International Islamic University Chittagong
Dept. of Electrical and Electronic Engineering

Experiment-1
Introduction to MATLAB

EEE-3604 Digital Signal Processing Sessional

Prepared By

Mohammed Abdul Kader

Assistant Professor, Dept. of EEE, IIUC

Objectives:

a) To familiarize with MATLAB and some basic commands of MATLAB.

b) To know about variable and variable types in MATLAB.

c) To know about Matrix manipulation in MATLAB.

d) To learn about plotting 2D graphs in MATLAB.

e) To become familiar with MATLAB script/editor.

f) To become familiar with conditional operators and loops in MATLB.

g) To learn about debugging program in MATLAB.

h) To know how to develop an user defined function in MATLAB

What is MATLAB?

- MATLAB Stands for MATrix LABoratory.
- MATLAB is a programming and numeric computing environment used by millions of engineers and scientists to analyze data, develop algorithms, and create models.
- It has many in built functions, tool boxes (signal and image processing, control systems, wireless communications, computational finance, robotics, deep learning and AI etc) and apps.

History of MATLAB

- Invented by **Prof. Cleve Moler** (American mathematician and computer programmer specializing in numerical analysis.) to make programming easy for his students.

  >> Late 1970.

  >> University of New Mexico.

- The MathWorks, Inc. was formed in 1984

  >> By Moler and Jack Little.

  >> One Product: MATLAB.

- Today

  >> 100 products

  >> As of 2020, MATLAB has more than 4 million users worldwide.

  >> Taught in 5,000 universities (2015).

- Matlab Version (Release history)

  >> 1st version: MATLAB 1.0 in 1984.

  >> Latest Version: MATLAB 9.9 (September 17, 2020).

Some useful MATLAB Commands

```
 >> version         % this will tell you the running MATLAB version
ans = 9.0.0.341360 (R2016a)
  >> help               % lists available packages/toolboxes on system.
  >> help elfun       % lists functions in elementary functions package
  >> help sin         % instructions on the sine function
  >> lookfor sine   %  if you don’t know the function name …
  >> doc      % start matlab help documentation
  >> doc sin          % for full details of function
>> Ctrl+C      (Press ‘Ctrl+C’ to stop execution of instruction)
  >> quit              % to quit MATLAB
```

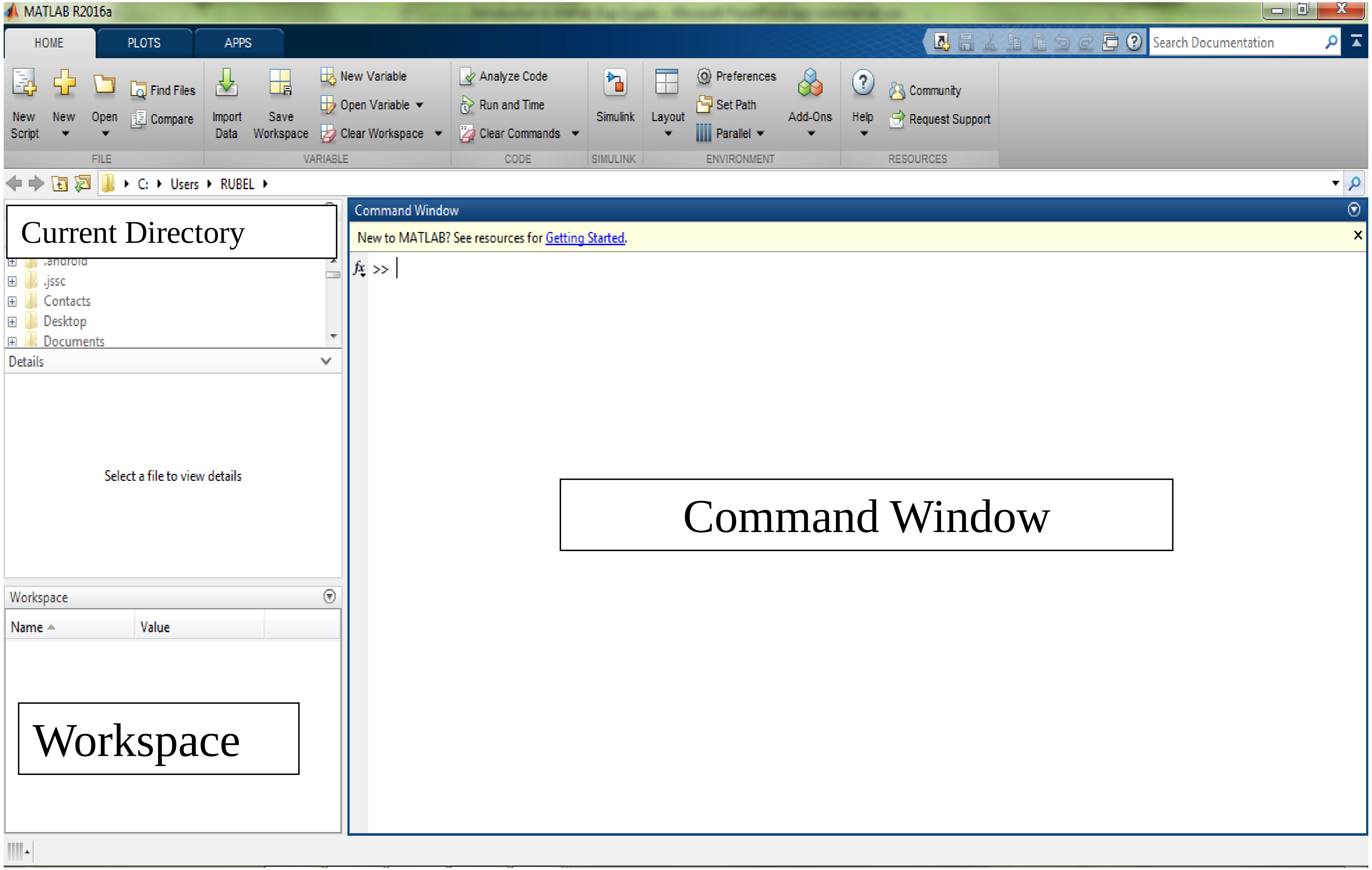
Some useful MATLAB Commands (Cont.)

```
>> format loose      % line space increased in command window
>> format compact   % line space decreased in command window
>> format long        % displays more digits after decimal points
>> format short    % displays less digits after decimal points
```

Note: ‘format long/short’ has no effect on accuracy during the calculation. The commands just show larger or smaller no of digits after decimal point in the display.

```
>> exist(‘name of variable/function’)    % Check if variables or functions are defined.
```

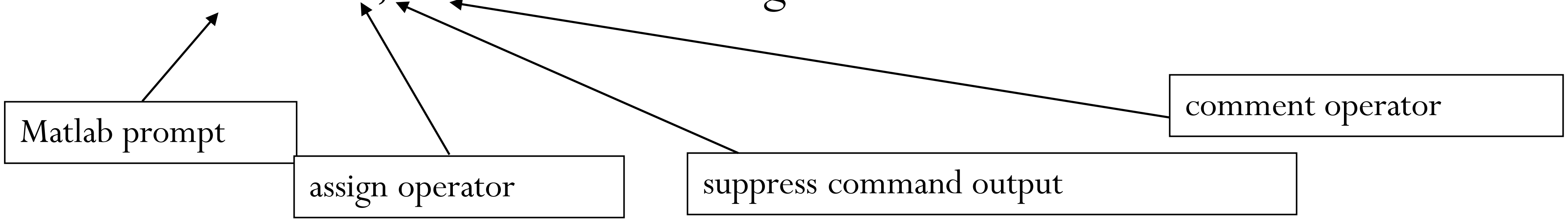
MATLAB Preview



Variables

- Don’t have to declare type, is case sensitive
- variable begins with a letter, e.g., A2z or a2z
- can be a mix of letters, digits, and underscores (e.g., vector_A)
- Variable name can be up to 63 characters
- Don’t even have to initialise
- Just assign in command window

```
>>
>> a=12; % variable a is assigned 12
```



Matlab prompt

assign operator

suppress command output

comment operator

Try the same line without the semicolon and comments

Size of Variables

- All numerical variables in MATLAB are matrices, a mathematical data type corresponding to a two-dimensional array of numbers.

```
 >> m=3;
>> size(m)
 ans =
        1     1
>> a=[1,2,3];
>> size(a)
   ans =
        1     3
```

Remember these terms

Scalar: Single element variable like 1,5,42 etc.

Vector: If you group (row or column wise) a number of scalars together you end up with a vector.

```
Example:  a=[1, 2, 3];
          b=[6 ,7 ,8];
```

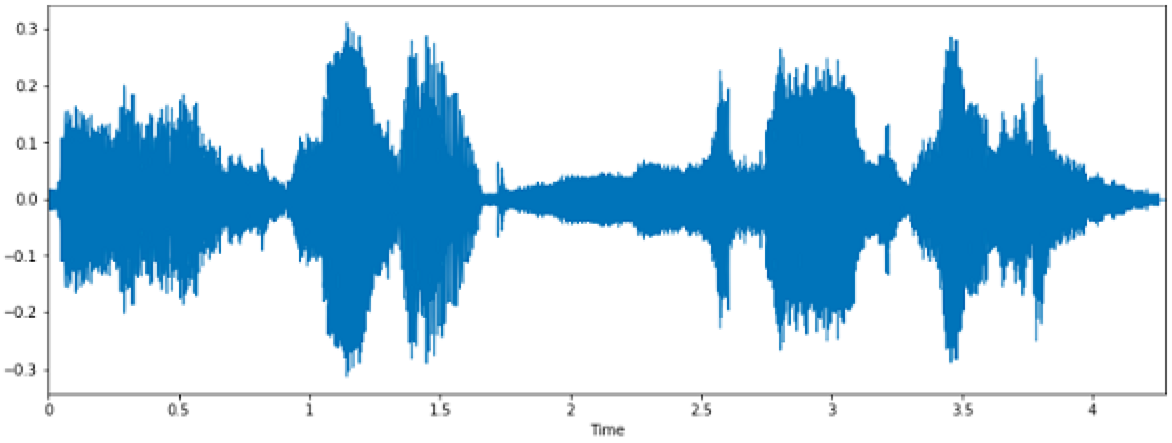
Matrix: A list of equal sized vector.

```
Example:  A= [a;b]
```

Tensor (Array): Three or more dimensional matrix

Example: Color Picture.

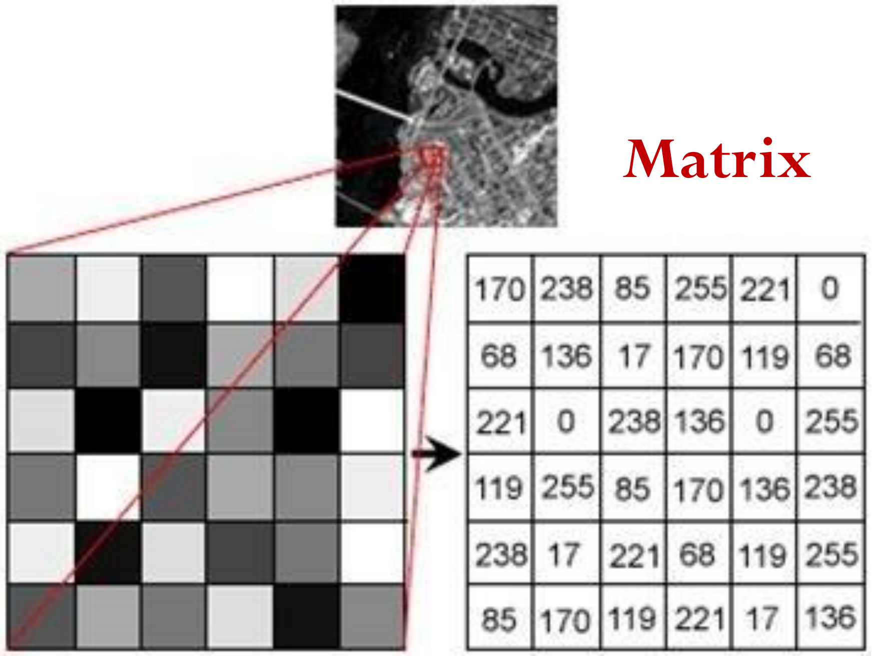
Audio Signal



Vector

{a1,a1,a3,a4,...............}

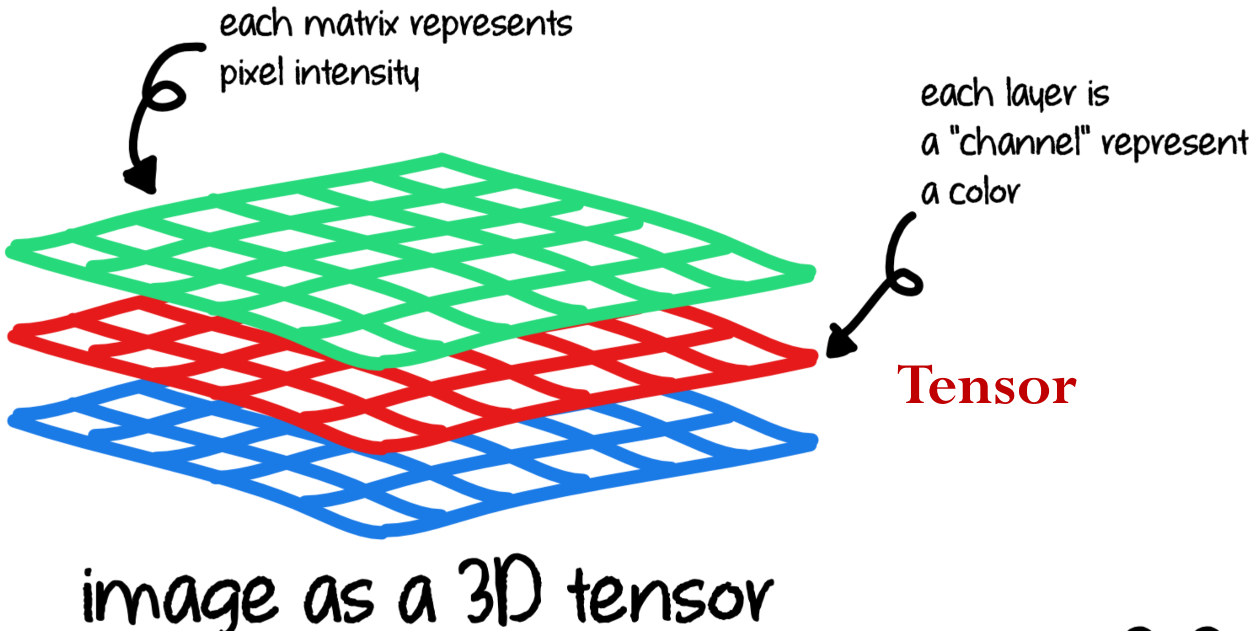
Gray Scale Image



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Matrix

Color Image



Tensor

Workspace

- The workspace is Matlab’s memory.
- Displaying contents of workspace.

```
>> a=12;
>> b=10;
>> c=a+b;
>> whos
  Name      Size            Bytes  Class     Attributes
    a         1x1               8  double
    b         1x1               8  double
    c         1x1               8  double
```

- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
>> whos
>> clear all; % delete all variables from workspace
>> whos
>> clc   % clear command window (workspace remain unchange)
```

Workspace (Cont.)

>> save   % save workspace variable to current directory. (before closing data one can save data for using in the next session)

>> load   % reload data.

>> save my_file a b   % create a new file named ‘my_file’in current directory and save variable a and b in that file.

>> load my_file   % load data from my_file to workspace.

Numeric Variable Types

- Floating Point Numbers (Double Precision and Single precision)
- Integer Numbers (signed and unsigned integer of 8,16,32,64-bits)

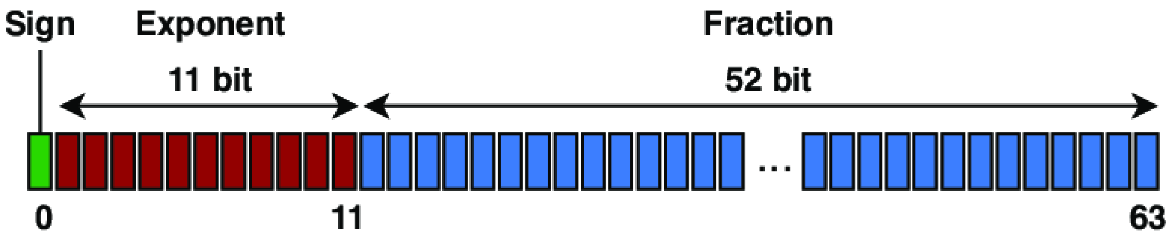
Floating-Point Numbers

MATLAB® represents floating-point numbers in either double-precision or single-precision format. The default is double precision.

Double-Precision Floating Point

- ✓ MATLAB constructs the double-precision (or double) data type according to **IEEE® Standard 754** for double precision.
- ✓ Any value stored as a double requires 64 bits.

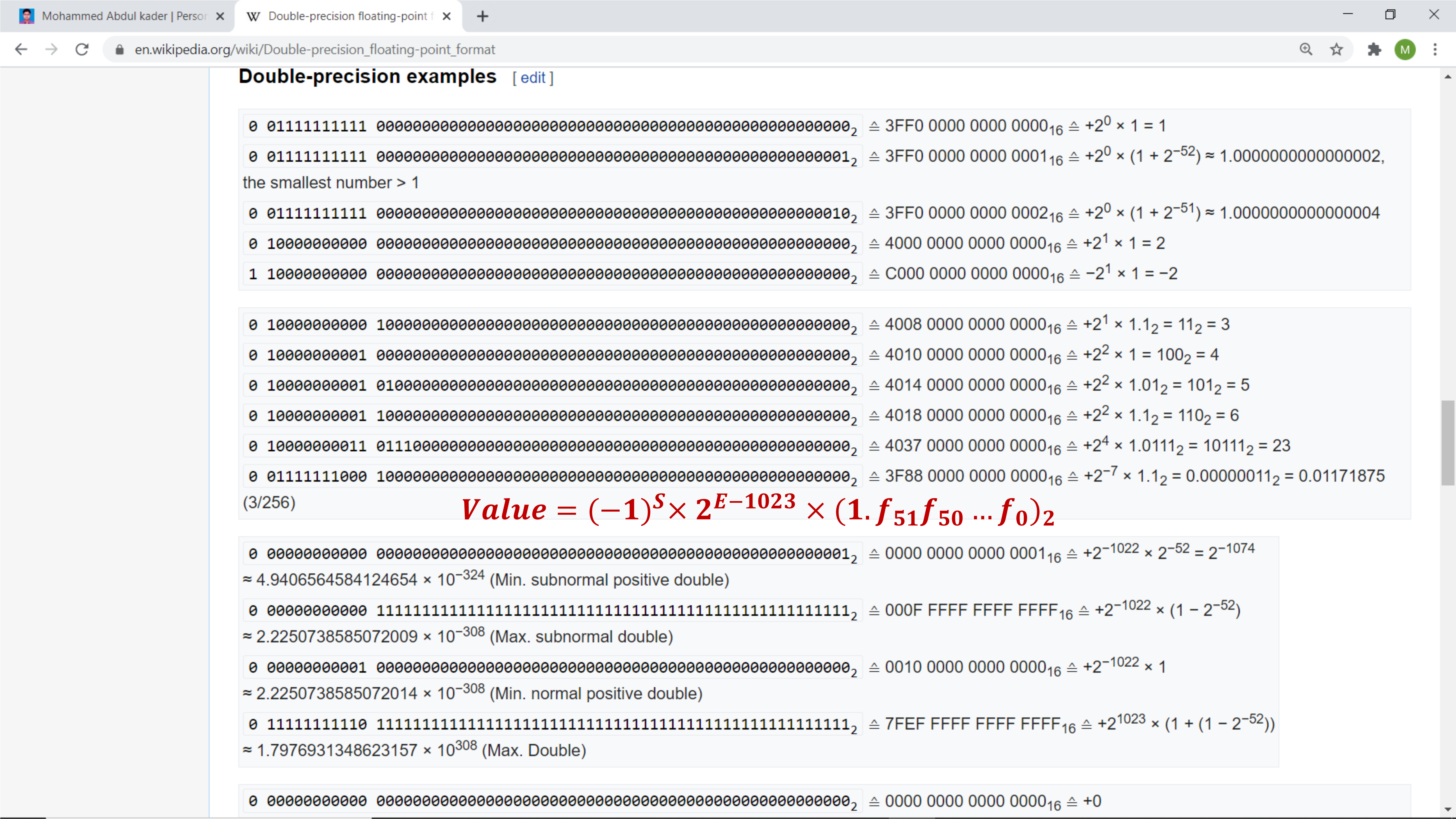
Bits	Usage
63	Sign (0 = positive, 1 = negative)
62 to 52	Exponent, biased by 1023
51 to 0	Fraction f of the number 1.f



$$\boldsymbol{Value = (-1)^{S} \times 2^{E-bias} \times (1.f)_2}$$

realmax or realmax('double')
realmin or realmin('double')

$$\boldsymbol{Value = (-1)^{S} \times 2^{E-1023} \times (1.f_{51}f_{50} \ldots f_0)_2}$$


Mohammed Abdul kader | Persor
Double-precision floating-point f
en.wikipedia.org/wiki/Double-precision_floating-point_format
M

Double-precision examples [edit]

0 01111111111 0000000000000000000000000000000000000000000000000000$_2$ ≙ 3FF0 0000 0000 0000$_{16}$ ≙ $+2^{0} \times 1 = 1$

0 01111111111 0000000000000000000000000000000000000000000000000001$_2$ ≙ 3FF0 0000 0000 0001$_{16}$ ≙ $+2^{0} \times (1 + 2^{-52}) \approx 1.0000000000000002$, the smallest number > 1

0 01111111111 0000000000000000000000000000000000000000000000000010$_2$ ≙ 3FF0 0000 0000 0002$_{16}$ ≙ $+2^{0} \times (1 + 2^{-51}) \approx 1.0000000000000004$

0 10000000000 0000000000000000000000000000000000000000000000000000$_2$ ≙ 4000 0000 0000 0000$_{16}$ ≙ $+2^{1} \times 1 = 2$

1 10000000000 0000000000000000000000000000000000000000000000000000$_2$ ≙ C000 0000 0000 0000$_{16}$ ≙ $-2^{1} \times 1 = -2$

0 10000000000 1000000000000000000000000000000000000000000000000000$_2$ ≙ 4008 0000 0000 0000$_{16}$ ≙ $+2^{1} \times 1.1_2 = 11_2 = 3$

0 10000000001 0000000000000000000000000000000000000000000000000000$_2$ ≙ 4010 0000 0000 0000$_{16}$ ≙ $+2^{2} \times 1 = 100_2 = 4$

0 10000000001 0100000000000000000000000000000000000000000000000000$_2$ ≙ 4014 0000 0000 0000$_{16}$ ≙ $+2^{2} \times 1.01_2 = 101_2 = 5$

0 10000000001 1000000000000000000000000000000000000000000000000000$_2$ ≙ 4018 0000 0000 0000$_{16}$ ≙ $+2^{2} \times 1.1_2 = 110_2 = 6$

0 10000000011 0111000000000000000000000000000000000000000000000000$_2$ ≙ 4037 0000 0000 0000$_{16}$ ≙ $+2^{4} \times 1.0111_2 = 10111_2 = 23$

0 01111111000 1000000000000000000000000000000000000000000000000000$_2$ ≙ 3F88 0000 0000 0000$_{16}$ ≙ $+2^{-7} \times 1.1_2 = 0.00000011_2 = 0.01171875$ (3/256)

$$Value = (-1)^{S} \times 2^{E-1023} \times (1.f_{51}f_{50}\ldots f_{0})_2$$

0 00000000000 0000000000000000000000000000000000000000000000000001$_2$ ≙ 0000 0000 0000 0001$_{16}$ ≙ $+2^{-1022} \times 2^{-52} = 2^{-1074}$ $\approx 4.9406564584124654 \times 10^{-324}$ (Min. subnormal positive double)

0 00000000000 1111111111111111111111111111111111111111111111111111$_2$ ≙ 000F FFFF FFFF FFFF$_{16}$ ≙ $+2^{-1022} \times (1 - 2^{-52})$ $\approx 2.2250738585072009 \times 10^{-308}$ (Max. subnormal double)

0 00000000001 0000000000000000000000000000000000000000000000000000$_2$ ≙ 0010 0000 0000 0000$_{16}$ ≙ $+2^{-1022} \times 1$ $\approx 2.2250738585072014 \times 10^{-308}$ (Min. normal positive double)

0 11111111110 1111111111111111111111111111111111111111111111111111$_2$ ≙ 7FEF FFFF FFFF FFFF$_{16}$ ≙ $+2^{1023} \times (1 + (1 - 2^{-52}))$ $\approx 1.7976931348623157 \times 10^{308}$ (Max. Double)

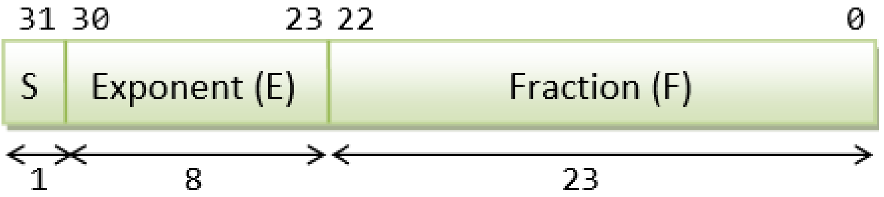
0 00000000000 0000000000000000000000000000000000000000000000000000$_2$ ≙ 0000 0000 0000 0000$_{16}$ ≙ +0

Numeric Variable Types (Cont.)

Single-Precision Floating Point

- ✓ MATLAB constructs the single-precision (or single) data type according to IEEE Standard 754 for single precision.
- ✓ Any value stored as a single requires 32 bits.

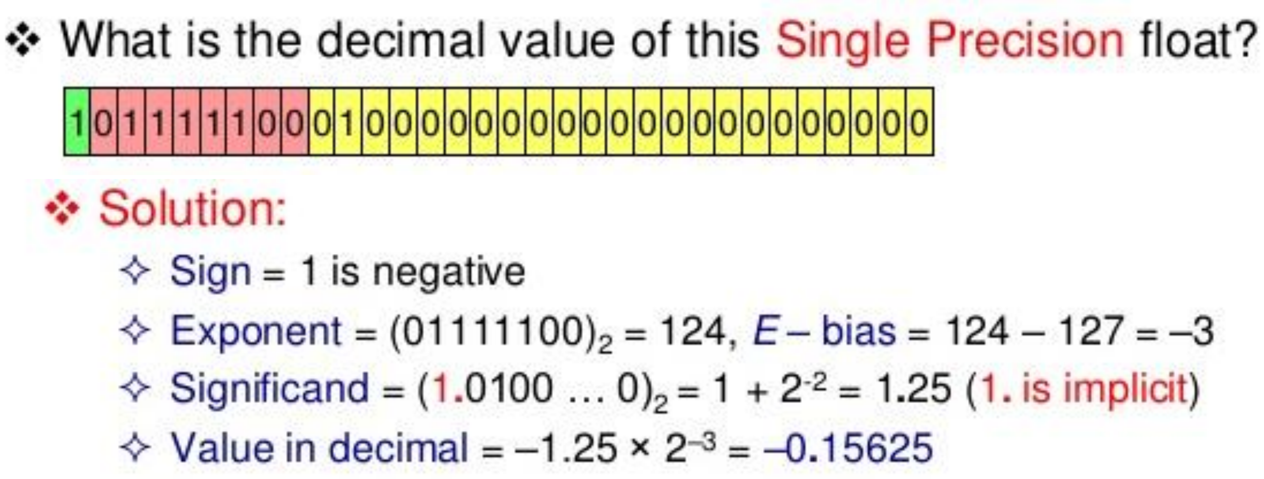
Bits	Usage
31	Sign (0 = positive, 1 = negative)
30 to 23	Exponent, biased by 127
22 to 0	Fraction f of the number 1.f



32-bit Single-Precision Floating-point Number

$$Value = (-1)^S \times 2^{E-bias} \times (1.f)_2$$

$$Value = (-1)^S \times 2^{E-127} \times (1.f_{22}f_{21}\dots f_0)_2$$



❖ What is the decimal value of this Single Precision float?

1	0	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

❖ Solution:

- ✧ Sign = 1 is negative
- ✧ Exponent = $(01111100)_2$ = 124, E – bias = 124 – 127 = –3
- ✧ Significand = $(1.0100 \dots 0)_2 = 1 + 2^{-2}$ = 1.25 (1. is implicit)
- ✧ Value in decimal = $-1.25 \times 2^{-3} = -0.15625$

Numeric Variable Types (Cont.)

Integer Type Variables (Signed and Unsigned)

int8	8-bit signed integer arrays
int16	16-bit signed integer arrays
int32	32-bit signed integer arrays
int64	64-bit signed integer arrays
uint8	8-bit unsigned integer arrays
uint16	16-bit unsigned integer arrays
uint32	32-bit unsigned integer arrays
uint64	64-bit unsigned integer arrays

Related functions:

intmax('type'), Example: intmax('uint8')

intmin('type'), Example: intmin('uint8')

S	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**

Range: $-2^7\ to\ (2^7 - 1) \gg -128\ to + 127$

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**

Range: $0\ to\ (2^8 - 1) \gg 0\ to\ 255$

Matrices and Access to matrix elements

- Don’t need to initialise type, or dimensions

```
>>A = [3 2 1; 5 1 0; 2 1 7]
A =
     3     2     1
     5     1     0
     2     1     7
```

square brackets to define matrices

semicolon for next row in matrix

- Access elements of a matrix

```
>> A = [3 2 1; 5 1 0; 2 1 7]
>>A(1,2)
ans= 2
```

- Remember Matrix(row,column)
- Naming convention: Matrix variables start with a capital letter while vectors or scalar variables start with a simple letter.

The colon (:) Operator and Matrices (Accessing Parts of Matrix)

- The colon is one of the most useful operators in MATLAB. It can create vectors, subscript arrays, and specify for iterations.

Use of colon (:) operator

✓ Create Unit-Spaced Vector

i : k

```
>> 1:10
ans =
    1    2    3    4    5    6    7    8    9   10
```

✓ Create Vector with Specified Increment

i : j : k

```
>> 1:2:10
ans =
    1    3    5    7    9
```

✓ Subscript vector- A(j:k) equivalent to the vector [A(j), A(j+1), …, A(k)].

```
>> m=[2  6  3  1  8  9  0  2  4];
>> p=m(2:7)
p = 6    3    1    8    9    0
```

The colon (:) Operator and Matrices (Accessing Parts of Matrix)

✓ Index Matrix Rows and Columns

- A(:,n) is the nth column of matrix A.
- A(m,:) is the mth row of matrix A.
- A(:) reshapes all elements of A into a single column vector. This has no effect if A is already a column vector.
- A(:,j:k) includes all subscripts in the first dimension but uses the vector j:k to index in the second dimension

```
A =

   3    2    1
   5    1    0
   2    1    7
```

```
>>A(:,2)
ans =
    2
    1
    1
```

```
>>A(3,2:3)
ans =

    1    7
>> A(3,:)
ans =
    2    1    7
```

```
>> A(:)
ans =3
     5
     2
     2
     1
     1
     1
     0
     7
```

What'll happen if you type

A(:,:) ?

A(1:end, 1) ?

A(end-1, end-2) ?

sum(A(:))

Manipulating Matrices

```
>> A '          % transpose
>> B*A          % matrix multiplication
>> B.*A         % element by element multiplication (Array Multiplication)
>> B/A          % matrix division
>> B./A         % element by element division (B over A)
>> B.\A         % element by element division (B under A)
>> [B A]        % Join matrices (horizontally)
>> [B; A]       % Join matrices (vertically)
```

```
A =

     3     2     1
     5     1     0
     2     1     7
```

```
B =

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

Task: Create matrices A and B and try out the matrix operators in this slide

MATLAB Graphics

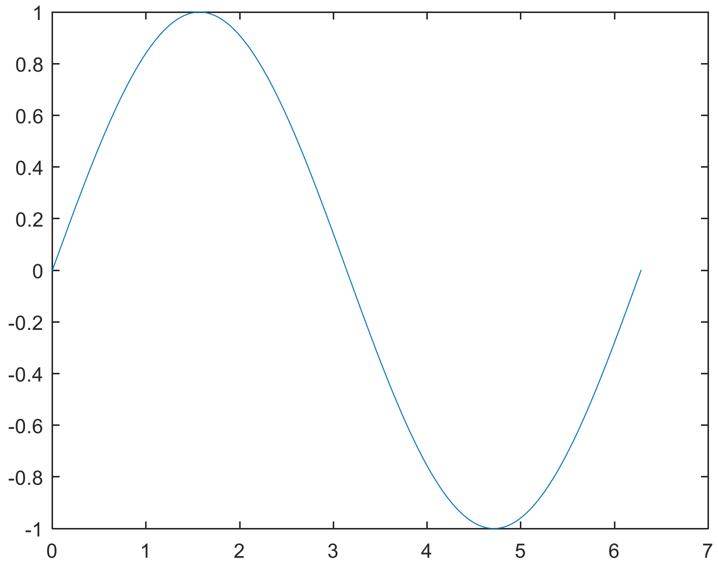
- Line plot
- Bar graph
- Surface plot
- Contour plot
- MATLAB has 2D, 3D visualization tools as well as other graphics packages.

MATLAB Graphics: 2D-Line Plot

plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

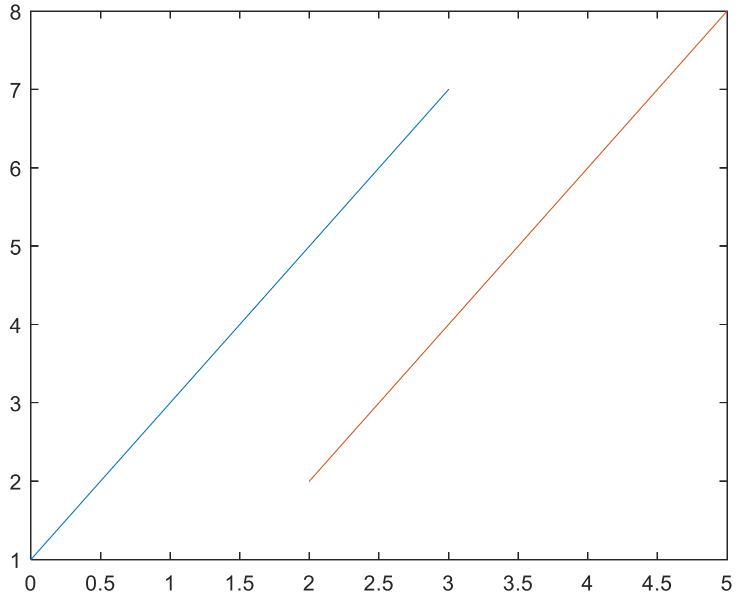
- ✓ If X and Y are both vectors, then they must have equal length. The plot function plots Y versus X.
- ✓ If X and Y are both matrices, then they must have equal size. The plot function plots columns of Y versus columns of X.

```
>> t = 0:pi/100:2*pi;
>> y = sin(t);
>> plot(t,y)
```



```
>> A=[1 2; 3 4; 5 6; 7 8]
>> B=[0 2; 1 3; 2 4; 3 5]
>> plot(B,A)
```

```
A =

     1     2
     3     4
     5     6
     7     8

B =

     0     2
     1     3
     2     4
     3     5
```

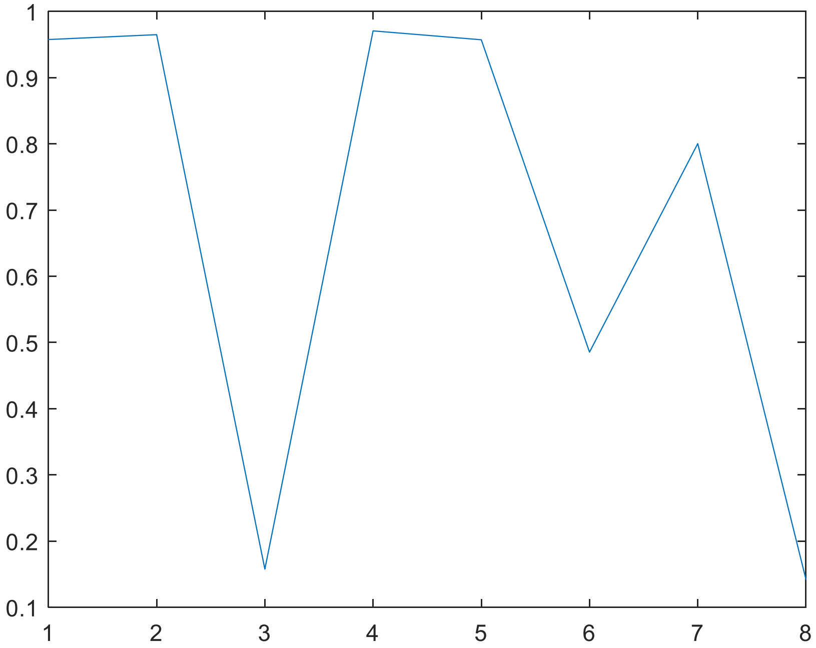


MATLAB Graphics: 2D-Line Plot (Cont.)

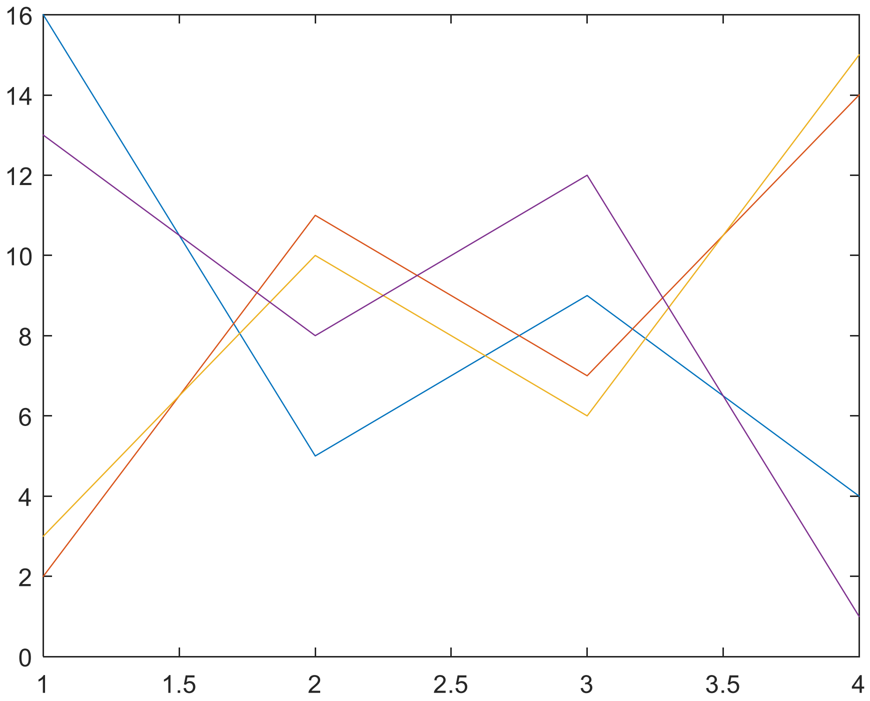
plot(Y) creates a 2-D line plot of the data in Y versus the index of each value.

- ✓ If Y is a vector, then the x-axis scale ranges from 1 to length(Y).
- ✓ If Y is a matrix, then the plot function plots the columns of Y versus their row number. The x-axis scale ranges from 1 to the number of rows in Y.

```
>> y=rand(1,8)
>> plot(y)
```



```
>> A=magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> plot(A)
```



MATLAB Graphics: Line Plot (Cont.)

```
>> xlabel(‘t’);
>> ylabel(‘sin(t)’);
>> title(‘The plot of t vs sin(t)’);
>> axis([0,12,-10,20]);  % axis([XMIN XMAX YMIN YMAX]) sets scaling for the x- and y-axes
                            on the current plot.
>> grid on
>> grid MINOR
>> grid off
>> axis off
>> axis on
>> close(1)
>> close all
```

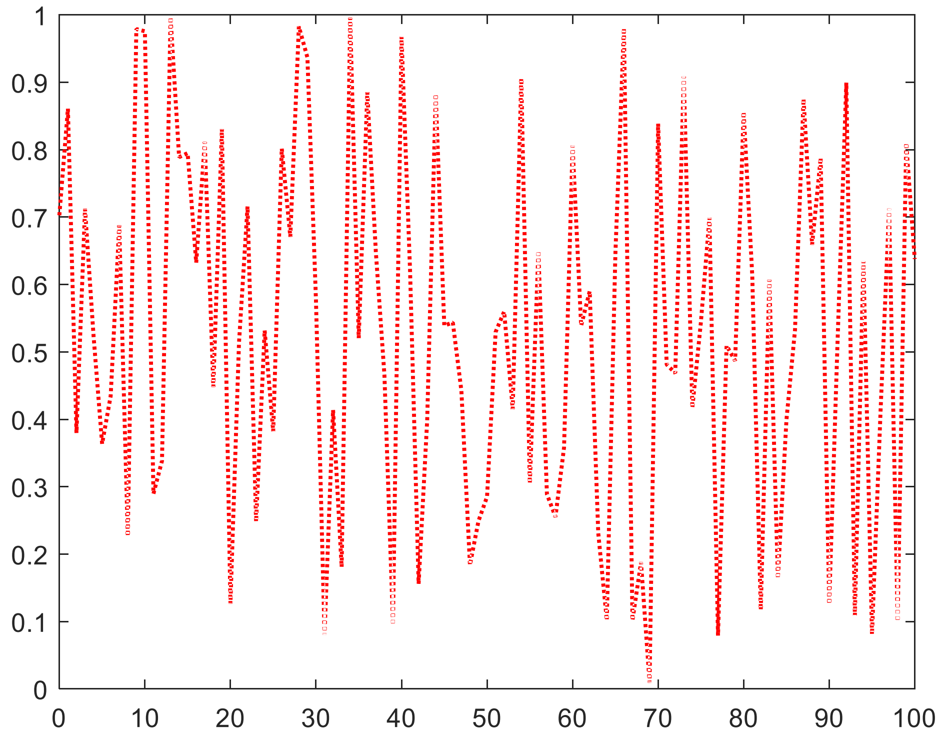
MATLAB Graphics: 2D-Line Plot (Cont.)

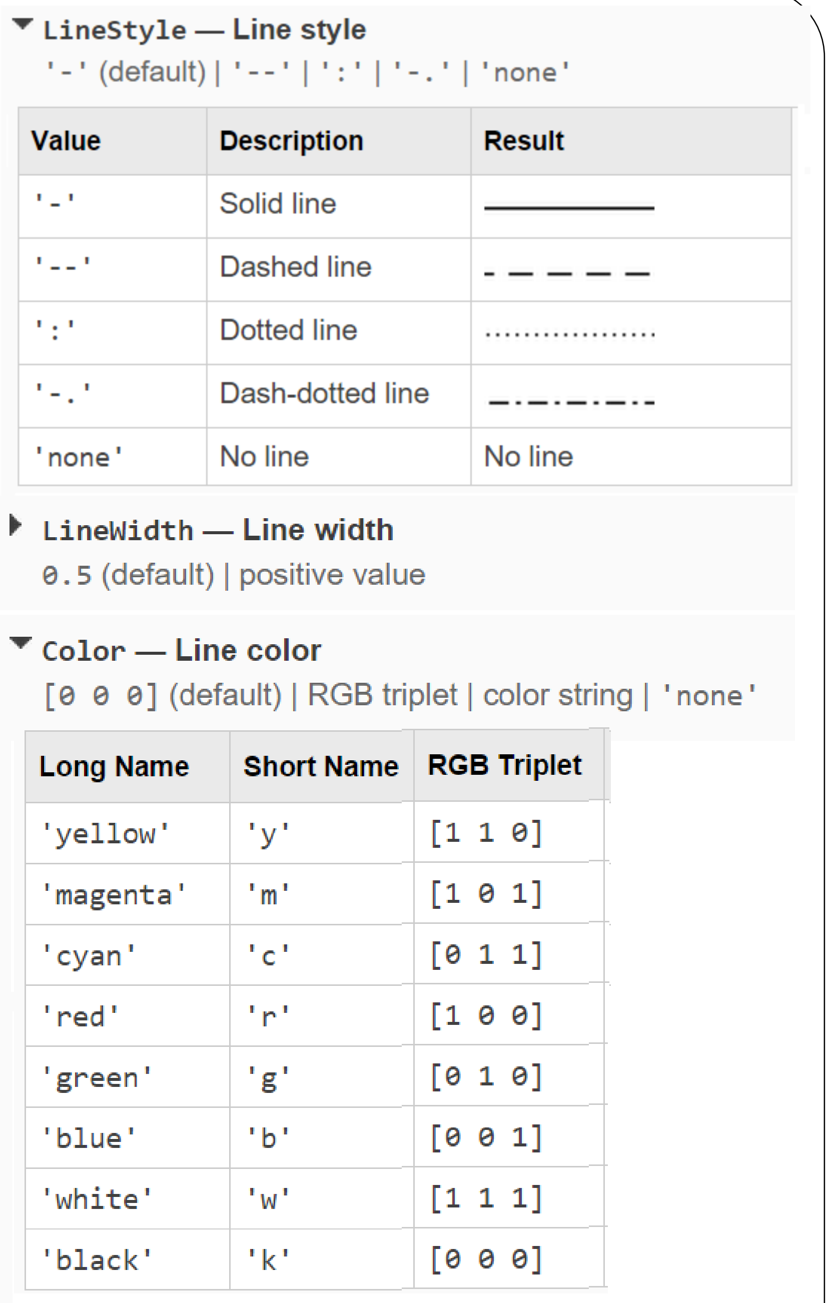
plot(X,Y,LineSpec)

plot(Y,LineSpec)

LineSpec: sets the line appearance and behavior (style, marker symbol, and color)

```
>> x=0:100;
>> y=rand(1,length(x));
>> plot(x,y,':r','LineWidth',1.5)
```





LineStyle — Line style

'-' (default) | '--' | ':' | '-.' | 'none'

Value	Description	Result
'-'	Solid line	————
'--'	Dashed line	– – – – –
':'	Dotted line	··················
'-.'	Dash-dotted line	—·—·—·—·
'none'	No line	No line

LineWidth — Line width

0.5 (default) | positive value

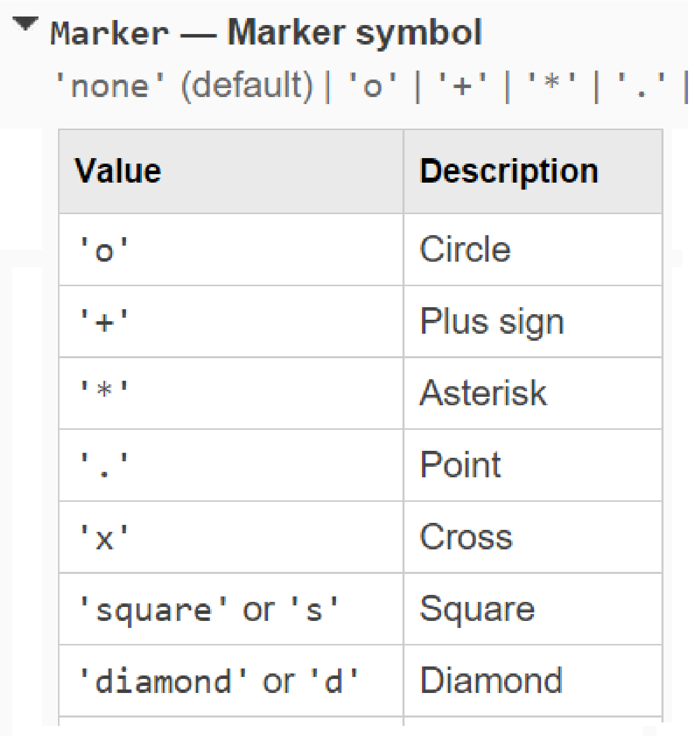
Color — Line color

[0 0 0] (default) | RGB triplet | color string | 'none'

Long Name	Short Name	RGB Triplet
'yellow'	'y'	[1 1 0]
'magenta'	'm'	[1 0 1]
'cyan'	'c'	[0 1 1]
'red'	'r'	[1 0 0]
'green'	'g'	[0 1 0]
'blue'	'b'	[0 0 1]
'white'	'w'	[1 1 1]
'black'	'k'	[0 0 0]

MATLAB Graphics: 2D-Line Plot (Cont.)

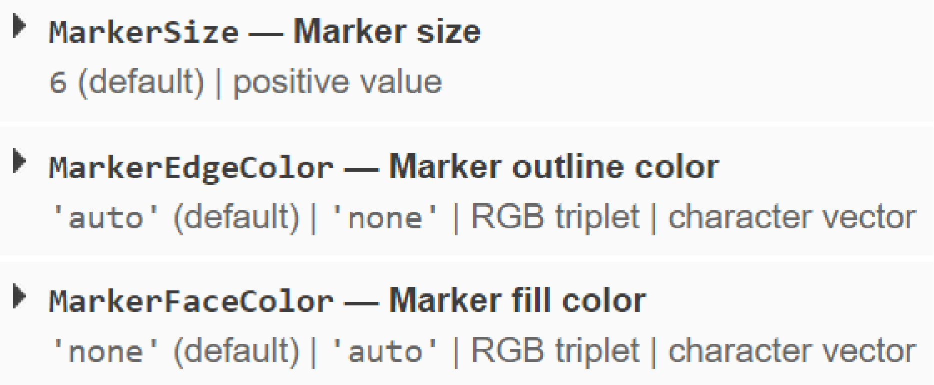
```
>> t=0:0.01:1;

>> y=sin(2*pi*3*t);

>>plot(t,y,'--gs','LineWidth',2,'MarkerSize',6,'MarkerEdgeColor','b','MarkerFaceColor',[0.5,0.5,0.5])
```



Marker — Marker symbol

'none' (default) | 'o' | '+' | '*' | '.' |

Value	Description
'o'	Circle
'+'	Plus sign
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond



MarkerSize — Marker size

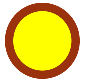
6 (default) | positive value

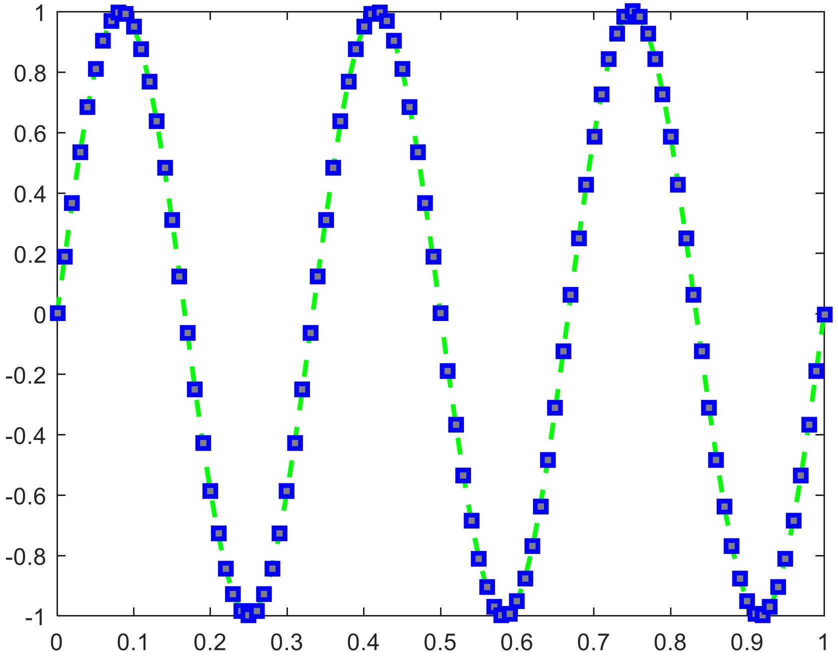
MarkerEdgeColor — Marker outline color

'auto' (default) | 'none' | RGB triplet | character vector

MarkerFaceColor — Marker fill color

'none' (default) | 'auto' | RGB triplet | character vector



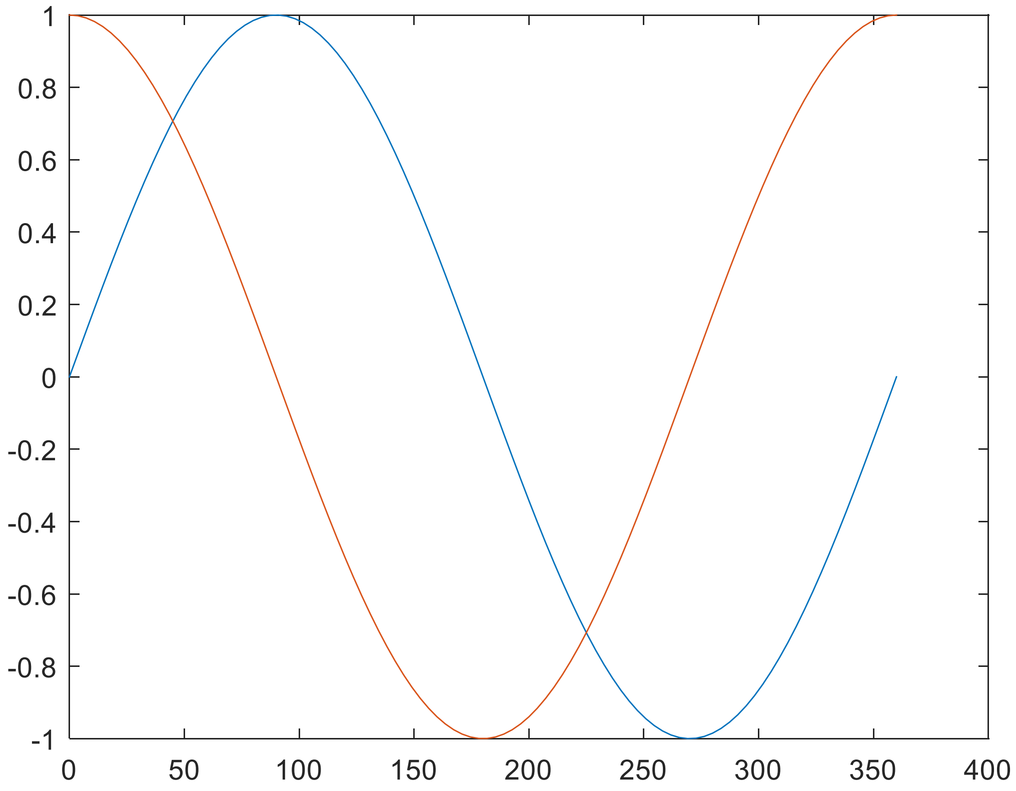


MATLAB Graphics: 2D-Line Plot (Cont.)

Multiple Vectors in Single Plot/Combine Plots in Same Axes (Method-1)

plot(X1,Y1,...,Xn,Yn) plots multiple X,Y pairs using the same axes for all lines.

```
>> x = linspace(0,360);
>> y1=sind(x);
>> y2=cosd(x);
>> plot(x,y1,x,y2)
```



Linearly spaced vector.

linspace(X1, X2) generates a row vector of 100 linearly equally spaced points between X1 and X2.

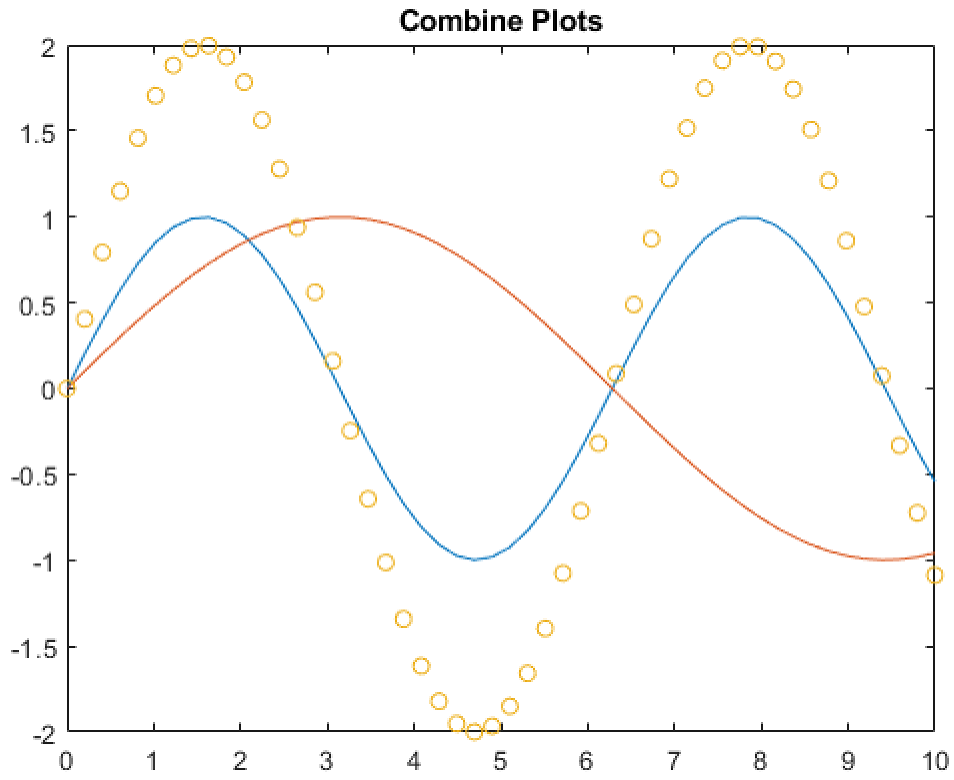
MATLAB Graphics: 2D-Line Plot (Cont.)

Multiple Vectors in Single Plot/Combine Plots in Same Axes (Method-2)

hold ON holds the current plot and all axis properties, including the current color and linestyle.

hold OFF returns to the default mode whereby PLOT commands erase the previous plots and reset all axis properties before drawing new plots.

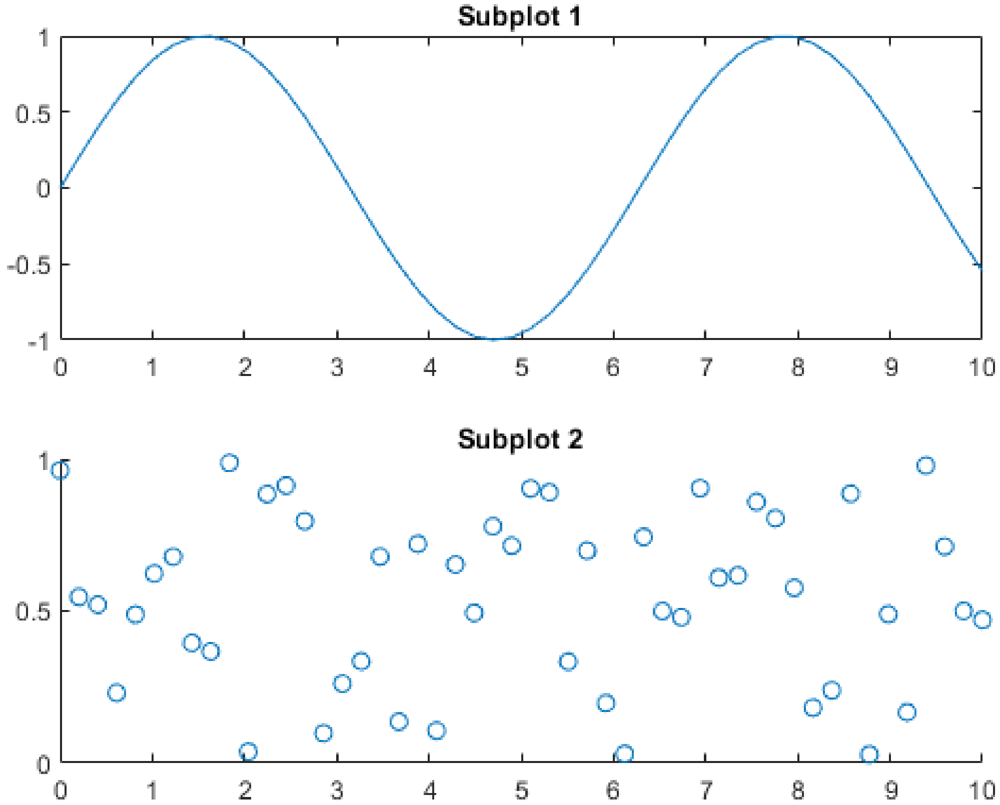
```
x = linspace(0,10,50);
y1 = sin(x);
plot(x,y1)
title('Combine Plots')
hold on
y2 = sin(x/2);
plot(x,y2)
y3 = 2*sin(x);
scatter(x,y3)
hold off
```



Combine Multiple Plots (Cont.)

Create Multiple Axes in Figure Using Subplots

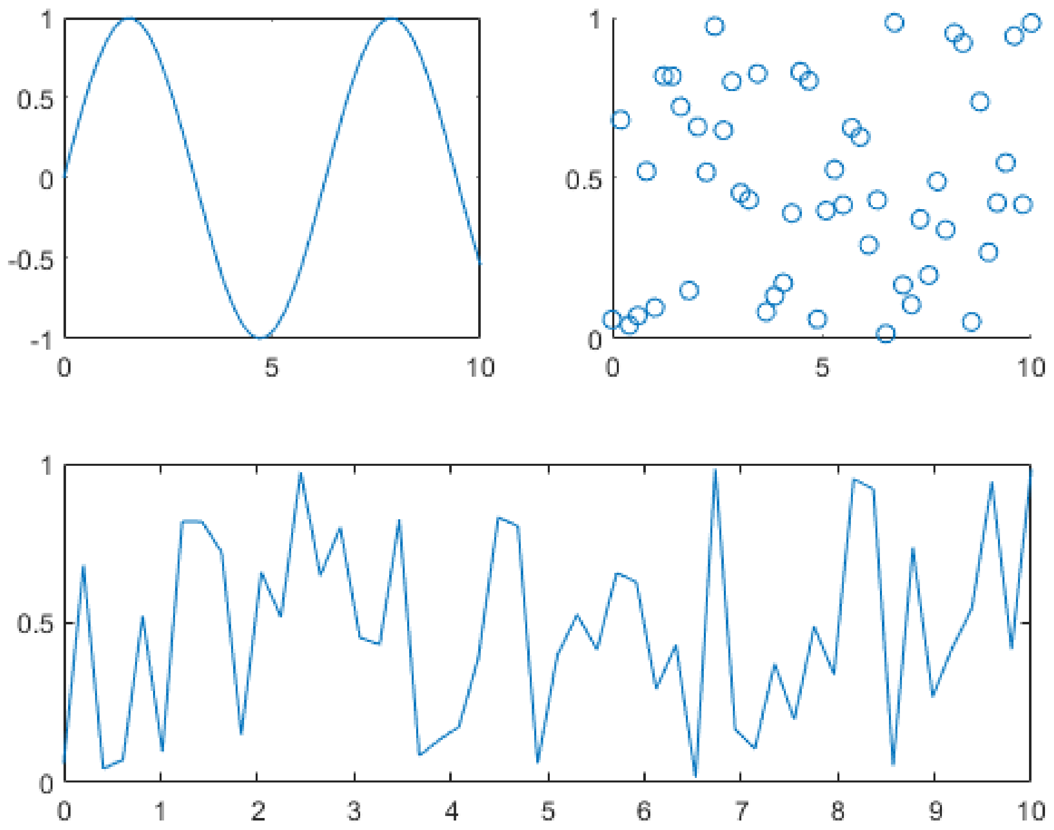
```
subplot(2,1,1);
x = linspace(0,10,50);
y1 = sin(x);
plot(x,y1)
title('Subplot 1')

subplot(2,1,2);
y2 = rand(50,1);
scatter(x,y2)
title('Subplot 2')
```



Combine Multiple Plots (Cont.)

Create Subplot that Spans Multiple Grid Positions

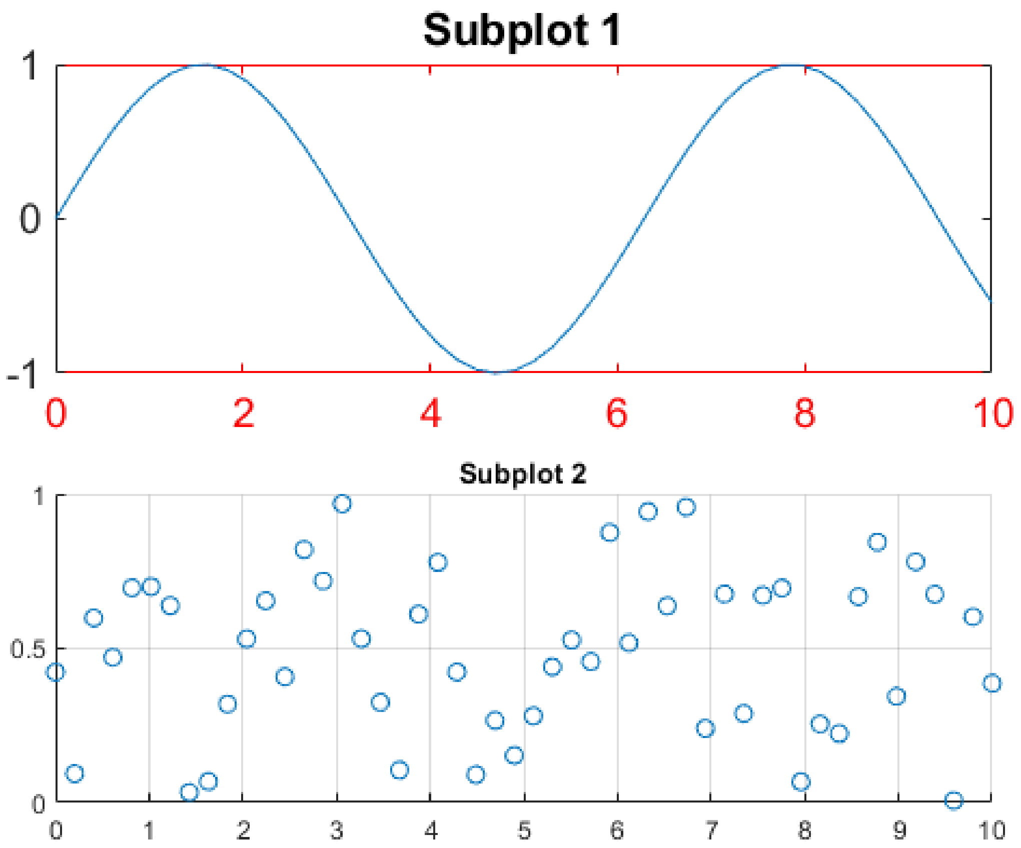
```
figure
subplot(2,2,1);
x = linspace(0,10,50);
y1 = sin(x);
plot(x,y1)

subplot(2,2,2);
y2 = rand(50,1);
scatter(x,y2)

subplot(2,2,[3 4]);
y3 = rand(50,1);
plot(x,y2)
```



Combine Multiple Plots (Cont.)

Modify Subplot Appearance

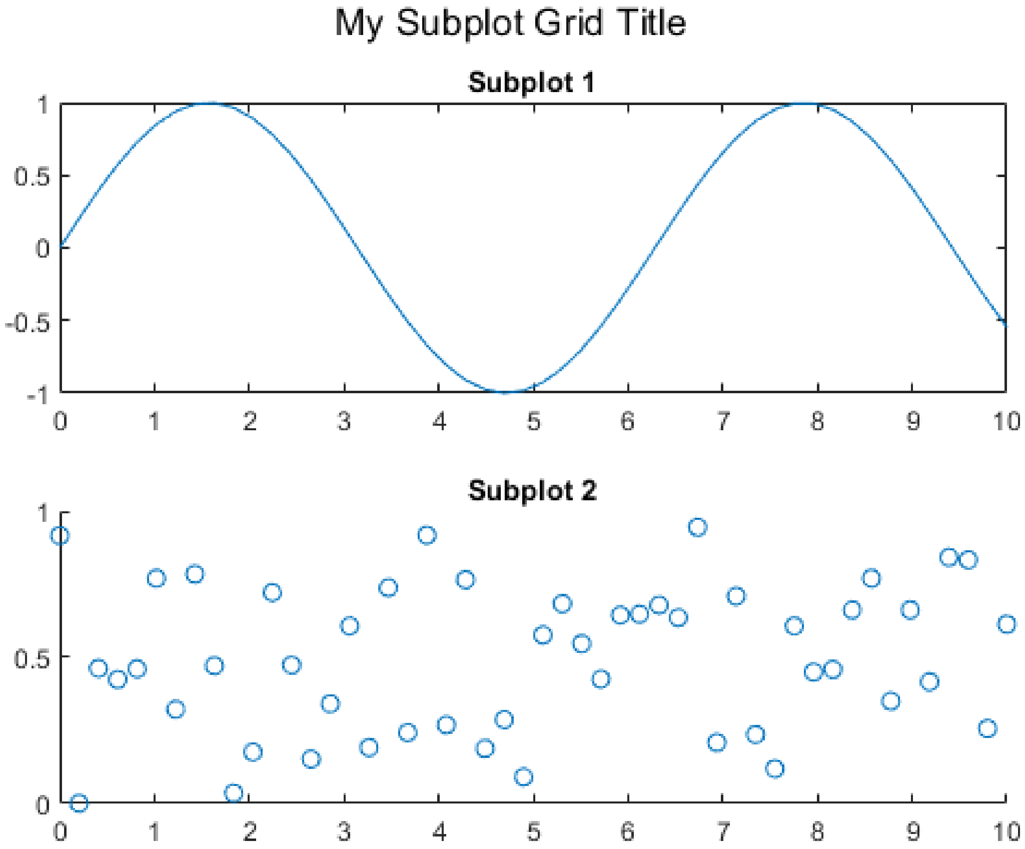
```
figure
ax1 = subplot(2,1,1);
x = linspace(0,10,50);
y1 = sin(x);
plot(ax1,x,y1)
title(ax1,'Subplot 1')
ax1.FontSize = 14;
ax1.XColor = 'red';

ax2 = subplot(2,1,2);
y2 = rand(50,1);
scatter(ax2,x,y2)
title(ax2,'Subplot 2')
grid(ax2,'on')
```



Combine Multiple Plots (Cont.)

Add SuperTitle to Grid of Subplots

```
subplot(2,1,1);
x = linspace(0,10,50);
y1 = sin(x);
plot(x,y1)
title('Subplot 1')

subplot(2,1,2);
y2 = rand(50,1);
scatter(x,y2)
title('Subplot 2')

sgtitle('My Subplot Grid Title')
```


My Subplot Grid Title
Subplot 1
Subplot 2