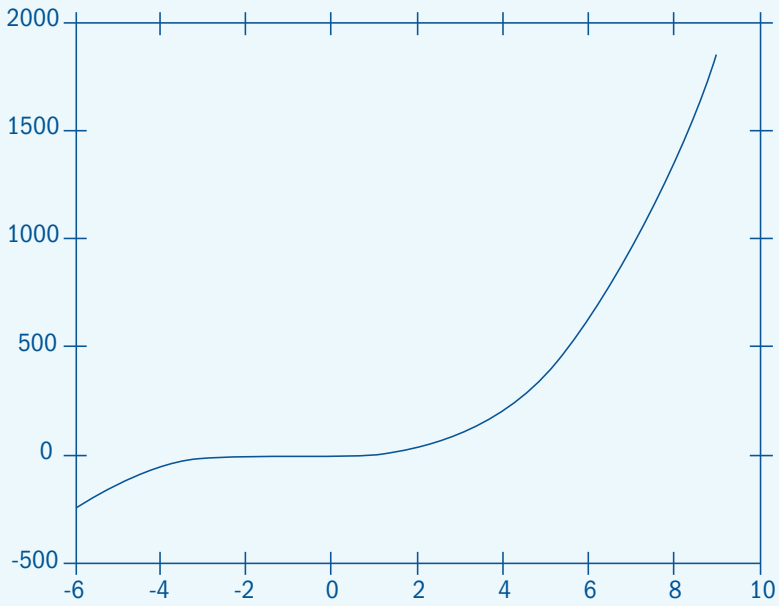
det(A)	*ans =* *12.0000*	%Finds the determinant of A
rref(B)	*ans =* *1 0 0* *0 1 0* *0 0 1*	%Row-reduced echelon form of B
A	*A =* *1 2 3* *3 2 1* *1 3 2*	%Matrix A
A(1,1)	*ans =* *1*	%Element of row 1 and column 1 of matrix A
A(1, 1:3)	*ans =* *1 2 3*	%Elements of row 1 and columns 1 to 3 of matrix A (i.e. first row of matrix A)
A(1:3, 3)	*ans =* *3* *1* *2*	%Elements of rows 1 to 3 and column 3 of matrix A (i.e. last column of matrix A)
C = eye(3)	*C =* *1 0 0* *0 1 0* *0 0 1*	%Creates a 3 x 3 identity matrix

PLOTTING

x=[-6:0.01:9];	%Creates a vector from -6 to 9 with values incrementing by 0.01; the semi-colon suppresses the output	%The incrementing step (in this case 0.01) is up to the user. The larger it is, the coarser the plot will appear.
*f=2*x.^3 + 5*x.^2;*	%Creates a vector f with corresponding values from the expression; the semi-colon supresses the output	%Make note of how the dot operator is used to achieve element by element operation

PLOTTING

plot (x , f)



%Plots f vs. x

v = [1 : 1 : 10]

A =

1 2 3 4 5 6 7 8 9 10

%Creates a row vector v

z =

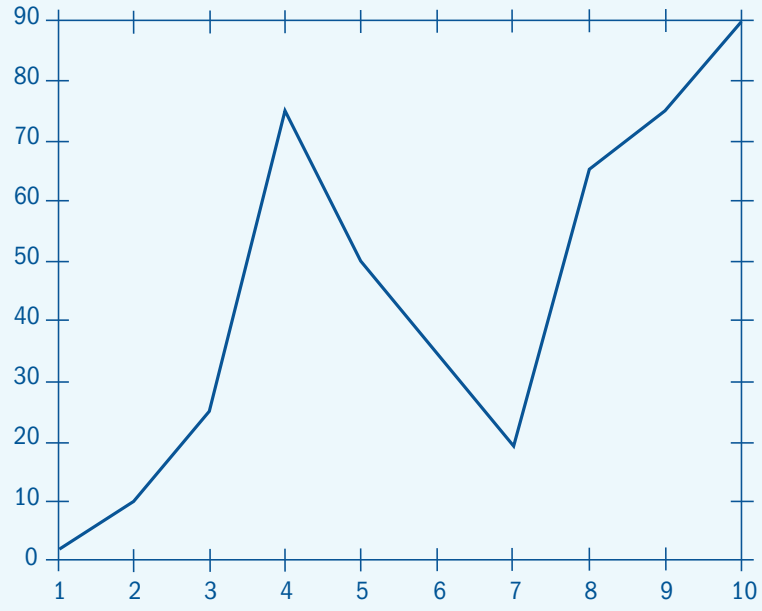
[2 10 25 75 50 35 19 65 75 90]

B =

2 10 25 75 50 35 19 65

%Creates a row vector z

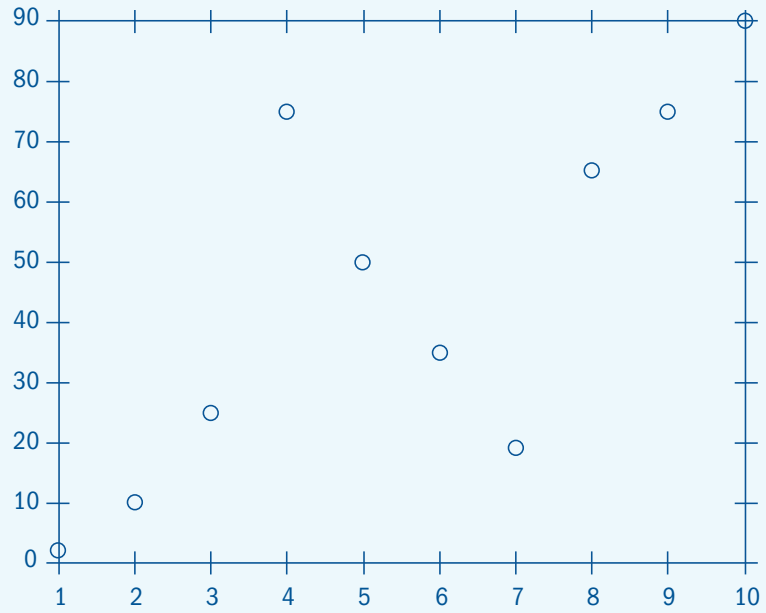
plot (v , z)



%Plots z vs. v

PLOTTING

scatter (v , z)




```
%Scatter plot of z vs. v
```

SCRIPT FILES

If you would like to save your work/commands and rerun/easily change them later on, use a MATLAB script file. It is advisable to use script files when working on assignments and projects.

NEW SCRIPT FILE

File → New → Script (Ctrl + N).

CONTENTS

Use the same commands within a script file as you would use directly in the Command Window. These commands will be run in series when the script file is run. The nice thing about script files is that they can be easily edited afterwards. If you were to make a mistake in the Command Window, there is no way you could erase the erroneous output. If you were to print it, the erroneous output would appear. With a script file, the commands could be edited after the mistake was noticed and run again. The output could then be printed.

One command of note is “echo on.” Using this in a script file, echoes the commands within the script file when the file is run. Without “echo on” only the output of the commands is visible.

In order to add comments or to comment out code use the ‘%’ symbol.

SAVING

When saving a MATLAB script file, make sure the filename contains no spaces. Also, make note of the directory you are saving it in as this will be important when running the script file.

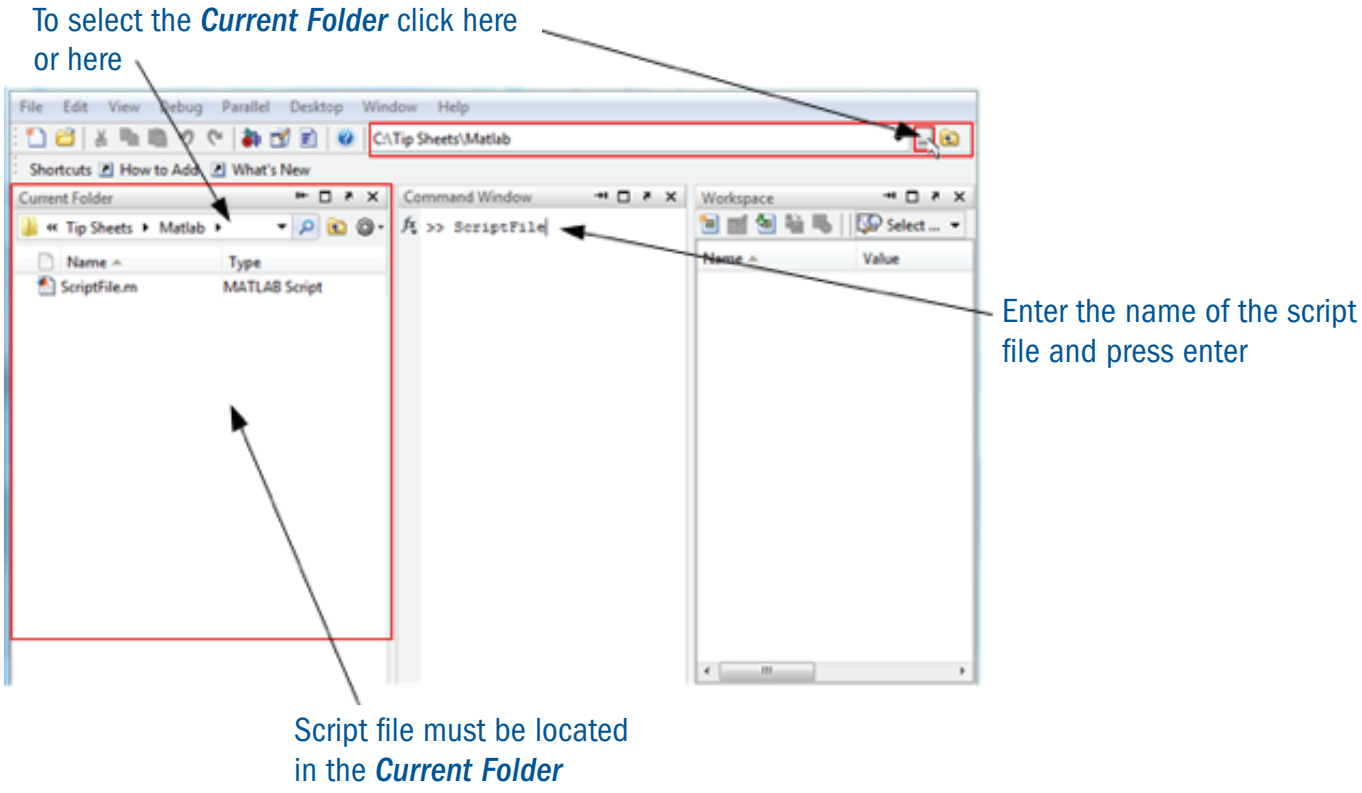
RUNNING A SCRIPT FILE

There are two ways of running a script file:

1. Press the Run button indicated in the Figure 1 below. The output of the file will then be displayed in the Command Window.

Figure 1: Running a Script File – Method 1

2. Write the name of the script file into the Command Window as shown in Figure 2 and hit enter. Make sure that the folder containing the script file is selected in the Current Folder's dialog. The output of the file will be displayed in the Command Window.



FUNCTIONS

If you would like your MATLAB code to accept an argument(s) and then do something with that argument, then a function may be what you need. Functions operate much like script files as they are a special type of script file.

NEW FUNCTION

File → New → Function

CONTENTS

Function files need to be of the following form:

```
function [ output_args ] = funName( input_args )
%   funName Summary of this function goes here
%   Detailed explanation goes here

%   Contents of function

end
```

where ***funName*** is the name of the function, ***input_args*** are the inputs the function takes, and ***output_args*** are the outputs of the function. The number of input arguments and output arguments can vary from none to many; it all depends on the tasks that you are trying to achieve. The ***output_args*** need to be set near the end of the function so that the function actually returns a value; otherwise, an error will occur. Also note that the function's name, ***funName***, cannot have spaces.

A sample function is given below. The function called ***quadraticEquation*** returns the two roots of any equation of the form $ax^2 + bx + c = 0$. The function, ***quadraticEquation,*** takes in three inputs, a, b, and c and returns two outputs, ***root1*** and ***root2***.

```
function [root1, root2] = quadraticEquation(a, b, c)
%quadraticEquation solves for the roots of any quadratic
of the form
%a*x^2 + b*x + c = 0

root1= (-b+sqrt(b^2-4*a*c))/(2*a);
root2= (-b-sqrt(b^2-4*a*c))/(2*a);

end
```

As you can see, the two output arguments, root1 and root2 are set within the function.

SAVING

MATLAB function file name's must not contain spaces and must have the same name as the defined function. Again, make note of the directory you are saving it in as this will be important when running the script file.

RUNNING A MATLAB FUNCTION FILE

A MATLAB function file can be run in either the command prompt or within a script file. In either case, running a function is usually of the following form:

```
[output_vars] = funName(input_vars)
```

where ***input_vars*** are the input variables placed into the function (these can be direct numerical values) and ***output_vars*** are the ***output variables*** that will be set by the function. The number of ***input_vars*** should match the number of ***input_args*** and the number of ***output_vars*** should match the number of ***output_args***. Like with script files make sure that the folder containing the function file is selected in the Current Folder's dialog.

In example, the ***quadraticEquation*** is run as follows:

```
[ x1, x2 ] = quadraticEquation(2, -8, -24)
```

where 2 is input for a, -8 is input for b, and -24 is input for c. x1 is set to root1 and x2 is set to the value of root2. The output is as follows:

```
x1 =
     6
x2 =
    -2
```


This document is licensed under Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). For more information visit creativecommons.org/licenses/by-nc/4.0/