Lecture 4 - Plotting

Example Plots

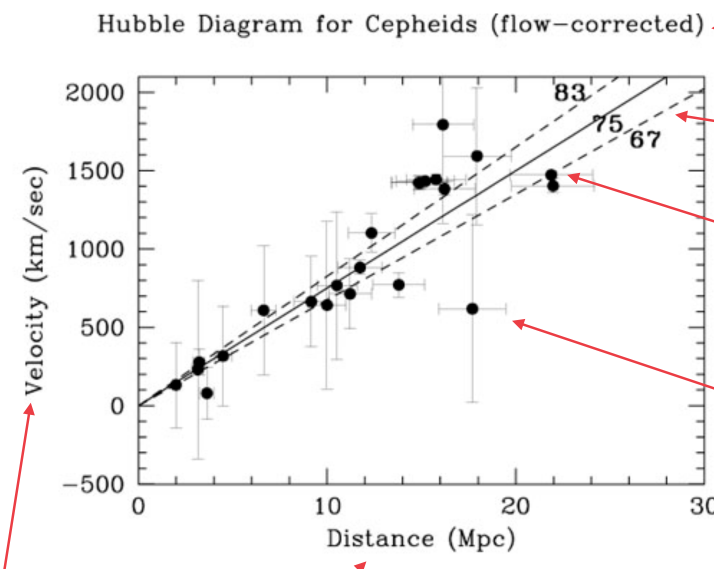
informative title

theoretical curves

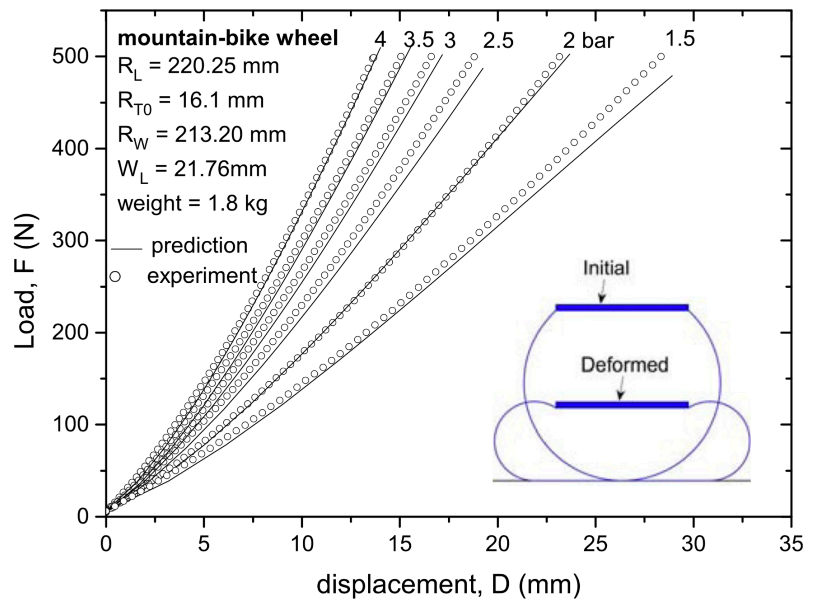
data points as filled circles (not connected with lines)

error bars (in x and y) plotted in gray so not distracting

axes labeled (with units)



Example Plots



Example Plots

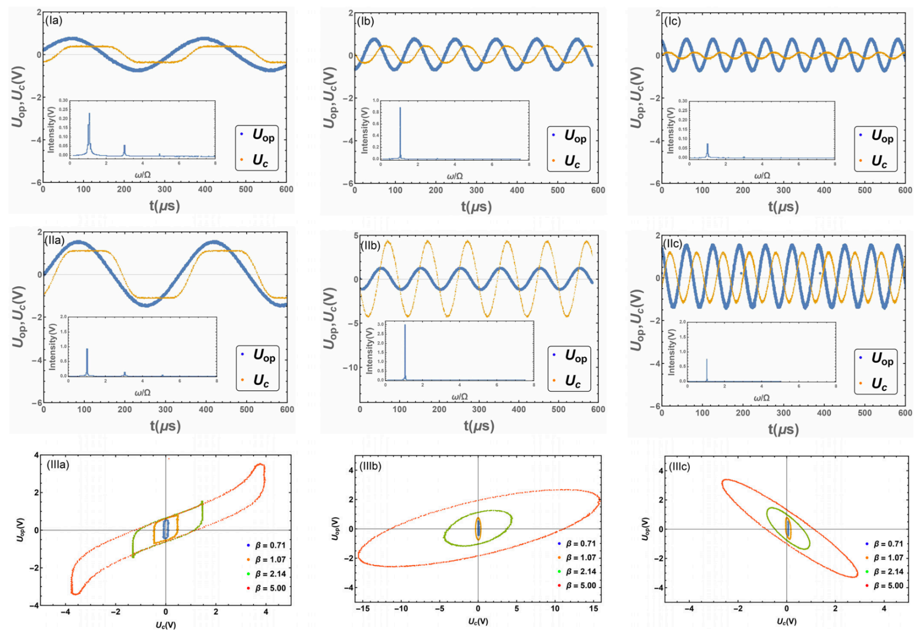
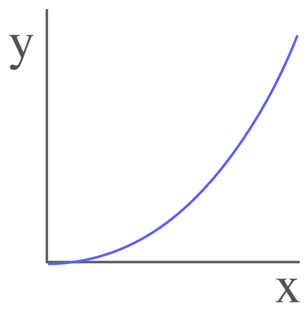


Fig. 3. (Ia)–(Ic) and (IIa)–(IIc) are the experimental results of the capacitor's voltage curves and their FFT spectrums under different driving voltages and frequencies. (IIIa)–(IIIc) are capacitor's voltage Uc with respect to the open-circuit voltage U_{op} in the X–Y mode under different driving frequencies. (Ia) $\beta = 1.07; \Omega/\omega_0 = 0.29$. (Ib) $\beta = 1.07; \Omega/\omega_0 = 0.95$. (Ic) $\beta = 1.07; \Omega/\omega_0 = 1.43$. (IIa) $\beta = 2.14; \Omega/\omega_0 = 0.29$. (IIb) $\beta = 2.14; \Omega/\omega_0 = 0.95$. (IIc) $\beta = 2.14; \Omega/\omega_0 = 1.43$. (IIIa) $\Omega/\omega_0 = 0.29$. (IIIb) $\Omega/\omega_0 = 0.95$. (IIIc) $\Omega/\omega_0 = 1.43$.

Plotting in Matlab

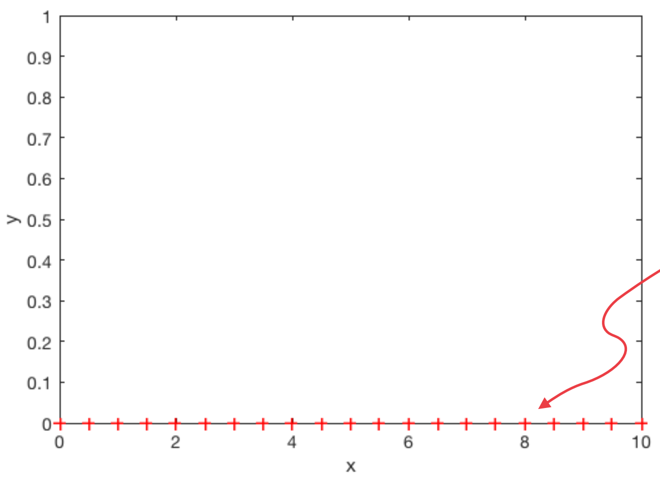
Let’s say you want to plot some function $y = f(x)$ over the domain $a \le x \le b$

As an example, we’ll use $f(x) = x^2$ on the domain $0 \le x \le 10$



Plotting in Matlab

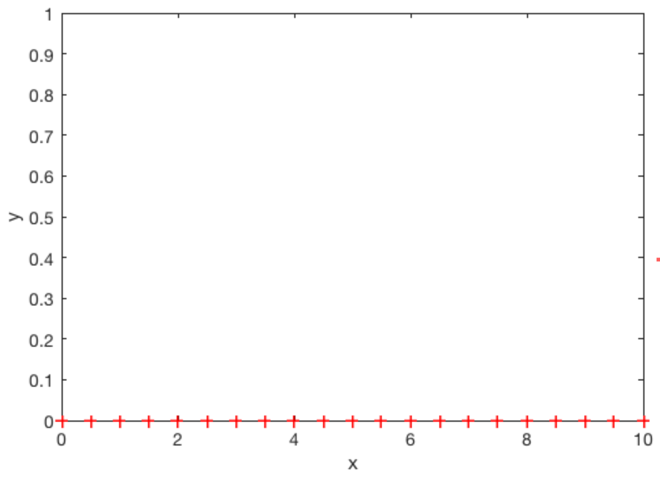
Step 1: define a set of points (stored as a row or column matrix) defining the domain along the x axis.



```
x = 0:0.5:10;
```

Plotting in Matlab

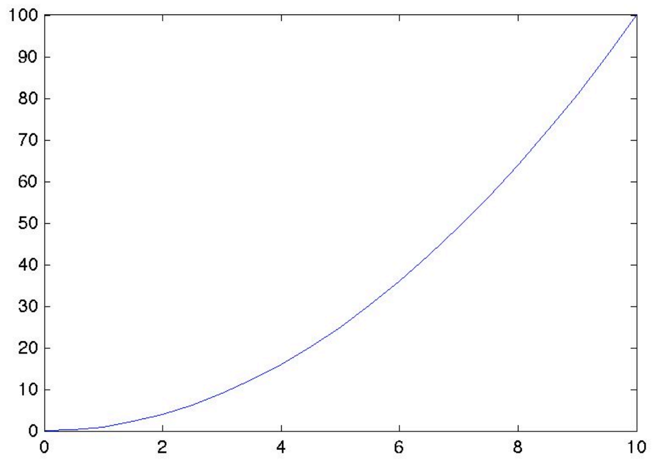
Step 2: create a second matrix y that evaluates the function $y = f(x)$ for each value of x.



```
x = 0:0.5:10;
y = x.^2;
```

Plotting in Matlab

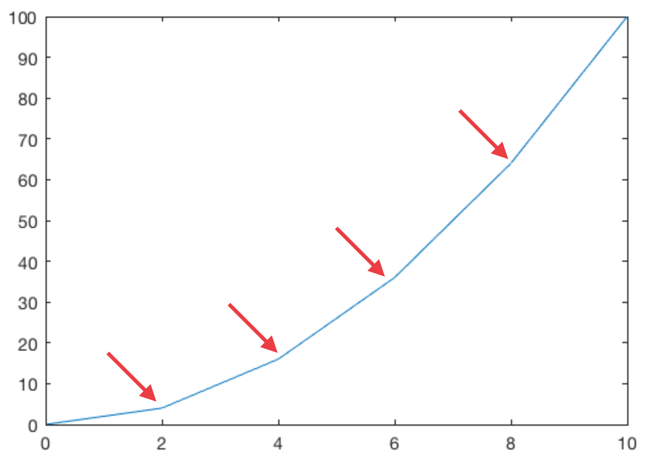
Step 3: Plot y vs x.



```
x = 0:0.5:10;
y = x.^2;
plot(x,y)
```

Plotting in Matlab

If the grid spacing is too course, the curve will not be smooth



```
x = 0:2:10;
y = x.^2;
plot(x,y)
```

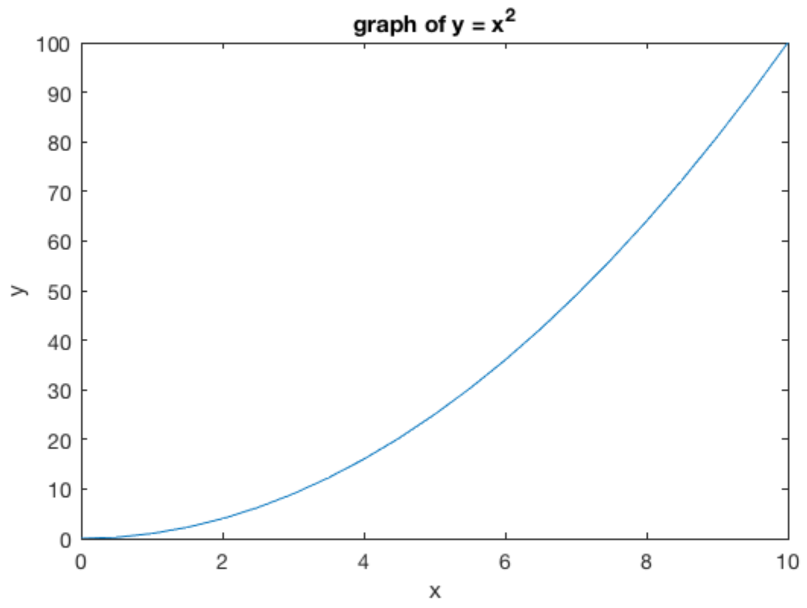
If the grid spacing is too dense, the curve will contain too many points and the final figure will take too long to plot.

Simple Plot Example: plot01A.m

```
x = 0:0.5:10;
y = x.^2;
plot(x,y)

xlabel('x (m)')       % label horizontal axis
ylabel('y (m^2)')     % label vertical axis
title('y = x^2')      % title
```

This code adds commands to label the x and y axes and a title.

Simple Plot Example: plot01A.m



Plot Options

Color Options:

```
'r' = red
'g' = green
'b' = blue
'y' = yellow
'c' = cyan
'm' = magenta
'k' = black
'w' = white
```

Marker Options:

```
'o' = circle
'+' = plus
's' = square
'*' = star
'^' = up triangle
'v' = down triangle
'x' = x
'd' = diamond
```

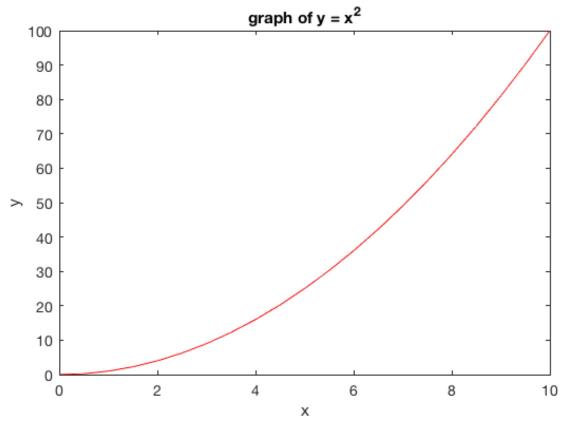
Line Options:

```
'-' = solid line
':' = dotted line
'--' = dashed line
'-.' = dash-dot line
```

Other Options:

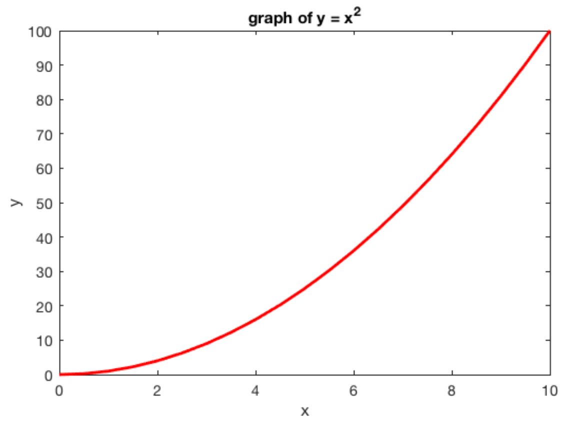
`‘LineWidth‘`	`line thickness`
`‘MarkerSize‘`	`size of marker`
`‘MarkerFaceColor’`	`solid marker color`

red line



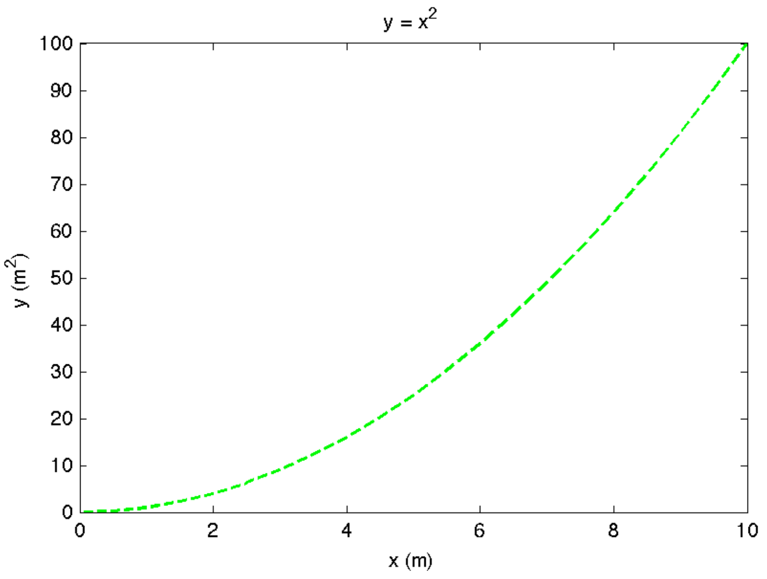
```
plot(x,y,'r-')
```

thick red line



```
plot(x,y,'r-', 'LineWidth', 2)
```

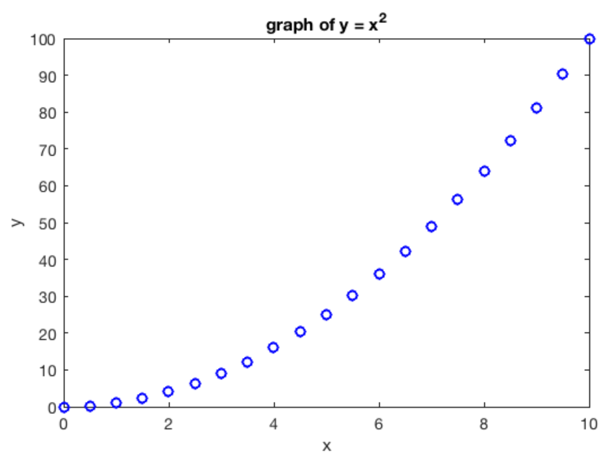
dashed green line

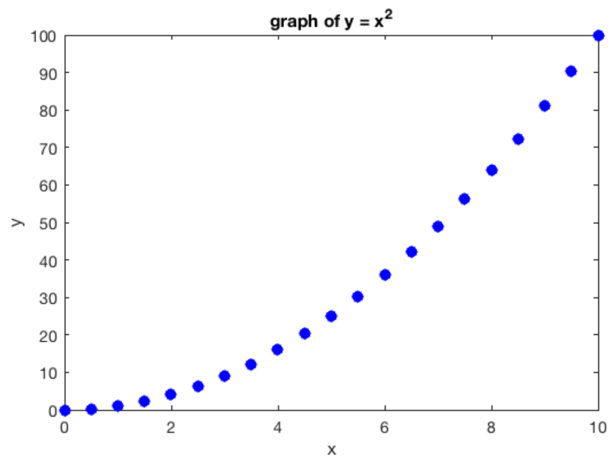


```
plot(x,y,'g--', 'LineWidth', 1.5)
```

blue circles

filled blue circles

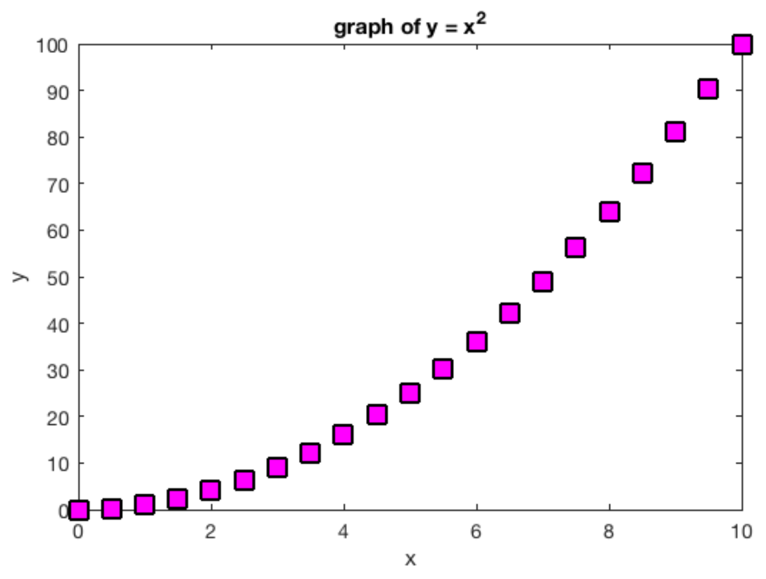




```
plot(x,y,'bo')
```

```
plot(x,y,'bo', 'MarkerFaceColor', ‘b’)
```

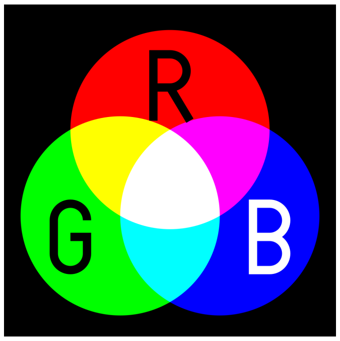
Big, magenta, filled-in squares with black borders



```
plot(x,y,'ks', 'MarkerFaceColor', ‘m’, ‘MarkerSize’, 12)
```

Custom Colors

Colors on computer screen are constructed by mixing Red, Green and Blue in different amounts:



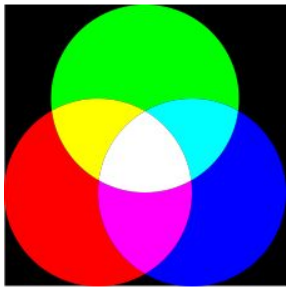
Matlab defines a 1x3 color matrix to store the RGB values.
RGB Color Matrix: `c = [R G B]`

Each RGB value is defined on the range [0-1].

Primary Colors

🟥	🟩	🟦		
[1	0	0	] = red	🟥
[0	1	0	] = green	🟩
[0	0	1	] = blue	🟦

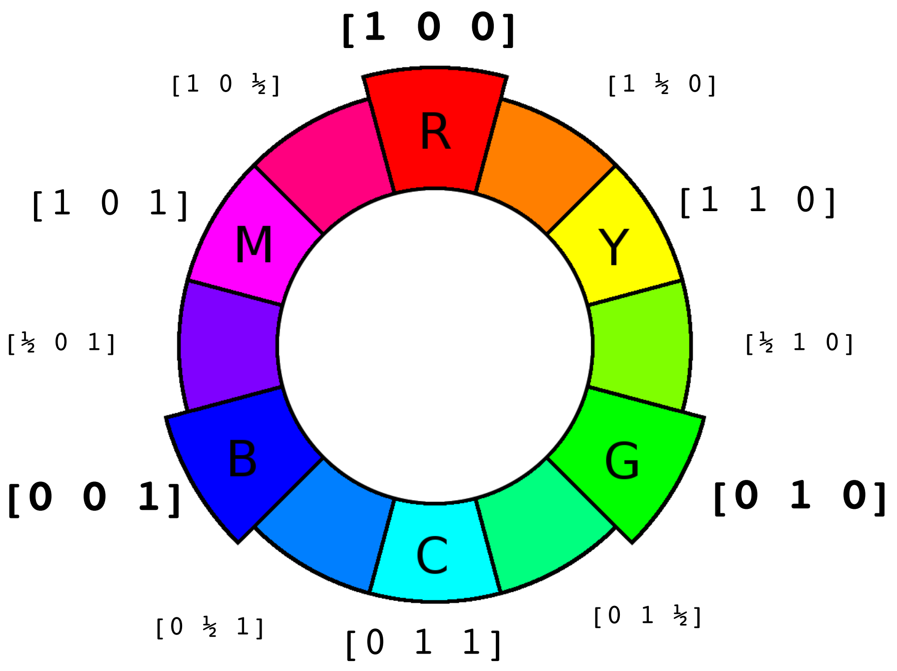
Secondary Colors



🟥	🟩	🟦		
[1	1	0	] = yellow	🟨
[0	1	1	] = cyan	🟦
[1	0	1	] = magenta	🟪

Color Wheel (Additive)

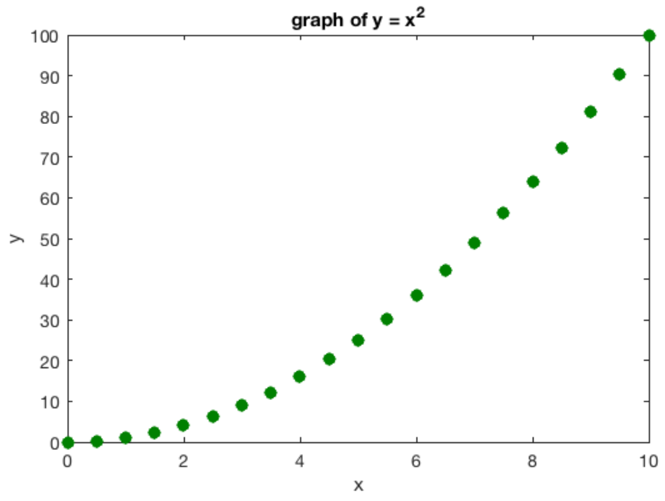
RGB Color Matrix: c = [R G B]



Gray Scales

🟥	🟩	🟦		
[1	1	1]	= white	⬜
[0.8	0.8	0.8]	= light gray	⬜
[0.5	0.5	0.5]	= middle gray	◼
[0.2	0.2	0.2]	= dark gray	⬛
[0	0	0]	= black	⬛

Custom Color: Dark Green

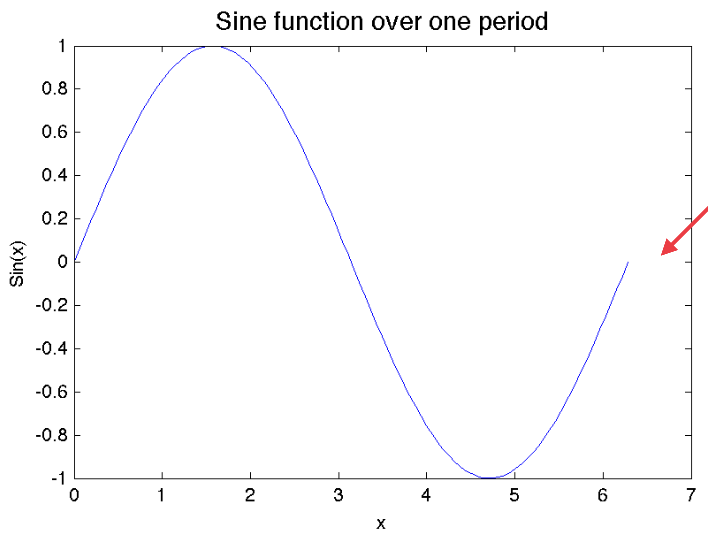


RGB Color Matrix: [R G B]

R = red (0-1)
G = green (0-1)
B = blue (0-1)

```
plot(x,y,'o', ‘Color', [0 0.5 0])
```

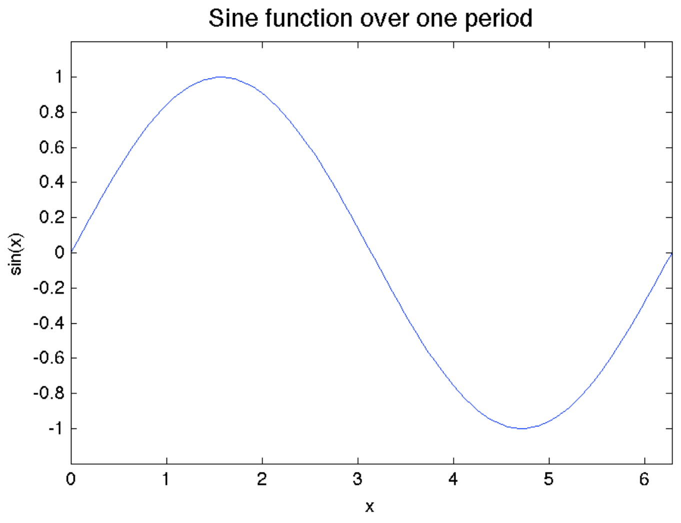
Plotting a Sine Function



gap 🙁

```
x = linspace(0,2*pi,100);     % define 100 x values from 0 to 2*pi
y = sin(x);                   % calculate sin(x)
plot(x,y)                     % plot y vs. x
```

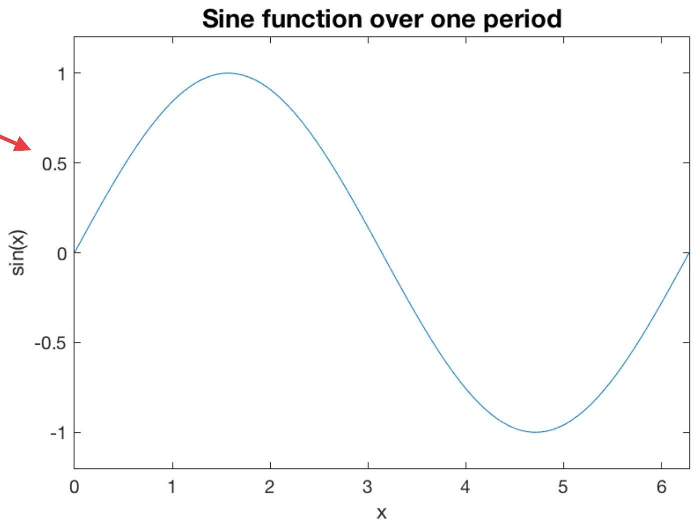
Specify plot limits (plot02A.m)



```
   x = linspace(0,2*pi,100);     % define 100 x values from 0 to 2*pi
   y = sin(x);                   % calculate sin(x)
   plot(x,y)                     % plot y vs. x
→  xlim([0 2*pi]);               % set limits on x axis
→  ylim([-1.2, 1.2]);            % set limits on y axis
```

Specify custom tick mark spacing (plot02B.m)

fewer labels



```
   x = linspace(0,2*pi,100);   % define 100 x values from 0 to 2*pi
   y = sin(x);                 % calculate sin(x)
   plot(x,y)                   % plot y vs. x
   xlim([0 2*pi]);             % set limits on x axis
   ylim([-1.2, 1.2]);          % set limits on y axis
→  yticks([-1.0:0.5:1.0]);     % tick labels every 0.5 units on y axis
```

Saving a plot to a file

I recommend saving your plots to a pdf file, and then printing that file. The following commands allow you to create a custom paper size. The plot will fill the page. You can change the width and height of the plot (in inches) by changing the `w` and `h` variables. In this example, the plot will be saved to a pdf file called `plot02.pdf`

```
w = 6;       % plot width in inches
h = 4;       % plot height in inches
set(gcf, 'PaperSize', [w h]);
set(gcf, 'PaperPosition', [0 0 w h]);
print ('-dpdf','plot02.pdf');
```

Other file types

Matlab supports saving plots using many file types. Here are a few:

Vector formats (looks sharp even when you blow it up):

```
print ('-dpdf','plot02.pdf');    % pdf file
print ('-deps','plot02.eps');    % encapsulated B&W postscript
print ('-depsc','plot02.eps');   % encapsulated color postscript
```

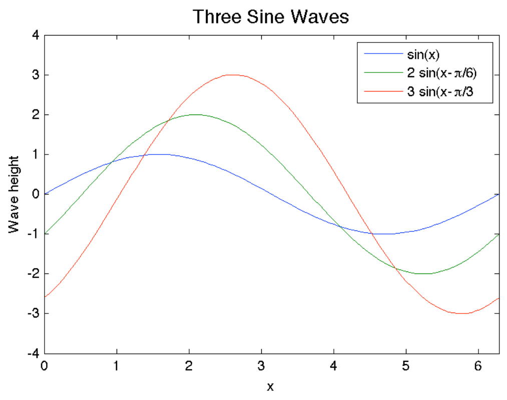
Bitmap formats (looks “pixelated” if blow up too big):

```
print ('-dtiff','plot02.tiff','-r300');   % tiff file
print ('-djpeg','plot02.jpeg','-r300');   % jpeg file
print ('-dpng','plot02.png','-r300');     % png file
```

Bitmap formats take a resolution option that specifies the number of dots per inch. In the above examples `'-r300'` uses the recommended value of 300 dots per inch.

Using 'hold on' to overlay graphs

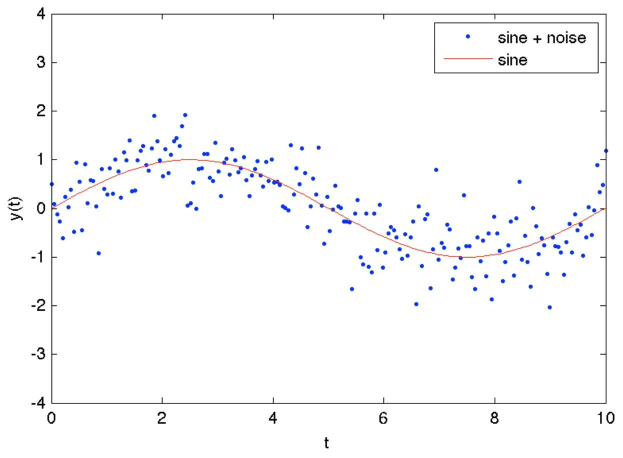
```
x = linspace(0,2*pi,100);
y = sin(x);
y2 = 2 * sin(x - pi/6);
y3 = 3 * sin(x - pi/3);

plot(x,y)

hold on
plot(x,y2)
plot(x,y3)

legend('sin(x)', '2 sin(x-\pi/6)','3 sin(x-\pi/3')
```



"hold on" lets you plot multiple plots on the same graph without overwriting previous plots. Try commenting out the "hold on" command and see what happens.

Also note the use of the legend() command.

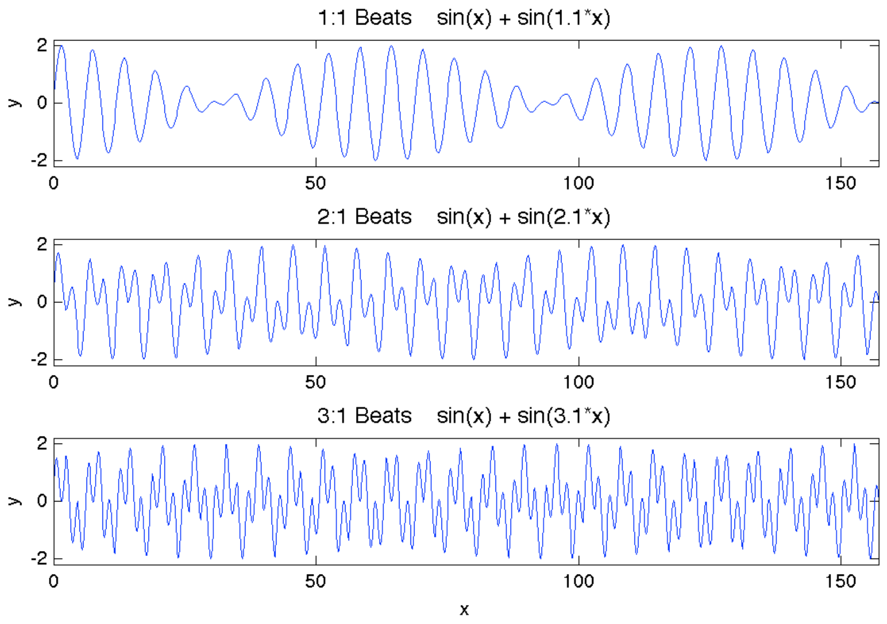
Plotting a smooth curve through noisy data (plot07.m)



You will often need to compare a theoretical curve against noisy data. Usually you plot the theoretical model as a smooth curve and the data as points. This example is a bit of a cheat: we "create" the data points by just adding Gaussian noise to a sine wave.

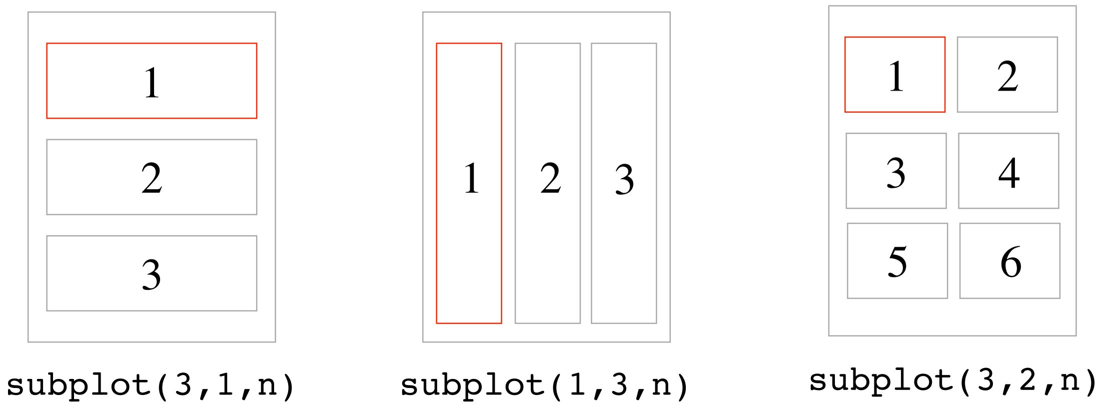
```
period = 10;                   % define the period of the signal
t = linspace(0,period,200);    % 200 data points evenly spaced over 1 period
hold all                       % allow multiple plots on same graph

% create a sine wave and add Gaussian noise to it (plot as blue dots)
y2 = sin(2*pi/period*t) + 0.5*randn(1,length(t));
plot(t,y2,'bo', 'MarkerSize',2, 'MarkerFaceColor', 'b');

% create a sine wave with no noise and plot as red curve
y = sin(2*pi/period*t);
plot(t,y,'r-')
```

Putting multiple plots on one page (plot08.m)



Here we use the `subplot(3,1,n)` command to create a 3x1 grid of plots, where n = 1, 2, and 3 for the top, middle and bottom plots respectively.

Use of `subplot()` command



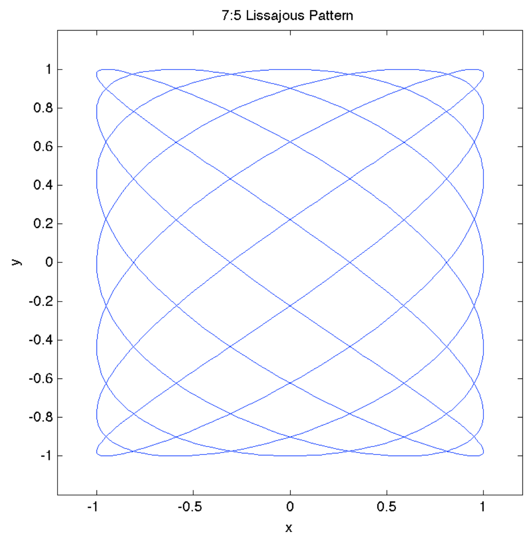
The command `subplot(ny, nx, n)` creates a grid of plots with `ny` rows and `nx` columns. The final parameter `n` selects which position to plot to (as indicated by the numbers on the diagrams). In each case, the red box shows the plot position when n = 1.

Putting multiple plots on one page (plot08.m)

Here's the code that generated the plot of the three waves (2 slides ago):

```
x = linspace(0,50*pi,1000);  % create 1000 evenly spaced x values from 0 to 50*pi

% two sine waves beating near 1:1 "resonance"
y = sin(x)+sin(1.1*x);
subplot(3,1,1);             % three horizontal panels, plot top one first
plot(x,y)
axis([0 50*pi -2.2 2.2])
ylabel('y')
title('1:1 Beats    sin(x) + sin(1.1*x)', 'FontSize',12)

% two sine waves beating near 2:1 "resonance"
y = sin(x)+sin(2.1*x);
subplot(3,1,2);             % now make a plot in the middle panel
plot(x,y)
axis([0 50*pi -2.2 2.2])
ylabel('y')
title('2:1 Beats    sin(x) + sin(2.1*x)', 'FontSize',12)

% two sine waves beating near 3:1 "resonance"
y = sin(x)+sin(3.1*x);
subplot(3,1,3);             % this plot will be in the bottom panel
plot(x,y)
axis([0 50*pi -2.2 2.2])
title('3:1 Beats    sin(x) + sin(3.1*x)', 'FontSize',12)
```

Plotting parametric curves: Lissajous Figures (plot09.m)



```
kx = 7;      % wavenumber of x wave
ky = 5;      % wavenumber of y wave

% define 1000 x values from 0 to 2*p
t = linspace(0,2*pi,1000);

x = sin(kx*t);            % x wave
y = sin(ky*t+pi/2);       % y wave

plot(x,y,'b-')            % plot blue curve
axis equal                % force axes to
                          % have same scaling

xlim([-1.2 1.2]);         % set x axis range
ylim([-1.2 1.2]);         % set y axis range

% label axes and add title
xlabel('x');
ylabel('y');
str = sprintf('%i:%i Lissajous Pattern',kx,ky);
title(str);
```

x(t) and y(t) are parametric functions that depend on t. The `axis equal` command scales the plot so unit length in x and y are equal

Reading Data From a File (plot11A.m)

Here's a data file called `run1.txt` that contains (x,y) values and errors in the y coordinate:

x	y	error
0	0.7	0.9
0.5	0.3	0.5
1	1.6	0.7
1.5	2.5	0.8
2	1.5	0.6
2.5	3.1	1.5
3	4.5	0.8
3.5	3.6	0.99
4	5.7	0.9
4.5	4.3	1.3
5	6.8	0.8
5.5	7.9	1.2
6	5.7	1.4
6.5	7.4	0.9
7	8.3	0.8
7.5	7.6	1.1

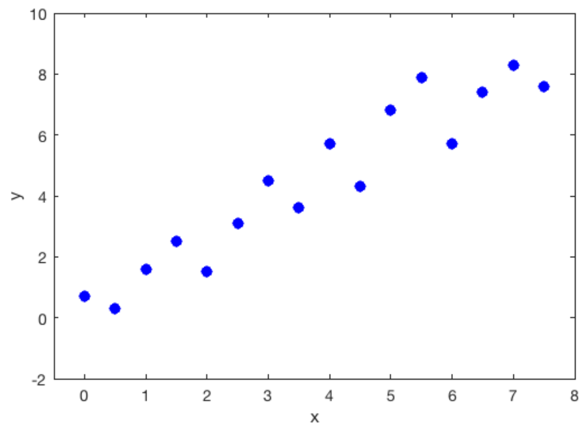
We read in the data like this:

```
data = load('run1.txt');
x = data(:,1);
y = data(:,2);
err = data(:,3);
```

Reading Data From a File (plot11A.m)

We can plot the data points like this:

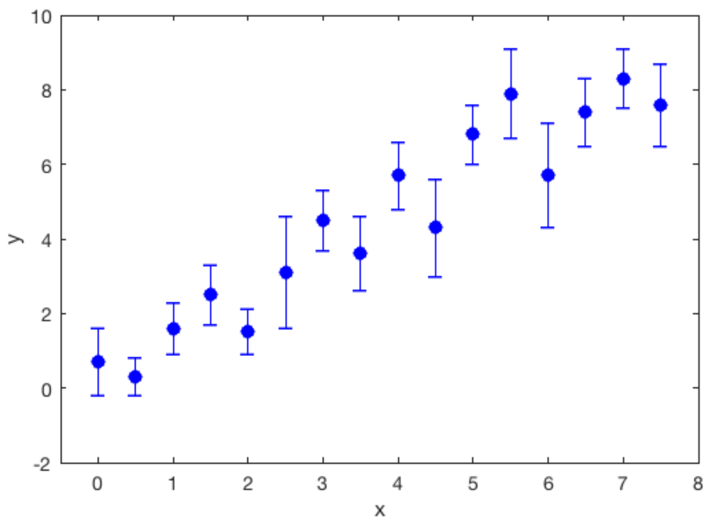
```
plot(x,y,'bo','MarkerFaceColor','b')

% Label axes
xlabel('x')
ylabel('y')

% Set axis limits for plot
xlim([-0.5 8]);
ylim([-2 10]);
```



Reading Data From a File (plot11B.m)

We can add error bars by replacing the `plot()` command with the `errorbar()` command.

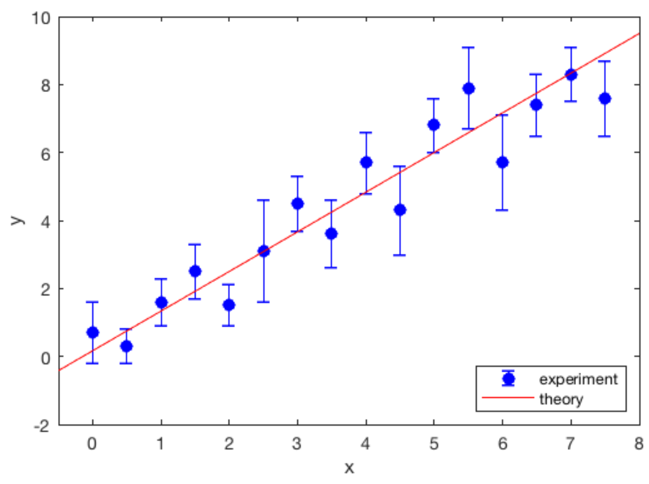
```
errorbar(x,y,'bo','MarkerFaceColor','b')
```



Reading Data From a File (plot11C.m)

Finally, let’s add a red reference line to compare our data with a theoretical linear relationship.

```
  errorbar(x,y,'bo','MarkerFaceColor','b')
→ hold on
→ plot([-1 8], [-1 9.5], 'r-')
```


y
x
experiment
theory