Introduction to MATLAB

Plotting Tools

I/O

Plotting in MATLAB

- 1D and 2D plotting

– plot/ semilogx / semilogy / loglog/ stem etc

- 3D plotting

– plot3 / bar3/comet3 / stem3 / pie3 / contour3 etc

- Useful function related to plotting

– grid / hold / axis / title / xlabel / ylabel/ mesh / surf /legend etc

- Other plotting tools

– You can do histograms, plot plots, image plot, scatter plot etc.
– To do those plots, you need to see corresponding commands

- GUI system design tools

– Type “guide” at Command window.
– It will give you many tools to design a GUI system such as radio button, slide bar, pop-up menu etc

Types of MATLAB Plots

There are various functions that you can use to plot data in MATLAB®. This table classifies and illustrates the common graphics functions.

Line Plots	Pie Charts, Bar Plots, and Histograms	Discrete Data Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots		Volume Visualization	Animation	Images
plot	area	stairs	polarplot	contour	quiver	surf	mesh	streamline	animatedline	image
plot3	pie	stem	polarhistogram	contourf	quiver3	surfc	meshc	streamslice	comet	imagesc
semilogx	pie3	stem3	polarscatter	contour3	feather	surfl	meshz	streamparticles	comet3	
semilogy	bar	scatter	compass	contourslice		ribbon	waterfall	streamribbon		
loglog	barh	scatter3	ezpolar	fcontour		pcolor	fmesh	streamtube		
errorbar	bar3	spy				fsurf		coneplot		
fplot	bar3h	plotmatrix				fimplicit3		slice		
fplot3	histogram	heatmap								
fimplicit	histogram2	geobubble								
	pareto	wordcloud								

Why Plot Data

- Large sets of data are usually difficult to interpret as tables of numbers

- Engineers and scientists use graphical techniques to reduce large sets of data to help gain insight

  – Observation of insightful trends
  – Identification of potential errors

2D (x, y) Plots

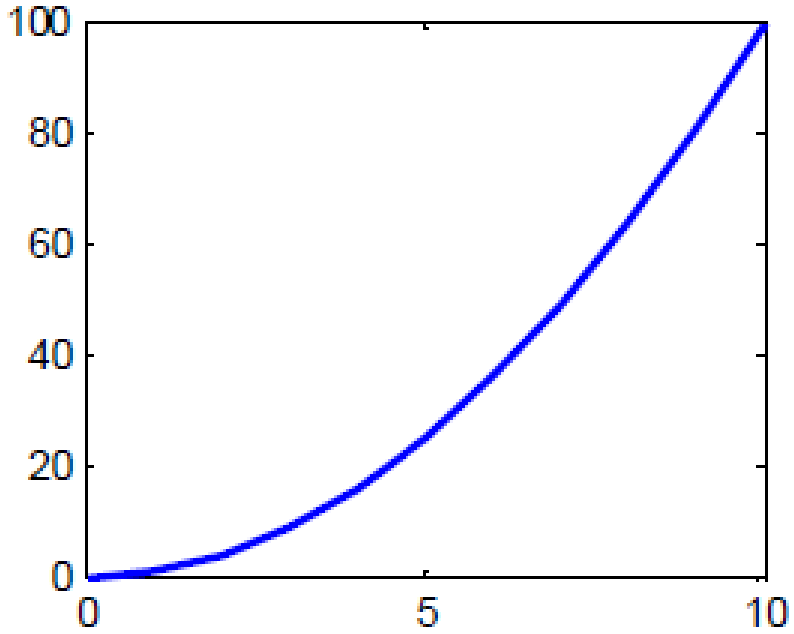
- Creating (x, y) plots is easy with plot, e.g.

- Generate data to plot:

```
>> x = 0:1:10;
>> y = x.^2;
```

  - Use the plot( ) command:

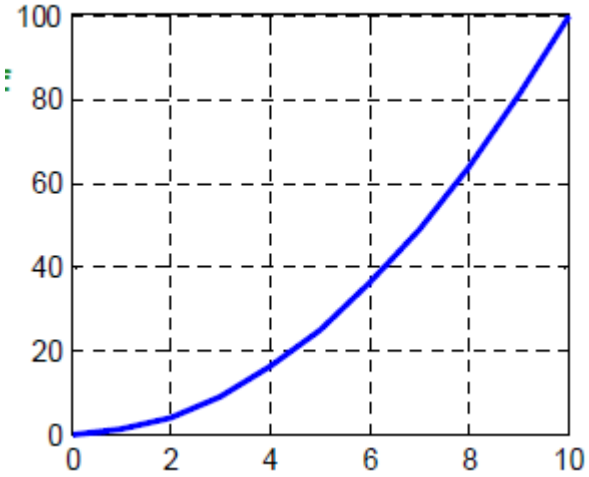
```
>> plot (x,y)
```



Adding a Grid to a Plot

Use the grid command to add a grid to the figure

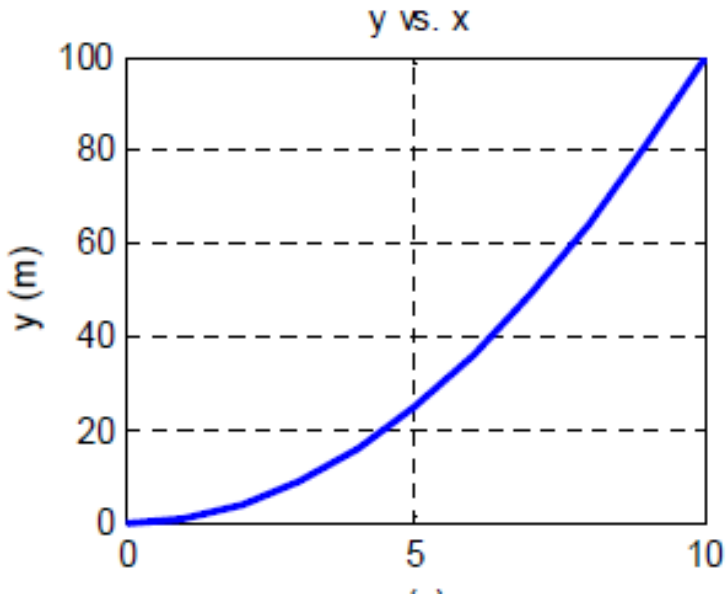
```
>> help grid % view all grid options an related
commands

>> grid on % turn grid on

>> grid off % turn grid off

>> grid % toggle grid display state
```



Plot Title and Axis Labels

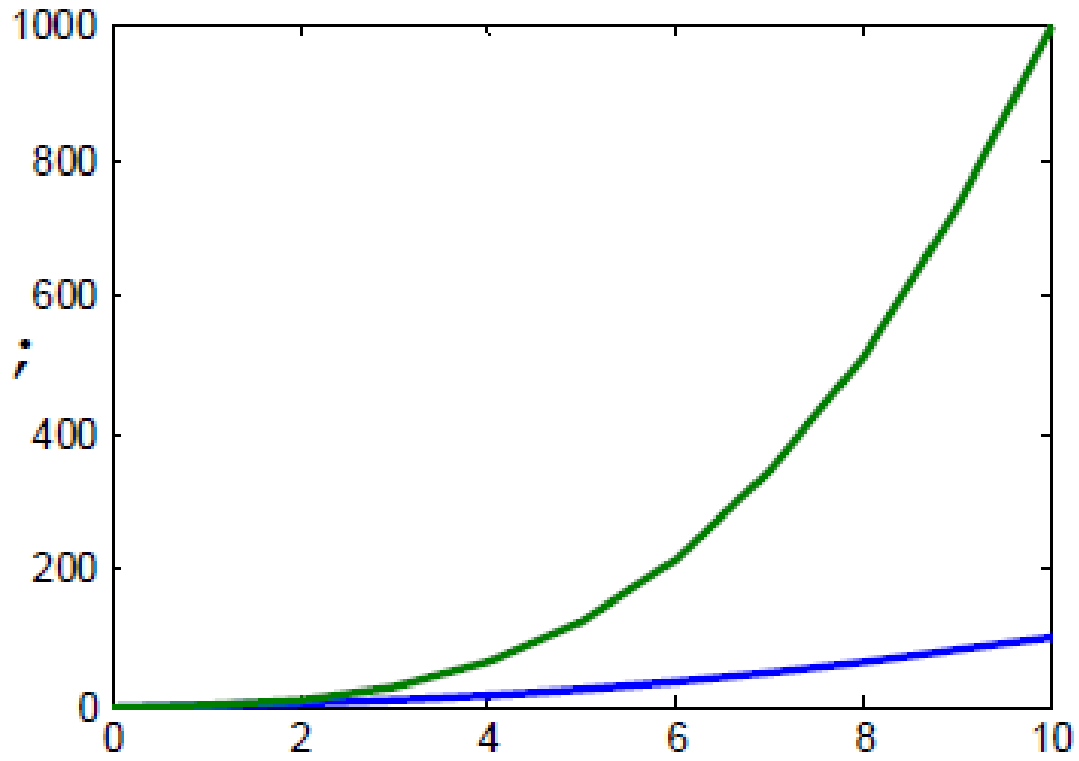
Add a title and axis labels to a plot as illustrated in the example:

```
>> title ('y vs. x')

>> xlabel ('x (s)')

>> ylabel ('y (m)')

% notice text strings are contained in
single quotes ' '
```



Plotting Multiple Curves on a Figure – Method 1

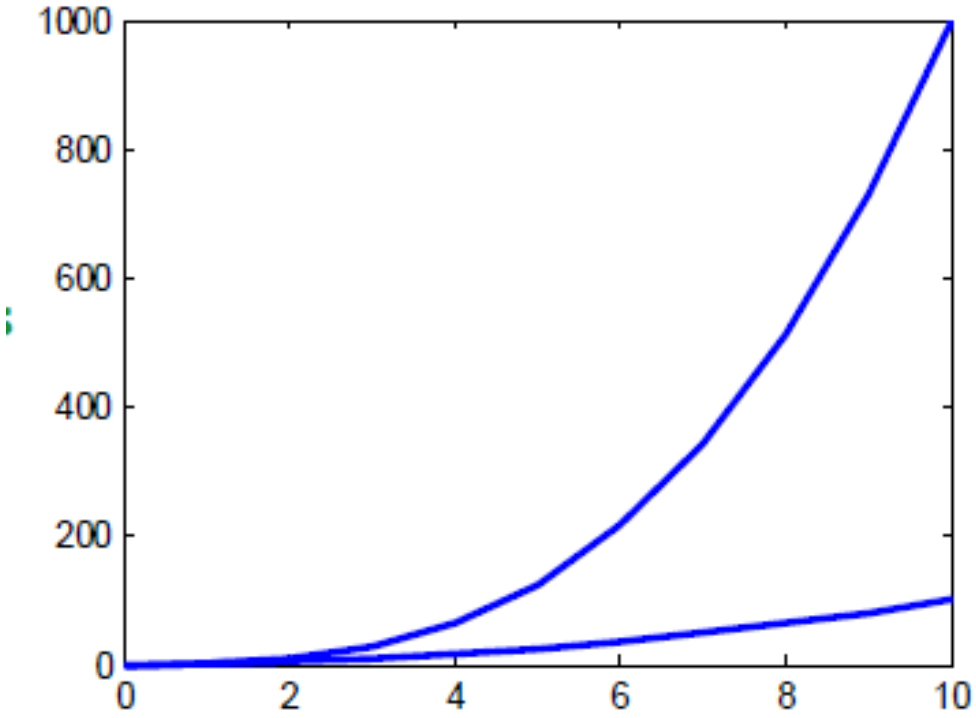
```
>> x = 1:1:10;

>> y1 = x.^2;

>> y2 = x.^3;

>> plot(x,y1,x,y2);
```



Plotting Multiple Curves on a Figure – Method 2

Multiple plots can be added to a figure using the hold command

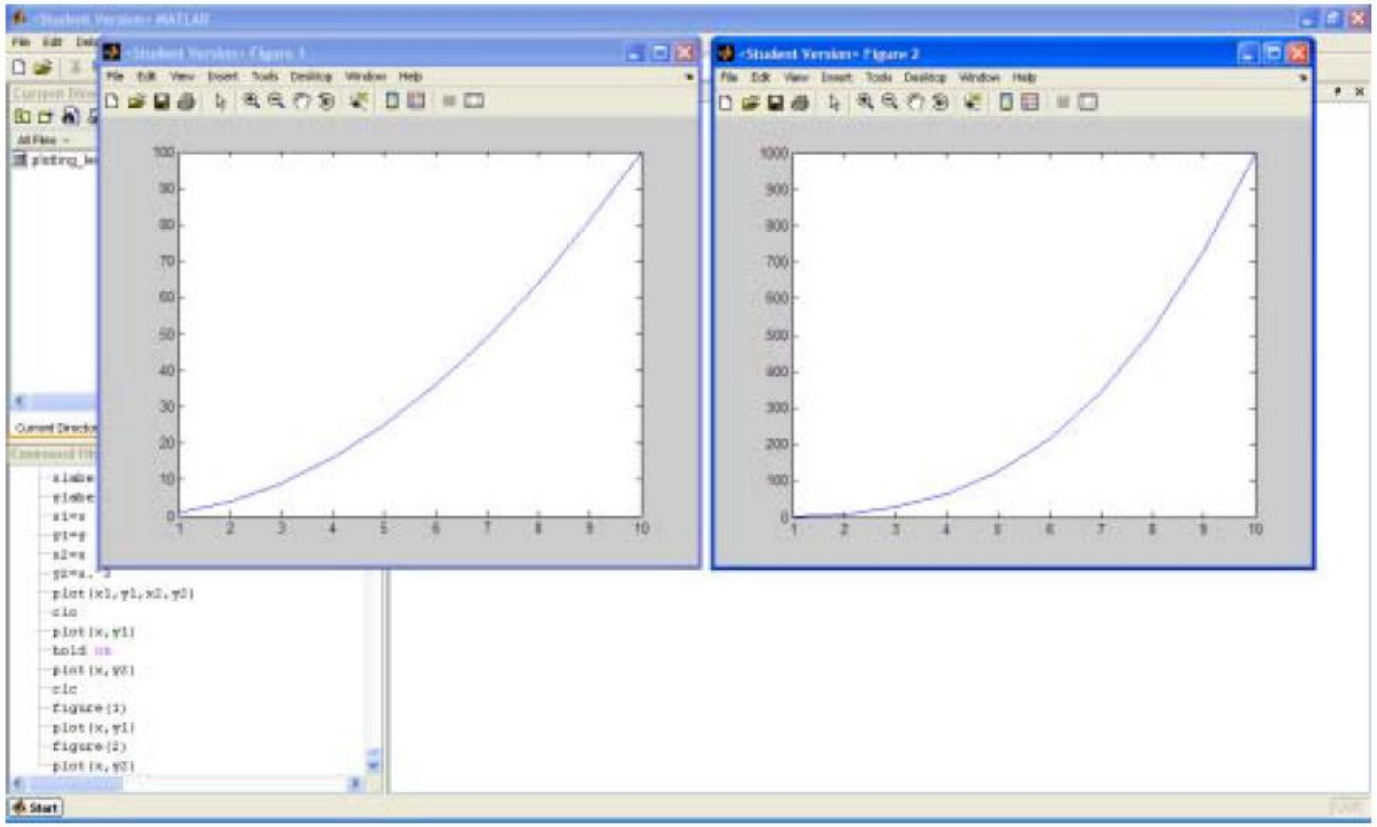
```
>> plot(x,y1)

>> hold on;

>> plot(x,y2)

>> … % other items

>> hold off
```



Multiple Figures

• Single (or multiple) plots can be created in multiple figures. A two-figures example:

```
>> figure(1);
>> plot(x,y1);
>> figure(2);
>> plot(x,y2);
```



Changing Line Style, Point Style, and Color

```
>> x = 1:1:10;

>> y = [4, 6.2, 8, 3.2, 11,
7.6, 5.4, 2.1, 9.3, 4];
```

- To plot the data with a red, dash-dot line and red stars for points

```
>> plot(x,y,'-.r*');

% Notice the single quotes again ' '
```

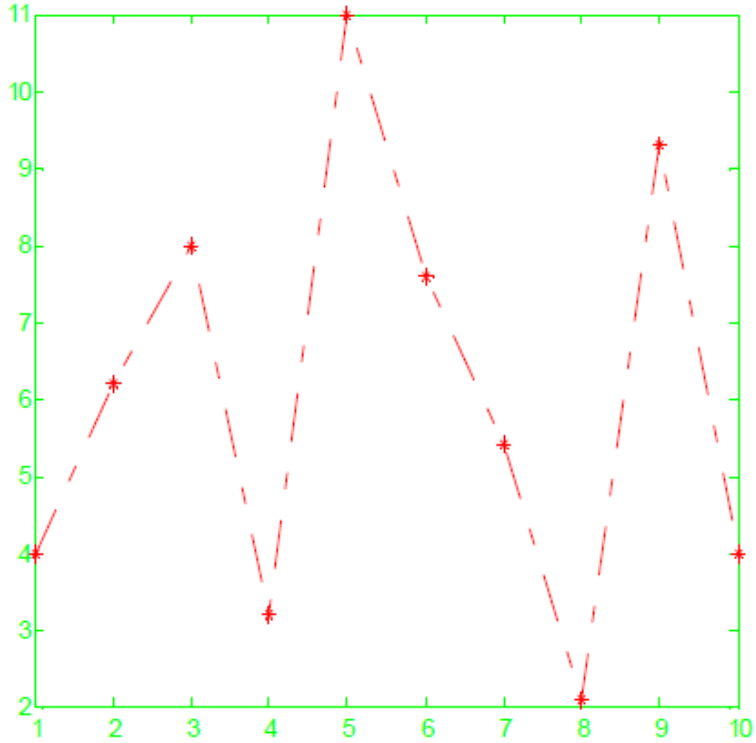


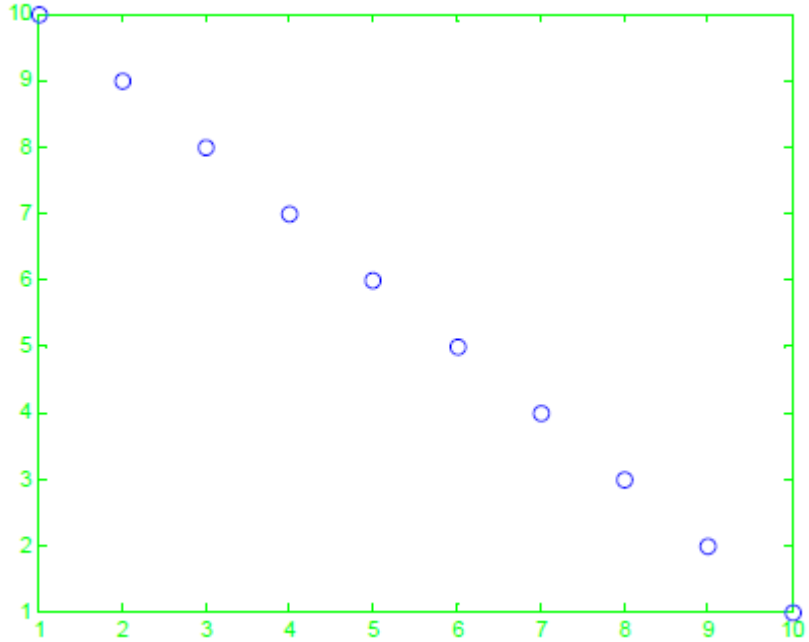
Table of Options

Line type	Indicator	Point type	Indicator	Color	Indicator
solid	-	point	.	blue	b
dotted	:	circle	o	green	g
dash-dot	-.	x-mark	x	red	r
dashed	--	plus	+	cyan	c
		star	*	magenta	m
		square	s	yellow	y
		diamond	d	black	k
		triangle down	v		
		triangle up	^		
		triangle left	<		
		triangle right	>		
		pentagram	p		
		hexagram	h		

Plotting Individual Data Points

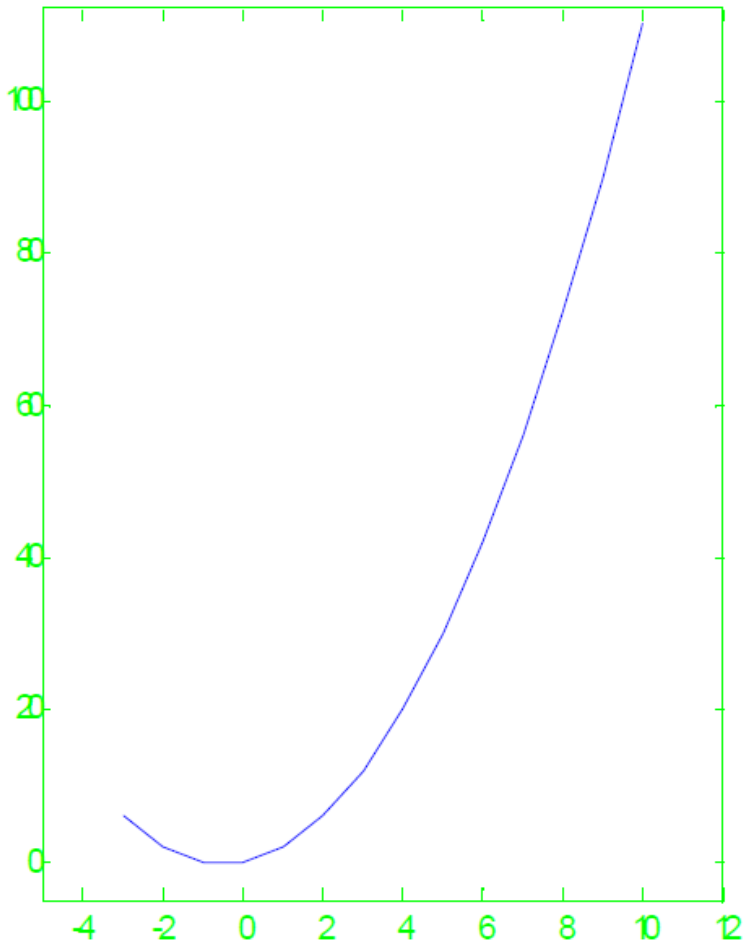
• Setting a marker type without specifying a line type will suppress the straight line drawn by default between the points that define the lines.

```
>> x = 1:10;

>> y = 10:-1,1;

>> plot(x,y);

>> plot(x,y,'o');
```



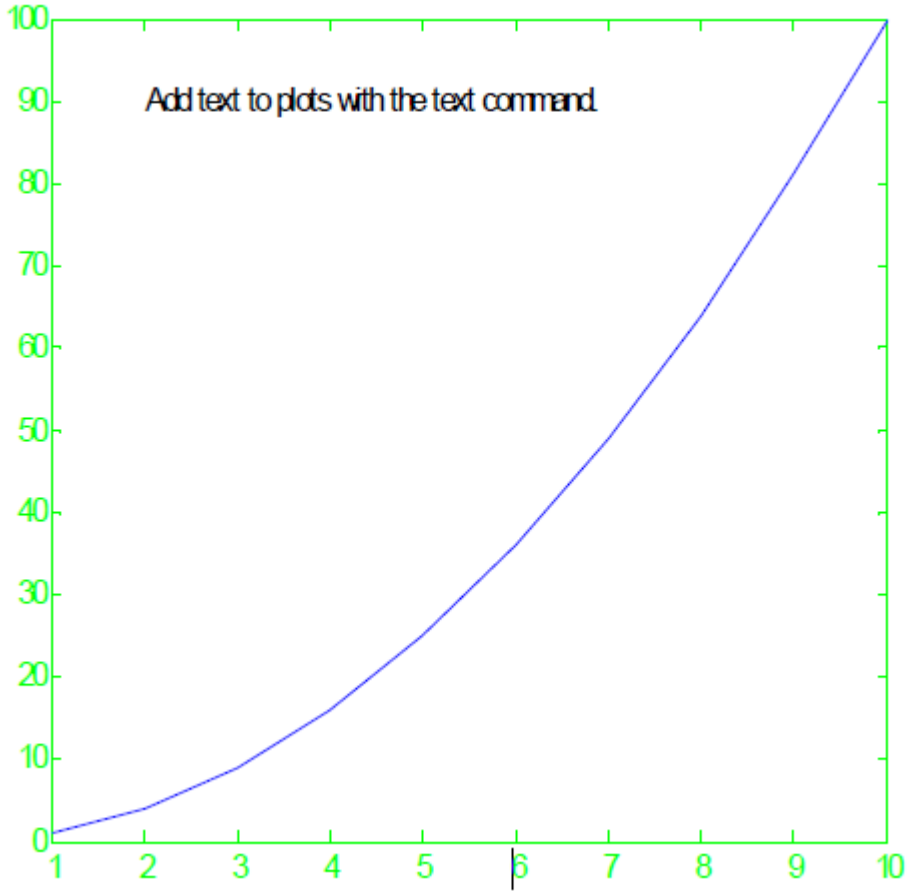
Axis Function

```
x = -3:1:10; y = x.^2 + x;

plot(x,y) %generate plot

y = axis %gets current axis
limits and assign to y

axis off %turn off axis
display

axis on

axis equal %set x and y to
same scale
%change axis limits

axis([-5,12, -5, 112]);

Help axis % more info
```



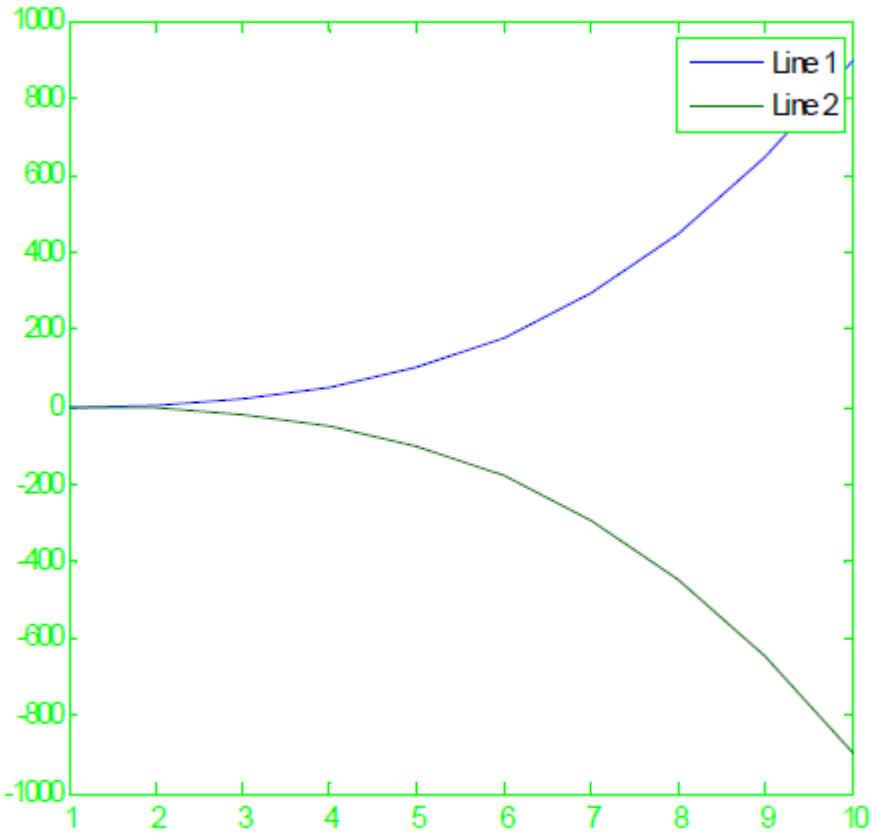
Annotations on Figures

```
>> x = 1:1:10;

>> y = x.^2;

>> plot(x,y);

>> text (2,90,'Add text to plots
with the text command');


>> % 2 & 90 is the location for
the text
```



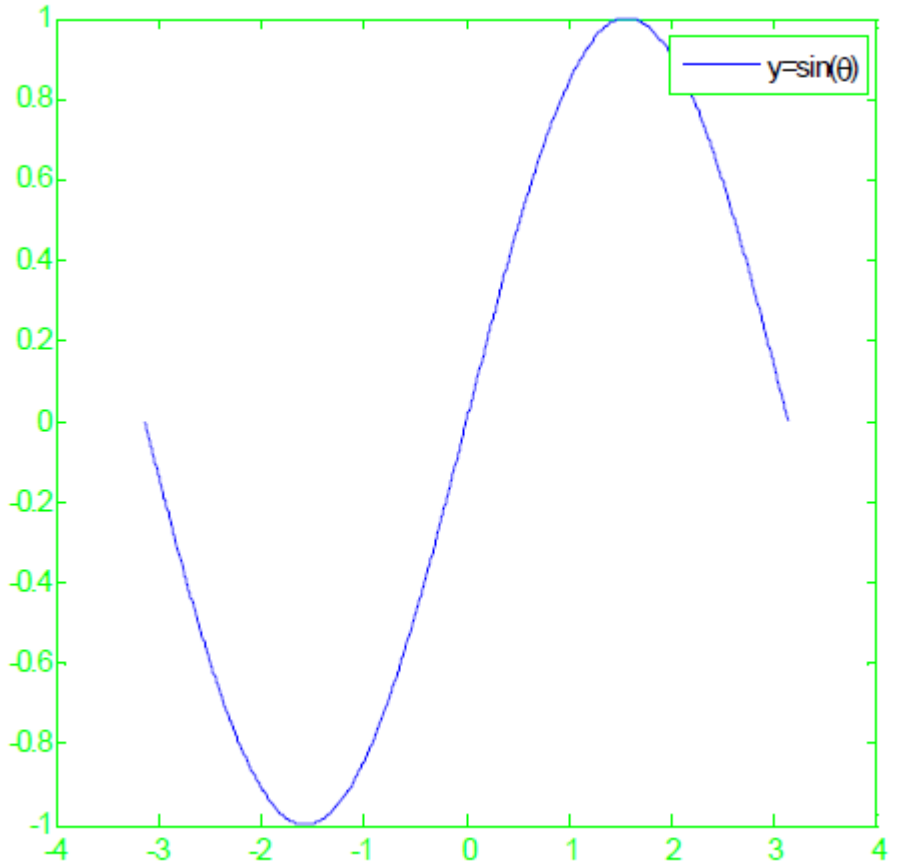
Addition of Legends

```
>> x = 1:1:10;

>> y1 = x.^3 – x.^2;

>> y2 = -x.^3 + x.^2;

>> plot(x,y1, x,y2);

>> legend (’Line 1’, ‘Line 2’);
```



Special Characters in Legends

```
>> theta = -pi:0.01:pi;

>> y = sin(theta);

>> plot(theta, y);

>> legend ('y = sin(\theta)');
```



- Use help to search for how to input other special characters

- Search for 'text properties'

Other Utilities for 2-D Plots

- Subplots
  - Multiple plots in the same figure

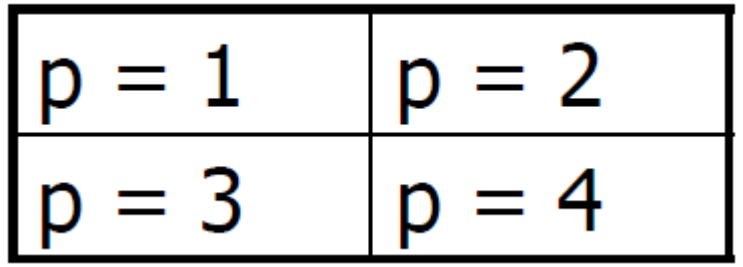
- MATLAB supports several 2-D plotting utilities:
  - Polar plots
  - Logarithmic plots
  - Bar graphs
  - Pie charts

Subplots

- Subplot command allows you to put multiple graphs on one figure window

- subplot(m,n,p) divides the figure window into a grid of m rows and n columns

- Variable p identifies the part of the window where the plot is placed

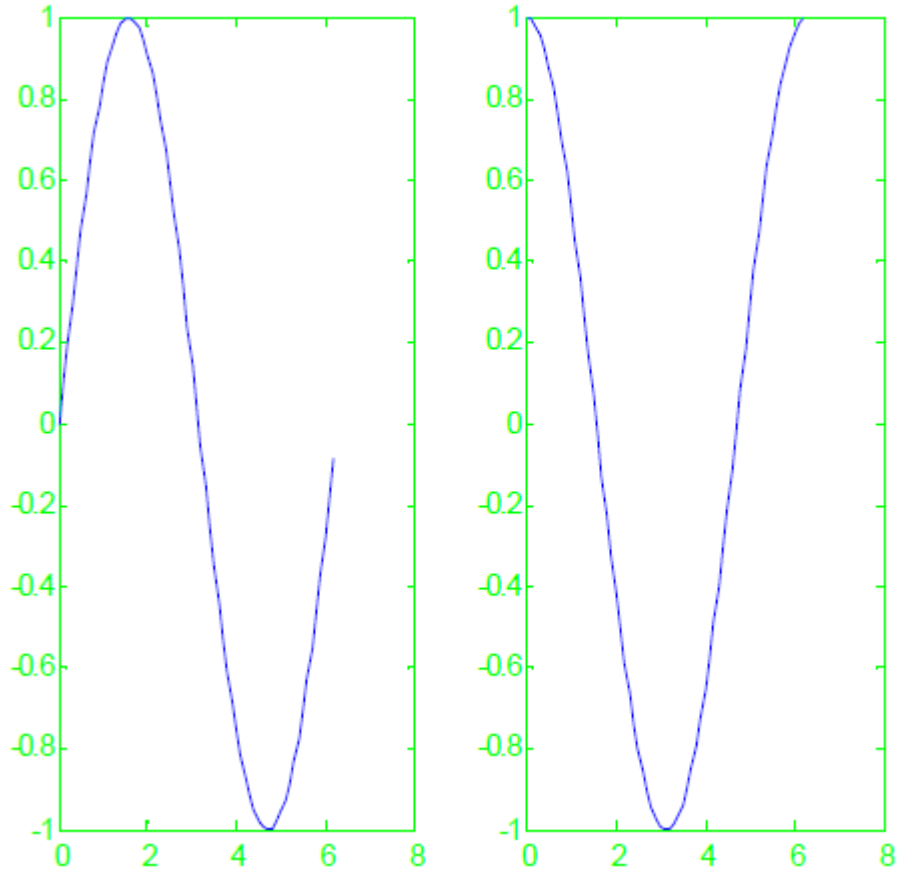


Examples of Subplots

- To graph sin(x) and cos(x) in the same figure side-by-side do the following

```
>> x = 0:0.1:2*pi;

>> subplot(1,2,1);

>> plot(x, sin(x));

>> subplot(1,2,2);

>> plot(x, cos(x));
```

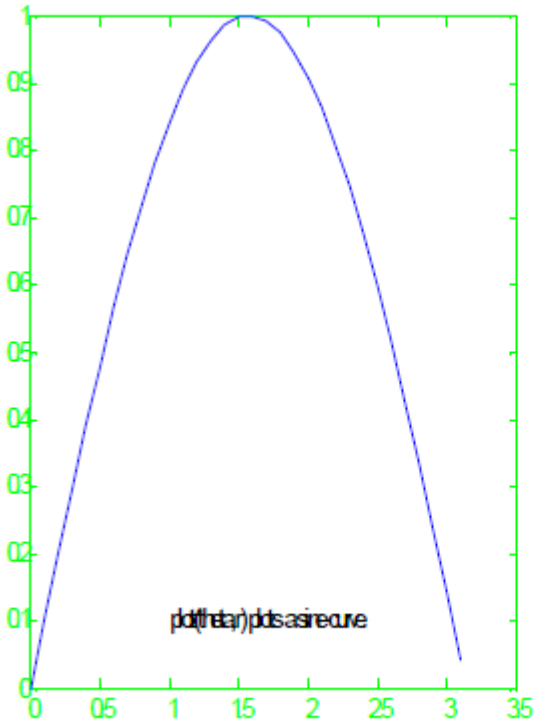
When a figure with a subplot is open, you must close it before opening a new figure in order for the new figure to display properly.

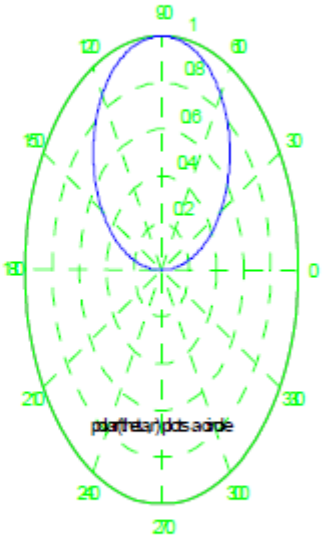


Polar Plots

- MATLAB supports tools for plotting data in polar coordinates

```
>> theta = 0:0.01:pi;
>> r = sin(theta);
>> subplot(1,2,1);
>> plot(theta, r);
>> subplot(1,2,2);
>> polar(theta, r);
```



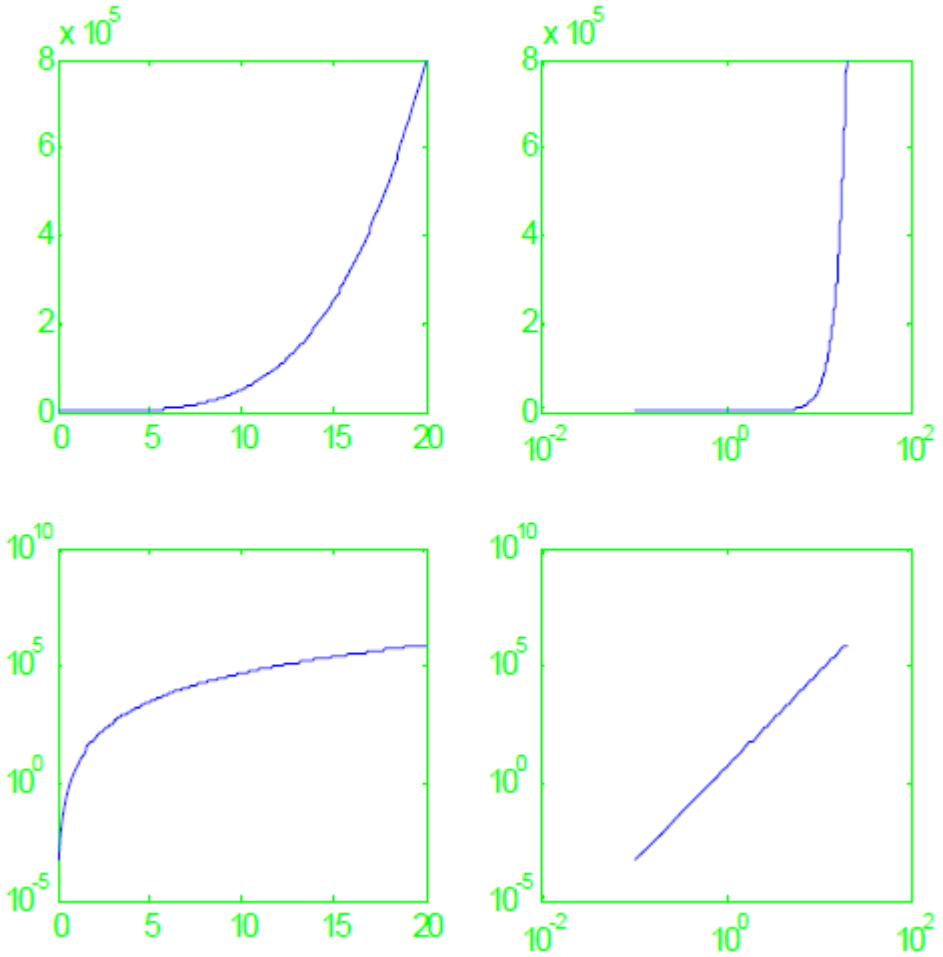


Logarithmic Plots

- MATLAB has tools for three kinds of logarithmic plots:
  - semilogx
  - semilogy
  - loglog
- These plotting utilities automatically replace linear scales with logarithmic scales.
- Logarithmic scales are useful when a variable ranges over many orders of magnitude.

Logarithmic Plots & Subplots

```
>> x = 0:0.1:20;
>> y = 5*x.^4;
>> subplot(2,2,1);
>> plot(x, y);
>> subplot(2,2,2);
>> semilogx(x, y);
>> subplot(2,2,3);
>> semilogy(x, y);
>> subplot(2,2,4);
>> loglog(x, y);
```



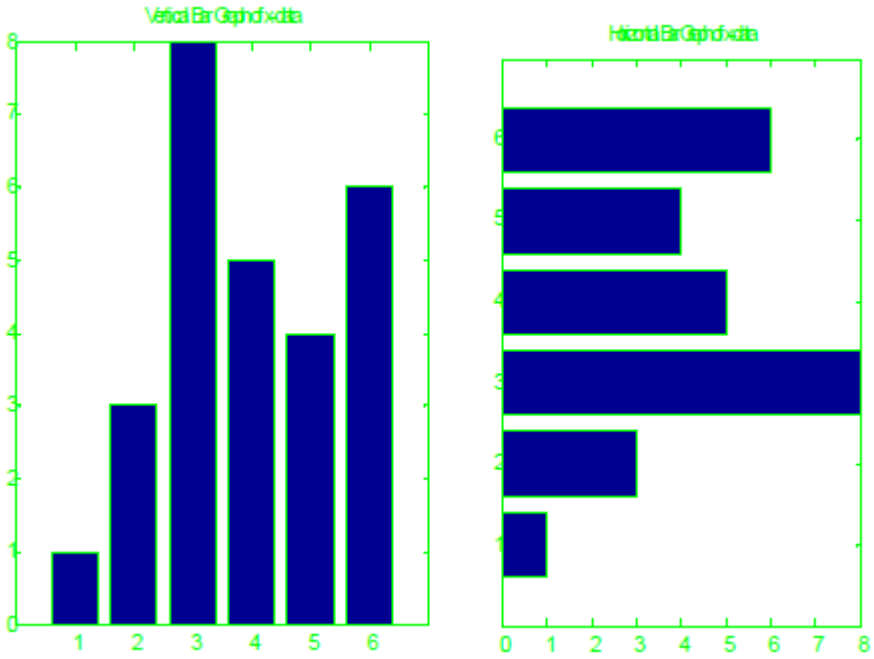
bar( ) Example

- Bar graphs are useful for reporting data.

- `x = [1,3,8,5,4,6];`

- bar(x); generates
a vertical bar graph.

- barh(x); generates
a horizontal bar graph.



pie( ) Example

• Pie charts are another useful way of reporting data.

```
>> pie(x);
```

$8/(1+3+8+5+4+6) \approx 30\%$

(cyan section)

$5/(1+3+8+5+4+6) \approx 19\%$

(yellow section)

etc.

3D Plots

plot3(x, y, z) – line plots in 3D

surf(x, y, z) – surface plot of z as a function of x & y

contour(x, y, z) – contour plot of z as a function of x & y

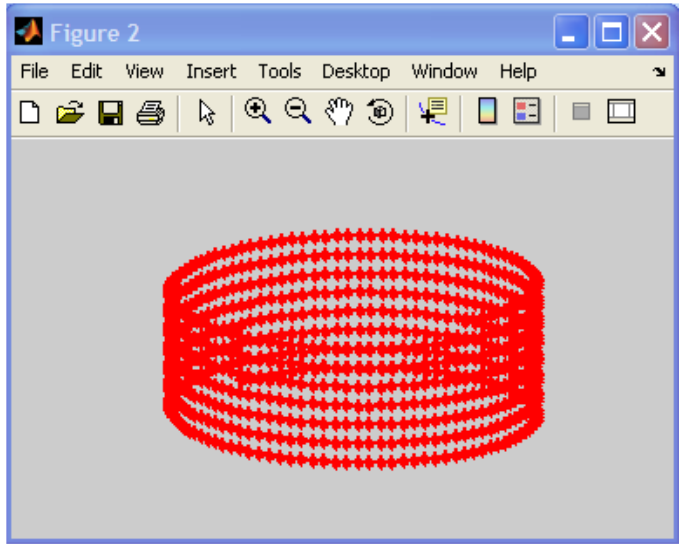
meshgrid( ) – creates a grid of points for use by surf and contour

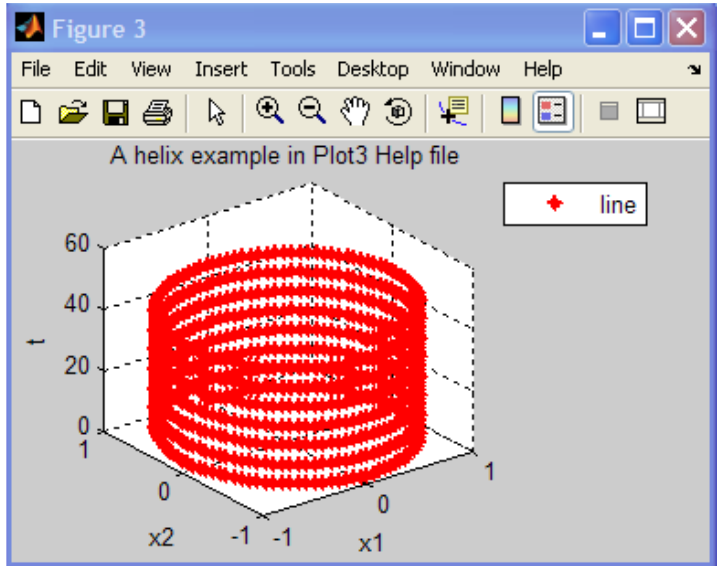
– x & y points do not have to be evenly spaced, but they do have to be structured as n x m arrays

All of these have multiple variations – see the on-line help for assistance

plot3( ) Example

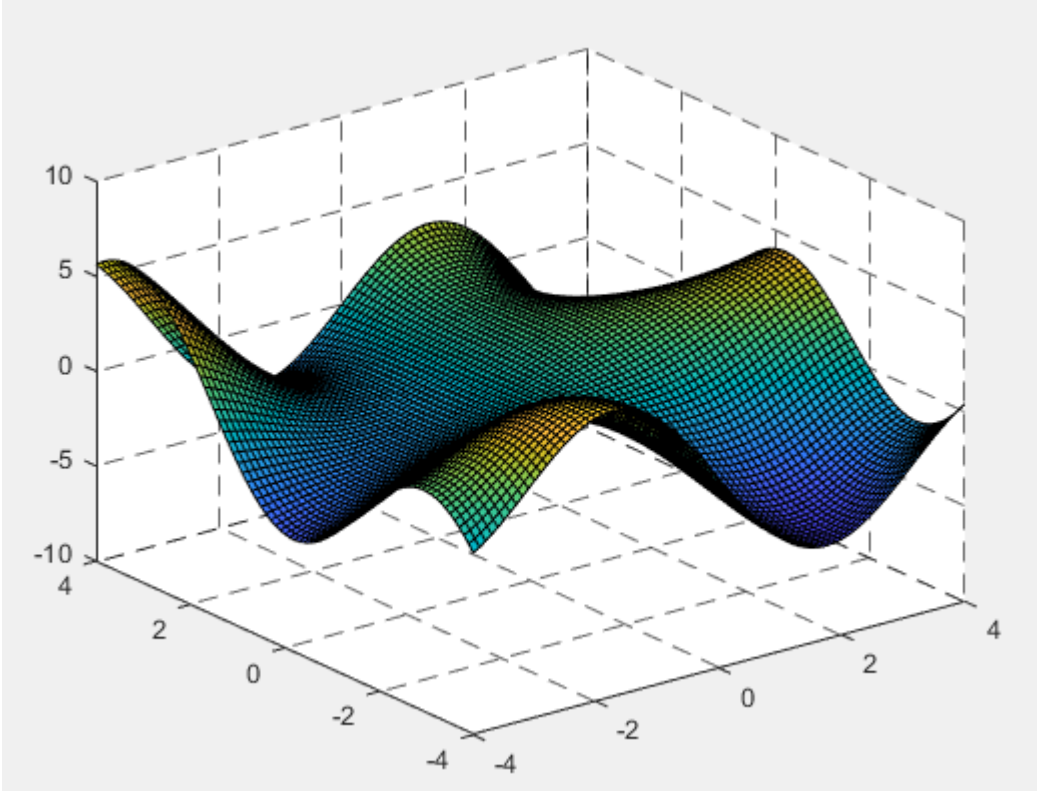
```
% Example of 3D Plotting
clc;clear
close
t=0:2*pi/100:15*pi;
x1=sin(t);
x2=cos(t);
figure(2)
plot3(x1,x2,t,'r*','Linewidth', 2);
axis off
figure(3)
plot3(x1,x2,t,'r*','Linewidth',2);
axis on; grid on;
xlabel('x1'); ylabel('x2'), zlabel('t');
title('A helix example in plots help file')
legendd('line')
```





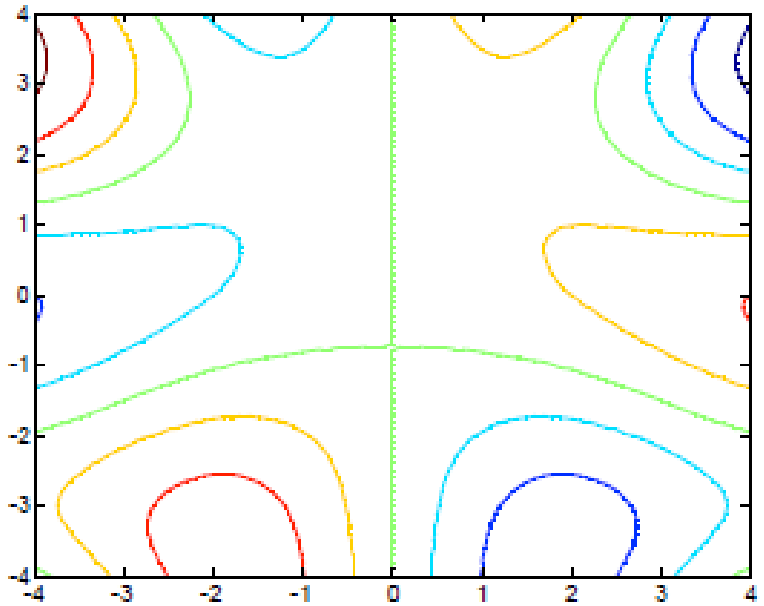
surf( ) Example

```
% define x & y range and spacing

x = -4:.1:4;y = -4:.1:4;

% generate x y grid data

[xgrid ygrid] = meshgrid(x,y);

% generate z data

zgrid = ygrid.*sin(xgrid) + ...
xgrid.*cos(ygrid);


surf(xgrid,ygrid,zgrid)
```



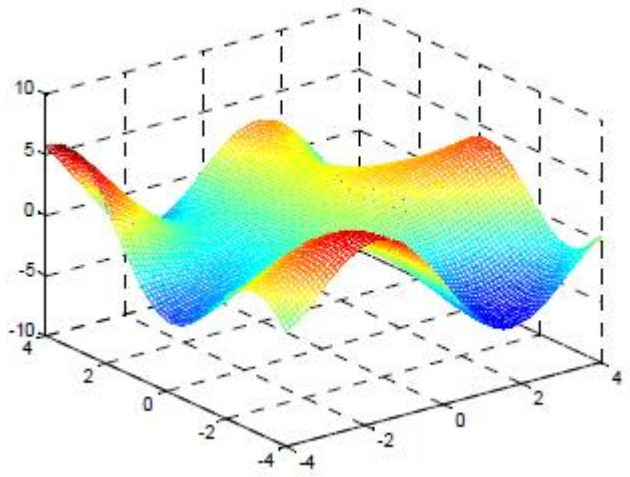
contour( ) Example

```
% define x & y range and spacing

x = -4:.1:4;y = -4:.1:4;


% generate x y grid data

[xgrid ygrid] = meshgrid(x,y);

% generate z data

 zgrid = ygrid.*sin(xgrid) + ...
 xgrid.*cos(ygrid);



contour(xgrid,ygrid,zgrid)
```



mesh( ) Example

```
% define x & y range and spacing

x = -4:.1:4;y = -4:.1:4;

% generate x y grid data


[xgrid ygrid] = meshgrid(x,y);

% generate z data

zgrid = ygrid.*sin(xgrid) + ...
xgrid.*cos(ygrid);


 mesh(xgrid,ygrid,zgrid)
```



Input/Output Operations

The **`input( )`** Function

- Key options

  – Assign values to variables in a command line using the equal (=) sign

  – Use the input function for interactive assignment of values to some of the variables within your computer program

  – Other uses
  - Great when using MATLAB as a super calculator
  - Useful when debugging scripts

input( ) Continued

- General format
  - >> variable = input('Please enter a value: ')

- The display in the Command window is:
  - Please enter a value:

- Type a value after the displayed prompt
  - The variable now has been assigned the value that was entered
  - If an array is entered, i.e. [1 2 3] or [1 2; 3 4] the variable will be an array

input( ) Example

- Area of a circle:

– Type this into an m-file:

```
radius = input('Please enter the radius: ')
area = pi * radius^2
```

– Execute and see this display in Command Window:

```
Please enter the radius: 5.2 % the 5.2 is entered by the user


radius =
5.2000

area =
84.9487
```

More Hands on – Free Fall

- Type this into an m-file:

```
height = input('Please enter the initial height in meters: ');
time = input('Please enter the fall time in seconds: ');
velocity = input('Please enter the initial velocity in m/s: ');
a = -9.81; % Positive is up, acceleration due to gravity is down
heightFin = height + velocity*time + 0.5*a*time^2
```

- Execute: See this display in Command Window:

```
Please enter the initial height in meters: 25
Please enter the fall time in seconds: 3
Please enter the initial velocity in m/s: 5
heightFin =-82.625
```

Input Strings of Text

- General format:

String = input('Please enter your name: ', 's')

- The display in the Command window:

Please Enter your name: Dr. Wei

This is used when the input is a string (e.g., names, months, etc.). The variable, in this case, is a 'char' class (or type)

– The 's' at the end tells input to accept a string as the variable

Output Options

- You can display the value of a variable in different ways.

– Typing the variable name without a final ";"

```
x = 500;
```

```
   x
x =
 500
```

– Using the disp function % a bit nicer

– Using the disp function % a bit nicer

The **disp( )** Function

- The disp() function displays either a string or a variable (but not both) to the command window
  - disp('This is a string')
  - disp(x)
  - Creates a line feed after displaying
  - Very useful for showing program flow
- To display a string and a number(s) in the same line, the numbers must be converted to strings using the num2str( ) function.
  - disp(['This is a combined string & variable: ', num2str(x)])
  - Note the use of num2str and [] to create a single string

disp( ) Example

>> clear; clc <Enter>

>> x = [0:0.2:1]; <Enter> % fills the vector

>> y = x.^2; <Enter> % y is a vector of x^2 values

>> disp(‘ x y '), disp([x’ y’]) <Enter>

% the 1st disp is a text header

% the 2nd disp is a transpose of the x and y vectors into column matrix

>> z = [x’ y’]; disp(z)

% does the same as disp([x’ y’])

The output is:

x	y
0	0
0.2000	0.0400
0.4000	0.1600
0.6000	0.3600
0.8000	0.6400
1.0000	1.0000

The **fprintf( )** Function

- The **fprintf( )** command is one way to display the value of a variable with a label

  – It can print to the command window as we are doing here and also to files once they have been opened

- General format:

  - MATLAB code:

    fprintf('format-string', variable)

Placeholders in fprintf( )

- To print a variable in your display on the command window, its place must be indicted by **%** in the format-string followed by the format of presentation (**d**, **f**, **e**, **g, s**).

- **%d**: integer notation
- **%f**: fixed point (decimal) notation
  - Most commonly used placeholder
- **%e**: exponential notation
- **%g**: whichever is shorter, %f or %e
- **%s**: string notation

fprintf( ) Example

- Type into an m-file:

```
smiles = 7

fprintf('Sarah smiles %d times a day', smiles)
```

Place holder #1 indicates where to insert variable #1 and how to format it

Variable #1

- Executing it displays in the command window:

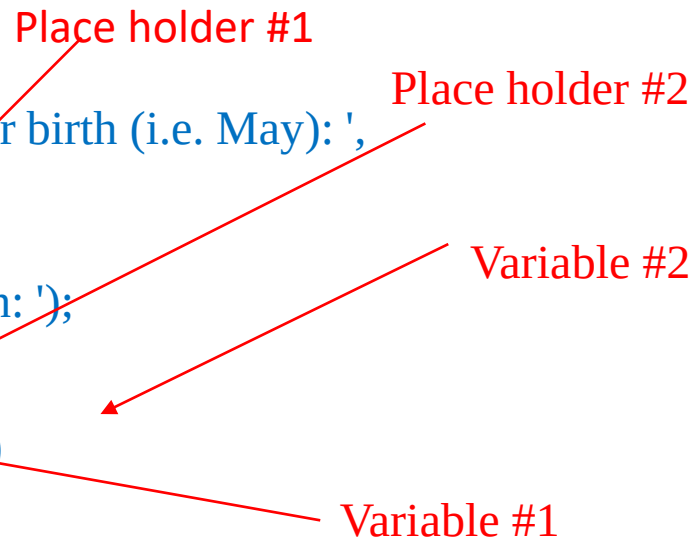
```
smiles =

     7

Sarah smiles 7 times a day
```

- Change the d to an f, e, or g and see what happens

Another fprintf( ) Example

- Type into an m-file:

```
month = input('Please enter the month of your birth (i.e. May): ',
's');

day = input('Please enter the day of your birth: ');

fprintf('Your birthday is %s %d!!', month, day)
```



- Executing it displays in Command Window:

Please enter the month of your birth (i.e. May): January

Please enter the day of your birth: 10

fprintf() line feeds

• When **fprintf** is used consecutively, MATLAB prints the results on the same line in the command window.

```
smiles = 7
fprintf('Sarah smiles %d times a day', smiles)

fprintf('Sarah smiles %d times a day', smiles)
```

• Executing this yields

```
Sarah smiles 7 times a daySarah smiles 7 times a day
```

• There is not an automatic line feed (new line) when using fprintf

The \n command

• This command is a linefeed. It commands MATLAB to start on a new line so that sequential outputs are on separate lines.

```
smiles = 7

fprintf('Sarah smiles %d times a day\n', smiles)

fprintf('Sarah smiles %d times a day\n', smiles)
```

• Executing this yields

```
Sarah smiles 7 times a day
Sarah smiles 7 times a day
```

• Be sure to use "\n" and NOT "/n"

Width & precision fields

- The width field specifies the minimum number of characters (including the ".") to be printed.

- The precision field specifies the number of those characters that will show up after the decimal point

- For example

– %8.2f specifies that there can be no less than 8 characters (width) in the displayed output, and that two of them (precision) are to follow the decimal point.

– %.3f specifies that three characters follow the decimal point but the total number of characters is not specified (it will adjust to fit as needed).

Even more fprintf() hands on

- Enter the following in an m-file and execute

```
>> weight = 57638.75453487621;
>> fprintf('The weight is %8.2f pounds \n', weight);
>> fprintf('The weight is %4.2f pounds \n', weight);
>> fprintf('The weight is %18.2f pounds \n', weight);
```

- The result will be:

```
The weight is 57638.75 pounds
The weight is 57638.75 pounds
The weight is                  57638.75 pounds
```

– Notice the blank spaces in the 3rd output.

– Use **%%** to have a % sign show up in output

– Use ‘’ to have a ‘ sign show up in output

Exercises

- Write a program to model a spring-mass system.

  – The spring constant k (N/m), the mass m (kg), amplitude x_o (m), and time, t, elapsed (s) are the input.

  – Display output [i.e., display the displacement x in meters of the system at the given conditions].

- The formula to use is:

  - $x=x_o\cos(\omega t)$, where $\omega=\text{sqrt}(k/m)$

I/O Summary

- input( )
  - variable = input('Please enter a value: ')
  - string = input('Please enter a value: ', 's')
- disp( )

disp(x); disp([‘The value of x is ‘ num2str(x)])

Again, notice the brackets to make a character/string array

- fprintf( )
  - fprintf('format-string', variable, variable2)
  - %f = fixed point (decimal) notation
  - %e = exponential notation
  - %g = whichever is shorter, %f or %e