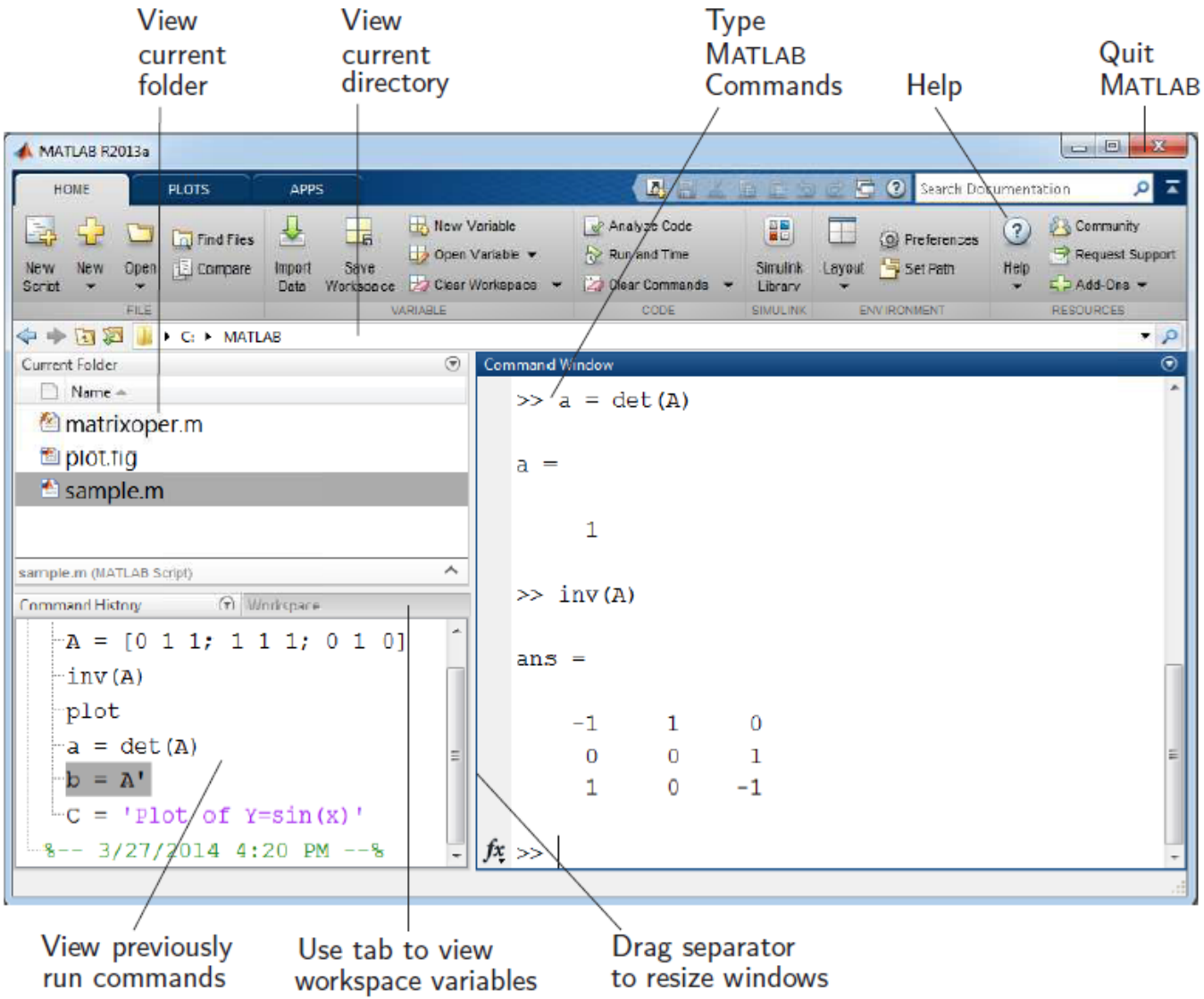
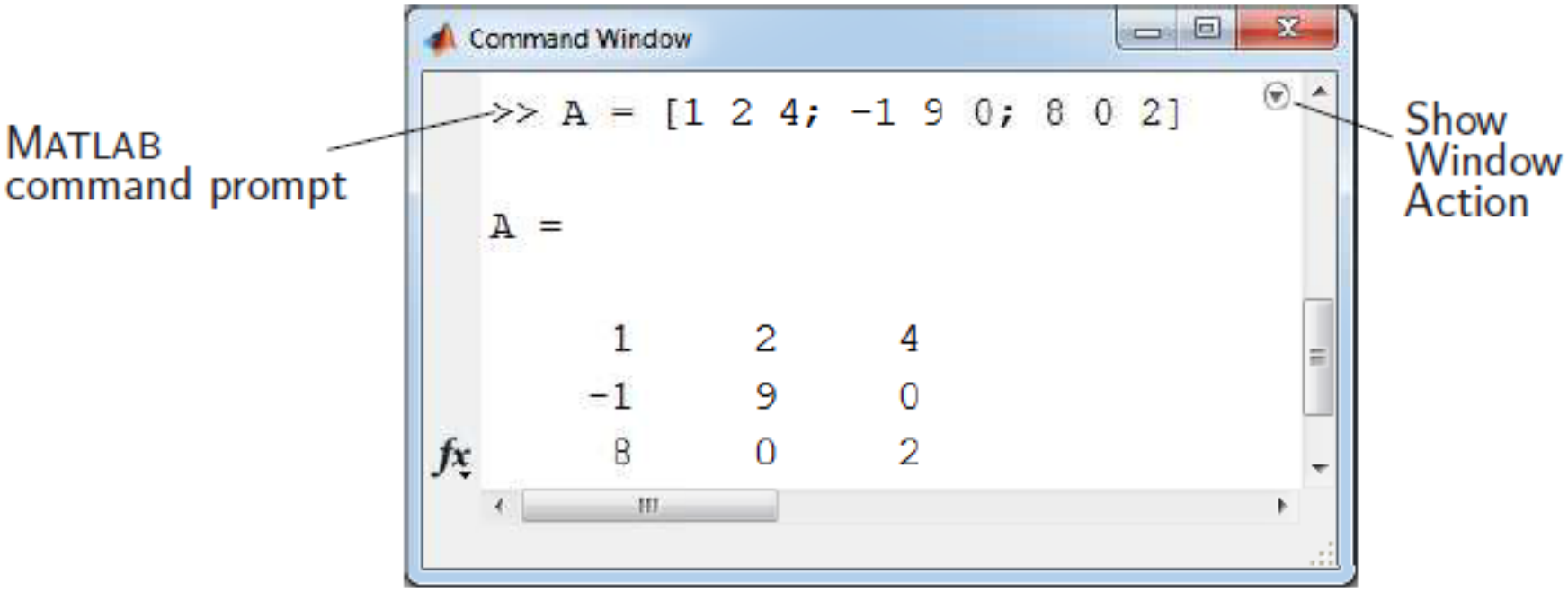
Lecture - 3
INTRODUCTION TO MATLAB

Reference: Book Chapter 1

MATLAB DESKTOP

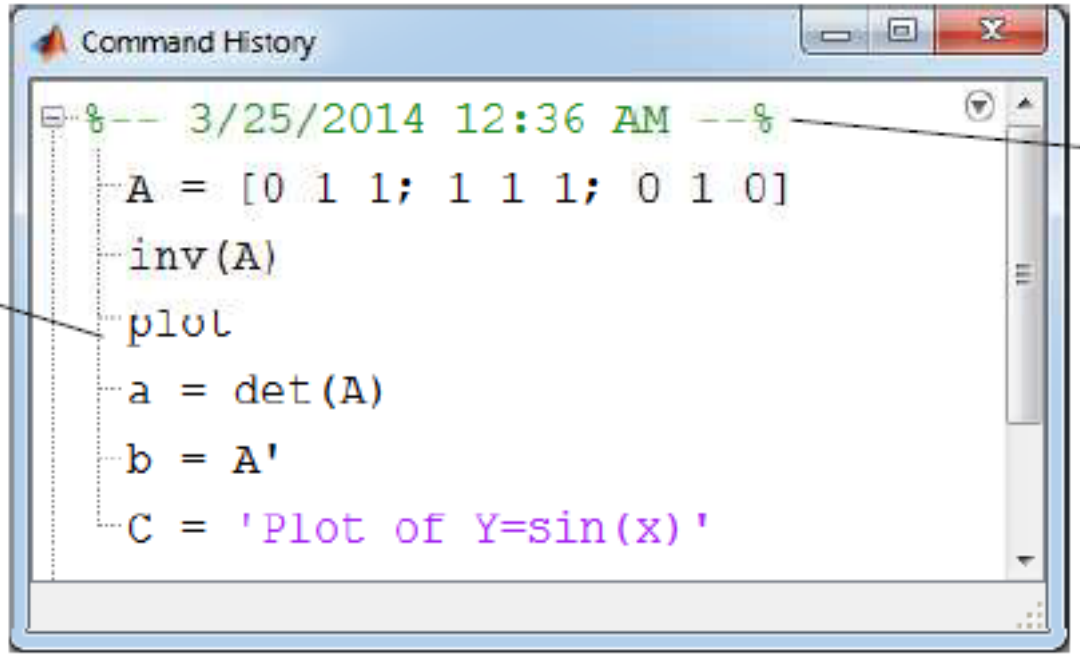


MATLAB COMMAND WINDOW



The Command Window is used to enter variables, evaluate MATLAB commands, and run M-files or functions.

MATLAB COMMAND HISTORY

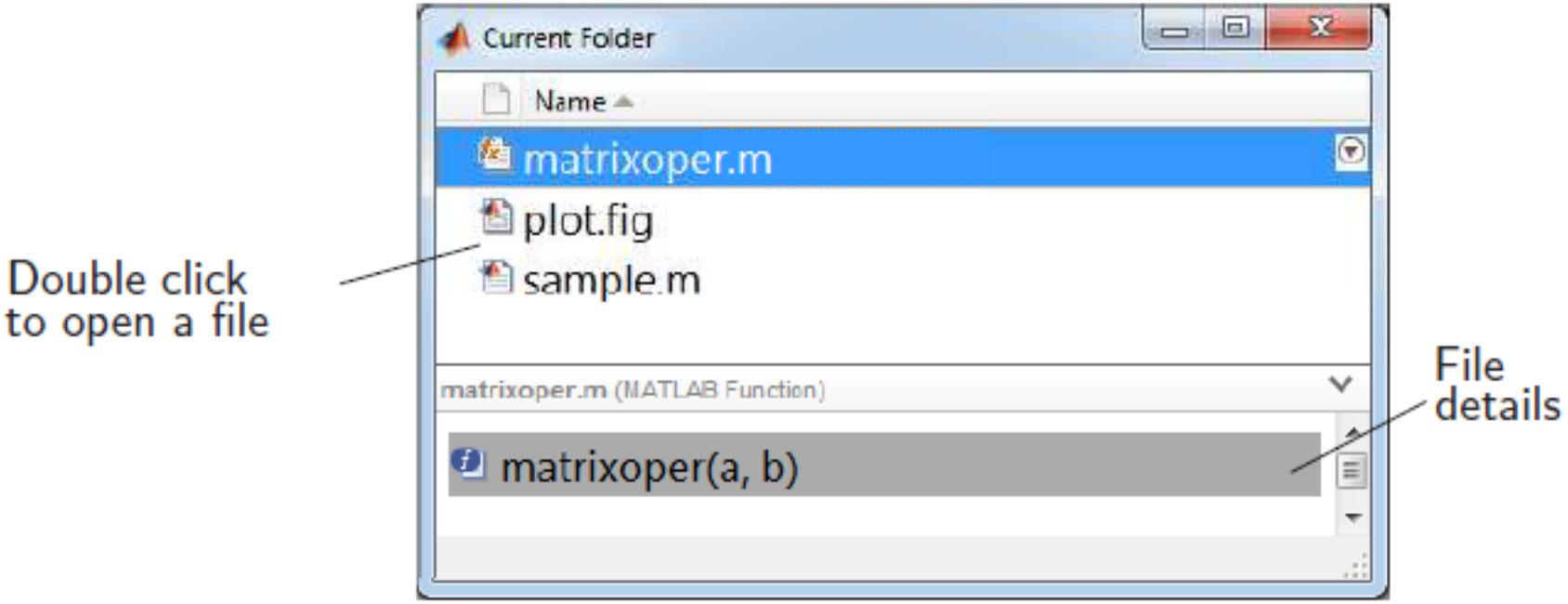


Select one or more lines and right click to copy, evaluate or create an M file from the selection

Time stamp marks start of a session

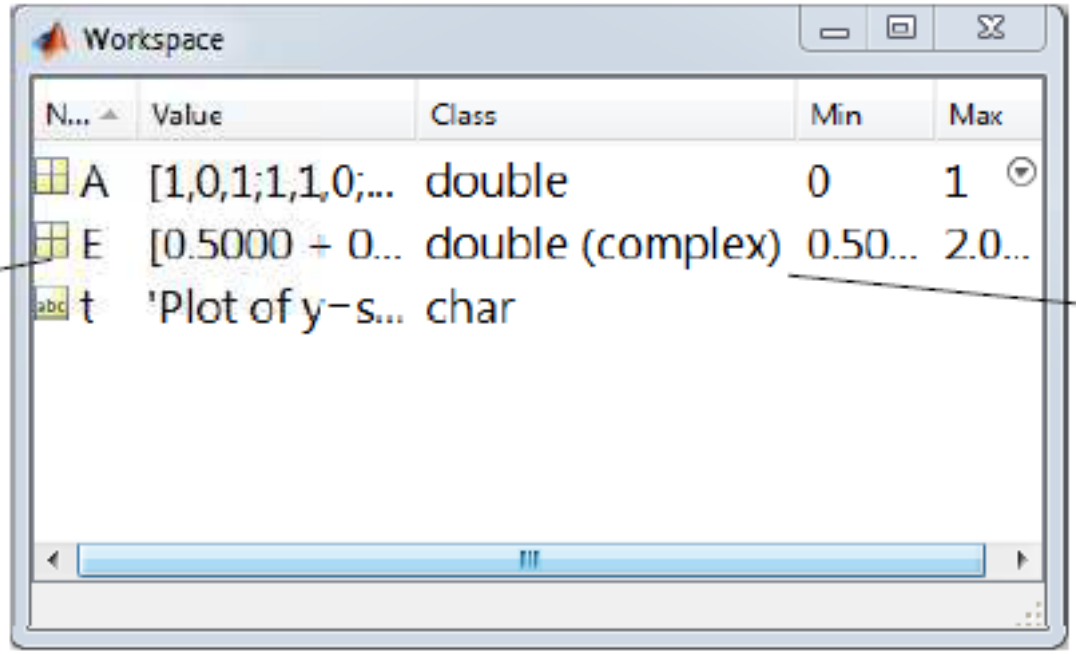
The Command History window is used to view previously-used functions, and to copy and execute selected lines from those functions.

MATLAB CURRENT DIRECTORY



The Current Directory browser can be used to view, open, and make changes to MATLAB related directories and files.

MATLAB WORKSPACE BROWSER

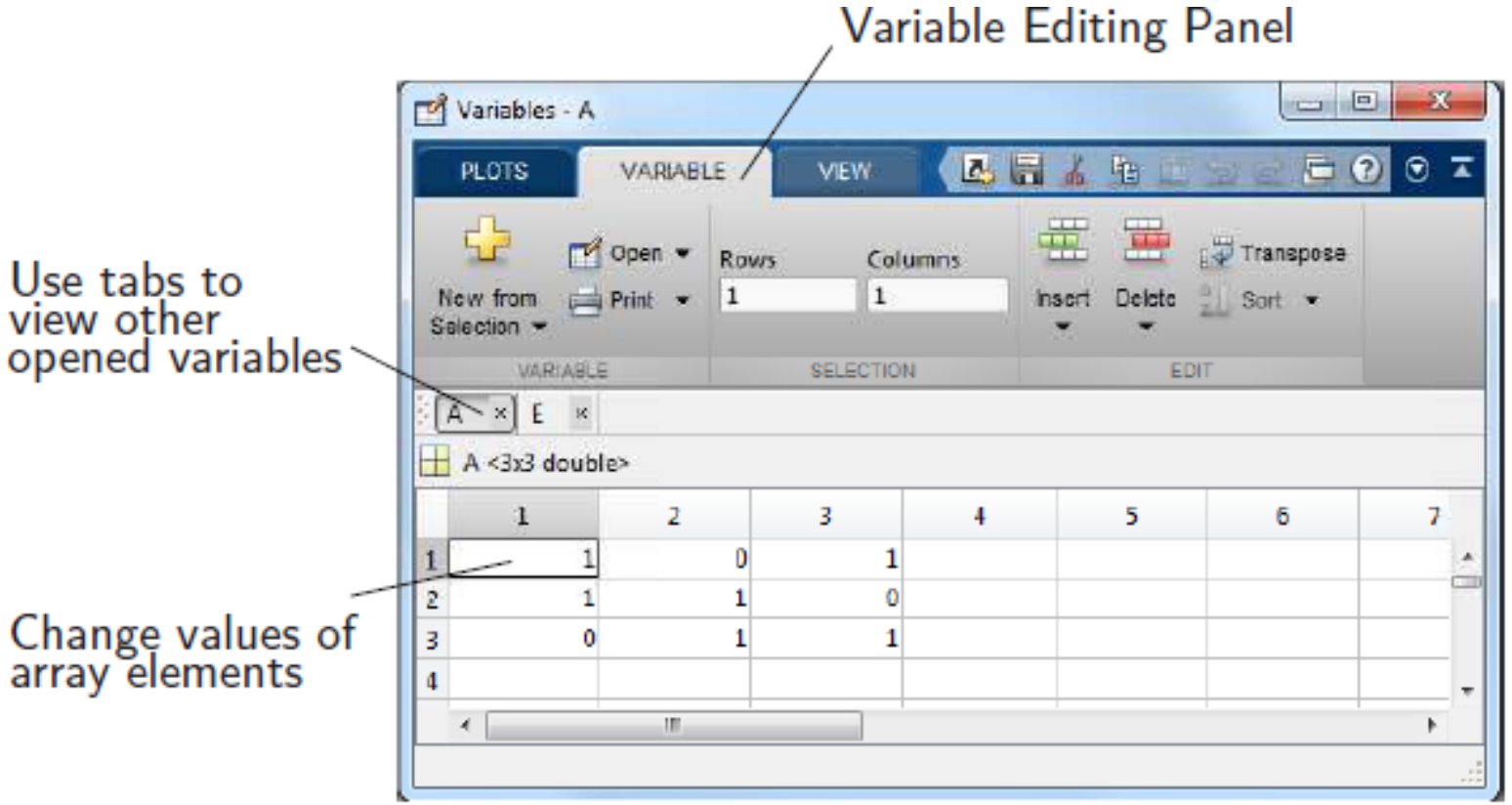


Double click a variable to view and change its content

Type of the variable

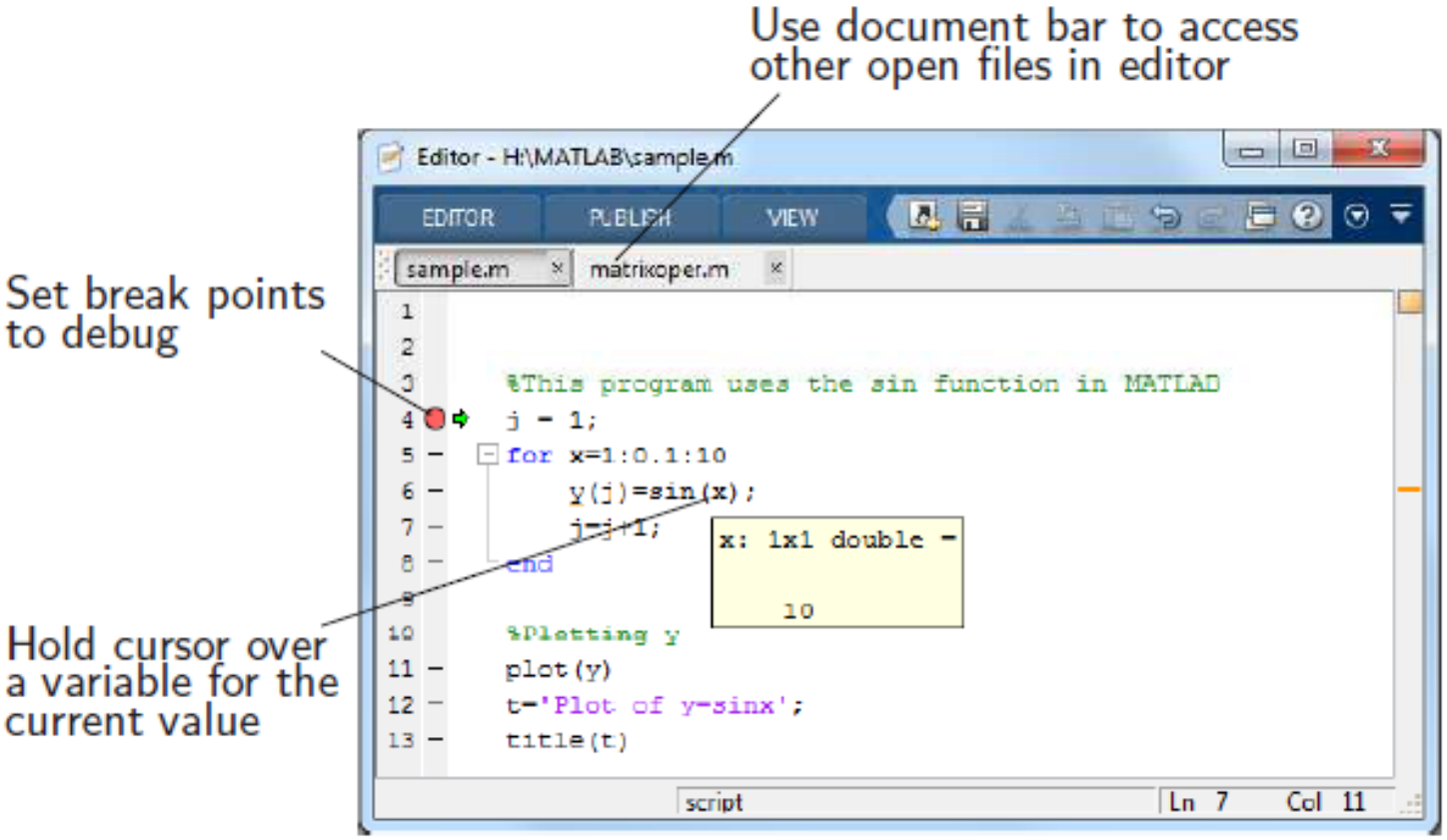
The Workspace Browser is used to view the workspace, and the information about each variable.

MATLAB ARRAY EDITOR



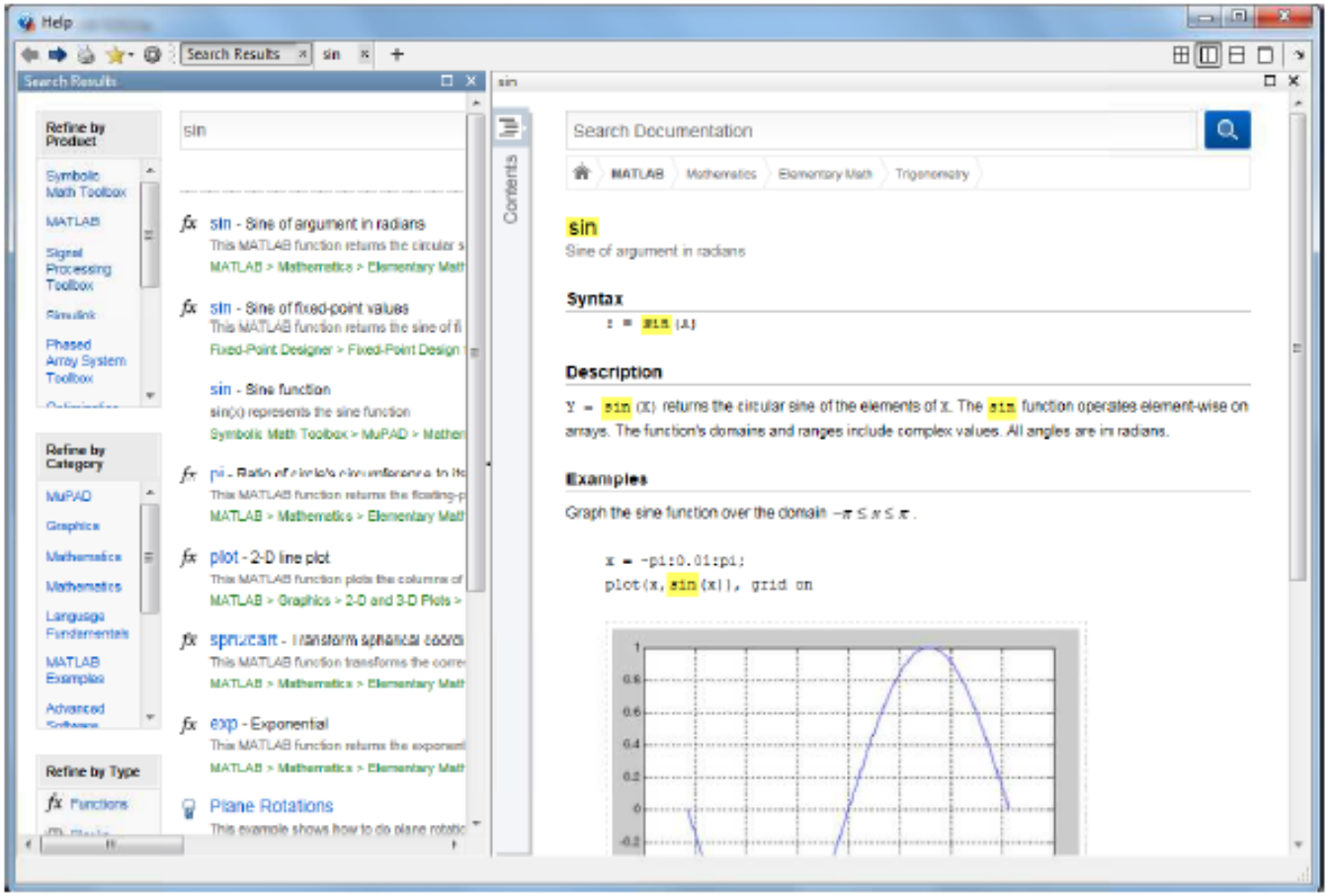
The Variable Editing Panel opens when you double-click on a variable in the Workspace Browser. It can be used to view and directly edit one or two dimensional numeric arrays, strings, and arrays of strings in the workspace.

MATLAB EDITOR OR DEBUGGER



The Editor/debugger provides a GUI to create and debug M-files.

MATLAB HELP BROWSER



The Help browser is used to search and view documentation for all MATLAB products

BASIC EXPRESSIONS

```
>> a = 1 + 2
a =
            3
>> b = 4
b =
            4
>> c = sqrt(a^2 + b^2)
c =
            5
>> x = a + b;
>> y = 2*x
y =
            14
```

LONG EXPRESSIONS/COMMANDS

As you type commands or expressions that are very long, you can simply type the command on several lines. To do this, begin the expression on one line, then type three periods followed by Enter or Return (where you want to break), and simply continue the expression on the following line.

```
>> a_long_variable = 2;
>> another_long_variable = 3;
>> x = a_long_variable + a_long_variable^2 ...
        + another_long_variable
x =
        9
```

TYPING MATRICES

```
>> M = [ 1 2 3 4; 5 6 7 8; 9 10 11 12]
M =
     1     2     3     4
     5     6     7     8
     9    10    11    12

>> v=[11 ; 22 ; 33]
v =
    11
    22
    33
>> mv = [M v]
mv =
     1     2     3     4    11
     5     6     7     8    22
     9    10    11    12    33
```

TYPING MATRICES (CONTINUED)

```
>> h = [11 22 33 44]
h =
          11          22          33          44
>> mh = [M ; h]
mh =
           1           2           3           4
           5           6           7           8
           9          10          11          12
          11          22          33          44
```

MATRICES GENERATORS

```
>> ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
>> ones(1,3)
ans =
     1     1     1
>> zeros(2)
ans =
     0     0
     0     0
>> zeros(1,3)
ans =
     0     0     0
>> A = eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
```

ones(3) produces a 3 by 3 matrix of ones.

ones(1,3) produces a 1 by 3 matrix of ones.

Zeros(2) produces a 2 by 2 matrix of zeros.

zeros(1,3) produces a 1 by 3 matrix of zeros.

eye(3) produces a 3 by 3 identity matrix.

SUBSCRIPTING

```
>> A = [1 3 5; 4 6 8]
A =
   1     3     5
   4     6     8
>> A(1,2)
ans =
   3
>> A(2,end)
ans =
   8
```

MATRIX ARITHMETIC

```
>> A = [1 3 5; 4 6 8]
A =
   1     3     5
   4     6     8

>> v=[1;1;1]
v =
         1
         1
         1
>> A*v
ans =
         9
        18
```

COLON OPERATOR

```
>> a = 2:8

a =

    2     3     4     5     6     7     8

>> 1:2:9

ans =

    1     3     5     7     9

>> 50:-5:30

ans =

    50    45    40    35    30

>> P=[1 2 3 4 ; 5 6 7 8]

P =

    1     2     3     4
    5     6     7     8

>> pp = P(2, 2:4)

pp =

    6     7     8
```

TRANSPOSE

```
>> a=[1 2 3;4 5 6]

a =

   1     2     3

   4     5     6

>> b = a'

b =

   1     4

   2     5

   3     6
```

THE COMMAND LINSPACE

The *linspace* function is a quick way to generate a row vector of *n* linearly spaced points between two given numbers.

The syntax is *linspace(a,b,n)*.

If the number, *n*, is not specified, it generates 100 points between a and b by default.

If *n* is not specified, the syntax is *linspace(a,b)*.

MORE MATLAB EXPRESSIONS

Below are some more examples of MATLAB expressions.

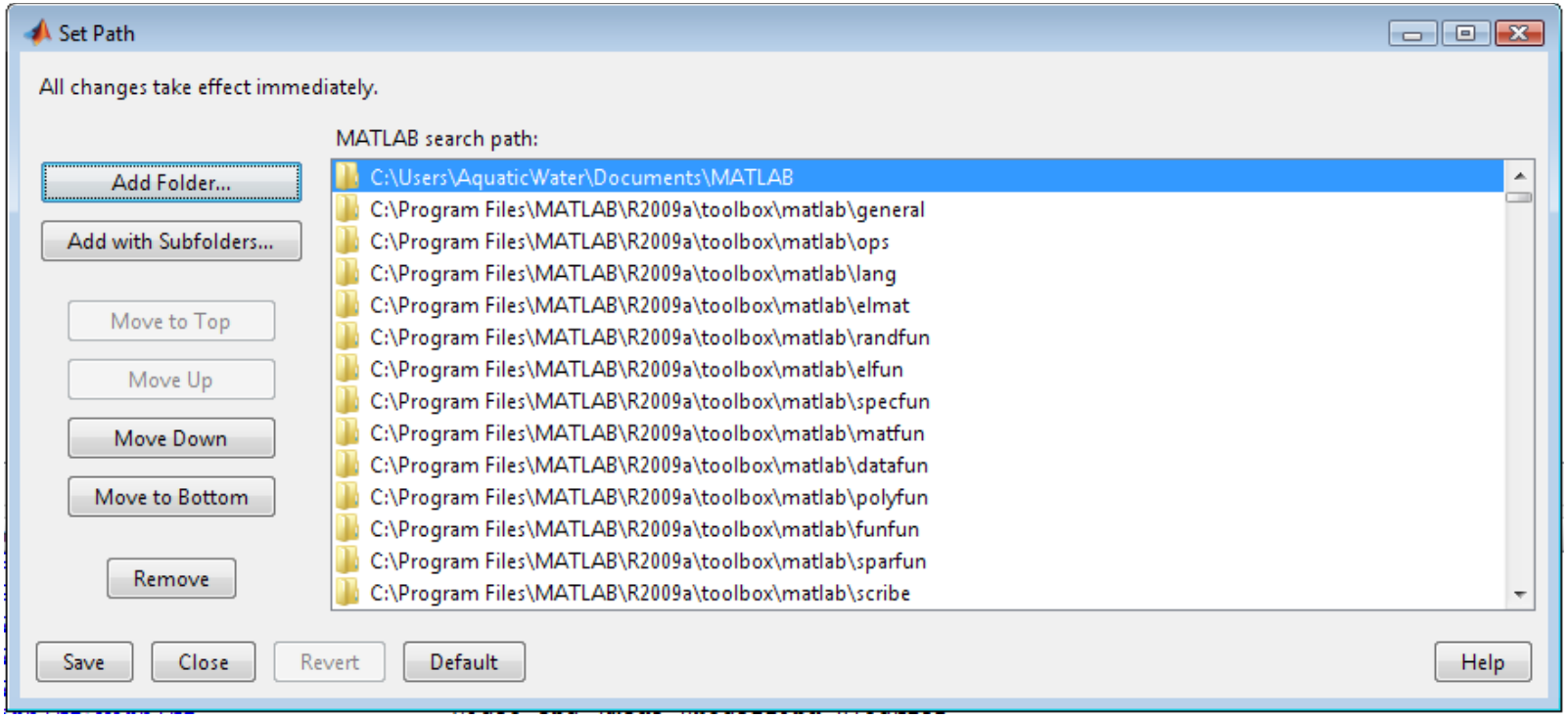
```
>> x = 2
x =
    2
>> y = exp(x) + log(x)
y =
    8.0822
>> z = (-sin(y)) + abs(-x)
z =
    1.0259
>> complex_number = 4 + 3i
complex_number =
    4.0000 + 3.0000i
>> magnitude = abs(complex_number)
magnitude =
    5
```

CURRENT DIRECTORY

The current directory is generally displayed in a text box towards the top of the MATLAB screen. Alternatively, you can type *pwd* at the command prompt to display the current directory.

SETTING THE PATH

You can set the path to directories that you use often. The files in these directories can then be directly accessed from any other directory. The path can be set from File >> Set Path menu.



SAVING AND LOADING VARIABLES

Any variable in the workspace can be saved to the hard disk using the *save* command.

save FILENAME x y

The above command saves the workspace variables x and y to a file named FILENAME.mat in the current directory. Note that there is no comma in the above syntax. To retrieve these variables, type the command

load FILENAME x y

THE "FOR" LOOP

The for loop executes a set of statements for a specified number of times.

```
>> for i = 1:5
        x(i)=1;
    end
>> x
x =
  1     1     1     1     1
```

THE "WHILE" LOOP

The STATEMENTS will be executed as long as the EXPRESSION is true.

```
while EXPRESSION
        STATEMENTS
end
```

THE "IF" STATEMENT

The *if* statement allows you to execute statements provided certain conditions hold. Nested *if* statements must each be paired with a matching end.

```
if EXPRESSION
  STATEMENTS
else
  STATEMENTS
end
```

THE "SWITCH-CASE" STATEMENTS

Switch provides a way to switch between several cases based on an expression.

```
>> x=2
   switch x
            case 1
                     disp('x is equal to 1')
            case 2
                     disp('x is equal to 2')
   end

x =

    2

x is equal to 2
```

LOGICAL AND RELATIONAL OPERATORS

A complete list of operators can be obtained by typing *help ops* at the command prompt.

Some commonly used relational operators are

EQUAL (==)

GREATER THAN, (>)

LESS THAN, (<)

NOT EQUAL (~=)

Some commonly used logical operators are:

AND (&)

OR (|)

NOT (~)

M-FILES

Script files are simply a series of MATLAB commands stored in a .m file. They do not take any arguments, or return any output.

```
% This is my first .m file
var = 5; new_var = var^2;
if new_var > 5
        disp('My first output');
end
```

```
>> myMFile
My first output
```

FUNCTION M-FILES

Function M-files can receive one or more variables as input arguments, and can return one or more variables as outputs. The input and output variables are available for use outside of the function, but the variables or parameters used inside the function are not available outside the function.

```
function output_var = cuberoot(input_var)
output_var = input_var^(1/3);
```

SUBFUNCTIONS

Subfunctions can be declared following the definition of the main function in the same M-file. Subfunctions are visible to the main function and other subfunctions in the same M-file, but are not visible outside the M-file in which they are declared.

GLOBAL AND LOCAL VARIABLES

All the variables created within a function M-file are local to that function, that is, they cannot be accessed outside the function. Similarly, workspace variables are local to the workspace, and are not available in any function. To make a workspace variable, x, globally available, we use the global command as follows.

```
global x
```

MATLAB HELP

helpdesk opens a MATLAB help GUI.

helpwin opens a hypertext help browser.

demo starts the MATLAB demonstration.

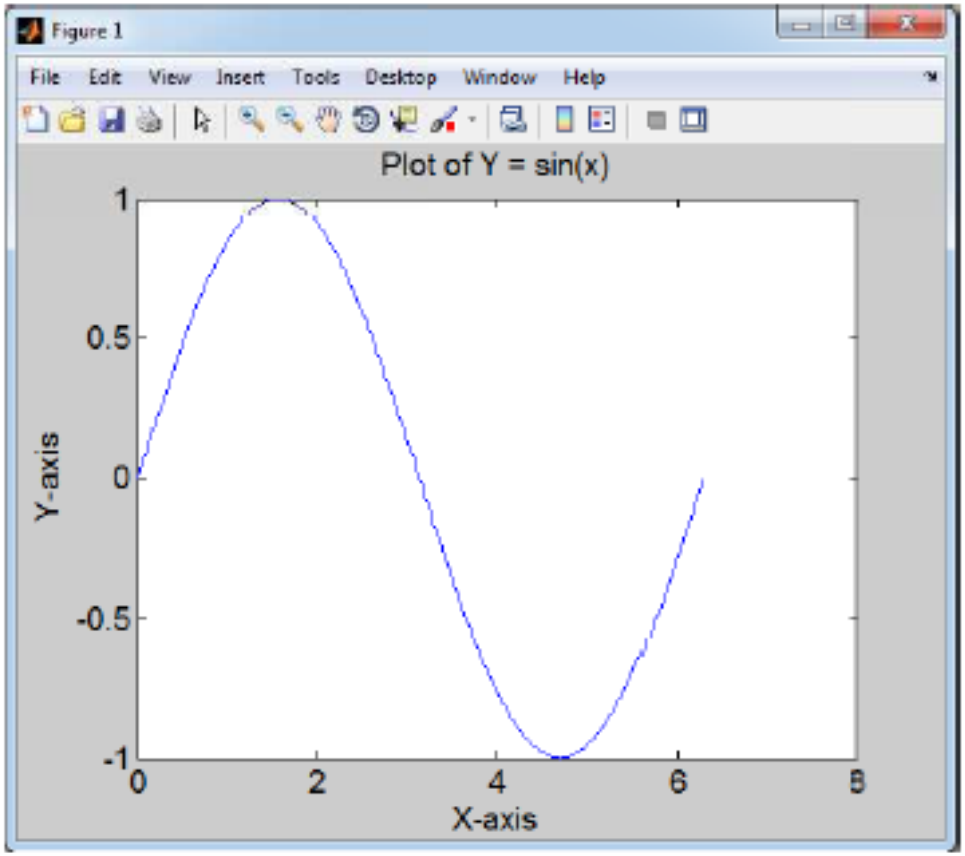
help prints on the screen the various help topics available.

help followed by a help topic, or any function name, provides help on the requested topic or function.

lookfor followed by a topic keyword, which gives the names of all the MATLAB functions that have that keyword on the first help line (That keywork does not have to be the name of a function, unlike for the help command)

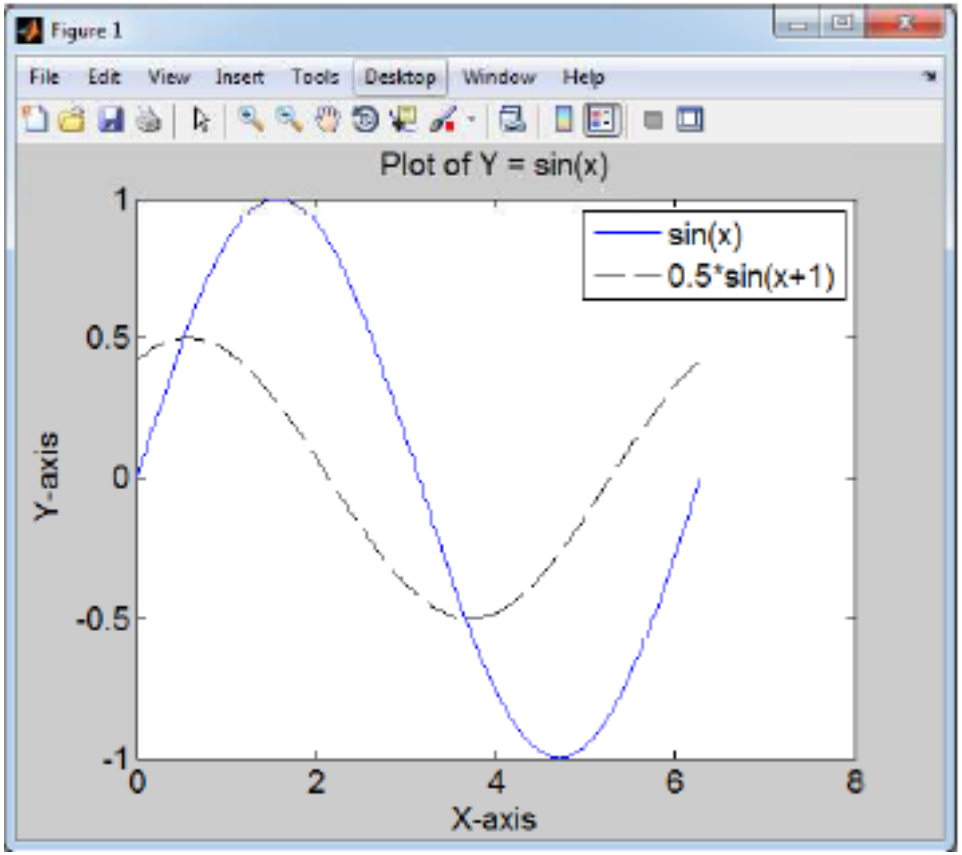
USE OF PLOT, AXES, AND LABELS COMMAND

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);
xlabel('X-axis')
ylabel('Y-axis')
title('Plot of Y = sin(x)')
```

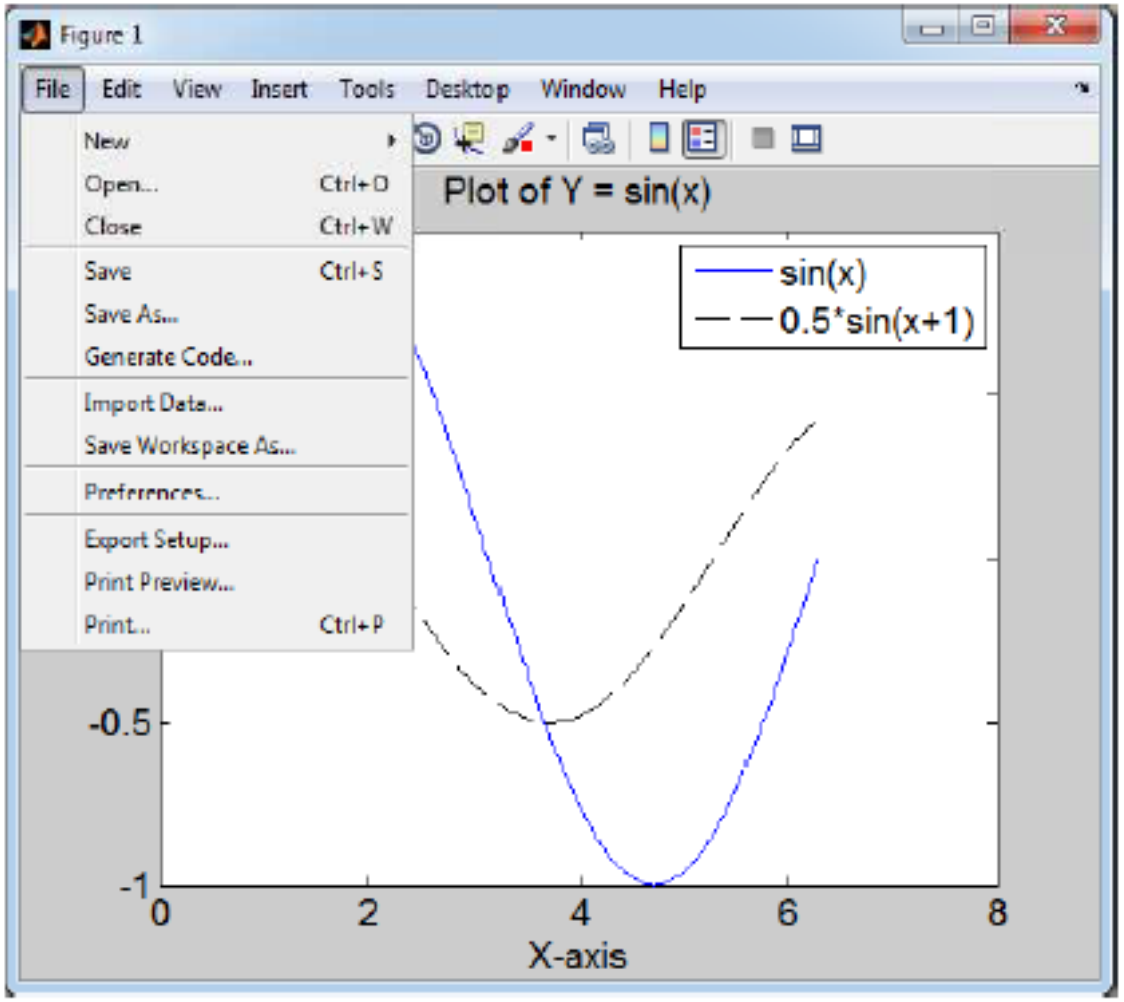


MULTIPLE PLOTS ON THE SAME FIGURE

```
clc
clear
x = 0:pi/100:2*pi;
y = sin(x);
y2 = 0.5*sin(x+1);
plot(x,y,x,y2,'--');
xlabel('X-axis')
ylabel('Y-axis')
title('Plot of Y = sin(x)')
legend('sin(x)','0.5*sin(x+1)')
```

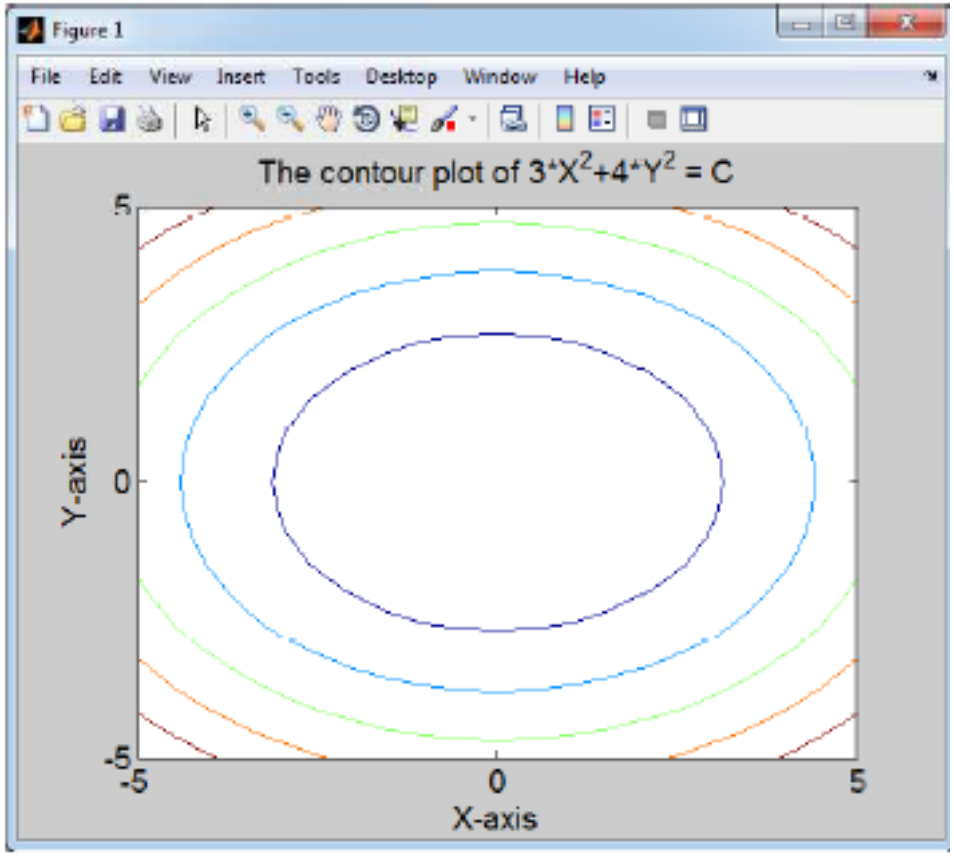


PRINTING AND SAVING PLOTS



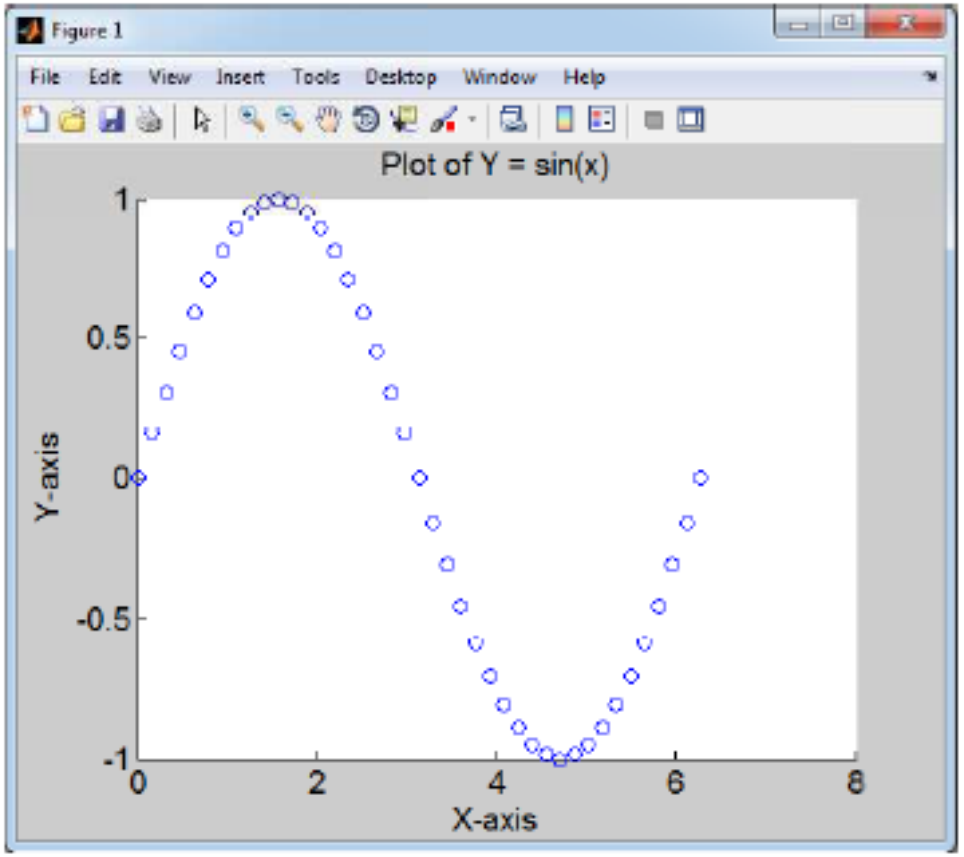
CONTOUR PLOT

```
[X,Y] = meshgrid(-5:.5:5,-5:.5:5);
Z = (3*X.^2+4*Y.^2);
[C,h] = contour(X,Y,Z,5);
xlabel('X-axis')
ylabel('Y-axis')
title('The contour plots …
of 3*X^2+4*Y^2 = C')
```



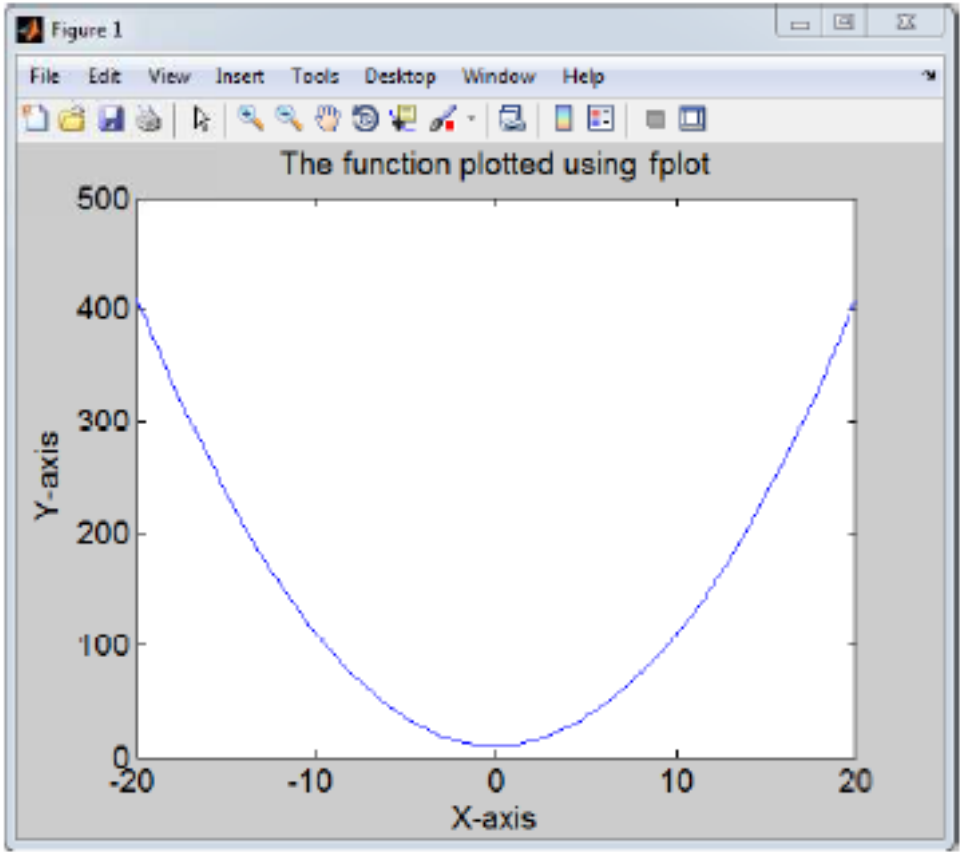
SCATTER PLOT

```
x = 0:pi/20:2*pi;
y = sin(x);
scatter(x,y);
xlabel('X-axis')
ylabel('Y-axis')
title('Plot of Y = sin(x)')
```



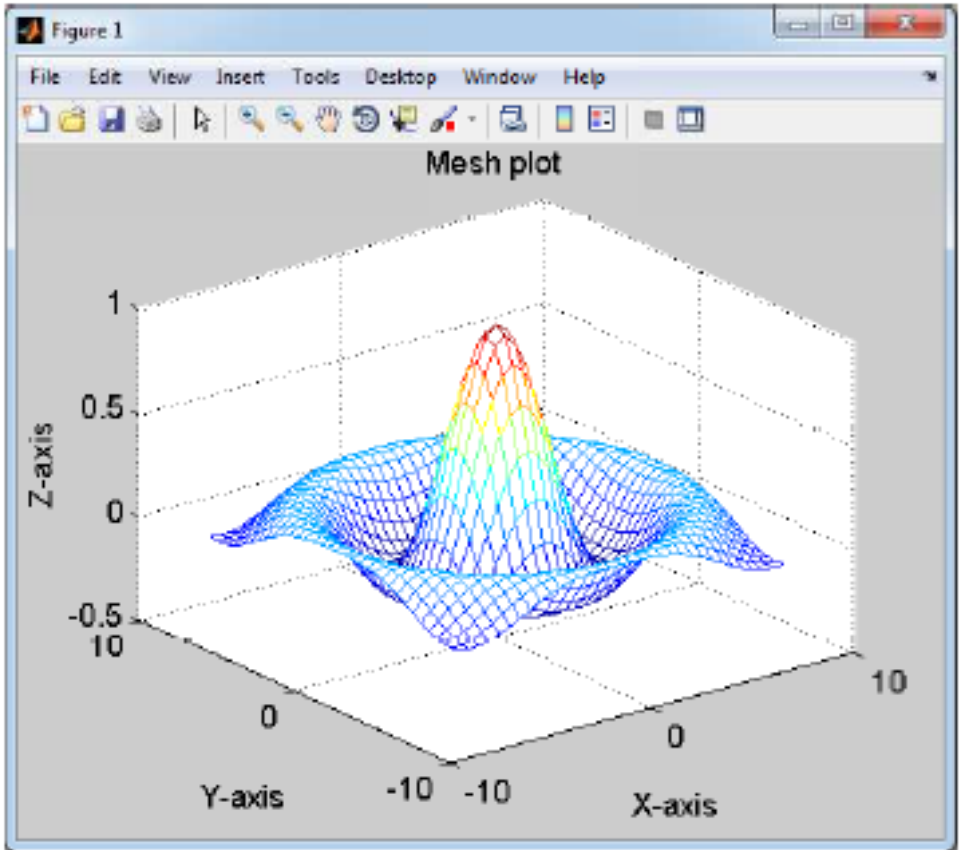
FPLOT: FUNCTION PLOTS

```
fplot('x^2+10',[-20 20])
xlabel('X-axis')
ylabel('Y-axis')
title('The function plotted …
using fplot')
```



MESH PLOT

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
xlabel('X-axis')
ylabel('Y-axis')
zlabel('Z-axis')
title('Mesh plot')
```



USING THE PLOT EDITING MODE


Start a new figure


Open existing file


Save this plot as figure


Print this plot


Start editing this plot


Zoom in


Zoom out


Pan figure in the plane


Rotate this plot in 3D


Data cursor


Insert Colorbar


Insert legend on the plot


Hide plot tools


Show plot tools

ARITHMETIC OPERATORS (HELP OPS)

Command	Symbol	Description
`plus`	+	Addition
`uplus`	+	Unary addition
`minus`	-	Subtraction
`uminus`	-	Unary minus
`mtimes`	*	Matrix mupliplication
`times`	.*	Array multiplication
`mldivide`	\	Left matrix divide
`mrdivide`	/	Right matrix divide
`ldivide`	.	Left array divide
`rdivide`	./	Right array divide
`mpower`	∧	Matrix power
`power`	.∧	Array power

LOGICAL OPERATORS (HELP OPS)

Command	Symbol	Description
`eq`	==	Equal
`ne`	=	Not equal
`lt`	<	Less than
`gt`	>	Greater than
`le`	<=	Less than or equal
`ge`	>=	Greater than or equal
`and`	&	Logical AND
`or`	|	Logical OR
`not`	~	Logical NOT
`xor`	N/A	Logical Exclusive OR
`any`	N/A	True if any element of vector is nonzero
`all`	N/A	True of all element of vector is nonzero

SPECIAL CHARACTERS (HELP OPS)

Command	Symbol	Description
`colon`	:	Colon
`punct`	..	Parent directory
`punct`	%	Comment
`punct`	=	Assignment
`transpose`	.'	Transpose

PROGRAM CONTROL FLOW CONSTRUCTS (HELP LANG)

Command	Description
`if`	Conditionally execute statement
`else`	If statement condition
`elseif`	If statement condition
`end`	Terminate scope for `while`, `switch`, `try`, and `if` statement
`for`	Repeat statement for a specific number of times
`while`	Repeat statement for indefinite number of times
`break`	Terminate execution of `for` or `for` loop
`continue`	Pass control to the next iteration of `for` or `while` loop
`switch`	Switch among several cases based on Expression
`case`	`switch` statement case

SCRIPTS, FUNCTIONS, AND VARIABLES (HELP LANG)

Command	Description
`global`	Define global variable
`mfilename`	Name of currently executing M-file
`exist`	Check if variable or function are defined
`isglobal`	True for global variable

ARGUMENT HANDLING (HELP LANG)

Command	Description
`nargin`	Number of function input argument
`nargout`	Number of function output argument

MESSAGE DISPLAY (HELP LANG)

Command	Description
`error`	Display error message and abort function
`warning`	Display warning
`disp`	Display and array
`fprintf`	Display formatted message
`sprintf`	Write format data to a string

ELEMENTARY MATRICES (HELP ELMAT)

Command	Description
`zeros`	Zeros array
`ones`	Ones array
`eye`	Identity matrix
`rand`	Uniformly distributed random numbers
`randn`	Normally distributed random numbers
`linspace`	Linearly spaced vector
`logspace`	Logarithmically spaced vector
`meshgrid`	x and y array for 3D plot
`:`	Regularly spaced vector and index into matrix

ARRAY INFORMATION AND MATRIX MANIPULATION (HELP ELMAT)

Command	Description
`size`	Size of matrix
`length`	Length of vector
`ndims`	Number of dimensions
`numel`	Number of elements
`isempty`	True for empty matrix
`isequal`	True of arrays are identical
`diag`	Diagonal matrices
`find`	Find indices of nonzero elements
`end`	Last index

SPECIAL VARIABLES AND CONSTANTS (HELP ELMAT)

Command	Description
`ans`	Most recent answer
`eps`	Floating-point relative accuracy
`realmax`	Largest positive floating-point number
`realmin`	Smallest positive floating-point number
`pi`	3.14159263
`i, j`	Imaginary unit
`inf`	Infinity
`Nan`	Not-a-number
`isnan`	True for not-a-number
`ininf`	True for infinite elements
`isfinite`	True for finite elements
`why`	Succinct number

TRIGONOMETRIC FUNCTIONS (HELP ELFUN)

Command	Description
sin	Sine
sinh	Hyperbolic sine
asin	Inverse sine
asinh	Inverse hyperbolic sine
cos	Cosine
cosh	Hyperbolic cosine
acos	Inverse cosine
acosh	Inverse hyperbolic cosine
tan	Tangent
tanh	Hyperbolic tangent
atan	Inverse tangent
atan2	Four quadrant inverse tangent
atanh	Inverse hyperbolic tangent

Command	Description
sec	Secant
sech	Hyperbolic secant
asec	Inverse secant
asech	Inverse hyperbolic secant
csc	Cosecant
csch	Hyperbolic cosecant
acsc	Inverse cosecant
acsch	Inverse hyperbolic cosecant
cot	Cotangent
coth	Hyperbolic cotangent
acot	Inverse cotangent
acoth	Inverse hyperbolic cotangenet

EXPONENTIAL AND COMPLEX FUNCTIONS (HELP ELFUN)

Command	Description
`exp`	Exponential
`log`	Natural logorithm
`log10`	Common (base 10) logorithm
`sqrt`	Square root
`abs`	Absolute value
`angle`	Phase angle
`complex`	Construct complex data from real and imaginary part
`imag`	Complex imaginary part
`real`	Complex real part
`isreal`	True for real array

ROUNDING AND REMAINDER (HELP ELFUN)

Command	Description
`fix`	Round towards zero
`floor`	Round towards minus infinity
`ceil`	Round towards plus infinity
`mod`	Modulus (signed reminder after division)
`rem`	Reminder after division
`sign`	Signum

SPECIALIZED MATH FUNCTIONS (HELP SPECFUN)

Command	Description
`cross`	Vector cross product
`dot`	Vector dot product

MATRIX ANALYSIS (HELP MATFUN)

Command	Description
`det`	Determinant
`trace`	Sum of diagonal elements
`inv`	Matrix inverse
`eig`	Eigenvalues and eigenvectors
`svd`	Singular value decomposition
`expm`	Matrix exponential
`logm`	Matrix logorithm
`sqrtm`	Matrix square root
`fnum`	Evaluate general matrix function

BASIC STATISTICAL OPERATIONS (HELP DATAFUN)

Command	Description
`max`	Largest component
`min`	Smallest component
`mean`	Average or mean value
`median`	Median value
`std`	Standard deviation
`var`	Variance
`sort`	Sort in ascending order
`sortrows`	Sort rows in ascending order
`sum`	Sum of elements
`prod`	Product of elements
`hist`	Histogram
`histc`	Histogram count
`trapz`	Trapezoidal numerical integration
`cumsum`	Cumulative sum of elements
`cumprod`	Cumulative product of elements

OPTIMIZATION FUNCTIONS

Command	Description
`bintprog`	Binary integer programming problems
`fgoalattain`	Multiobjective goal attainment problems
`fminbnd`	Minimum of single-variable function on fixed interval
`fmincon`	Minimum of constrained nonlinear multivariable function
`fminimax`	Minimax constraint problem
`fminsearch`	Minimum of unconstrained multivariable function using derivative-free method
`fminunc`	Minimum of unconstrained multivariable function
`fseminf`	Minimum of semi-infinitely constrained multivariable nonlinear function
`ktrlink`	Minimum of constrained or unconstrained nonlinear multivariable function using KNITRO
`linprog`	Linear programming problems
`quadprog`	Quadratic programming problems

EQUATION SOLVING FUNCTIONS

Command	Description
`fsolve`	Solve system of nonlinear equations
`fzero`	Find root of continuous function of one variable

LEAST SQUARES (CURVE FITTING)

Command	Description
`lsqcurvefit`	Solve nonlinear curve-fitting problems in least-squares sense
`lsqlin`	Solve constrained linear least-squares problems
`lsqnonlin`	Solve nonlinear least-squares problems
`lsqnonneg`	Solve nonnegative least-squares constraint problem