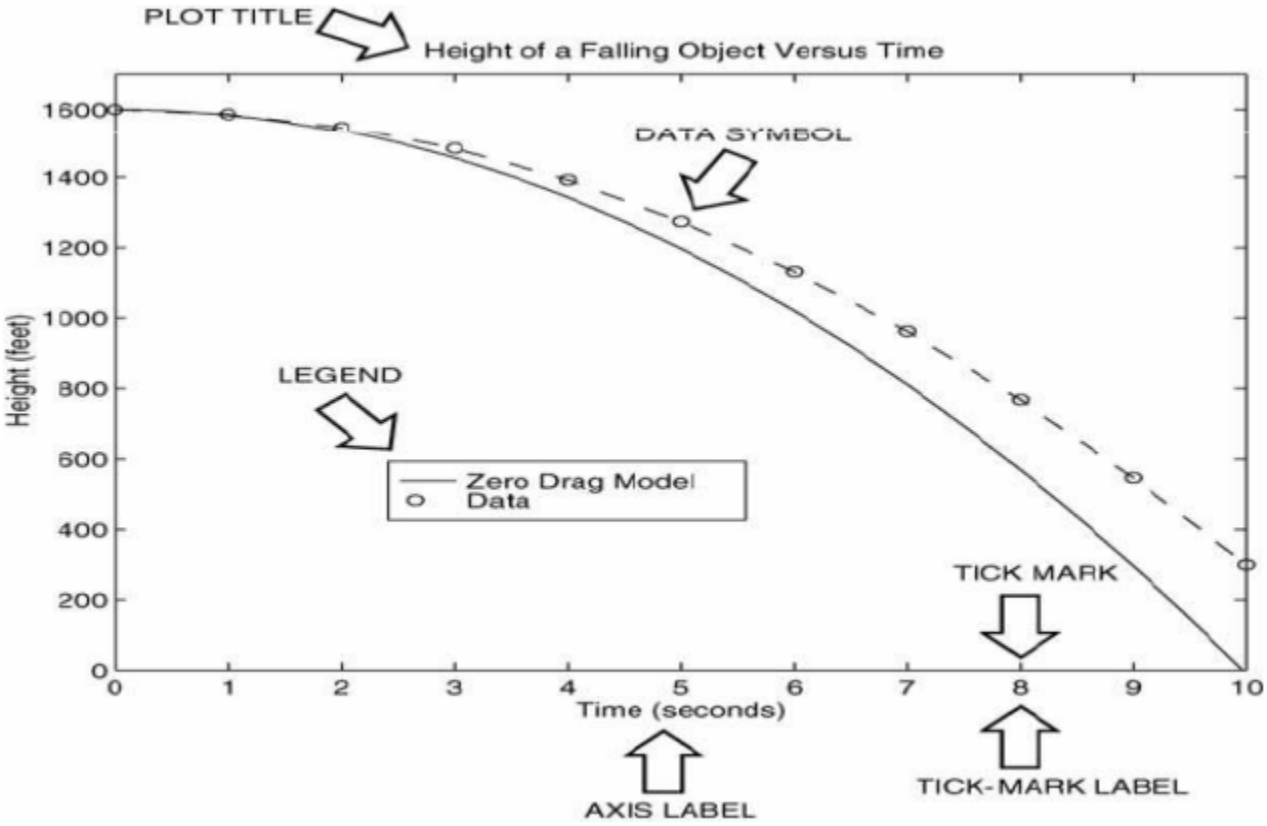
Lab 5: Plotting

xy Plotting:

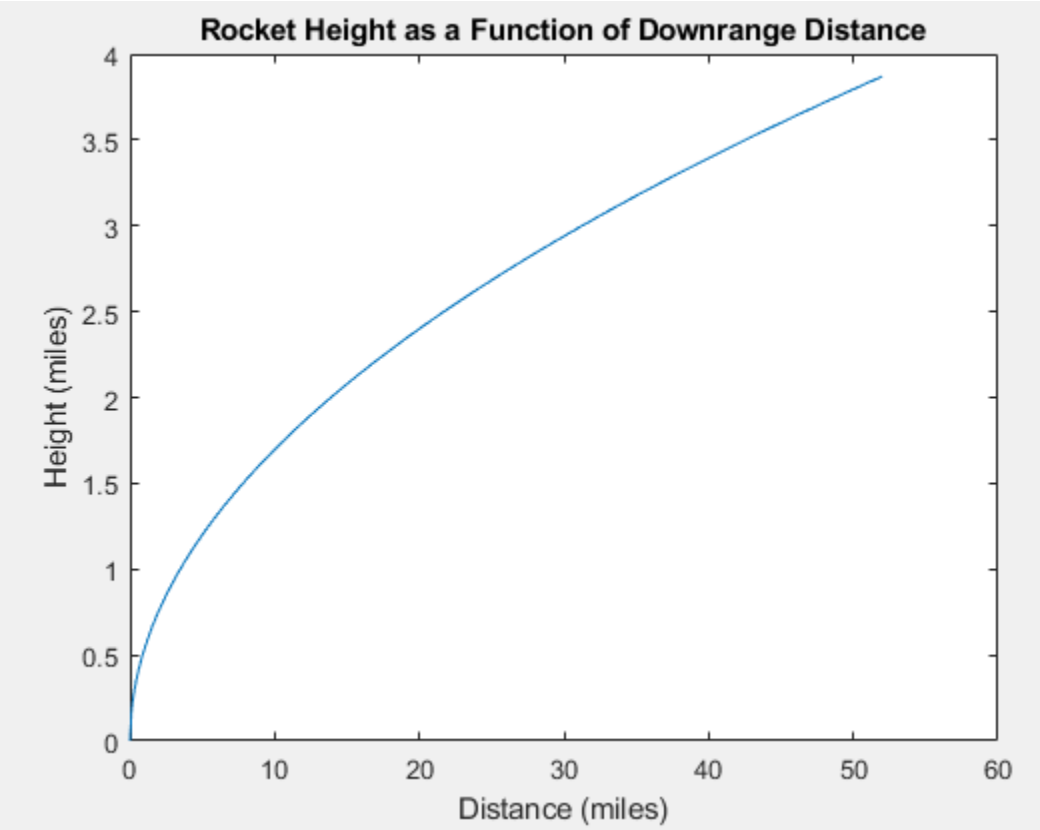
- The most common types of plots.
- Plots the function y = f(x). (Other symbols may be used)
- The command that is used for xy plots in Matlab is:
  `plot(x,y)`
  where both x and y are vectors of the same length.

Anatomy of a Plot



Example:

```
>> x = [0:0.1:52];
>> y = 0.4*sqrt(1.8*x);
>> plot(x,y)
>> xlabel('Distance (miles') % Adds label to the x-axis
>> xlabel('Distance (miles)') % Adds label to the x-axis
>> ylabel('Height (miles)')   % Adds label to the y-axis
>> title('Rocket Height as a Function of Downrange Distance')
>> % Adds title to the plot
```



Requirements for a Correct Plot:

1. Each axis must be labeled with the name of the quantity being plotted and its units!
2. Each axis should have regularly spaced tick marks at convenient intervals—not too sparse, but not too dense— with a spacing that is easy to interpret and interpolate. For convenient intervals—not too sparse, but not too dense— with a spacing that is easy to interpret and interpolate. For example, use 0.1, 0.2, and so on, rather than 0.13, 0.26, and so on.
3. If you are plotting more than one curve or data set, label each on its plot or use a legend to distinguish them.
4. If you are preparing multiple plots of a similar type or if the axes' labels cannot convey enough information, use a title.
5. If you are plotting measured data, plot each data point with a symbol such as a circle, square, or cross (use the same symbol for every point in the same data set). If there are many data points, plot them using the dot symbol.
6. Sometimes data symbols are connected by lines to help the viewer visualize the data, especially if there are few data points. However, connecting the data points, especially with a solid line, might be interpreted to imply knowledge of what occurs between the data points. Thus, you should be careful to prevent such misinterpretation.
7. If you are plotting points generated by evaluating a function (as opposed to measured data), do not use a symbol to plot the points. Instead, be sure to generate many points, and connect the points with solid lines.

The Figure Window:

- It has different menus to save, export, and edit generated figures.
- We can:
  1. Save the figure to be used later on (.fig extension)
  2. Print the plot
  3. We can insert text boxes, arrows, shapes, and legends.
  4. Zoom in and out and rotate plot
  5. Clear the figure
  6. Copy the plot to be inserted in other documents
  7. Export the figure in other data formats using File>Export

More Commands:

- grid:
  The grid command displays gridlines at the tick marks corresponding to the tick labels. Type `grid on` to add gridlines; type `grid off` to stop plotting gridlines. When used by itself, `grid` toggles this feature on or off, but you might want to use grid on and grid off to be sure.

- axis:
  Is used to override the MATLAB selections for the axis limits. The basic syntax is `axis([xmin xmax ymin ymax])`. This command sets the scaling for the x-and y-axes to the minimum and maximum values indicated. Note that, unlike an array, this command does not use commas to separate the values.

- figure:
  Use this command to open a new figure window to be used for new plots. Using the plot command more than once update the same figure window that was used by the first plot command, unless the hold command is used

Some variations of the plot command:

- If y is a vector and we use `plot(y)`, MATLAB plots the elements of y versus their indices.
- If y is a vector of complex numbers, the use of plot(y) plots the imaginary part of the elements versus the real part. In other words, plot(y) in case y contains complex values is equivalent to `plot(real(y),imag(x))`.

```
>> z = 0.1 + 0.9i;
>> n = [0:0.01:10];
>> plot(z.^n)
>> xlabel('Real')
>> ylabel('Imaginary')
```

The function plot fplot:

- It is a smart command that takes the function to be plotted and determine the required number of points to show most of the features of the function.
- Syntax:
  `[x,y] = fplot('string',[xmin xmax ymin ymax])`

```
>> fplot('cos(tan(x))-tan(sin(x))',[1 2])
```

Compare with:

```
>> x = [1:0.01:2];
>> y = cos(tan(x))-tan(sin(x));
>> plot(x,y)
```

Plotting Polynomials:

- The `polyval(coeff,x)` function in MATLAB can be used to evaluate a polynomial whose coefficients are given in coeff evaluated at values specified by x

```
>> x = [-6:0.01:6];
>> p = [3 2 -100 2 -7 90];
>> plot(x,polyval(p,x))
```

Subplots:

- You can use the subplot command to obtain several smaller "subplots" in the same figure.
- Syntax:
  `Subplot(m,n,p)`
  This command divides the Figure window into an array of rectangular panes with m rows and n columns.
- The variable p tells MATLAB to place the output of the plot command following the subplot command into the pth pane.
- For example, subplot(3,2,5) creates an array of six panes, three panes deep and two panes across, and directs the next plot to appear in the fifth pane (in the bottom-left corner).

```
>> x = [0:0.01:5];
>> y = exp(-1.2*x).*sin(10*x+5);
>> subplot(1,2,1)
>> plot(x,y)
>> axis([0 5 -1 1])
>> x = [-6:0.01:6];
>> y = abs(x.^3-100);
>> subplot(1,2,2)
>> plot(x,y)
>> axis([-6 6 0 350])
```

Data Markers and Line Types:

- We can change the data markers, line styles, and colors for our plots by using a string in a plot command
- For example, plot(x,y,'b-.o') plots y versus x using a blue dash-dotted line with circles as data markers.
- More data markers, line styles, and colors are listed below:

Data markers†		Line types		Colors	
Dot (.)	.	Solid line	—	Black	k
Asterisk (*)	*	Dashed line	– –	Blue	b
Cross (×)	×	Dash-dotted line	– .	Cyan	c
Circle (○)	○	Dotted line	....	Green	g
Plus sign (+)	+			Magenta	m
Square (□)	s			Red	r
Diamond (◇)	d			White	w
Five-pointed star (w)	p			Yellow	y

†Other data markers are available. Search for "markers" in MATLAB help.

Controlling Tick-Mark Spacing and Labels:

- The `set` command is a powerful command for controlling objects
- One of these objects is the axes in the plot.
- We can change the tick mark spacing and use labels instead of numbers as tick marks.
- Syntax:
  ```
  set(gca,'XTick',[xmin:dx:xmax],'YTick',[ymin:dy:ymax])

  set(gca,'XTick',[0:0.1:2],'YTick',[0:0,1:1])
  ```
- We can assign labels to tick marks instead of numbers.

```
>> x = [1:6];
>> y = [13,5,7,14,10,12];
>> set(gca,'Xticklabel',['Jan';'Feb';'Mar';Apr';'May';'Jun'])
>> set(gca,'XTick',[1:6]), axis([1 6 0 15])
>> xlabel('Month')
>> ylabel('Monthly Sales $1000')
>> title('Printer Sales for January to June, 1997')
```

Labeling Curves and Data

- When we plot more than one curve on the same plot, we have to distinguish them with:

  1. Legends:
     `legend('string1','string2',…)`

  2. Labels:
     `gtext('string')`
     (it requires user interaction to specify the location of the text by the mouse)

     `text(x,y,'string')`
     (x and y specify the location of the text)

```
>> x = [0:0.01:2];
>> y = sinh(x);
>> z = tanh(x);
>> plot(x,y,x,z,' -- ')
>> xlabel('x')
>> ylabel('Hyperbolic Sine and Tangent')
>> legend('sinh(x)','tanh(x)')
```

The hold Command

- We can plot y versus x and v versus u on the same figure using the plot command by `plot(x,y,u,v)`
- If we use the plot command once more, the figure gets updated with the new plot and the old plot is erased.
- To preserve earlier plots, use the `hold on` command.
- This command will allow the new plot to be superimposed over the old plot.
- Use `hold off` to disable the hold command.

Annotating Plots

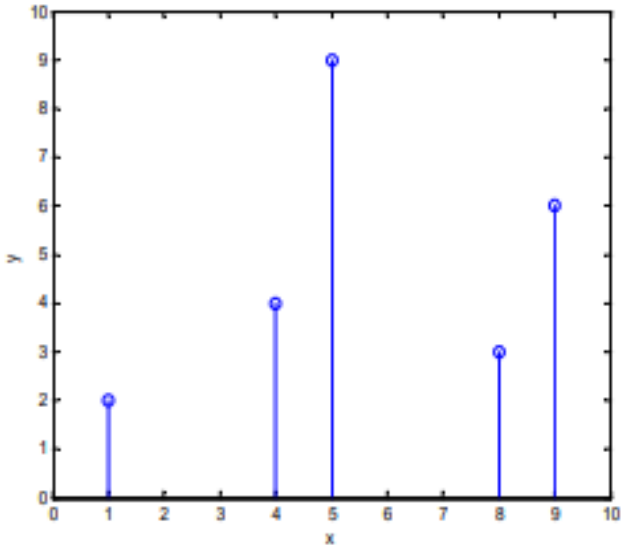
- We already saw the `title('string')` that is used to insert a title for the plot.
- We can insert mathematical symbols in the title, axes labels, and text labels by using the backslash character \ followed by the symbol name.
- Superscripts and subscripts are denoted by ^ and _ , respectively. To set multiple characters as superscripts or subscripts, enclose them by {}.
- Example: the command >> title('Ae^{-t/\tau} sin(\omega t)') produces in the figure title.

$$Ae^{-t/\tau}\ sin(\omega t)$$

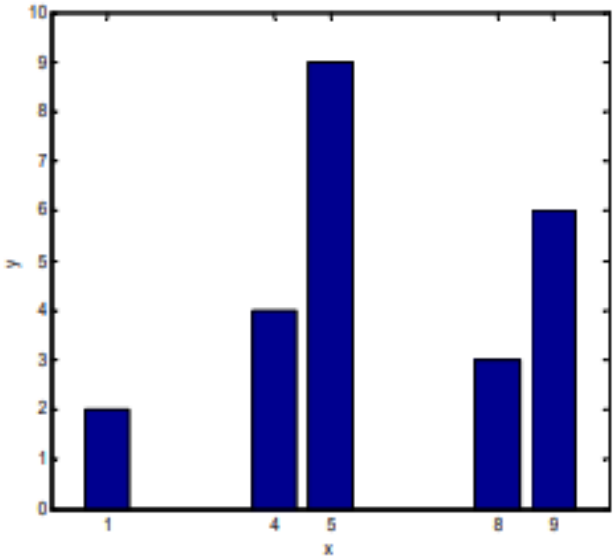
Hints for Improving Plots

1. Start scales from zero whenever possible. This technique prevents a false impression of the magnitudes of any variations shown on the plot.
2. Use sensible tick-mark spacing. If the quantities are months, choose a spacing of 12 because 1/10 of a year is not a convenient division. Space tick marks as close as is useful, but no closer. If the data is given monthly over a range of 24 months, 48 tick marks might be too dense, and also unnecessary.
3. Minimize the number of zeros in the data being plotted. For example, use a scale in millions of dollars when appropriate, instead of a scale in dollars with six zeros after every number
4. Determine the minimum and maximum data values for each axis before plotting the data. Then set the axis limits to cover the entire data range plus an additional amount to allow convenient tick-mark spacing to be selected.
5. Use a different line type for each curve when several are plotted on a single plot and they cross each other; for example, use a solid line, a dashed line, and combinations of lines and symbols. Beware of using colors to distinguish plots if you are going to make colors to distinguish plots if you are going to make black e black-and-white printouts and photocopies.
6. Do not put many curves on one plot, particularly if they will be close to each other or cross one another at several points.
7. Use the same scale limits and tick spacing on each plot if you need to compare information on more than one plot.

Special Types of Plots



Stem Plots: use *stem(x,y)*



Bar Plots: use *bar(x,y)*

- Use the help to check the Stairs Command

3D Plotting

Plotting of Curves:

- Use the command `plot3(x,y,z)` to plot three-dimensional curves.
- We can use the grid command and the attributes for selecting the line style, color and data markers.

```
>> t = [0:pi/50:10*pi];
>> x = exp(-0.05*t).*sin(t);
>> y = exp(-0.05*t).*cos(t);
>> plot3(x,y,t)
>> xlabel('x')
>> ylabel('y')
>> zlabel('t')
>> grid on
```

Surface Mesh Plots

- The function z = f(x,y) represents a surface when plotted on xyz axes, and the mesh function provides the means to generate a surface plot.
- Before you can use this function, you must generate a grid of points on the xy-plane, and then evaluate the function f(x,y) at these points. The `meshgrid` function generates the grid.
- Syntax:
  `[X,Y] = meshgrid(x,y);`
  Where x = [xmin:xspacing:xmax] and y = [ymin:yspacing:ymax]
- This computes the Cartesian product between x and y.
- The function `[X,Y] = meshgrid(x)` is the equivalent to `[X,Y] = meshgrid(x,x)` and can be used if x and y have the same minimum values, the same maximum values, and the same spacing. Using this form, you can type `[X,Y] = meshgrid(min:spacing:max)`, where min and max specify the minimum and maximum values of both x and y and spacing is the desired spacing of the x and y values.
- Once the grid is obtained, we can plot the three dimensional surface using the `mesh(X,Y,Z)` function. As always, the grid, label, and text functions can be used with the mesh function.

```
>> [X,Y] = meshgrid([-2:0.1:2]);
>> Z = X.*exp(-((X-Y.^2).^2+Y.^2));
>> mesh(X,Y,Z)
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```

Contour Plots

- Contour plot contains lines that indicate constant elevation (values).
- Contour plots can be generated in Matlab by the `contour(x,y,z)` command. We need first to use `meshgrid` as well.

```
>> [X,Y] = meshgrid([-2:0.1:2]);
>> Z = X.*exp(-((X-Y.^2).^2+Y.^2));
>> contour(X,Y,Z)
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```

Other 3D Plots

Function	Description
`contour(x,y,z)`	Creates a contour plot.
`mesh(x,y,z)`	Creates a 3D mesh surface plot.
`meshc(x,y,z)`	Same as `mesh` but draws contours under the surface.
`meshz(x,y,z)`	Same as `mesh` but draws vertical reference lines under the surface.
`surf(x,y,z)`	Creates a shaded 3D mesh surface plot.
`surfc(x,y,z)`	Same as `surf` but draws contours under the surface.
`[X,Y] = meshgrid(x,y)`	Creates the matrices `X` and `Y` from the vectors `x` and `y` to define a rectangular grid.
`[X,Y] = meshgrid(x)`	Same as `[X,Y]=meshgrid(x,x)`.
`waterfall(x,y,z)`	Same as `mesh` but draws mesh lines in one direction only.