Mid-term exam

13:25 – 14:25

May 21

- **Short answer questions: 9 problems, 58 points;**
- **Find the errors: 1 problems, 10 points;**
- **Written response questions: 2 problems, 22 points.**

Introduction to Scientific Computing

File Input / Output

2D & 3D Plotting

Input Commands

- `input` **command**

```
in1 = input('Enter data: ');
```

Stores input value as a numerical value or array of values of type double

```
in2 = input('Enter data: ', 's');
```

Stores input value as a string (an array of characters)

L14.m

Output Commands

- `disp` **command**

```
disp(variable)
disp('text to print')
```

- `fprintf` **command**

```
fprintf(format,data)
```

- Formatted output placed on screen or saved to a file
- Lots of options; see Help
  - More flexible way to display/handle output data
  - Many parameters control the way data is presented

  (Preview of something we will also examine later with C)

Files in MATLAB

- **Files are stored as sequences of bytes on the HD**
  - The file format is used to interpret the contents of a file, sometimes indicated by the file suffix

    `.txt, .m, .Mat, .doc, .ppt`

- **Files are stored in the context of a 'File System' which allows us to organize information in a directory structure**

- **Scripts and functions are stored in `.m` files which can be conveniently created and edited using the MATLAB `editor`. These are actually simple text files**

- **The variables in a MATLAB session can be saved and retrieved using the '`save`' and '`load`' commands respectively.**

`save` and `load` Commands

- `save`
  - Save all variables in the workspace to a file

    `save('file_name'); or, save file_name;`

  - Writes `file_name.mat` file in binary format

  - Save a few variables

    `save file_name var1 var2;`

  - Save in readable format

    `save -ascii file_name;`

    - Useful to read data from other applications
    - Platform independent
    - Easy way to transfer data

`save` and `load` Commands (cont.)

- `load`
  - Load the variables in a file

    ```
    load('file_name'); or, load file_name;
    ```

  - Reads `file_name.mat` and restores variables
  - Load a few variables

    ```
    load file_name var1 var2;
    ```

  - Load from a text (.txt) file

    ```
    load file_name.txt;
    var = load('file_name.txt');
    ```

  - Options

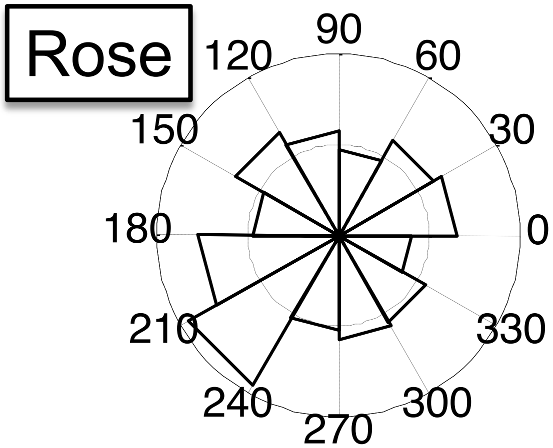
    ```
    mat, -ascii
    ```

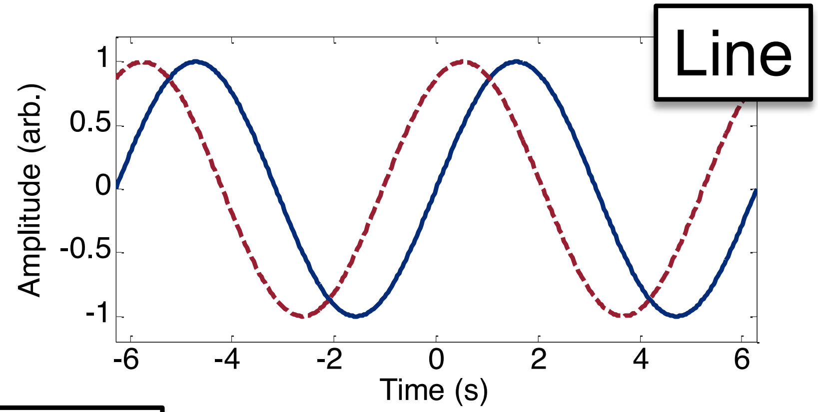
Cell Arrays

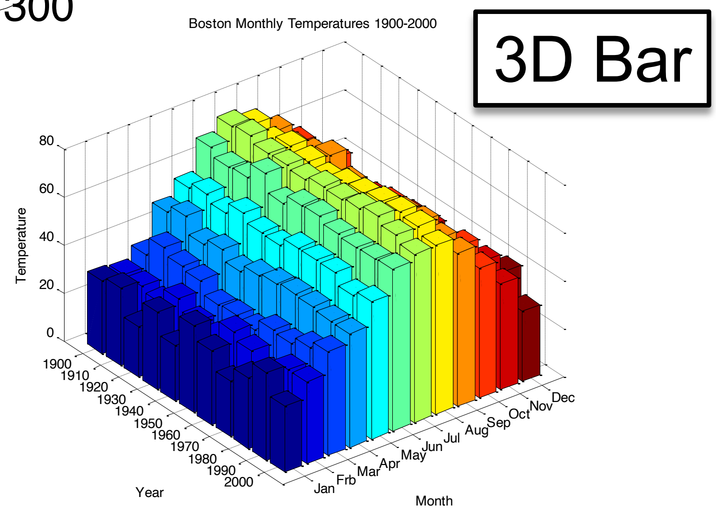
- A cell is the most general data object in MATLAB.

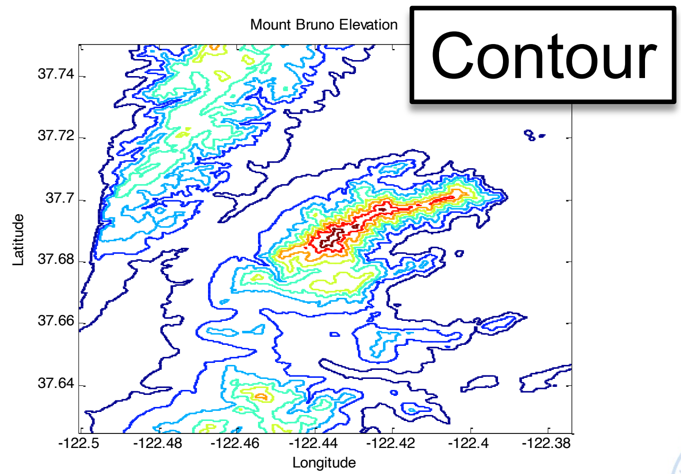
- An example. 14-11

Graphical representation of data



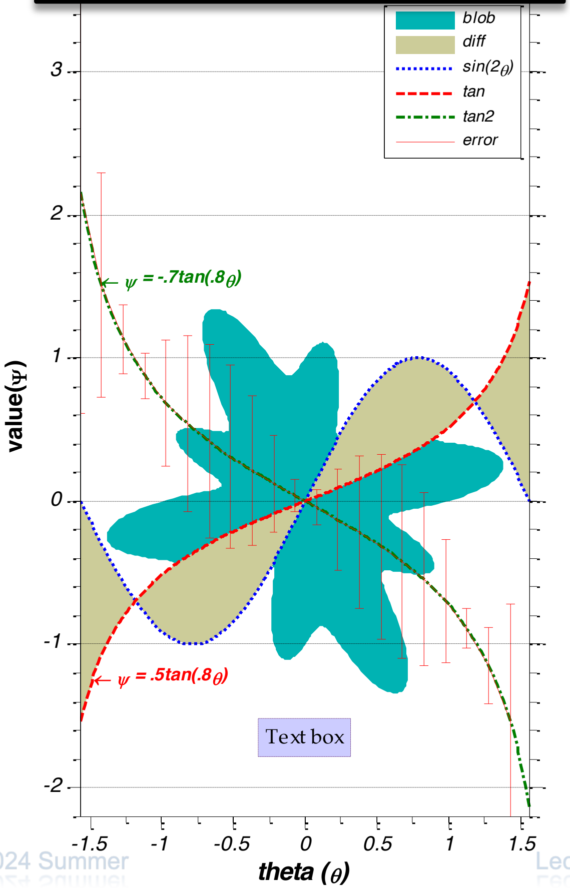




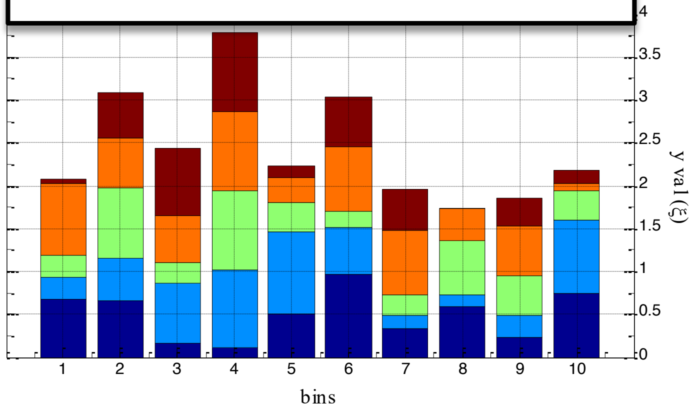


Graphical representation of data

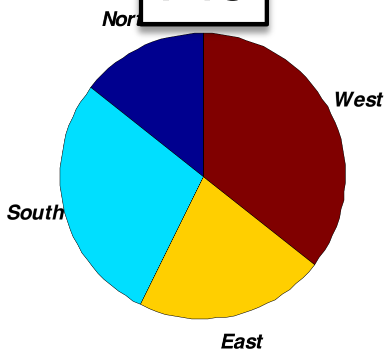
Multiple Overlays



Bar and Stacked Bar



Pie



What defines a pleasing plot?

- **Although "pleasing" is subjective, there are several objective measures of a good plot**
  - Labels with units
  - Minimum amount of information to convey the objective (don't double label objects unless necessary, extraneous callouts should be avoided)
  - Avoid overuse of eye candy
  - Scatter plots for measured data and line plots for fits or closed form equations
  - Sufficient resolution on domain to represent solution
  - Line widths, data point sizes, and text large enough to read

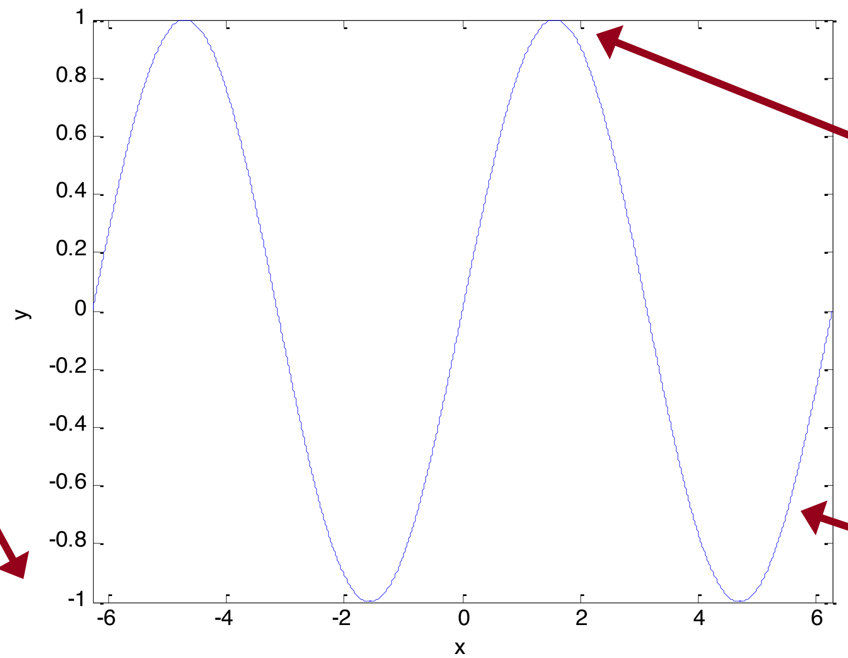
Examples of “pleasing” plots

Data generated from a “continuous” function

$y = \sin(x)$

Small labels

Can’t read text on x/y axes



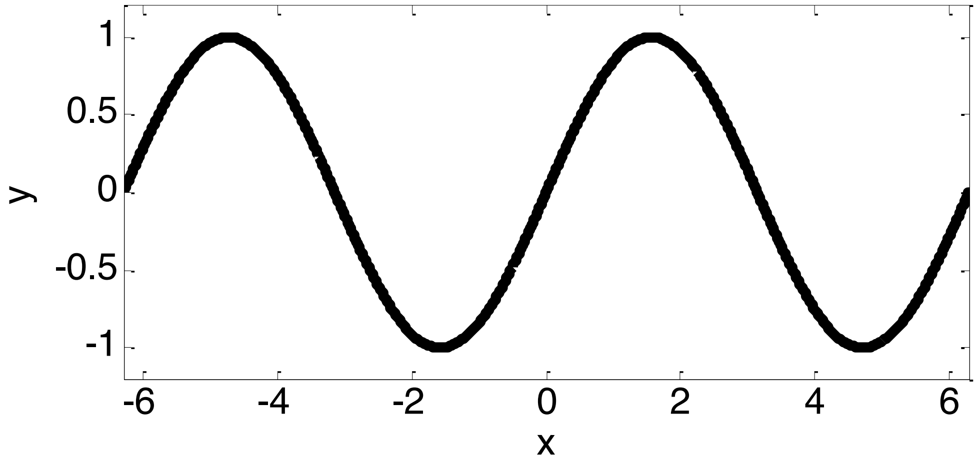
No “breathing” room at top or bottom of graph

Small linewidth and unnecessary use of color

Examples of “pleasing” plots

Data generated from a “continuous” function

y = sin(x)



A better version of the previous plot

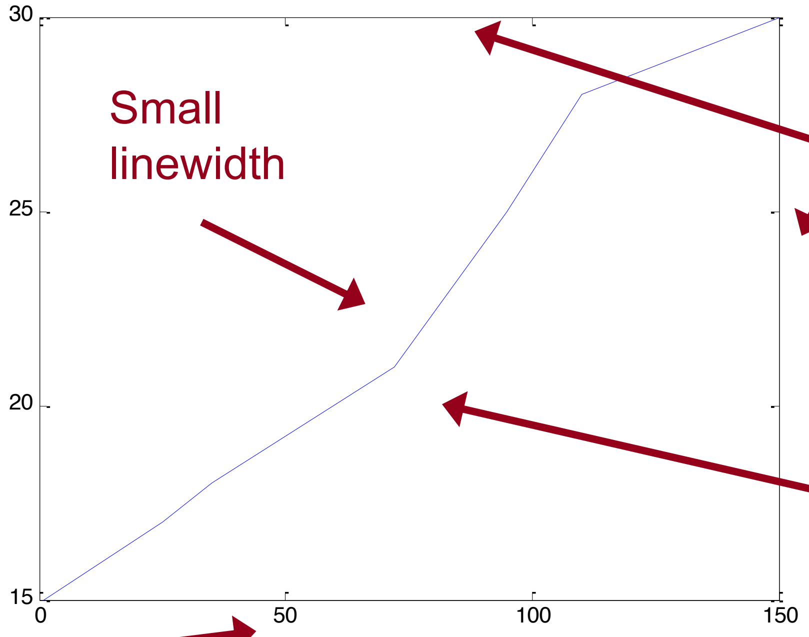
Examples of “pleasing” plots

Data from discrete measurements

No data labels

Small linewidth

Unnecessary borders on reflected x and y axes

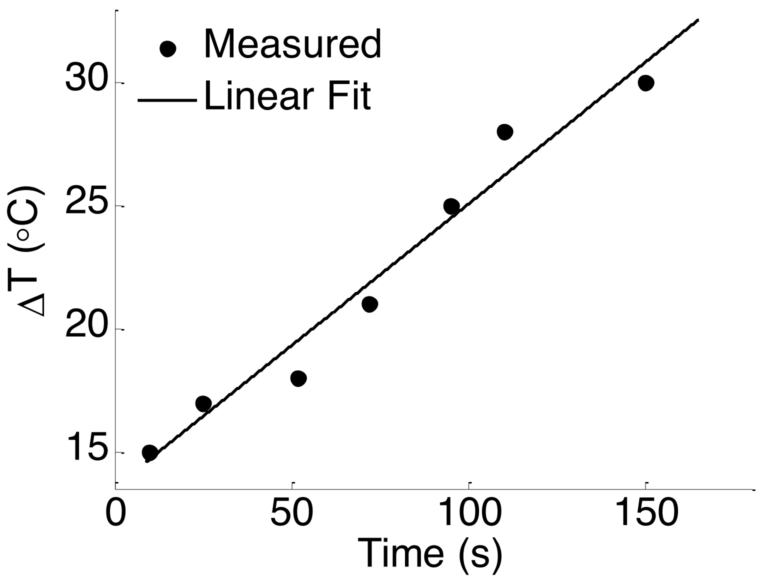


Generally, scatter plots are used for experimental data

Can’t read text on x/y axis values - font too small

Examples of “pleasing” plots

Data from discrete measurements



A better version of the previous plot

Examples of “pleasing” plots

Code used to generate the previous plot:

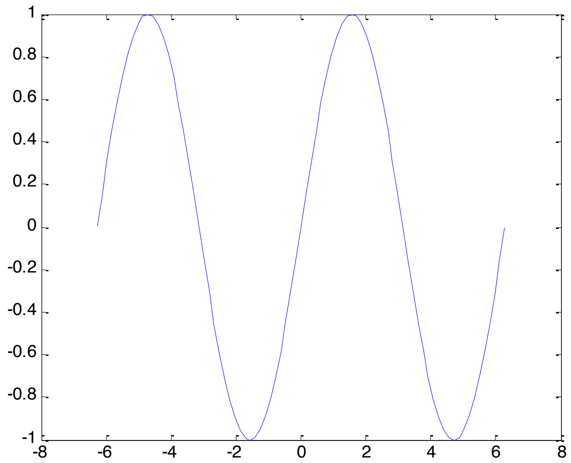
```
x = [10 25 52 72 95 110 150];
y = [15 17 18 21 25 28 30];
p1 = plot(x,y,'ko');
set(p1,'MarkerFaceColor','k')
set(p1,'MarkerSize',8)
axis([0 1.2*max(x) 0.9*min(y) 1.1*max(y)])
set(gca,'FontSize',20)
xlabel('Time (s)')
ylabel('\DeltaT (\circC)')
xfit = 0.9*min(x):.1:1.1*max(x);
fitComp = polyfit(x,y,1);
yfit = fitComp(1)*xfit+fitComp(2);
hold on
p2 = plot(xfit,yfit,'k-');
set(p2,'LineWidth',2)
hold off
lg = legend('Measured','Linear Fit');
set(lg,'Location','NorthWest')
legend boxoff
set(gcf,'Color','w')
set(gca, 'box', 'off')
```

Basic plot commands – `plot()` and `scatter()`

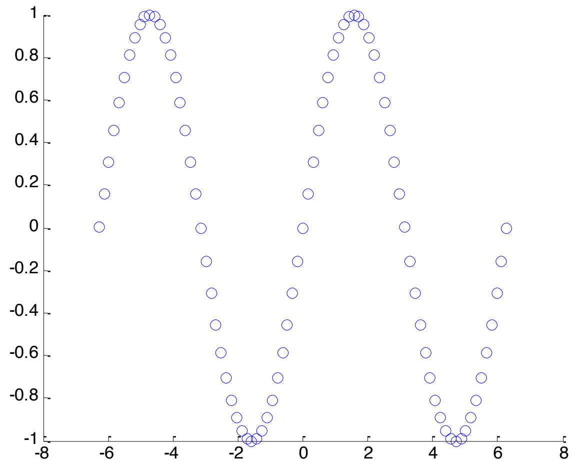
The two most commonly used 2D plot commands are `plot()` and `scatter()`

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
```

`plot(x,y)`



`scatter(x,y)`



Basic plot commands – `plot()` and `scatter()`

`plot()` and `scatter()` require vectors of equal length

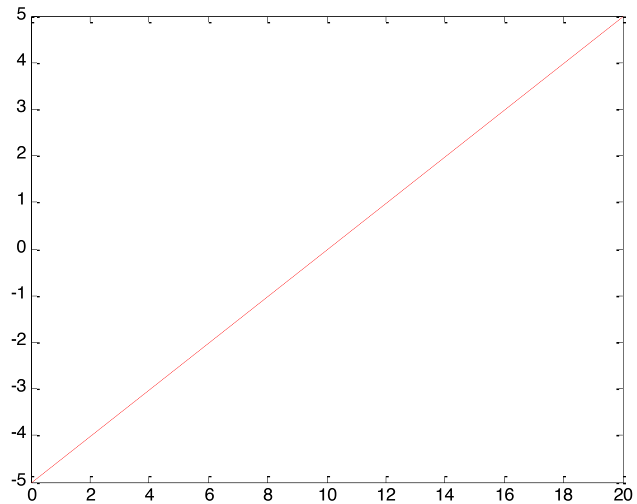
The default plot attributes are blue lines or circular blue data points, respectively

These attributes can be modified be adjusting the line specifications of the plot (`linespec`)

Basic plot commands – `plot()`

Straight lines may also be plotted by specifying the beginning and end points of the line

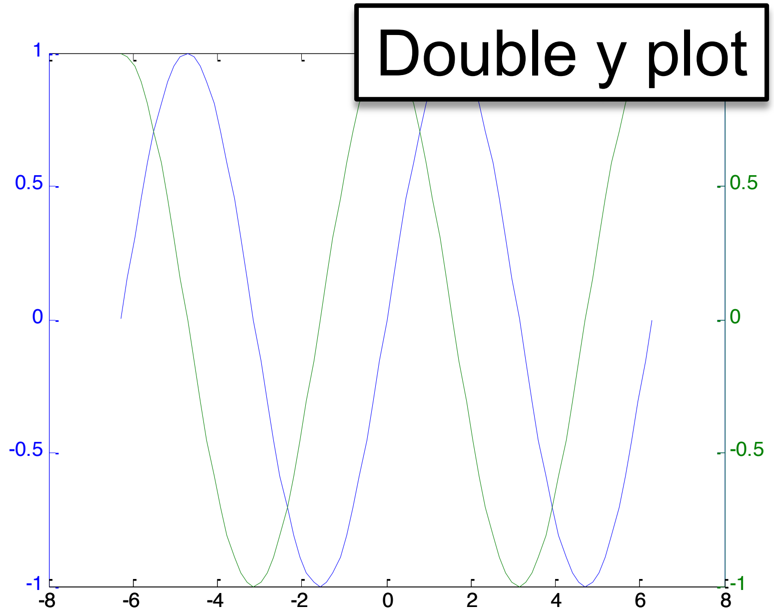
```
plot([0 20],[-5 5],'r--')
```



Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

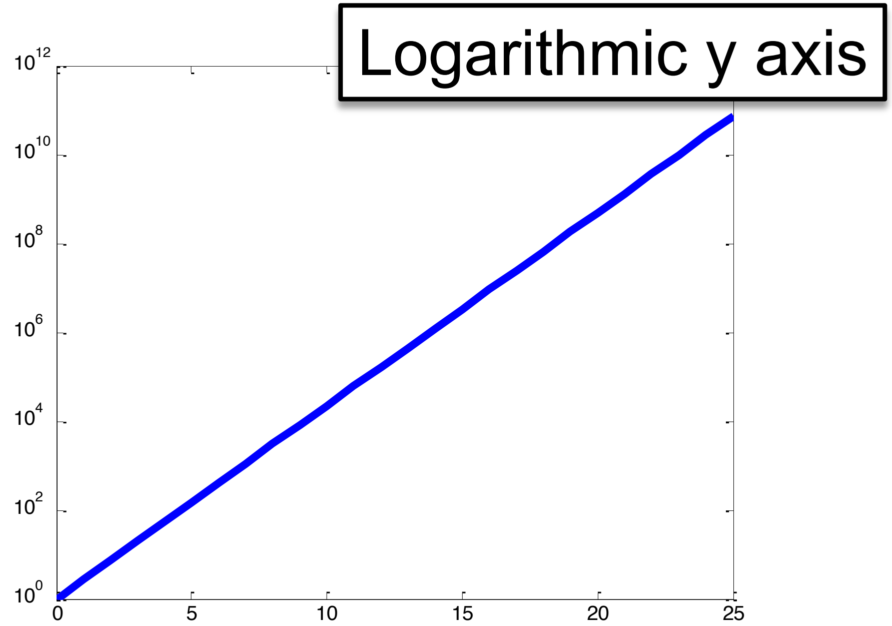
```
x = -2*pi:pi/20:2*pi;
y1 = sin(x);
y2 = cos(x);
plotyy(x,y1,x,y2)
```



Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

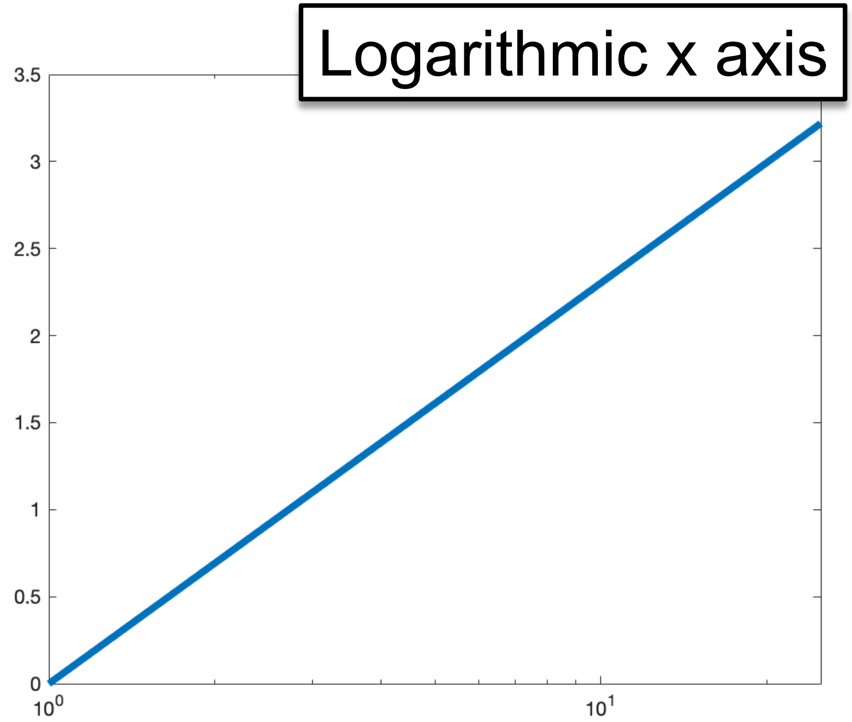
```
x = 0:1:25;
y = exp(x);
semilogy(x,y ,'LineWidth',4);
```



Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

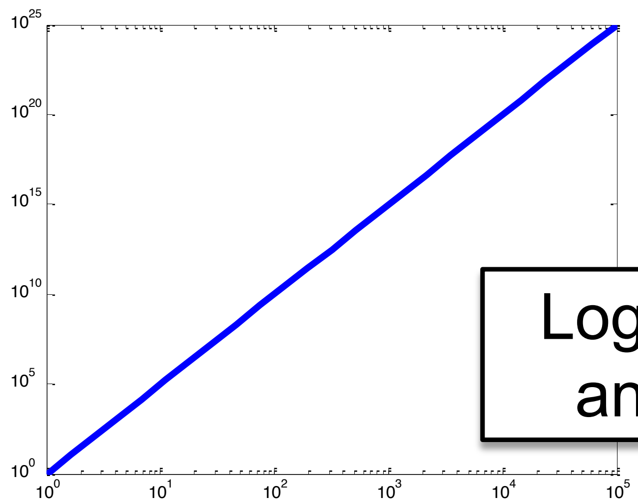
```
x = 0:1:25;
y = log(x);
semilogx(x,y,'LineWidth',4);
```



Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

```
x = logspace(0,5,25);
y = x.^5;
loglog(x,y,'LineWidth',4)
```

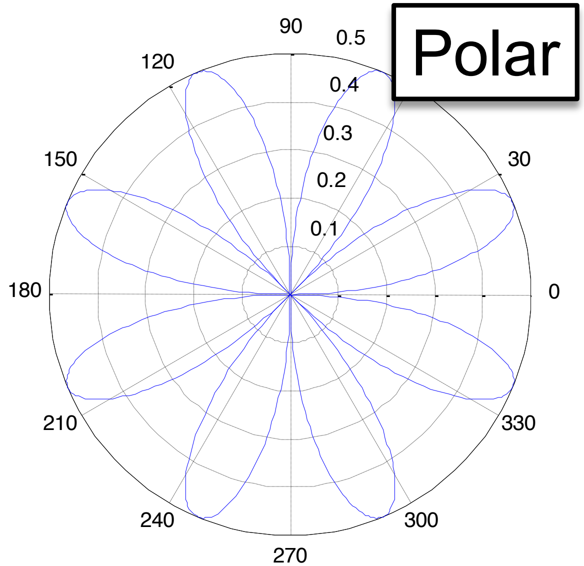


Logarithmic x and y axes

Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

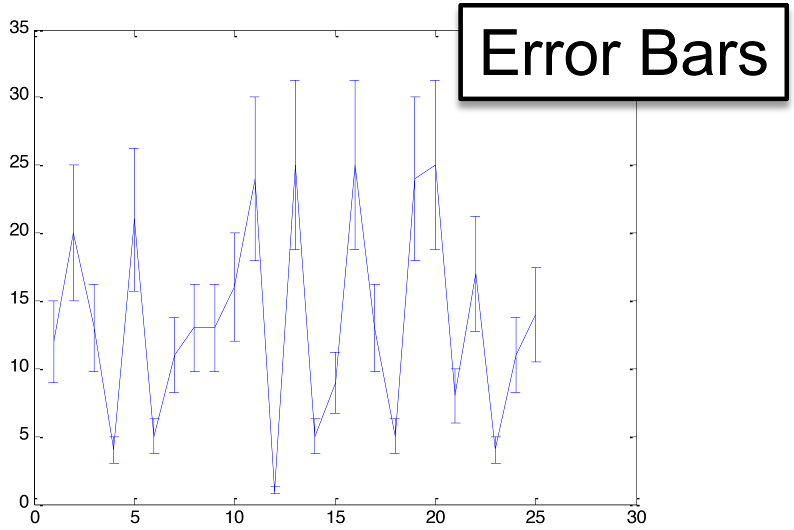
```
theta = 0:0.01:2*pi;
rho = sin(2*theta).*cos(2*theta);
polar(theta,rho)
```



Other 2D plot commands

Matlab offers a variety of pre-built 2D plot types

```
x = 1:25; y = randi(25,1,length(x));
y_err = 0.25*y;
errorbar(x,y,y_err)
```

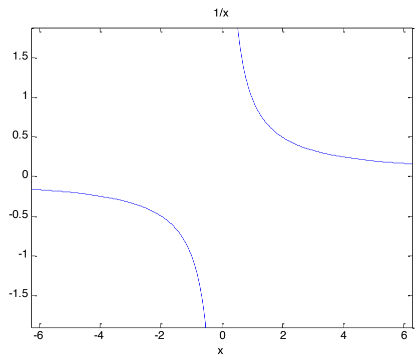


Other 2D plot commands

`ezplot` produces simple plots without the need for discrete input of x/y vectors

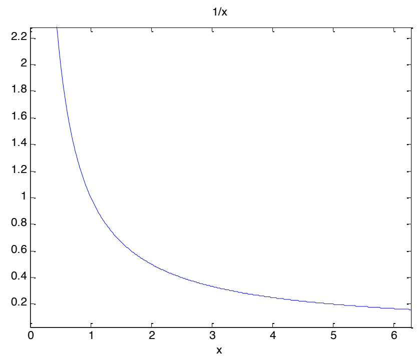
The default domain is
-2π to 2π

```
ezplot('1/x')
```



The domain may also be specified by a two element vector

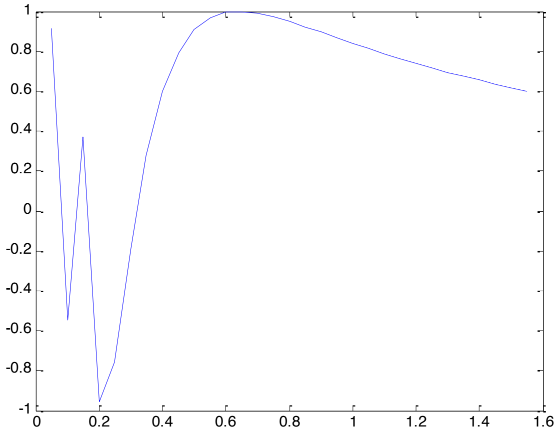
```
ezplot('1/x',[0 2*pi])
```



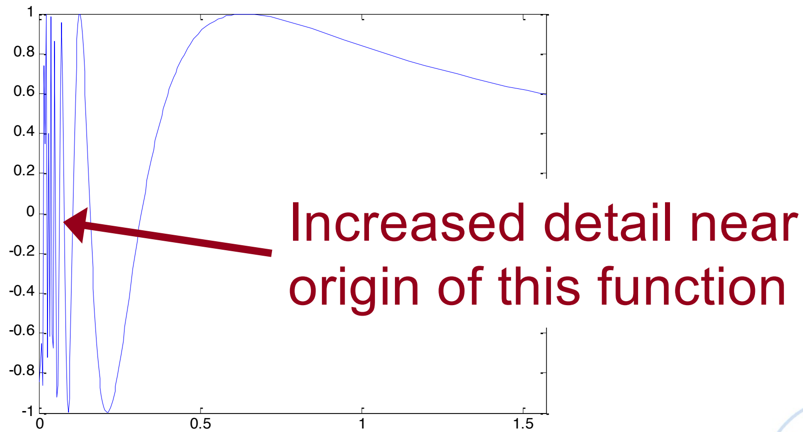
Other 2D plot commands

`fplot` intelligently selects `x` values based on changes to a function and may be used for rapidly changing functions

```
x = 0:.05:0.5*pi;
y = sin(1./x);
plot(x,y)
```




```
fplot('sin(1/x)',...
    [0 0.5*pi]);
```

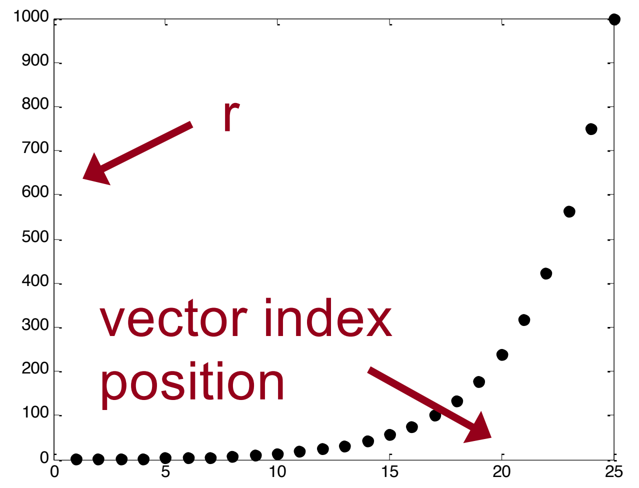


Plotting a vector

Most 2D plot commands can be called with a single vector, which is often useful for debugging

The `x` values are then assigned the index location

```
r = logspace(0,3,25)
plot(r,'ko',...
     'MarkerFaceColor','k')
```



Specifying plot attributes

The canonical plot call is

```
plot(x,y,'LineSpecifiers', ...
    'PropertyName',PropertyValue)
```

Line specifiers

- Line style: solid, dash, dot, dash-dot
- Line color: red, green, blue, cyan, etc.
- Marker: plus, circle, asterisk, etc.

Property names

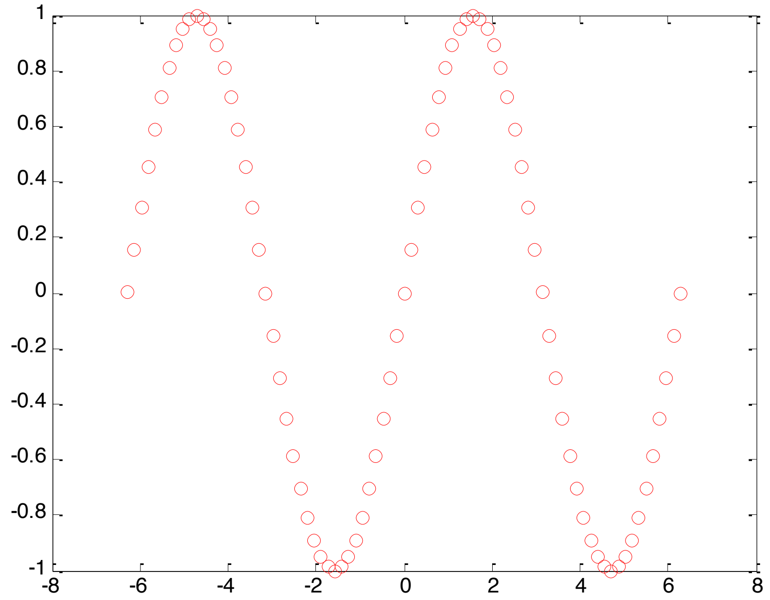
- LineWidth
- MarkerSize
- MarkerEdgeColor
- MarkerFaceColor

Specifying plot attributes

Specifying marker and line styles

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y,'ro')
```

Red circles

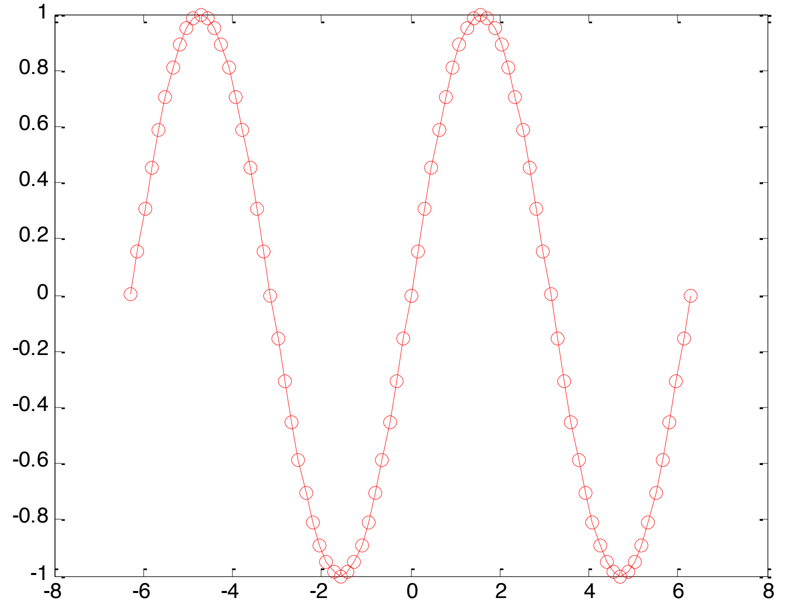


Specifying plot attributes

Specifying marker and line styles

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y,'ro-')
```

Red circles with connected lines

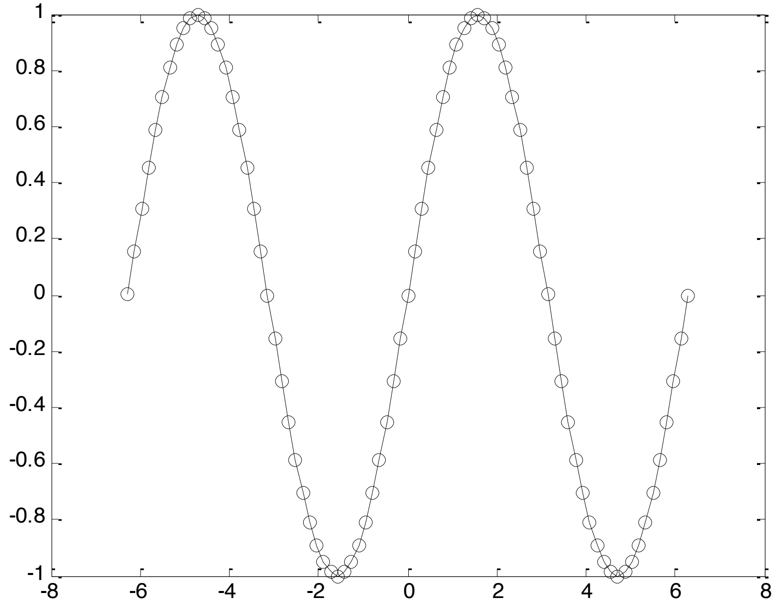


Specifying plot attributes

Specifying marker and line styles

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y,'ko-')
```

Black circles with connected lines

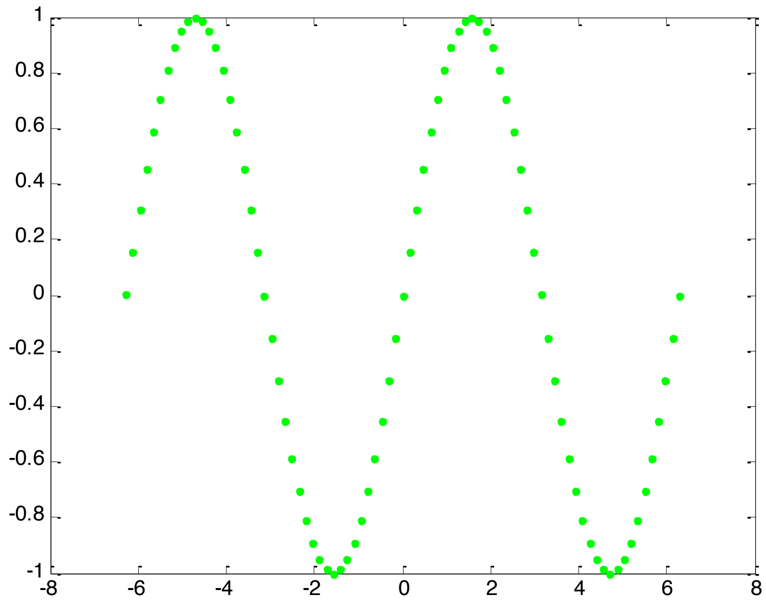


Specifying plot attributes

Specifying marker and line styles

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y,'g.')
```

Green dots

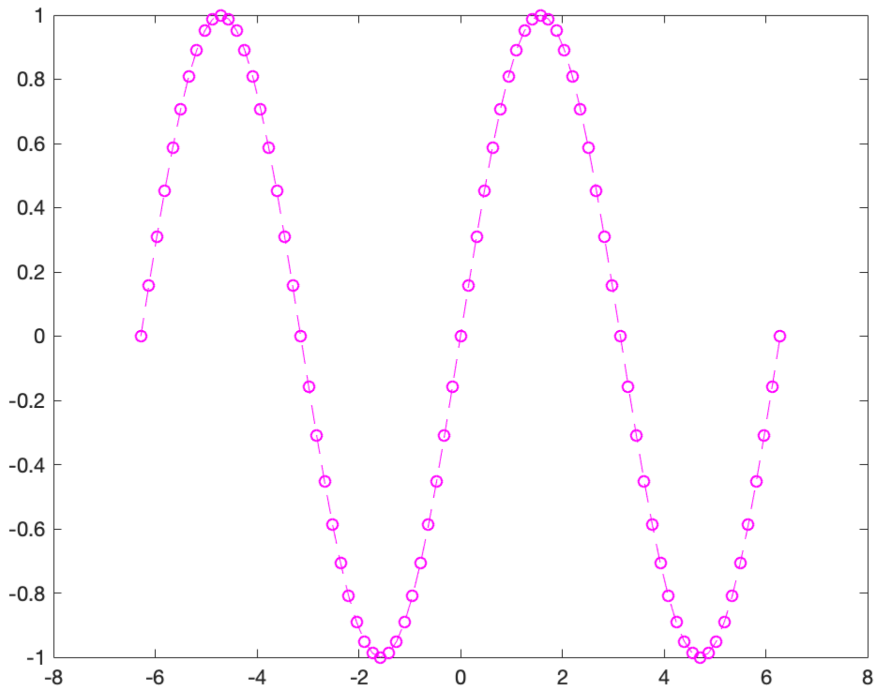


Specifying plot attributes

Specifying marker and line styles

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y,'m--o')
```

Dashed magenta line with circles

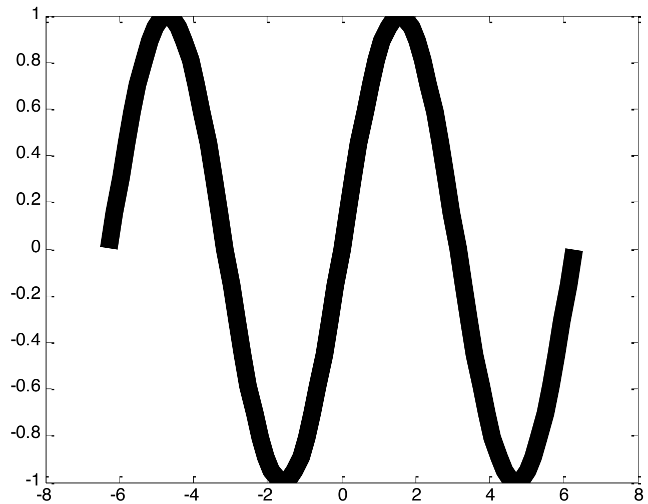


Specifying plot attributes

Many other attributes may be specified within the plot call

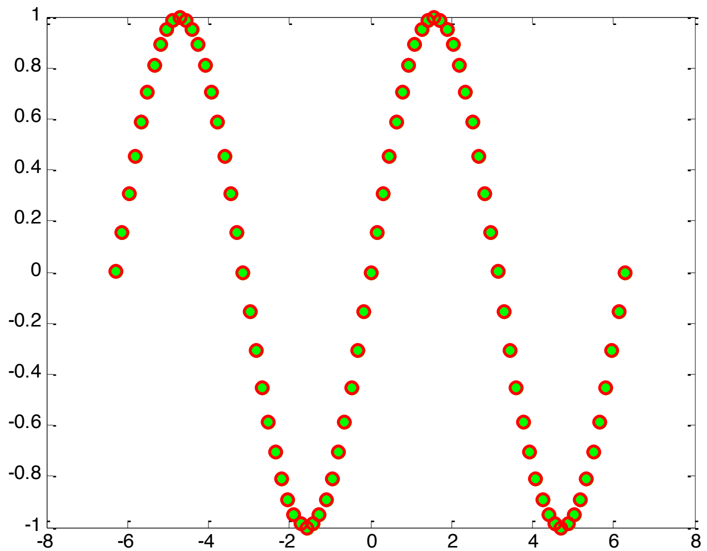
```
plot(x,y,'k-','LineWidth',10)
```

Black plot line set to a 10 pt width



Specifying plot attributes

```
plot(x,y,'o','MarkerFaceColor','g',...
    'MarkerEdgeColor','r','LineWidth',2)
```



Circles with a green face color, red edge color, and 2 pt edge width

Specifying plot attributes

MATLAB includes many predefined colors

However, you are not limited to just this color set (it's rather ugly anyways)

Colors may be specified using the `Color` (or other appropriate) property and specifying the 3-digit RGB color set

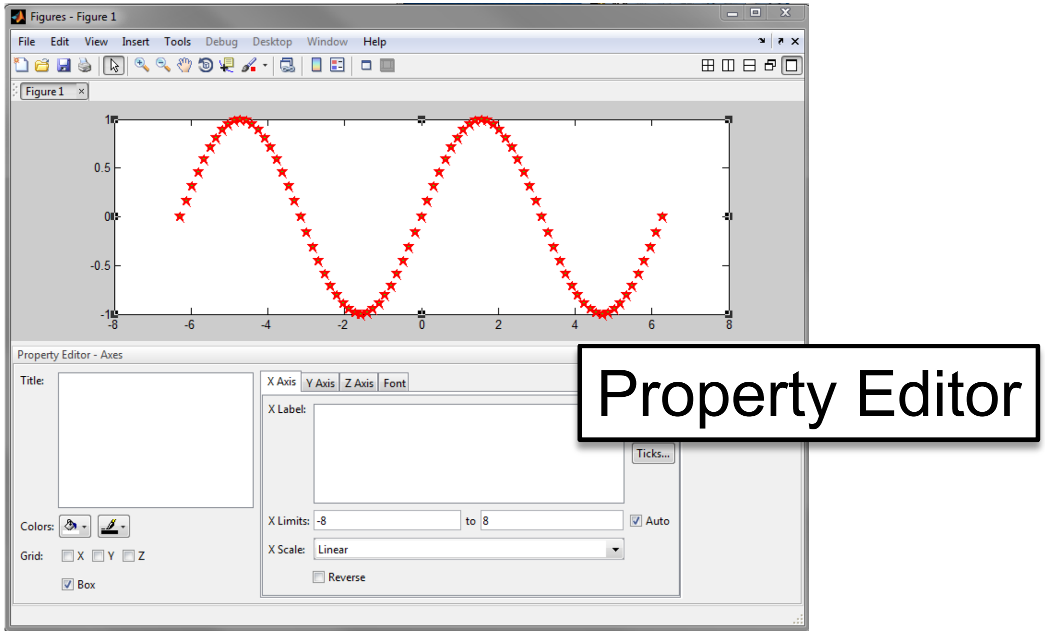
MATLAB's predefined color set

r = Red
g = Green
b = Blue
c = Cyan
m = Magenta
y = Yellow
k = Black
w = White

RGB colors typically range from 0 to 255, however, MATLAB takes in color values rescaled from 0 to 1

Specifying plot attributes

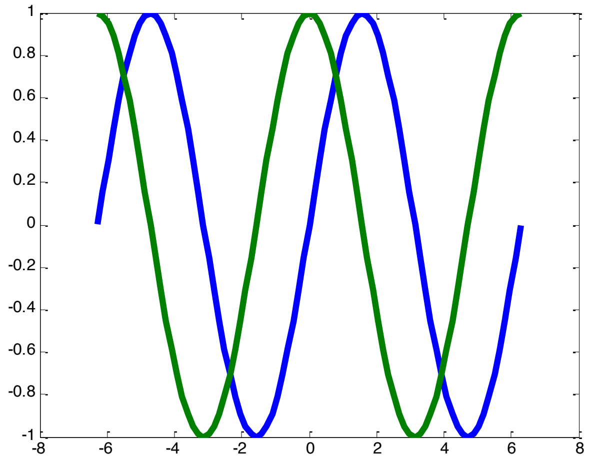
Line specifications may be investigated by searching `linespec` in the MATLAB documentation (DEMO), by choosing view/property editor from the plot options bar (DEMO), or by choosing view/property editor/more properties (DEMO)



Overlaid plots

Multiple plots may be overlaid by specifying multiple vectors

```
x = -2*pi:pi/20:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,x,y2,'LineWidth',4)
```

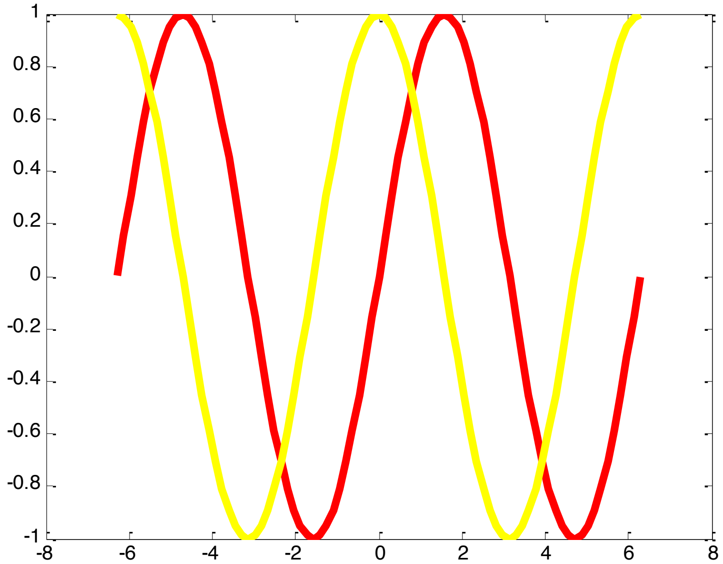


Note that x must be specified for each plot instance, MATLAB automatically chooses the plotting colors unless otherwise specified, and line specifications are adopted by all of the plot vectors

Overlaid plots

Plot color may be specified for each vector, but line specifications such as `LineWidth` invoked within the plot function will be applied to all data series

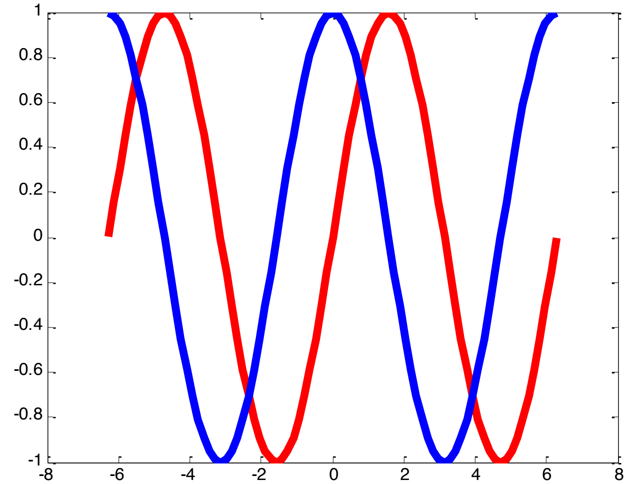
```
plot(x,y1,'r-',x,y2,'y-','LineWidth',4)
```



Overlaid plots

Multiple plots may also be overlaid by using the `hold on` / `hold off` command

```
x = -2*pi:pi/20:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,'r-','LineWidth',5)
hold on
plot(x,y2,'b-','LineWidth',5)
hold off
```



`hold on` is invoked after the first plot command and indicates that more data will be attributed to the existing axes;
`hold off` indicates that additional plotting will overwrite the axes

Setting axes or figure properties

Characteristics of the plot may be modified by referring to the current axes or figure

`gca`: get current axes

`gcf`: get current figure

This may be combined with the `set` command to modify a plot characteristic

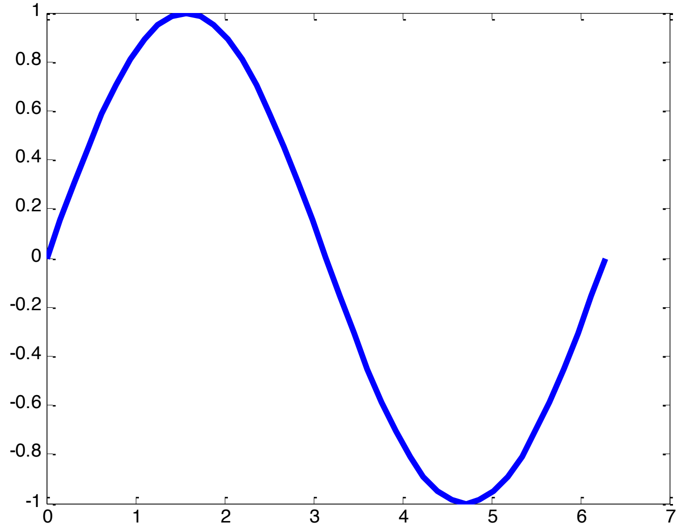
ex. `set(gca,'FontSize',20)`

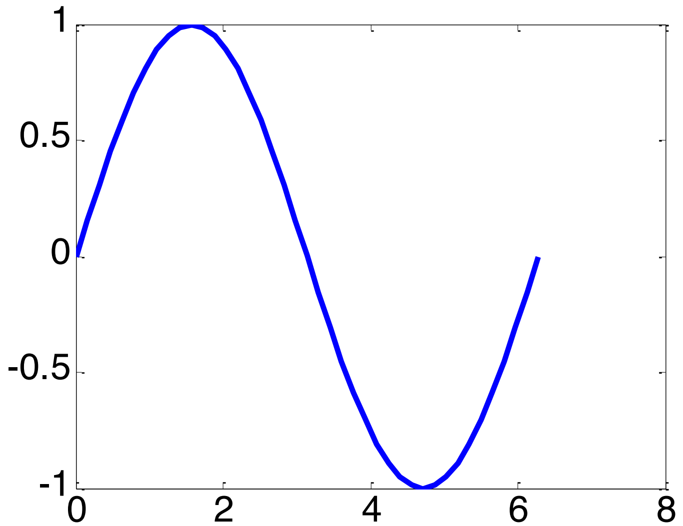
The general form for the `set` command is

```
set(<ref to handle>,<property name>,<property value>)
```

Setting plot characteristics using `gca` and `gcf`

```
x = 0:pi/20:2*pi;
y = sin(x);
plot(x,y,'LineWidth',3)
```

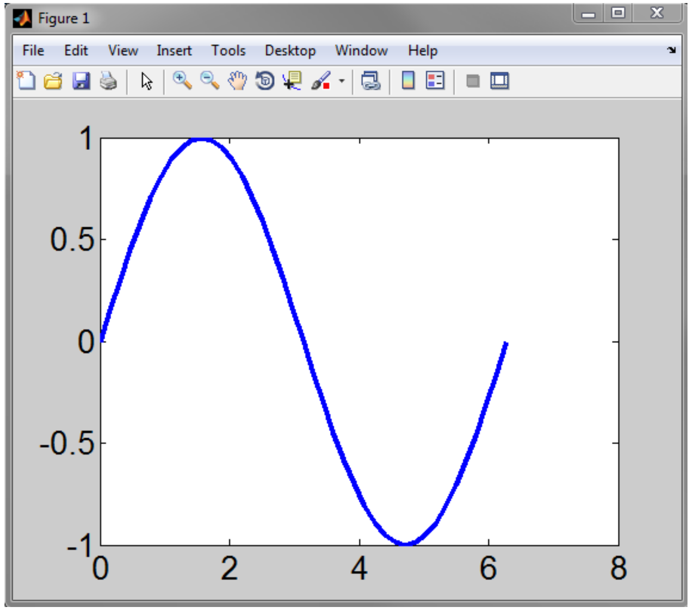


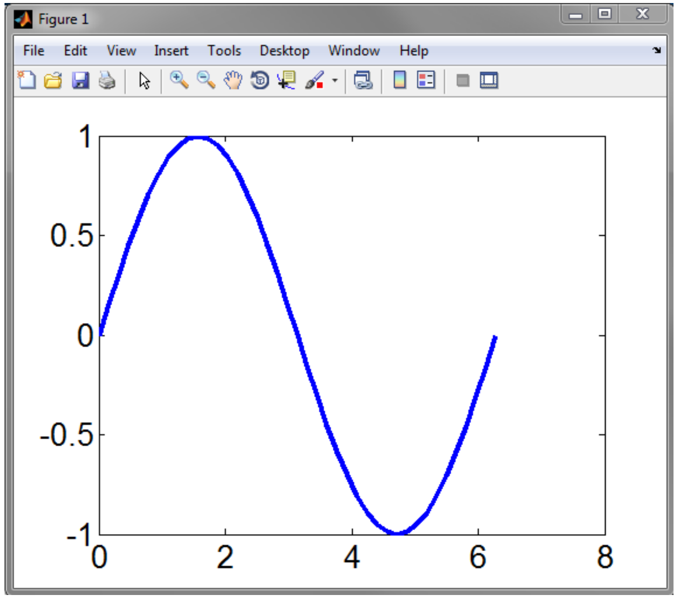



```
x = 0:pi/20:2*pi;
y = sin(x);
plot(x,y,'LineWidth',3)
set(gca,'FontSize',20)
```

Setting plot characteristics using gca and gcf

```
x = 0:pi/20:2*pi;
y = sin(x);
plot(x,y,'LineWidth',3)
set(gca,'FontSize',20)
```

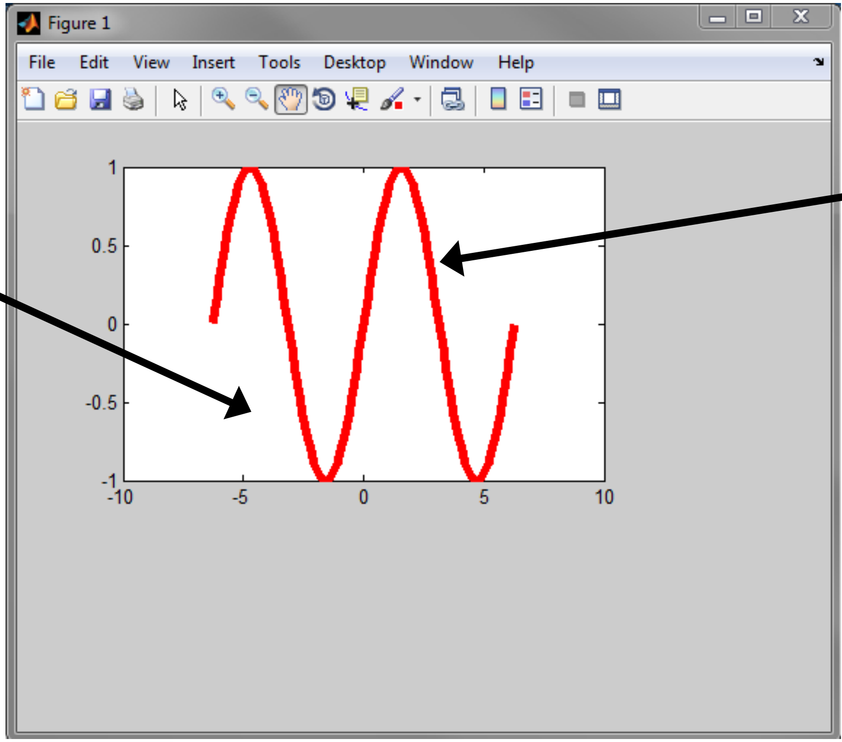





```
x = 0:pi/20:2*pi;
y = sin(x);
plot(x,y,'LineWidth',3)
set(gca,'FontSize',20)
set(gcf,'Color','w')
```

Plot parent / child hierarchy

axes = "child" of "parent" figure window



data set = "child" of "parent" axes

figure

Multiple axes can exist on a single figure window!!

Plot parent / child hierarchy

When invoking the `plot()` command MATLAB recognizes that you want to create a set of axes and a figure window to contain the plot information

When invoking the `axes()` command MATLAB recognizes that you want to create a figure window to contain the axes

Close all figures and try typing `axes` or `figure` at the command line

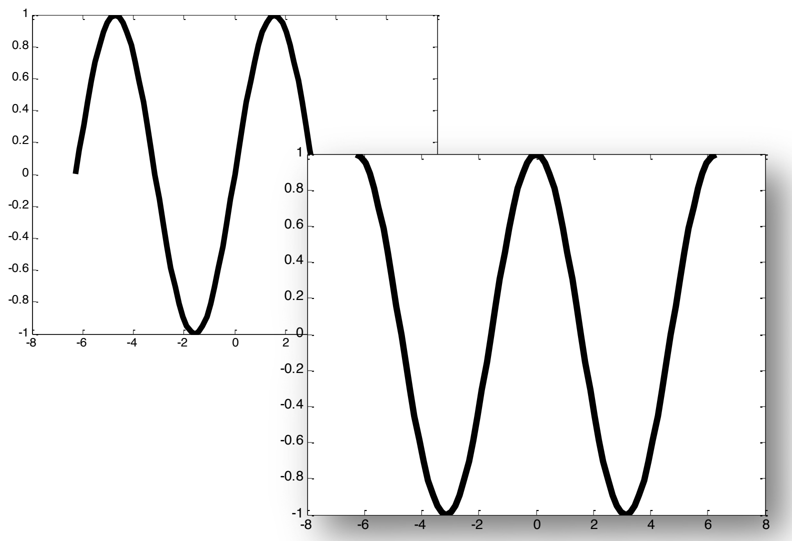
Plotting multiple figures

Say you want to plot multiple figures

The second plot command executed (without specifying hold on) will overwrite the first plot

You need to specify a new figure establish a new parent figure window

```
x = -2*pi:pi/20:2*pi;
y1 = sin(x);
y2 = cos(x);

plot(x,y1,'k-','LineWidth',5)

figure
plot(x,y2,'k-','LineWidth',5)
```



Plotting multiple figures

Figures numbers may also be specified with a figure number, ex. `figure(3)`

Plots may then be closed using the close command

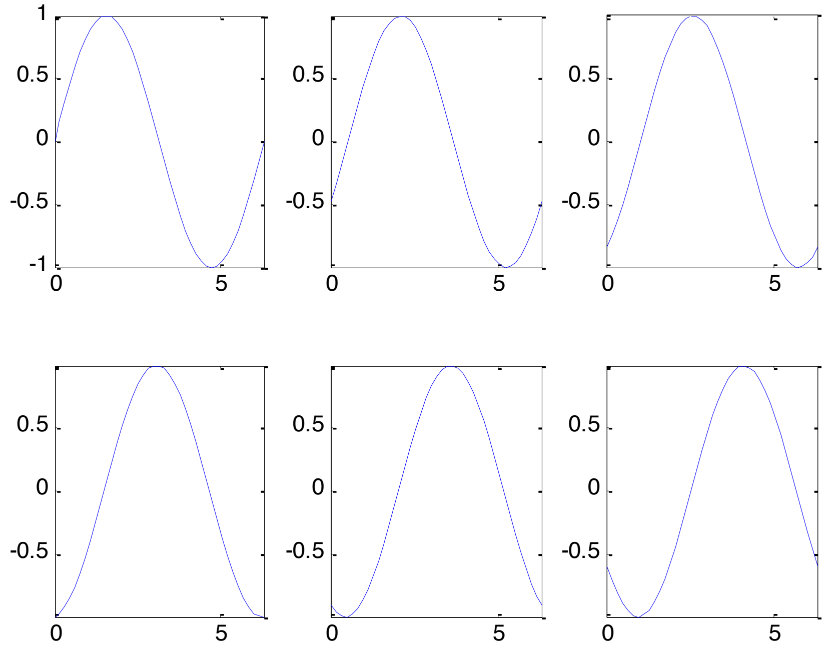
`close`: closes the last figure

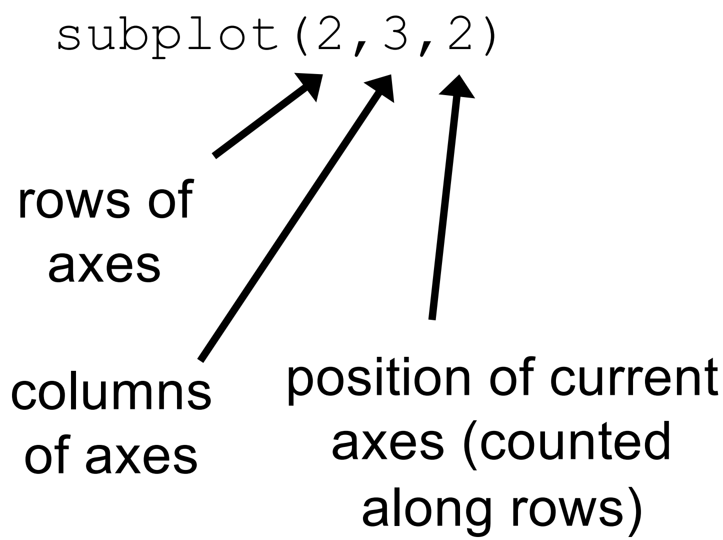
`close(n)`: closes figure number n

`close all`: closes all figures

Multiple axes on a figure

Multiple axes may be assigned to a single figure using the `subplot` command or by specifying the `position` of multiple axes





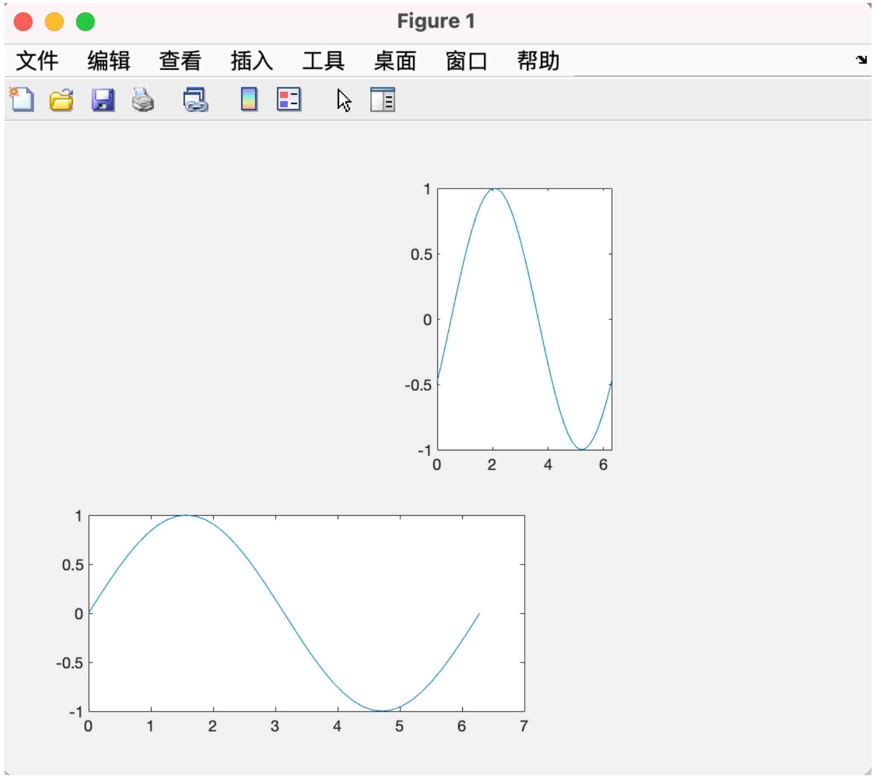
Multiple axes on a single figure

Code from the previous example

```
x = 0:pi/20:2*pi;
y1 = sin(x-0);
y2 = sin(x-.5);
y3 = sin(x-1);
y4 = sin(x-1.5);
y5 = sin(x-2);
y6 = sin(x-2.5);

subplot(2,3,1)
plot(x,y1)
axis tight

subplot(2,3,2)
plot(x,y2)
axis tight

subplot(2,3,3)
plot(x,y3)
axis tight

subplot(2,3,4)
plot(x,y4)
axis tight

subplot(2,3,5)
plot(x,y5)
axis tight

subplot(2,3,6)
plot(x,y6)
axis tight
```

Multiple axes on a single figure

The position of axes can be directly specified using the axes command



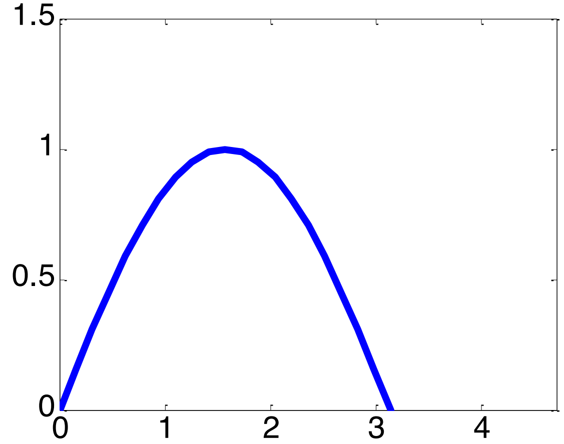
four element vector describing the lower left hand corner coordinates (x and y) and the width and height of the axes - units are in fractions of the entire window space [0 -> 1]

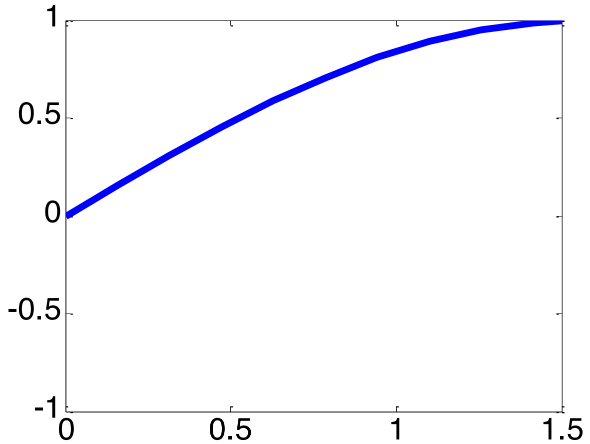
```
axes('Position',[.1 .1 .5 .3])
plot(x,y1)

axes('Position',[.5 .5 .2 .4])
plot(x,y3)
```

Specifying the domain and range

MATLAB usually specifies reasonable bounds for the plot - but a specific domain and range may be specified using `axis` or `xlim/ylim`

```
plot(x,y1,'LineWidth',5)
axis([0 1.5*pi 0 1.5])
set(gca,'FontSize',20)
```

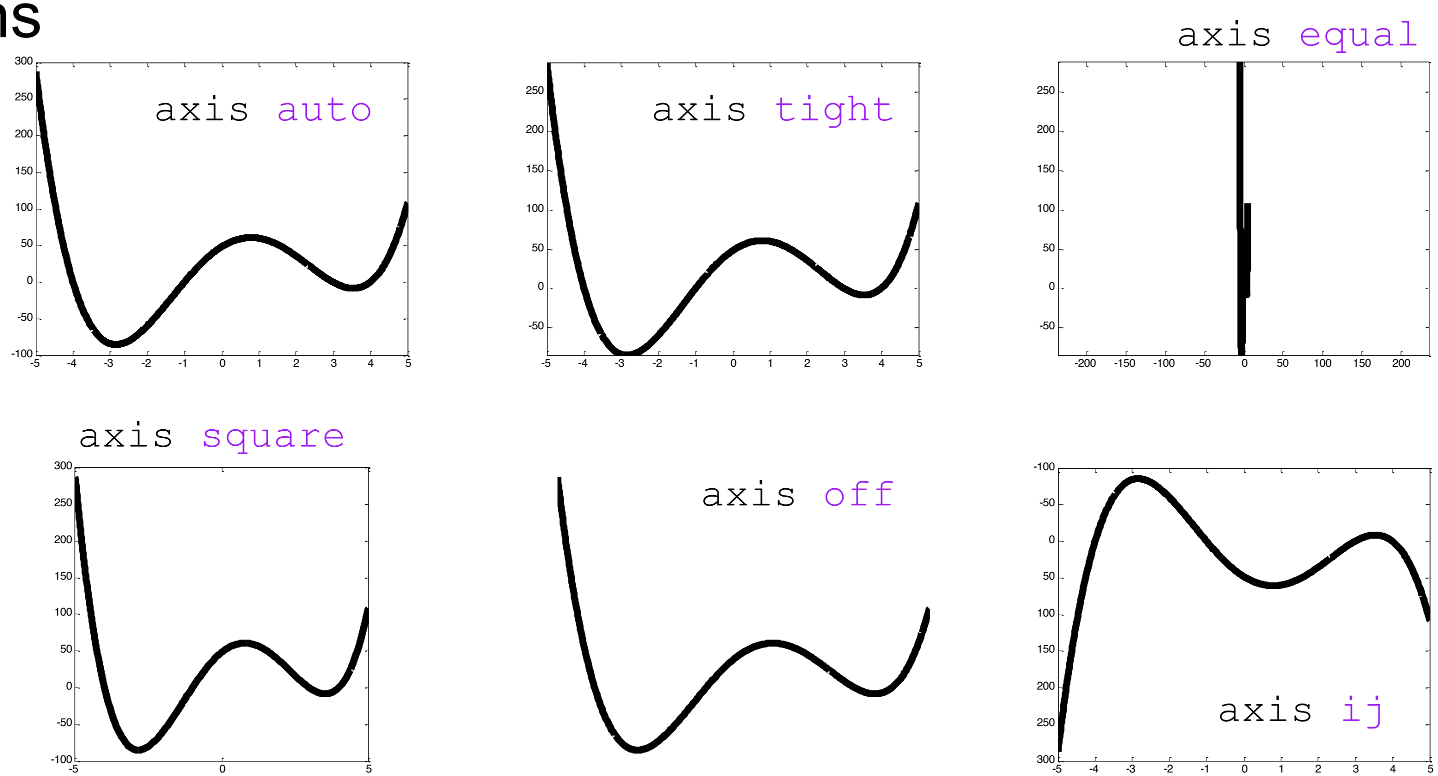





```
plot(x,y1,'LineWidth',5)
xlim([0 1.5])
ylim([-1 1])
set(gca,'FontSize',20)
```

Specifying the domain and range

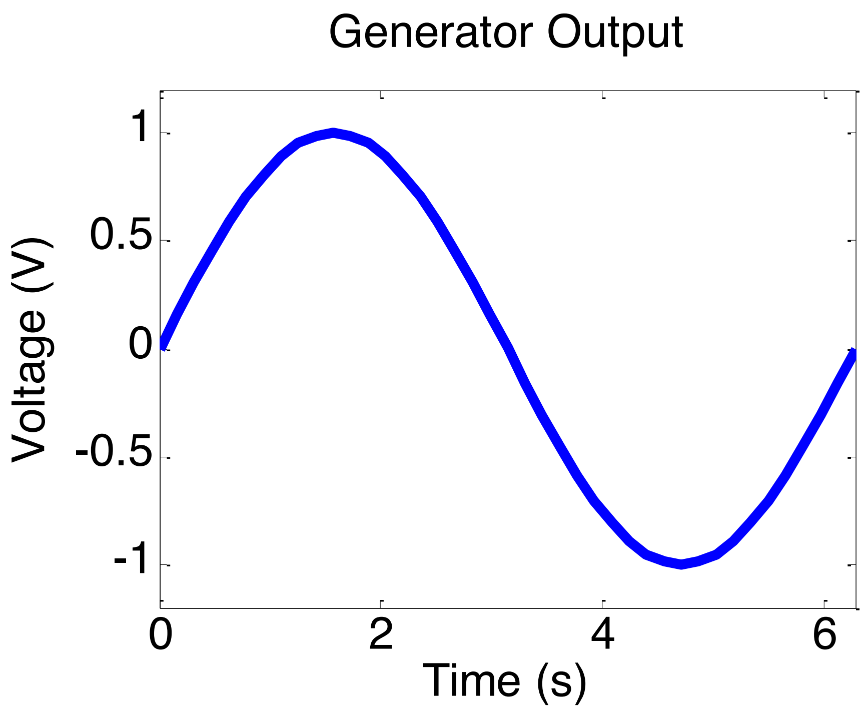
The axis command offers many predefined axis scaling/sizing operations



Labeling and titling axes

Axis labels and titles may be placed on the graph

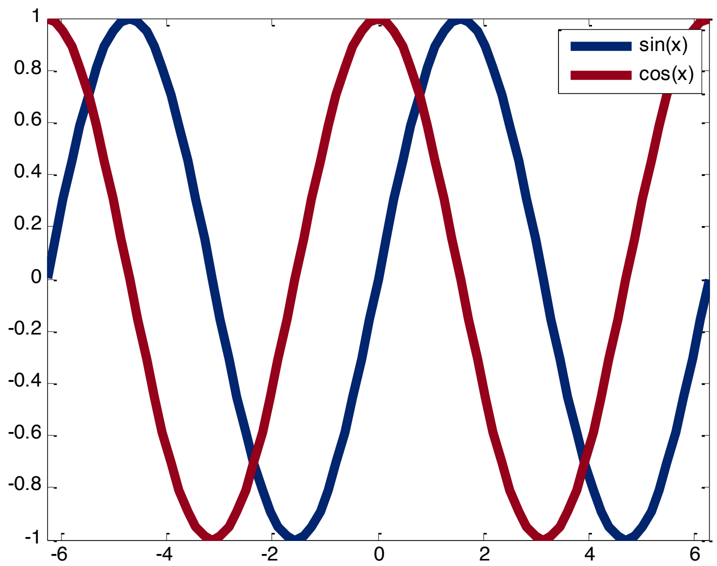
```
plot(x,y1,'LineWidth',5)
set(gca,'FontSize',20)
xlabel('Time (s)')
ylabel('Voltage (V)')
title('Generator Output')
axis([0 2*pi -1.2 1.2])
```



Adding a legend

Legends may be added and the position of the legend specified

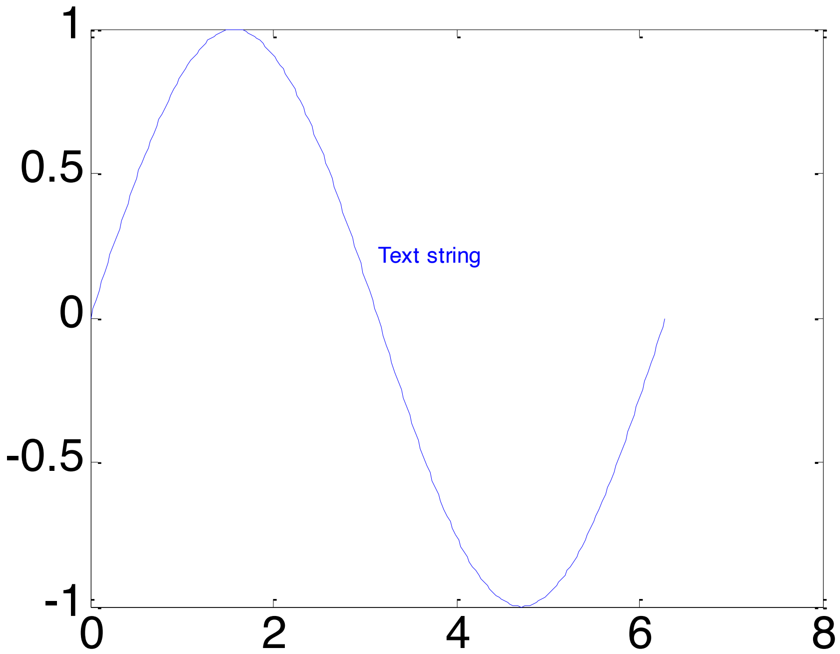
```
x = -2*pi:pi/20:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,'Color',...
    [1 37 110]./255,...
    'LineWidth',5);
hold on
plot(x,y2,'Color',...
    [149 0 26]./255,...
    'LineWidth',5);
hold off
axis tight
legend('sin(x)','cos(x)',...
    'Location','NorthEast')
```



Placing text on a plot

Text may be placed on a plot using the text command

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
set(gca,'FontSize',20)
text(pi,0.2,'Text
string','Color','b')
```



-or using features in "Insert/textBox"

Seeing in 3D

```
plot3(x, y, z, 'line specifiers',
'PropertyName', PropertyValue)
```

`x, y, z`: vectors of points

`line specifiers`: type and color

`PropertyName & PropertyValue`: other attributes

16_1

$$x = \sqrt{t}\sin(2t)$$

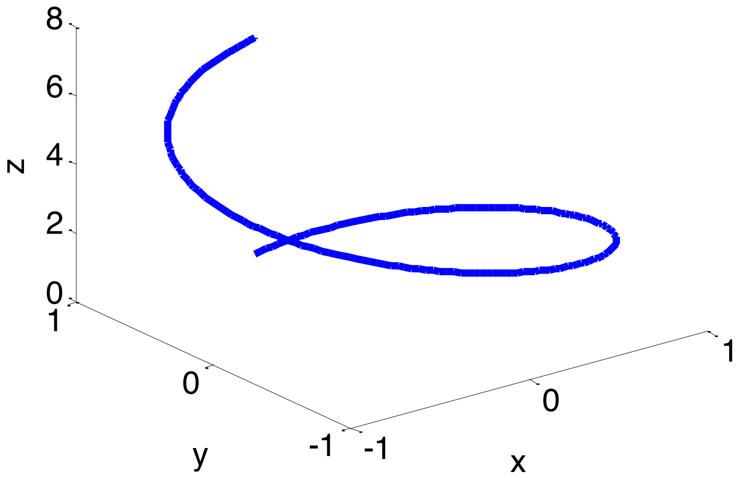
$$y = \sqrt{t}\cos(2t)$$

$$z = 0.5t$$

3D plotting (vectors)

Plotting vectors using `plot3`

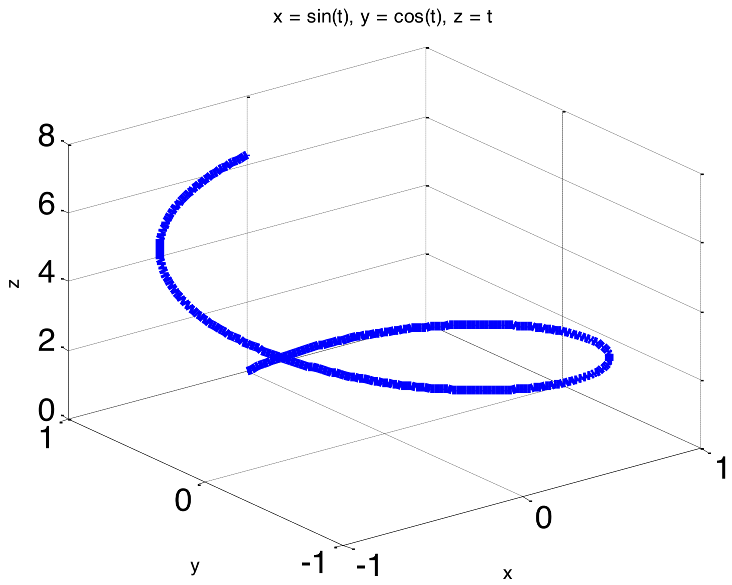
```
t = 0:pi/100:2*pi;
x = sin(t);
y = cos(t);
z = t;
plot3(x,y,z,'LineWidth',4)
set(gca,'FontSize',16)
xlabel('x')
ylabel('y')
zlabel('z')
```



3D plotting (vectors)

Plotting vectors using `ezplot3`

```
h = ezplot3('sin(t)',...
    'cos(t)','t',[0 2*pi]);
set(gca,'FontSize',16)
set(h,'LineWidth',5)
```

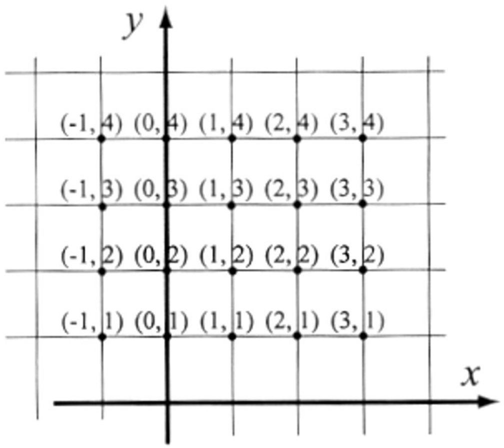


Mesh and Surface Plots

- `mesh` lines connecting the points
- `surf` areas colored

- Useful for functions of the form $z = f(x,y)$
- Three steps
  1. Make a grid in the x-y plane
  2. Find the value of z for each point of the grid
  3. Draw the mesh or surface plot

Step 1: Make the Grid



$$X = \begin{bmatrix} -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \end{bmatrix} \text{ and } Y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

```
[X, Y] = meshgrid (x, y)
```

`X, Y`: matrices of x and y coordinates, respectively

`x, y`: vectors that define the domain of the grid

Step 2: Find the z Values

- Element-by-element calculation

$$z = \frac{xy^2}{x^2 + y^2}$$

Step 3: Draw the Plot

- `mesh(X, Y, Z)` or `surf(X, Y, Z)`

16-2

Variations on mesh and surface plots

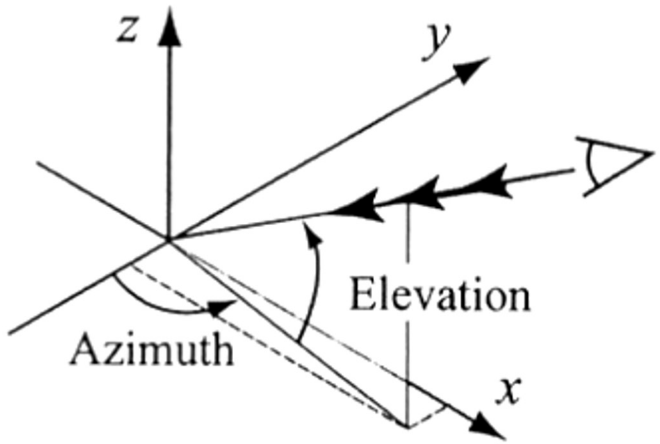
16-3

Other 3D Plots

- Sphere
- Cylinder
- 3D bar
- 3D stem
- 3D scatter
- 3D pie

16-4, text 7.2.9

Controlling the View – in 3D



`view(az, el)` or `view([az, el])`

`az`: azimuth relative to negative y-axis

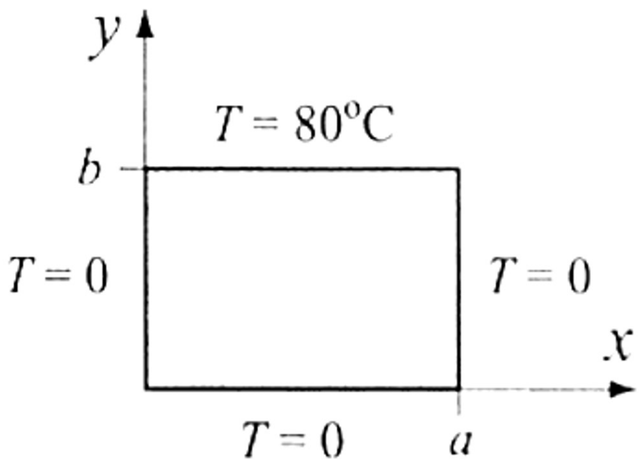
(default is -37.5°)

`el`: elevation from x-y plane

(default is 30°) 16-5

Example Heat Conduction on a Plate

- Three sides of a rectangular plate (a = 5m, b = 4m) are kept at 0° C.
- The fourth side is kept at 80° C.
- Plot the temperature distribution T(x, y) across the plate.



Example Heat Conduction on a Plate

- Two-dimensional heat equation

$$T(x,y) = \frac{4T_1}{\pi} \sum_{n=1}^{\infty} \frac{\sin\left[(2n-1)\frac{\pi x}{a}\right]}{2n-1} \frac{\sinh\left[(2n-1)\frac{\pi y}{a}\right]}{\sinh\left[(2n-1)\frac{\pi b}{a}\right]}$$

- Steps
  - Create the grid on the x and y domain dividing width *a* into 20 segments and width *b* into 16.
  - Calculate temperature at each point
  - Plot the results

  Ex16_2.m

Example Electric Field of Two Point Charges

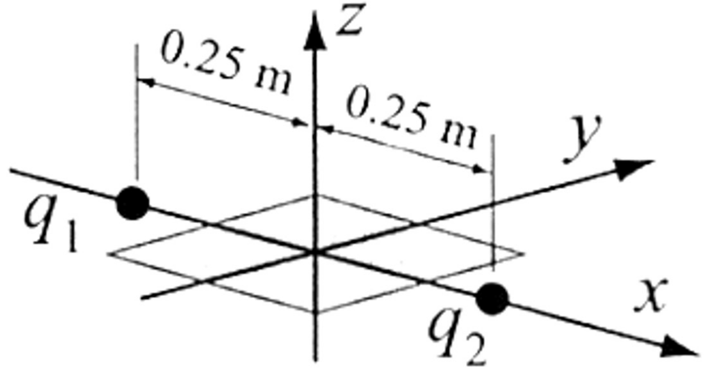
- Electric potential V around a charged particle is

$$V = \frac{1}{4\pi\varepsilon_0}\frac{q}{r}$$

$$\text{where } \varepsilon_0 = 8.8541878 \times 10^{-12}\,\frac{\mathrm{C}}{\mathrm{N}\cdot\mathrm{m}^2}$$

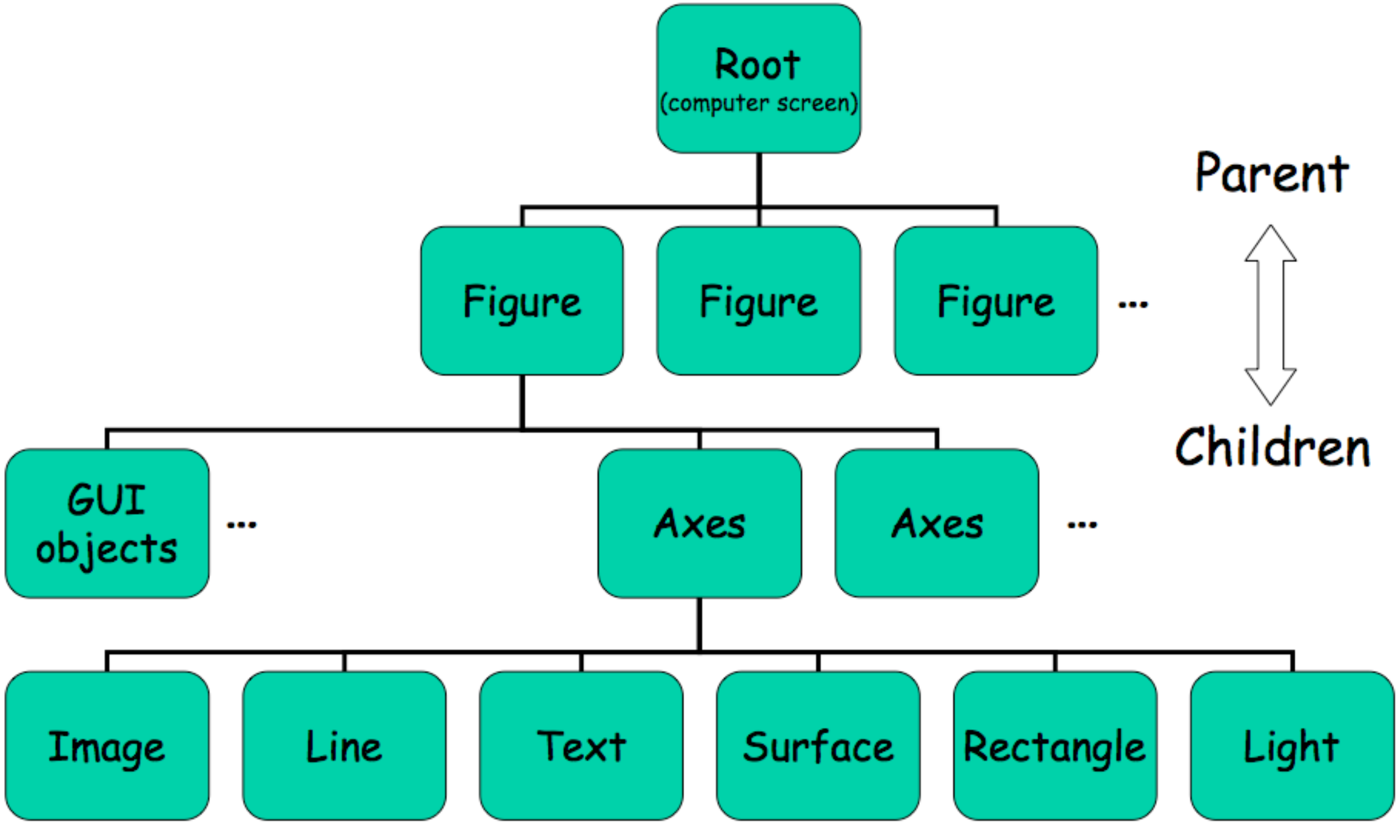
$$q = \text{charge in Coulombs}$$

$$r = \text{distance from particle (m)}$$



- Two particles with charges $q_1 = 2\text{x}10^{-10}$ and $q_2 = 3\text{x}10^{-10}$ are positioned as shown.
- Calculate and plot the electric potential due to the particles. Ex16_3.m

MATLAB Graphics System



Properties of Objects

- Control how the object looks or behaves
- Each has `property name` and `property value`
- Default values applied when object is created
  – can be overridden

```
hnd1 = plot(x, y, 'LineWidth', 2);
```

- Use `get` and `set` to retrieve and change properties of existing objects

L17-1.m

Graphics handles

Figures, axes, and plots can all be assigned handles

These handles look a lot like "variables" and serve as a pointer to the object

ex.

```
myHandle1 = plot(x,y);

myFig = figure;

h = axes;

foo = axes;
```

Graphics handles

The `set` command is then used to set properties associated with the handle

The general form of the set command is

```
set(<handle>,<property>,<propertyValue>)
```

ex.

```
pl1 = plot(x,y);
set(pl1,'LineWidth',2)
```

Graphics handles

The `get` command may be used to get properties associated with the handle

The general form of the set command is

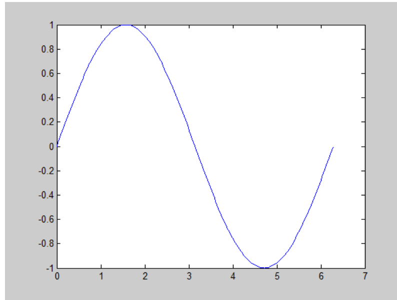
```
get(<handle>,<property>)
```

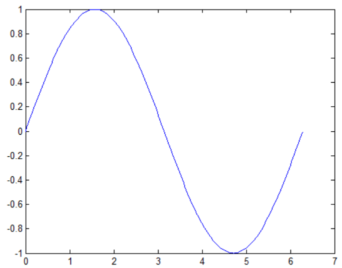
ex.

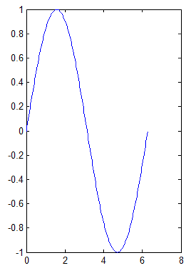
```
fig1 = figure
pl1 = plot(x,y);
get(fig1,'Position')
```

An example of graphics handles

```
% Example handle graphics use

x = 0:pi/100:2*pi;
y = sin(x);

myFigure = figure;
plot(x,y)

% Set the background of the figure
to
% white
set(myFigure,'Color','w')

% Take the default plot position
and
% reduce the width by 1/2
pos = get(myFigure,'Position');
pos(3) = 0.5*pos(3);
set(myFigure,'Position',pos);
```







Example - Customizing a Plot

$$\mathrm{sinc}(x)=\begin{cases}\dfrac{\sin x}{x} & x\neq 0\\ 1 & x=0\end{cases}$$

- Plot the function from -3πι ≤ x ≤ 3πι then
  - – make the background pink
  - – show the y-axis grid lines only
  - – change the line to orange and 2-points wide

Ex17-1

Example - Selecting Objects

- Write a program that displays information about selected objects in a plot until a key is pressed

Ex17-2

Example - Positioning Objects

- The program creates two overlapping sets of axes with a single figure
- The first has a text comment attached to the line itself
- The second has a text comment in the lower left corner

  Ex17-3

Example - Animating Plots

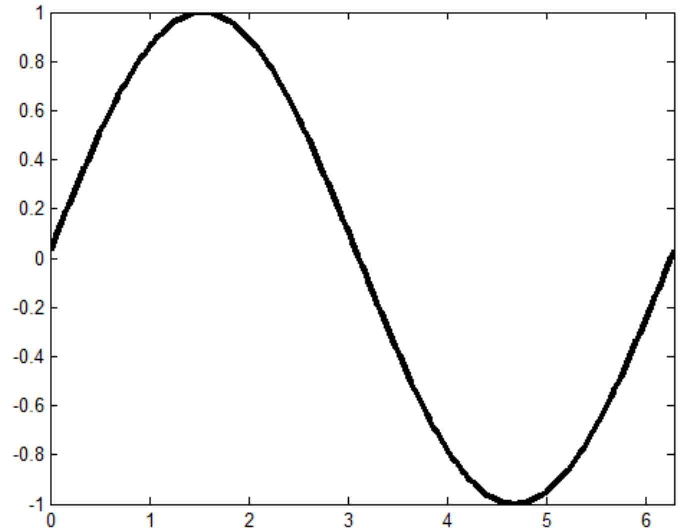
- Fine control over plotting allows a form of animation

Ex17-4

Animating figures

Example: 2D shifting a sine wave

```
x = 0:pi/100:2*pi;
y = sin(x);

p1 = plot(x,y,'k-','LineWidth',3);
axis tight

for jj = 1:100
    pause(0.01)
    y = sin(x+jj*pi/100);
    set(p1,'XData',x,'YData',y)
end
```

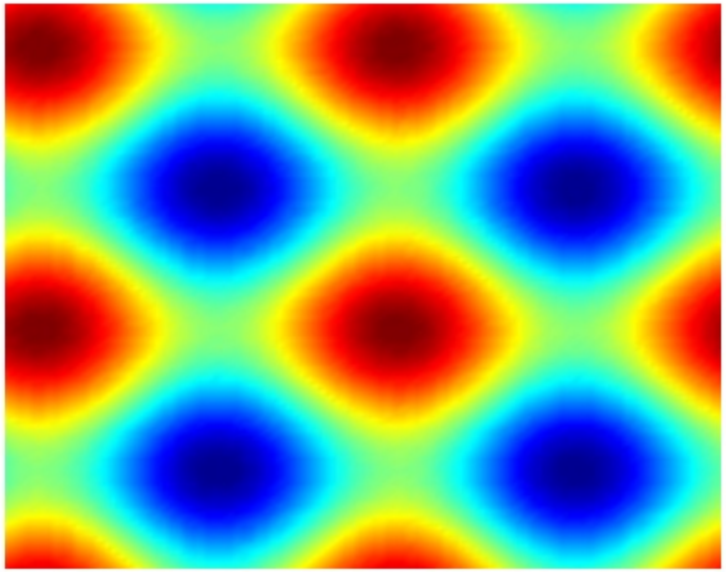


A combination of pausing and updating the axis data results in an animation

Animating figures

Example: 3D shifting trigonometric field

```
x = linspace(-2*pi,2*pi,100);
y = x;
[X,Y] = meshgrid(x,y);
Z = sin(X)+cos(Y);
h = surface(Z);
shading interp
axis off
set(gcf,'Color','w')

for jj = 1:100
    pause(0.05)
    Z = sin(X+jj*pi/100) + cos(Y+jj*pi/100);
    set(h,'CData',Z)
end
```



A combination of pausing and updating the axis data results in an animation