2D Graphs in MatLab

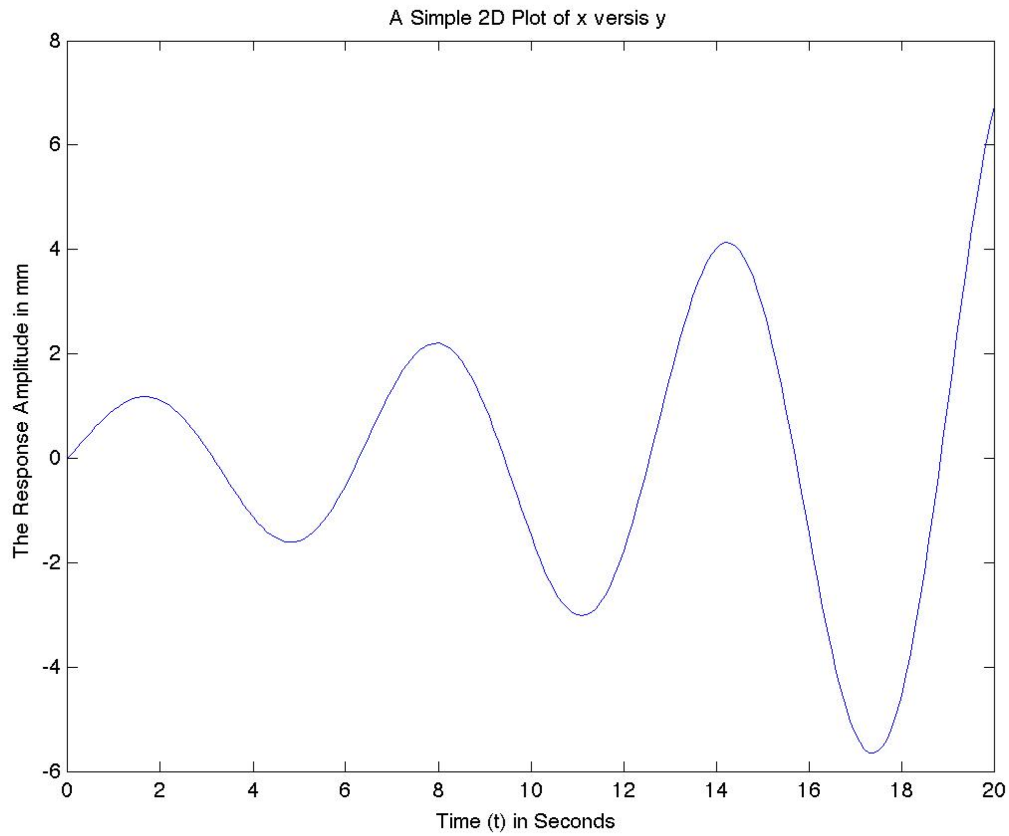
The plot function

- If a variable `ydata` has n values corresponding to n values of variable `xdata`, then `plot(xdata,ydata)` produces a graph with `xdata` on the horizontal axis and `ydata` on the vertical axis.
- Consider the following MatLab code using `plot` in file **L07plot0.m**:

```
% create vector x
x = 0:0.1:20;
% calculate y
y = exp(0.1*x).*sin(x);
```

```
% plot x vs. y
plot(x,y)
% label x axis
xlabel('Time (t) in Seconds')
% label y axis
ylabel('The Response Amplitude in mm')
% put a title
title('A Simple 2D Plot of x versis y')
% file L07plot0.jpg
print('L07plot0.jpg','-djpeg');
```

It generates the plot shown below:



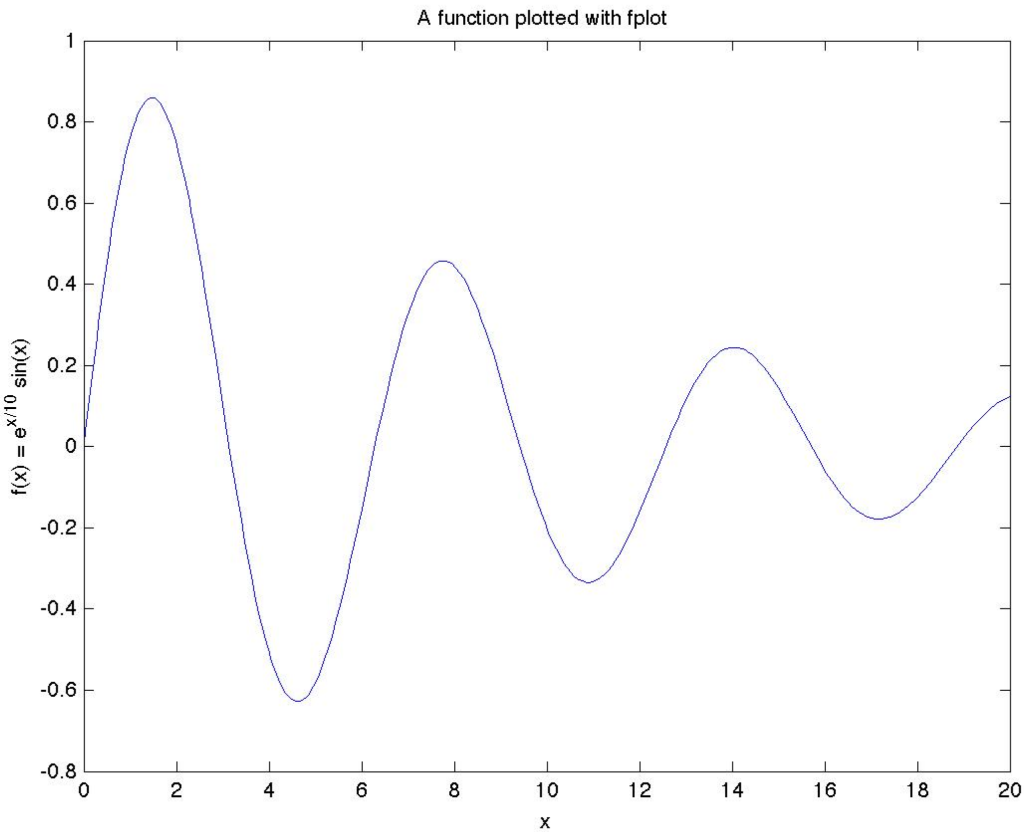
Simple 2D plot of $f(x) = e^{t/10}\sin(t)$ using **plot**.

- If a figure window already existed, this plot command would draw this figure over it. To keep the original figure contents it is necessary to use a `figure` command first. A little later there is an example.

- Consider using `fplot` in **L07plot1.m** to plot this function:

```
fplot('exp(-0.1*x).*sin(x)',[0,20])
% Note the use of latex to do the
% equation for the y axis
xlabel('x'),ylabel('f(x) = e^{x/10} sin(x)')
title('A function plotted with fplot')
print('L07plot1.jpg','-djpeg')
```

The 1^{st} argument to `fplot` is the function to be plotted while the 2^{nd} argument gives the range for the `x` values of the function. The `print` command generates a `jpg` file of the figure displayed in the figure window. The figure below shows the graph produces by these commands.

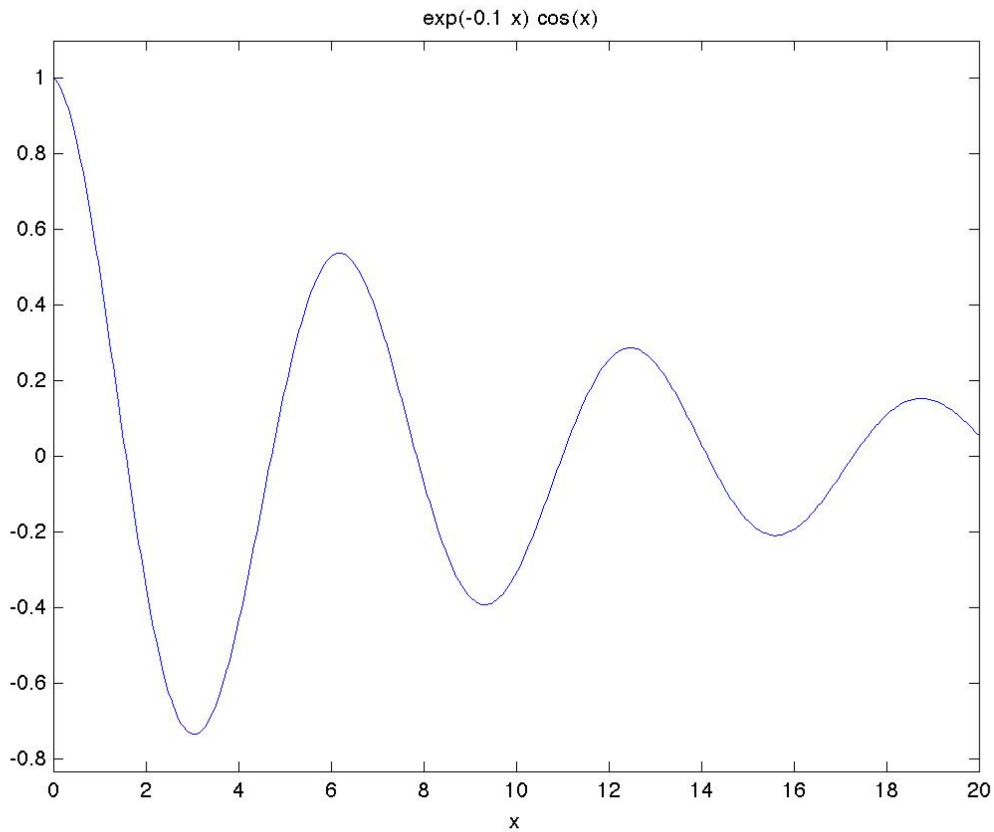


Simple 2D plot of $f(x) = e^{t/10}\sin(t)$ using **fplot**.

- `ezplot` can be used to plot $f(x) = e^{-x/10}\cos(x)$ in a very simple way (using all default values). MatLab code in **L07plot.m** to do this:

```
ezplot('exp(-0.1*x).*cos(x)',[0,20])
print L07plot2.jpg -djpeg
```

The figure below shows the output graph.



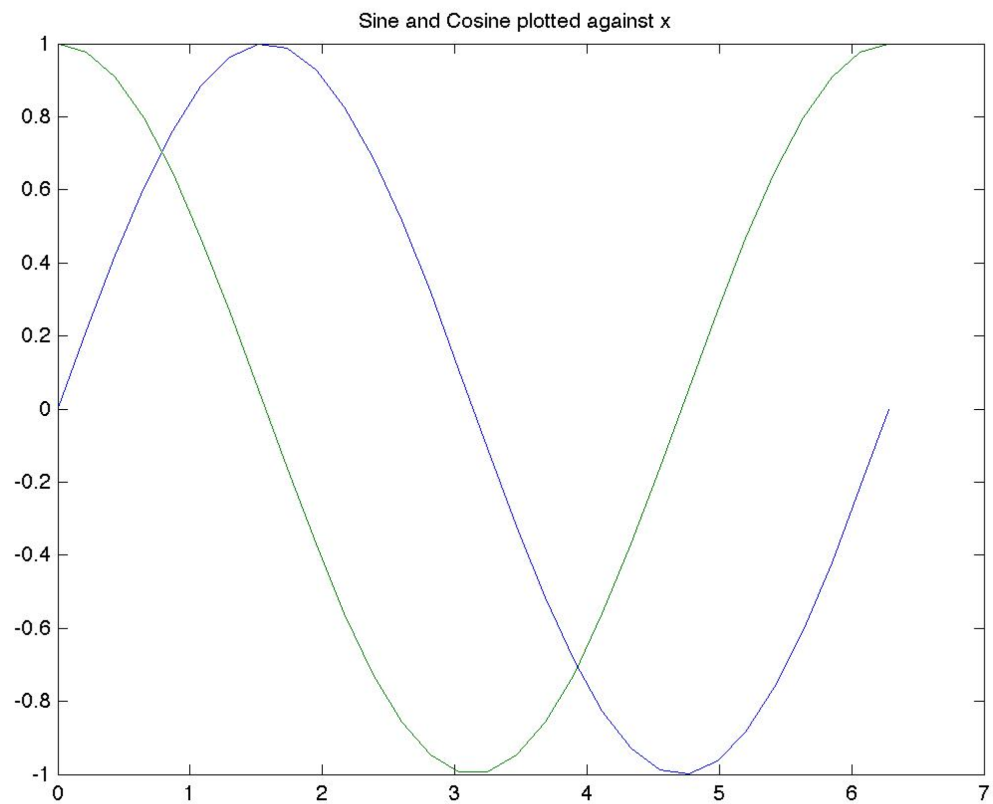
Simple 2D plot of $f(x) = e^{t/10} \cos(t)$ using **ezplot**.

- Consider multiple graphs plotted at the same time (as in **L07plotA.m**):

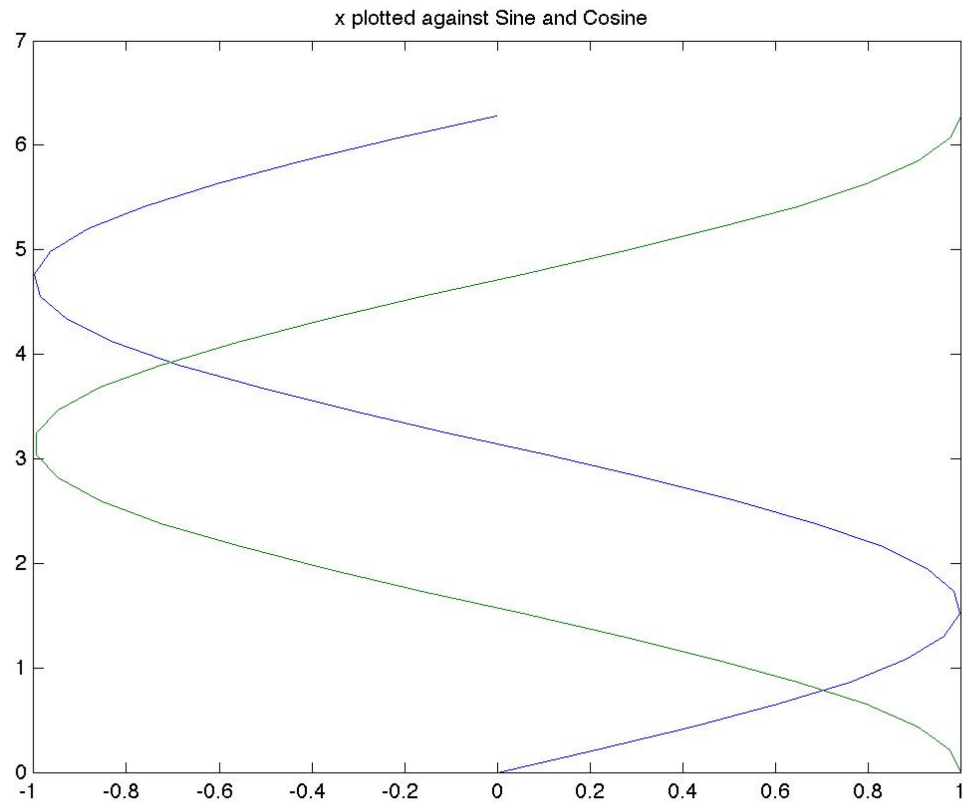
```
% Generate 30 values between 0 and 2*pi
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
% create W as a 30*2 array
% 1st column are the sine values
% 2nd column are the cosine values
W=[y;z];
plot(x,W)
title('Sine and Cosine plotted against x');
print L07plotA1.jpg -djpeg
```

```
% Use a new figure window
figure;
plot(W,x)
title('x plotted against Sine and Cosine');
print L07plotA2.jpg -djpeg
```

- The first figure below shows the sine and cosine values (stored as a 2D array `W` with the 1^{st} column as the sine values and the 2^{nd} column as the cosine values) plotted against `x`. Note the first curve is plotted in default colour **blue** and the second curve is plotted in default colour **green**.
- The next figure below shows `x` plotted against these sin and cosine values (hence, the graphs are sideways!).



Simple 2D multiple plot of sin and cosine values against 30 equally spaced values between 0 and 2π.



Simple 2D multiple plot of 30 equally spaced values from 0 to 2π against their sin and cosine values.

Customizing Graphs

- Notice that in the figures for the sine and cosine, MatLab chose solid linetype and the colours blue and green for the plots.

- You can customize your plots in MatLab to use different colours, markers and linestyles by using a 3^{rd} argument in `plot`. This 3^{rd} argument is a character string with one or more characters from the Table below:

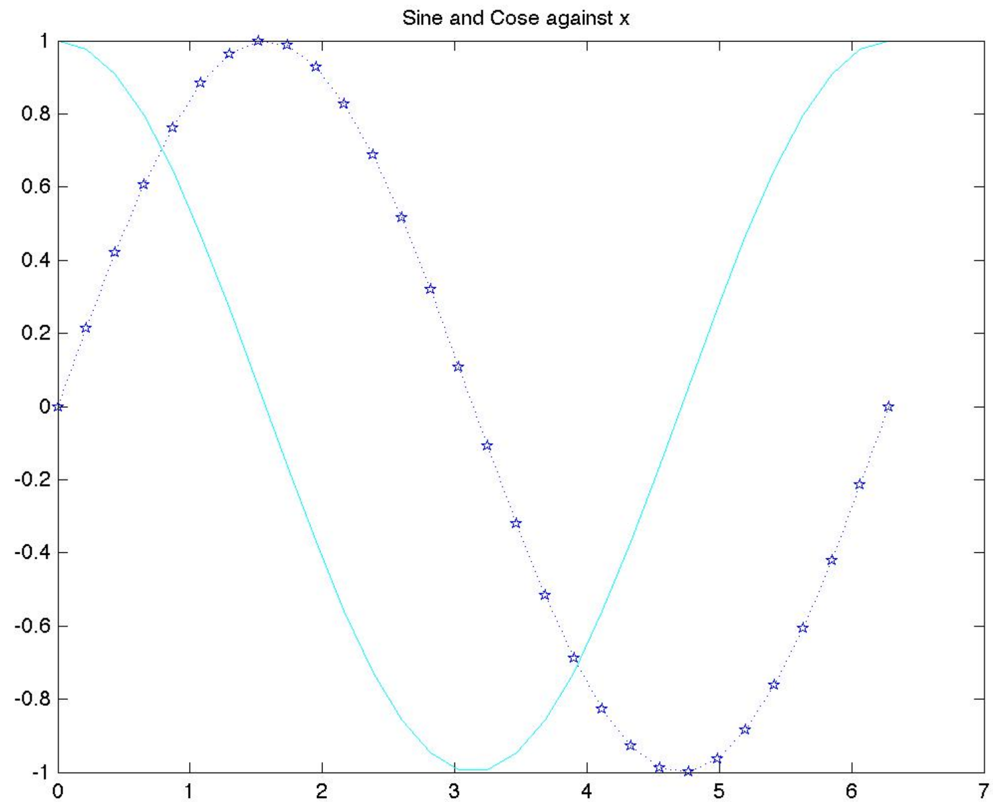
Symbol	Colour	Symbol	Marker	Symbol	Linestyle
b	Blue	.	Point	-	Solid line
g	Green	o	Circle	:	Dotted line
r	Red	x	Cross	-.	Dash-dot line
c	Cyan	+	Plus sign	–	Dashed line
m	Magenta	*	Asterisk	none	no line
y	Yellow	s	Square		
k	Black	d	Diamond		
w	White	v	Triangle (down)		
		^	Triangle (up)		
		<	Triangle (left)		
		>	Triangle (right)		
		p	Pentagram		
		h	Hexagram		

Setting colours, markers and linetypes

- The following MatLab code gives the **L07plotB.m** code:

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(y);
plot(x,y,'b:p',x,z,'c-');
title('Sine and Cose against x');
print L07plotB.jpg -djpeg
```

produces:



30 equally spaced values from 0 to 2π against their sin and cosine values. The sine plot is dotted blue with pentagram markers ('b:p') while the cosine plot is a solid cyan line ('c-').

Grids, Axes and Labels

- The `grid on` commands add grid lines to the current plot at the tick marks. By default MatLab has `grid off`.
- Normally, a graph is enclosed by solid lines called an **axes box**. Use `box off` to turn this off and `box on` to turn it on again.
- `xlabel` and `ylabel` can be used to label the x and y axes. The `title` command allows a title to be added above the plot.
- An example, consider the MatLab code for the plot we made earlier with some changes (as shown in **L07plotC.m**):

```
close all
```

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
W=[y;z];
plot(x,W)
box off
xlabel('X','Fontsize',16);
ylabel('Y and Z','Fontsize',16);
% 2 line title
grid on
text(2.5,0.7,'sin x','Fontsize',14);
text(1.1,0.7,'cos x','Fontsize',14);
```

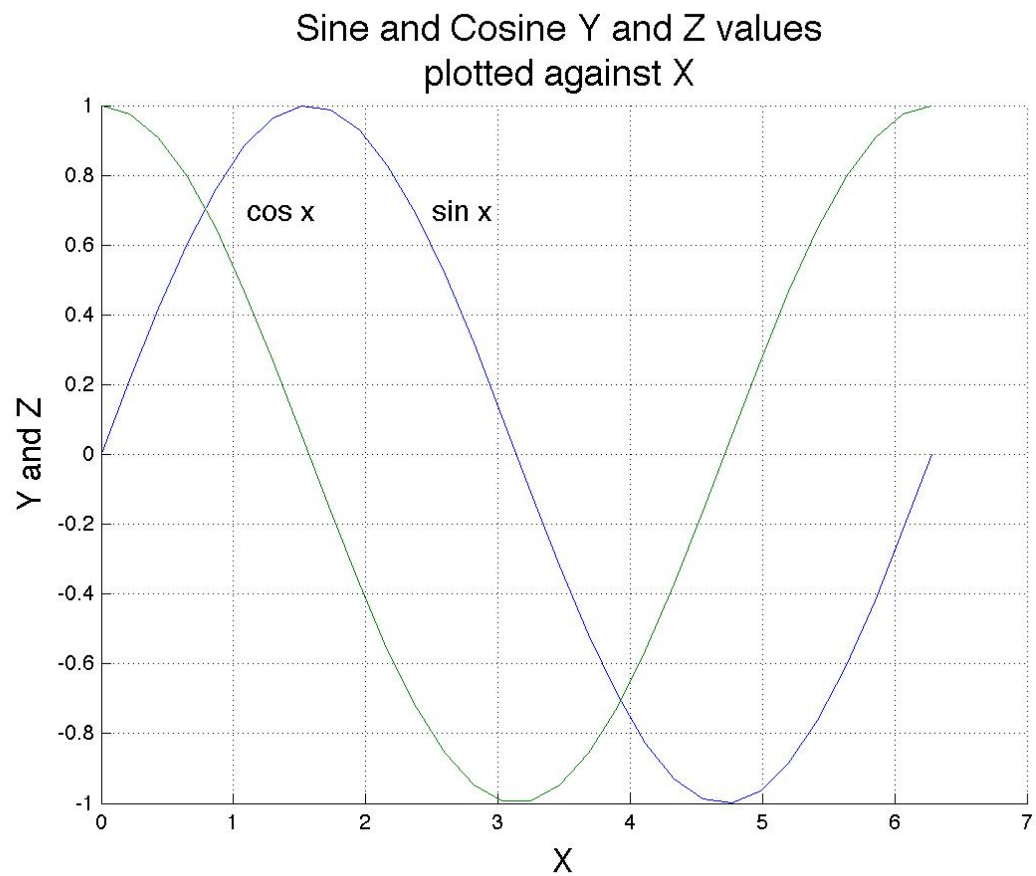
```
title({'Sine and Cosine Y and Z values';...
       'plotted against X'},'Fontsize',18);
print L07plotC.jpg -djpeg
```

- Note that we use `close all` to close all windows currently open in this session.
- We control the fontsize in `title`, `xlabel`, `label` and `text` using attribute `Fontsize` and values 18, 16 and 14.
- Note the grid printed as dotted lines. `box off` means there are no top or right solid lines on the graph.
- We print two (or more) strings (1 per line) in the `title` command (or in an `xlabel`, `ylabel` or `text` commands) by enclosing these strings

separated by semi-colons with curly braces. These curly braces tell MatLab each string is an element of a cell array and to print each element as a separate line. The 2D positions of the text strings have to be determined by trial and error with a good initial guess possible is you know the `x` abd `y` axes limits.

- The `...` allows a MatLab command to broken at a white space and continued on the next line. This is good for preventing long command lines from wrapping around onto the next line.

- The figure below shows what is plotted:



Sine and Cosine X and Y values plotted against X values, box off, grid on, double string title, x and y labels, and text strings (all with different fontsizes).

- MatLab gives you control over the appearance of your plot axes. A few options available:

  1. `axis[(xmin xmax ymin ymax])` - control the axes limits - good when making comparisons across multiple plots.
  2. `axis auto` - lets MatLab determine axis defaults.
  3. `axis on` - Turn on axis labelling, tic marks and background.
  4. `axis off` - Turn off axis labelling, tic marks and background.

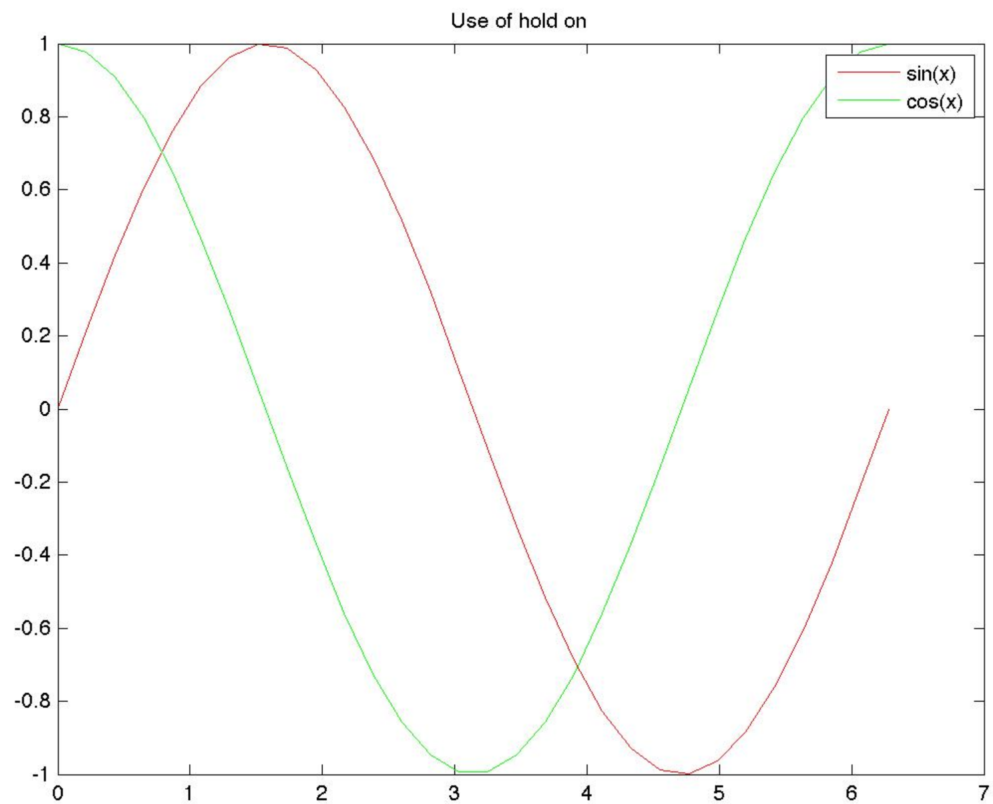
- Another way to make the multiple sine and cosine plots would be to use `hold on` (as shown in **L07plotD.m**):

```
x=linspace(0,2*pi,30);
y=sin(x);
```

```
% sin(x) as red line
plot(x,y,'r');
z=cos(x);
hold on
% cos(x) as green line
plot(x,z,'g');
legend('sin(x)','cos(x)');
title('Use of hold on');
print L07plotD.jpg -djpeg
```

which generates the following graph:

- Without `hold on` the sine graph would have been overwritten by the cosine graph.



Use of hold on.

- `legend` allows each plot line to be associated with a character string. In this example, a red solid line is associated with 'sin(x)' (the first curve printed) while a green solid line is associated with 'cos(x)' (the second curve printed).

Subplots

- A single MatLab window can hold more than 1 plot. `subplot(n,m,p)` divides a window up into an `m-by-n` matrix and indicates the $\mathtt{p}^t h$ subplot is active. For `m=2` and `n=2` we have a 2×2 array with 4 `p` values:

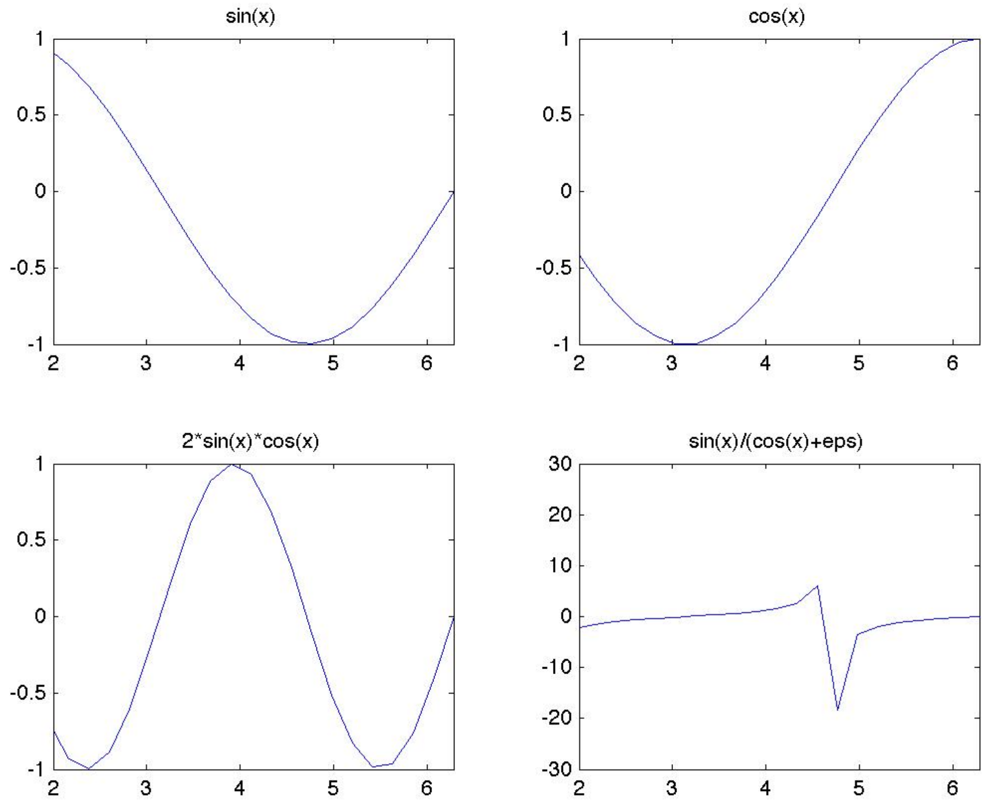
$$\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}$$

- A MatLab example shown in **L07plotE.m**:

```
x=linspace(0,2*pi,30);
y=sin(x);
z=cos(x);
a=2*sin(x).*cos(x);
```

```
% eps - machine epsilon
b=sin(x)./(cos(x)+eps);
% upper left of 2 by 2 subplots
subplot(2,2,1)
plot(x,y);
axis([2 2*pi -1 1]);
title('sin(x)');
% upper right of 2 by 2 subplots
subplot(2,2,2)
plot(x,z);
axis([2 2*pi -1 1]);
title('cos(x)')
```

```
% lower left of 2 by 2 subplots
subplot(2,2,3)
plot(x,a);
axis([2 2*pi -1 1]);
title('2*sin(x)*cos(x)');
% lower right of 2 by 2 subplots
subplot(2,2,4)
plot(x,b);
axis([2 2*pi -30 30]);
title('sin(x)/(cos(x)+eps)');
print L07plotE.jpg -djpeg
```



4 subplots of $\sin(x)$, $\cos(x)$, $2\sin(x)(cos(x)$ and $\sin(x)/(\cos(x)+eps)$.

- `eps` or `eps('double')` is is the distance from 1.0 to the next largest double-precision number, which is `2^(-52)` or about 2.2204e-16 in base 10. `eps('single')` is the distance from 1.0 to the next largest single-precision number, which is `2^-23` or about 1.1921e-07 in base 10. Adding `eps` prevents division by zero from happening when cos(x) is zero.

- `xlabel`, `ylabel`, `title`, `text`, `hold on`, `grid`, etc apply to the active subplot.

- A `dramnow` command forces MatLab to draw all figures computed so far (MatLab may have buffered some).

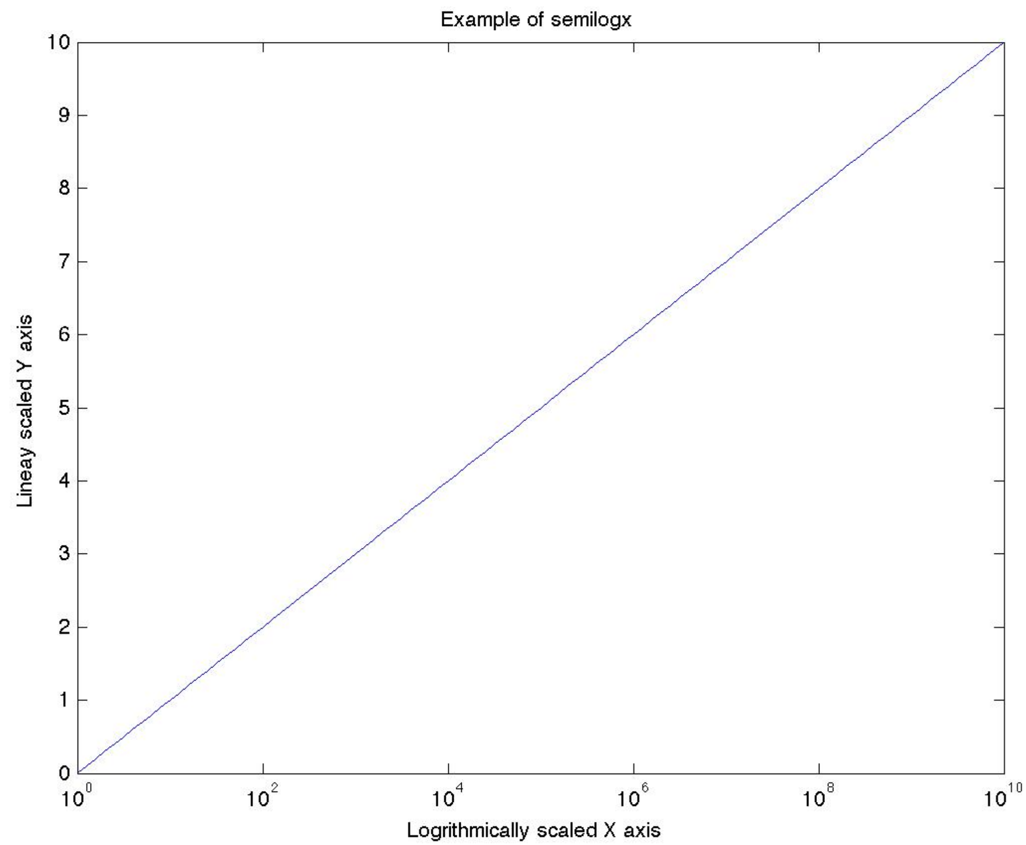
Log Plots

- We can use `semilogx` to plot a logarithmically scaled x-axis, `semilogy` to plot a logarithmically scaled y-axis and `loglog` to plot a graph with both axes logarithmically scaled.

- For example (as shown in **L07plotF.m**):

```
x = 0:0.1:10;
semilogx(10.^x,x);
xlabel('Logrithmically scaled X axis');
ylabel('Lineay scaled Y axis');
title('Example of semilogx');
print L07plotF.jpg -djpeg
```

produces the graph:



Logarithmically scaled x axis example.

Stacked Area Plots

- The `area` function can be used to build stacked plots. `area(x,y)` and `plot(x,y)` are the same for vectors `x` and `y` except except the area under the plot is filled with colour when using `area`.

- To stack areas, use `area(X,Y)`, where `Y` is a matrix and `X` is a matrix or vector whoses length equals the number of rows in `Y`. If `X` is ommitted, `X=1:size(Y,1)` is used.

- If `Y` consists of n vectors of data, then the i^{th} vector is added to the first $i-1$ before it before being plotted. Each area is coloured differently than the other areas.

- Consider the following MatLab code in **L07stacked_area_plot.m**r:

```
% stacked area plot
% x axis 2011:1:2020
X=linspace(2011,2020,10);
% y1 axis 20000 to 300000
% y2 axis 155000 50 1000000
% Make column vectors
y1=linspace(50000,300000,10)';
y2=linspace(100000,1000000,10)';
Y=zeros(size(y1,1),size(y1,2)*2);
Y(:,1)=y1;
Y(:,2)=y2;
% y2 is stacked on top of y1
```

```
area(X,Y);
colormap('summer');
set(gca,'XLim',[2011 2020])
set(gca,'YLim',[0 1800000])
set(gca,'YTick',0:200000:1400000)
set(gca,'YTickLabel',{'','200000',...
        '400000','600000','800000',...
        '1000000','1200000','1400000'})
stg=['\fontsize{20}\color{black}\bf '...
     '1,000,000 more jobs than students '...
     'by 2020'];
text(2011,1600000,stg);
```

```
stg=['\fontsize{18}\color{black}\bf '...
     '1.4 million'];
text(2017,700000,stg);
stg=['\fontsize{18}\color{black}\bf '...
     'computing jobs'];
text(2017,550000,stg);
stg=['\fontsize{16}\color{black}\bf '...
     '400,000 computer science students'];
text(2014,65000,stg);
stg=['\fontsize{24}\color{black}\bf '...
     '$500 billion'];
text(2012,1200000,stg);
```

```
stg=['\fontsize{24}\color{black}\bf '...
     'opportunity'];
text(2012,1100000,stg);
% draw a line with coordinates
% (x1,y1) and (x2,y2)
% use line([x1 x2],[y1 y2])
line([2014 2016],[1000000 600000],...
     'linewidth',2);
print L07stacked_area_plot.jpg -djpeg
```

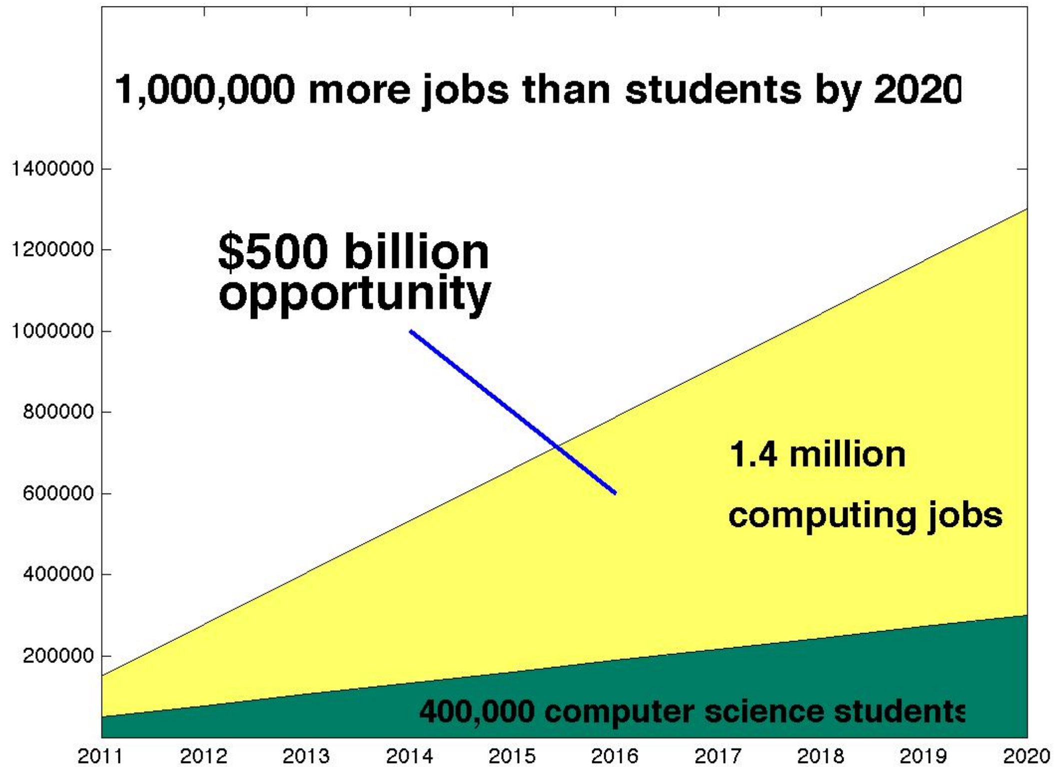
- Note the use of latex commands `fontsize` and `color` to control the size and colour of text. `\bf` boldfaces the text.

- The x coordinates are the years 2011 to 2020 while the y coordinates the numbers 0 to 1800000. The data only goes to 1400000, the extra space is for the title.

- `gca` is the MatLab function that gets the current axis handle (pointer). Then the set commands is used to set the `XLim`, `YLim`, `YTick` and `YTickLabel` properties of the current axis. We use the default `XTick` and `XTickLabel` properties. It probably wasn't necessary to explicitly set the `XLim` property as this is probably the default value. In the case of the y axis, the ticks are numbers that would have been printed in scientific notation instead of as whole integers (if we had not specified the integers in the `set` for the `YTickLabel` property as above).

- The blue line as drawn using `line([x1 x2],[y1 y2])` (it is blue because the is the first colour MatLab selects).

- This code produces the following plot (**stacked_area_plot.jpg**):



Some Computer Science propaganda: Number of computing jobs versus the number of computer science students for 2011 to 2012. From http://www.code.org/stats.

Pie Charts

- Standard pie charts can be created using `pie(a,b)`, where `a` is vector of values and `b` is an optional argument describing a slice or slices to be pulled out from the pie chart. `pie3` gives a 3D effect.
- A standard pie chart can be produces by the following MatLab code (in **L07pie1.m**):

```
a=[0.024 0.976];
pie(a,{'2.4%','97.6%'});
colormap summer
stg='\fontsize{36}\color{red} Students';
text(-0.5,-0.3,stg);
```

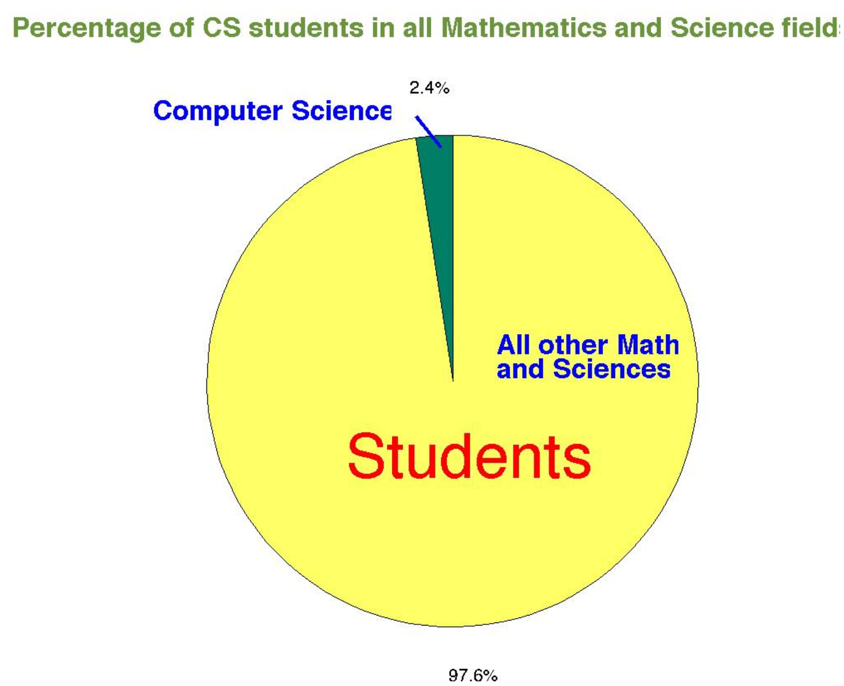
```
stg='\fontsize{16}\color{blue}\bf All other Math';
text(0.15,0.15,stg);
stg='\fontsize{16}\color{blue}\bf and Sciences';
text(0.15,0.05,stg);
stg='\fontsize{16}\color{blue}\bf Computer Science';
text(-1.25,1.1,stg);
line([-0.15 -0.05],[1.075 0.95],'linewidth',2);
title({['\fontsize{16}\color{darkgreen}\bf' ...
        'Percentage of CS students in all ' ...
        'Mathematics and Science fields'],'   '});
print L07pie1.jpg -djpeg
```

- A pie chart with a part “pulled” out (highlighted) as shown by the following code (in **pie2.m**):

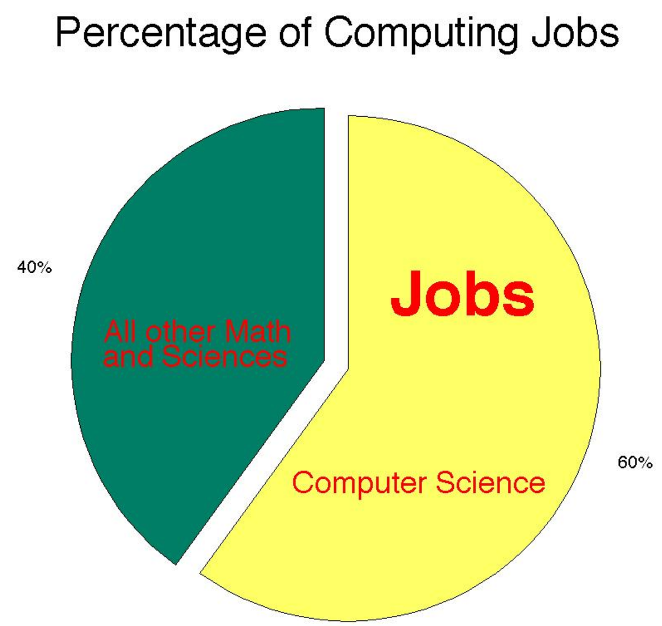
```
a=[0.4 0.6];
explode=(a==min(a));
pie(a,explode,{'40%','60%'});
colormap summer
% coordinates found by trial and error
stg='\fontsize{36}\color{red}\bf Jobs';
text(0.1,0.3,stg);
stg='\fontsize{18}\color{red} All other Math';
text(-1.0,0.15,stg);
stg='\fontsize{18}\color{red} and Sciences';
```

```
text(-1.0,0.05,stg);
stg='\fontsize{18}\color{red} Computer Science';
text(-0.25,-0.45,stg);
title('\fontsize{24} Percentage of Computing Jobs');
print L07pie2.jpg -djpeg
```

The following pie charts are produced:



(a)



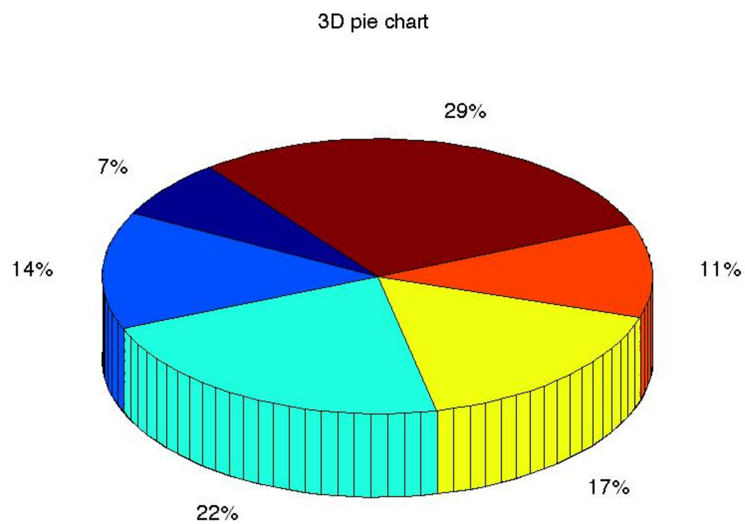
(b)

More Computer Science propaganda: (a) Percentage of computer science students versus science and mathematics students and (b) Percentage of jobs for computer science students and science and mathematics students. From http://www.code.org/stats.

- Finally, following MatLab code (in file **L07plotH.m**) produces:

```
a=[0.5 1 1.6 1.2 0.8 2.1];
pie3(a);
title('3D pie chart');
print L07plotH.jpg -djpeg
% Sum a values
s=sum(a(:));
% print out normalized rounded ratios
% these appear on the pie chart
a=cast(a/s*100,'int32')
```

produces the graph:



3D pie chart.

- `a` prints with values:

```
a = 7  14  22  17  11  29
```

which are exactly the percentages on the pie chart.

Filled Polygons

- Polygons can be filled with a colour using `fill`. The command `fill(x,y,'color')` fills the polygon defined by column vectors `x` and `y` with colour `color`.
- Consider the following MatLab code (in file **L07stop_sign.m**):

```
% 8 points on circle to
% make the stop sign polygon
t=(1:2:15)'*pi/8;
x=sin(t);
y=cos(t);
% fill the polygon with red
```

```
fill(x,y,'red')
axis square off
% print stop in white
text(0,0,'STOP',...
     'Color',[1 1 1],...
     'FontSize',80,...
     'FontWeight','bold',...
     'HorizontAlalignment','center');
title('Stop Sign');
print L07stop_sign.jpg -djpeg
```

- The following figure is printed (with STOP in black for some reason, although it is white when run in MatLab?):

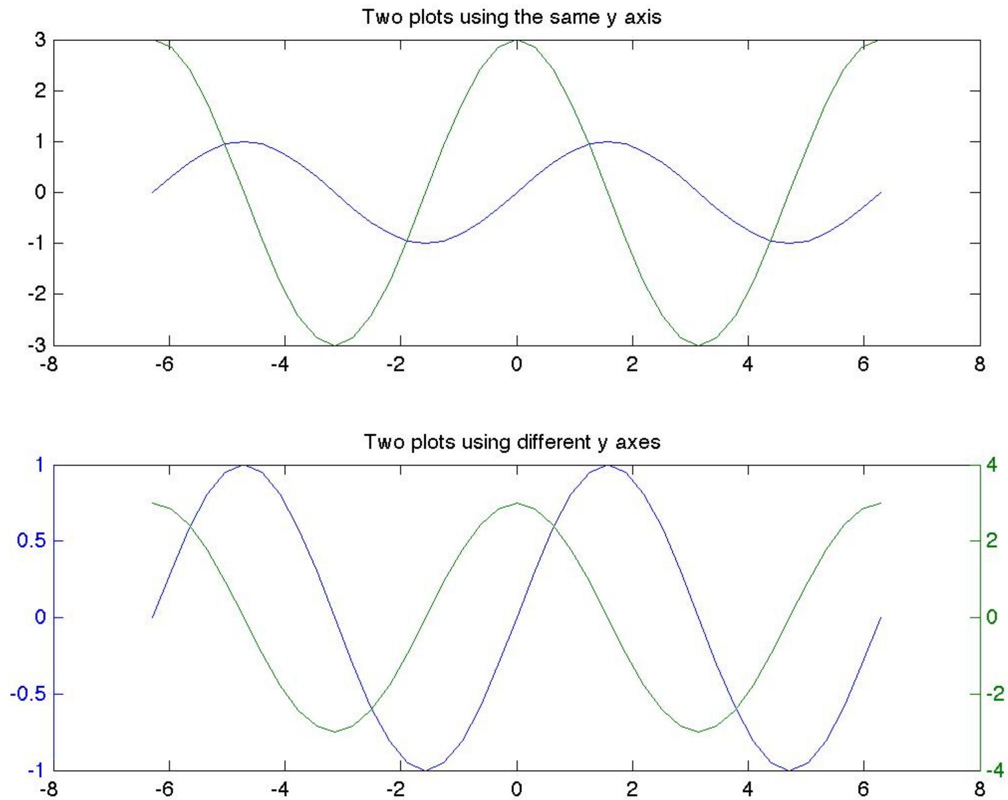
A “stop sign” filled polygon.

Plots using the Same or Different y Axes

- Sometimes we may want to plot two different functions on the same x axis but using different y axes.
- The MatLab code (in file **L07plotI.m**) to illustrate this is given as:

```
x=-2*pi:pi/10:2*pi;
y=sin(x);
z=3*cos(x);
subplot(2,1,1), plot(x,y,x,z);
title('Two plots using the same y axis');
subplot(2,1,2), plotyy(x,y,x,z);
title('Two plots using different y axes');
```

and produces the following output:



Two plots on the same y axis and on different axes.

- Note that the 2 plots of the first graph have the same y axis but the second plot has 2 different y axes.

Bar and Stairs Plots

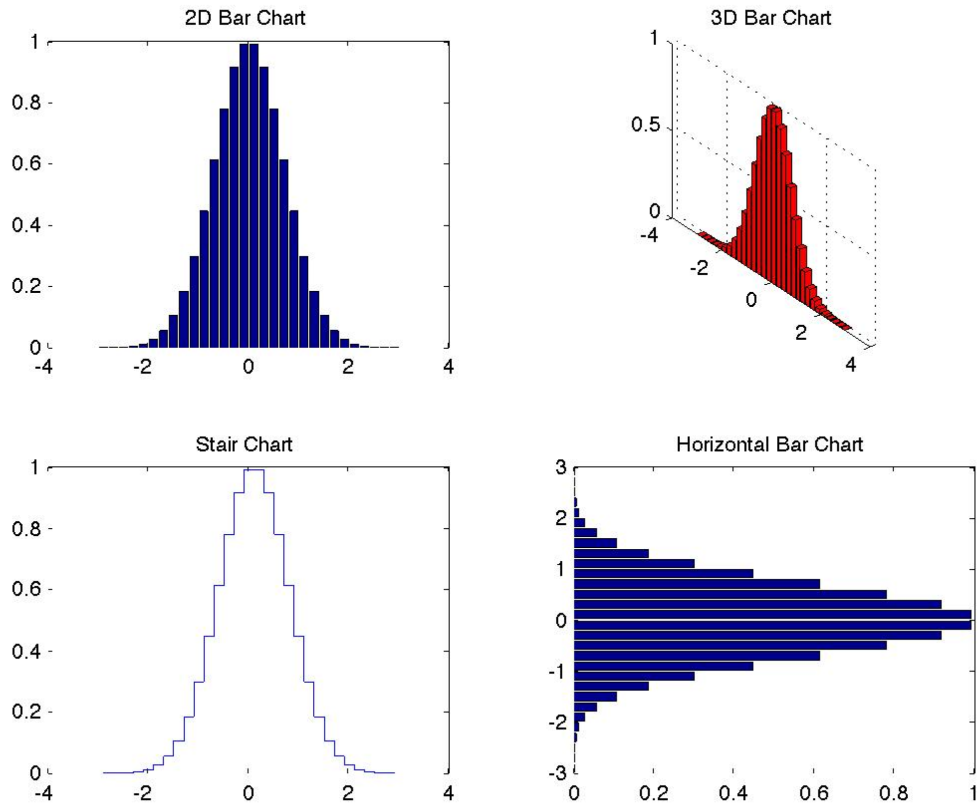
- Bar and stairs plots can be generated using `bar`, `barh` and `stairs` functions, with `bar3` and `bar3D` rendering the bar charts with a 3D effect.

- The following MatLab code (in file **L07plotJ.m**):

```
x=-2.9:0.2:2.9;
y=exp(-x.*x);
subplot(2,2,1)
bar(x,y)
title('2D Bar Chart');
subplot(2,2,2)
```

```
bar3(x,y,'r');
title('3D Bar Chart');
subplot(2,2,3)
stairs(x,y)
title('Stair Chart');
subplot(2,2,4)
barh(x,y)
title('Horizontal Bar Chart');
print L07plotJ.jpg -djpeg
```

produces the graphs:



Upper left: 2D bar Chart, Upper right: 3D bar chart, lower left: stairs chart and lower right: horizontal bar chart.

Histogram Data

- Histograms counts the number of times a variable in a bin (a small range of values) occurs and presents this data as a plot.
- The MatLab code in file **L07plotK.m**):

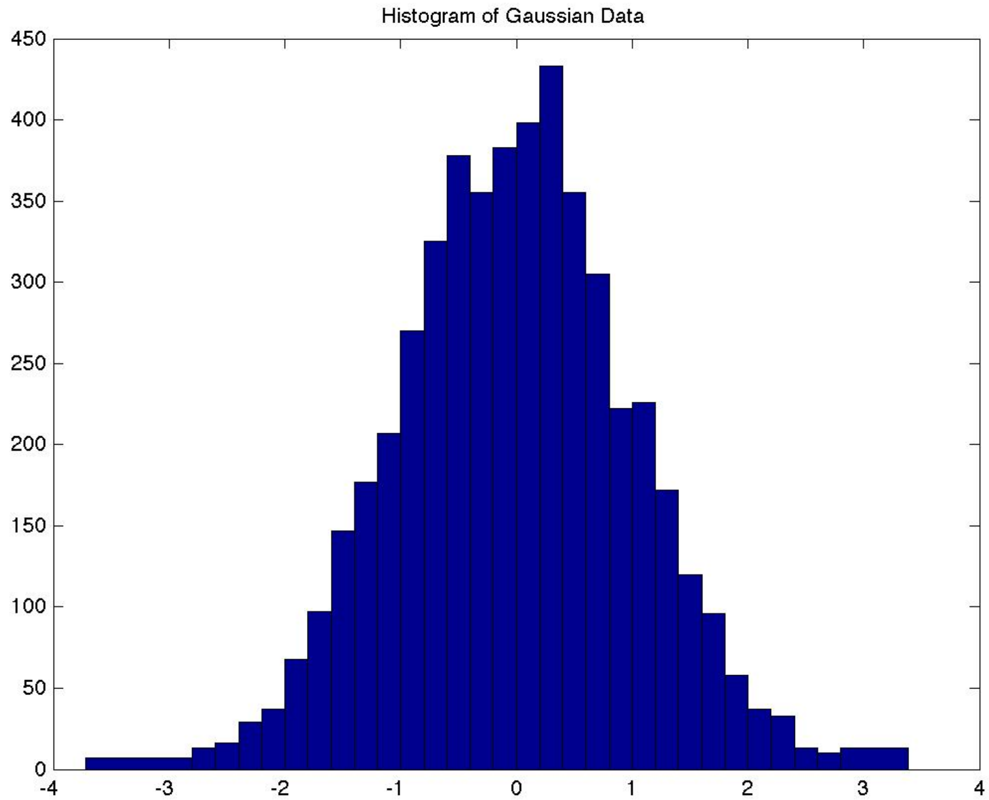
```
% Specify the bins to use
x=-2.9:0.2:2.9;
% generate random normal data points
% normal data is Gaussian data
% r = randn(m,1) returns an m-by-1 matrix (a vector)
y=randn(5000,1);
% Draw histogram
```

```
hist(y,x)

title('Histogram of Gaussian Data')

print L07plotK.jpg -djpeg
```

produces:



Histogram of Gaussian data.

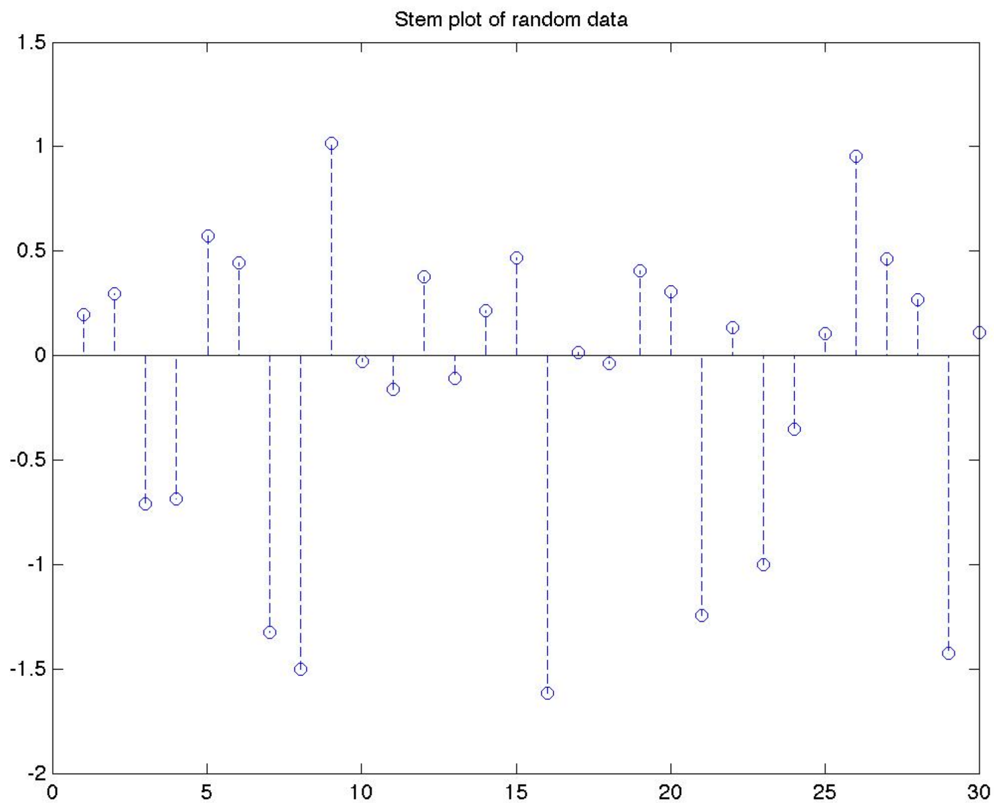
Stem Plots

- Discrete data can be plotted by using the `stem` function. The function `stem(x,z)` creates a plot of the data points in vector `z` connected to the horizontal axis at values of `x`. An optional argument can used to specify the linestyle.

- The following MatLab code in file **L07plotL.m**) does this:

```
% create a 30*1 matrix of random normal data
z=randn(30,1);
stem(z,'--');
title('Stem plot of random data');
print L07plotL.jpg -djpeg
```

plots:



Stem plot of random data.

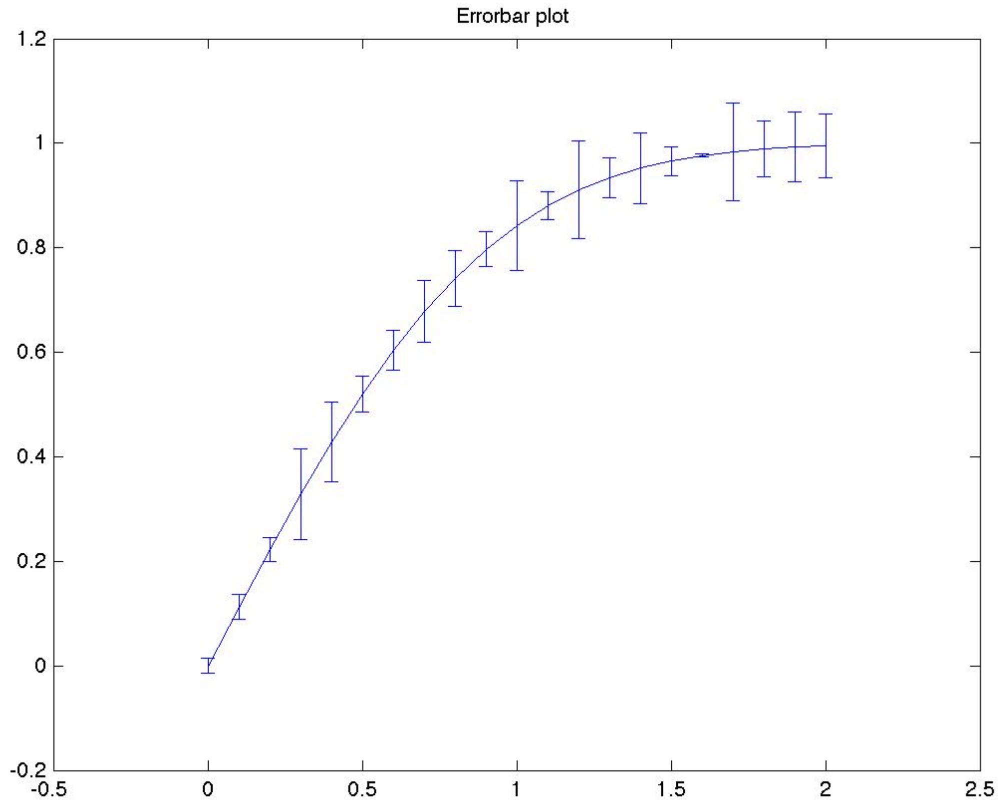
Plots with Error Bars

- Often we would like to plot a function with its standard deviation shown as error bars.
- The following MatLab code in file **L07plotM.m** does this:

```
x=linspace(0,2,21);
% Use erf function to generate some
% Gaussian error function values for x
% ==> a smooth set of increasing values
y=erf(x);
% Generate an error array, e,
% with the same size as x
```

```
% Scale e by 10 to make these values small
% This simulates standard deviation measurements
e=rand(size(x))/10;
% Plot x versus y with error bars 2*e(i)
% for each point x(i),y(i) on the plot
errorbar(x,y,e);
title('Errorbar plot');
print L07plotM.jpg -djpeg
```

produces:



Errorbar plot.

- From wikipedia (for your information): `erf` is defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

- When the results of a series of measurements are described by a normal distribution with standard deviation σ and expected mean value 0, then $erf\left(\frac{a}{\sigma\sqrt{2}}\right)$ is the probability that the error of a single measurement lies between -a and +a, for some positive a. This is useful, for example, in determining the bit error rate of a digital communication system.

Scatter Plots

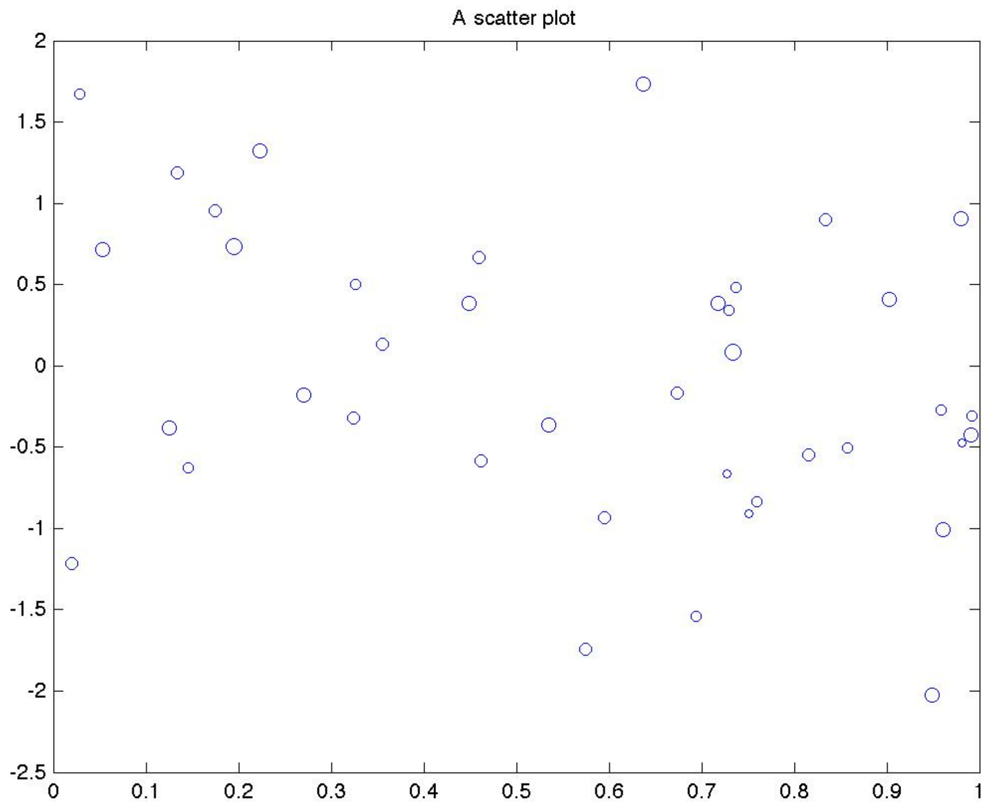
- A scatter plot of data shows the distribution of the data in a 2D area. Also know as a "bubble plot" it draws circles of varying sizes at each data point.

- The following MatLab code in file **L07plotN.m** show how to do this:

```
% Generate 40 random numbers in [0,1]
x=rand(40,1);
% Generate 40 random numbers with
% a Gaussian (normal) distribution
y=randn(40,1);
% Generate 40 areas starting at 21
```

```
% and uniformly increasing to 60
area=20+(1:40);
% Scatter plot the data
scatter(x,y,area);
box on
title('A scatter plot');
print L07plotN.jpg -djpeg
```

produces:



A scatter plot.

Text Formatting

- In the current versions of MatLab, all Tex (Latex) formatting commands can be used. Some of the most useful include:

    1. Superscripts and subscripts are specified by `^` and _ respectively.
    2. `\fontname` and `\fontsize` specify font type and size.
    3. Font style can be controlled by `\bf` (bold), `it` (italic), `\sl` (slant or oblique) or `\rm` (roman).
    4. Colour can be controlled by `\color{colorname}` or `\color[rgb]{r g b}`. `r`, `g` and `b` specify the amount as red, green and blue as floating point numbers between 0 and 1. Each values is rounded to the nearest $1/256$ value between 0 and 1; thus

there are at most 256 values for each colour or $256^3 = 16,777,216$ colours in total (24 bit colour).

5. Backslash special Tex (Latex) characters to print them: `\\` (1 backslash), `\{` and `\}` for curly braces, `\_` for underscore and `\^` for carrot.

- The next 2 tables show a large subset of symbols (including Greek symbols) that can be embedded on MatLab text strings.

`\alpha`	α	`\pi`	π	`\upsilon`	υ
`\angle`	$\angle$	`\rho`	ρ	`\phi`	ϕ
`\ast`	$\ast$	`\sigma`	σ	`\chi`	χ
`\beta`	β	`\varsigma`	ς	`\psi`	ψ
`\gamma`	γ	`\tau`	τ	`\omega`	ω
`\delta`	δ	`\equiv`	$\equiv$	`\Gamma`	Γ
`\epsilon`	ϵ	`\Im`	$\Im$	`\Delta`	Δ
`\zeta`	ζ	`\otimes`	$\otimes$	`\Theta`	Θ
`\eta`	η	`\cap`	$\cap$	`\Lambda`	Λ
`\theta`	θ	`\supset`	$\supset$	`\Xi`	Ξ
`\vartheta`	ϑ	`\int`	$\int$	`\Pi`	Π
`\iota`	ι	`\rfloor`	$\rfloor$	`\Sigma`	Σ
`\kappa`	κ	`\lfloor`	$\lfloor$	`\Upsilon`	Υ
`\lambda`	λ	`\perp`	$\perp$	`\Phi`	Φ
`\mu`	μ	`\wedge`	$\wedge$	`\Psi`	Ψ
`\nu`	ν	`\rceil`	$\rceil$	`\Omega`	Ω
`\xi`	ξ	`\vee`	$\vee$	`\forall`	$\forall$

Some Tex (Latex) characters.

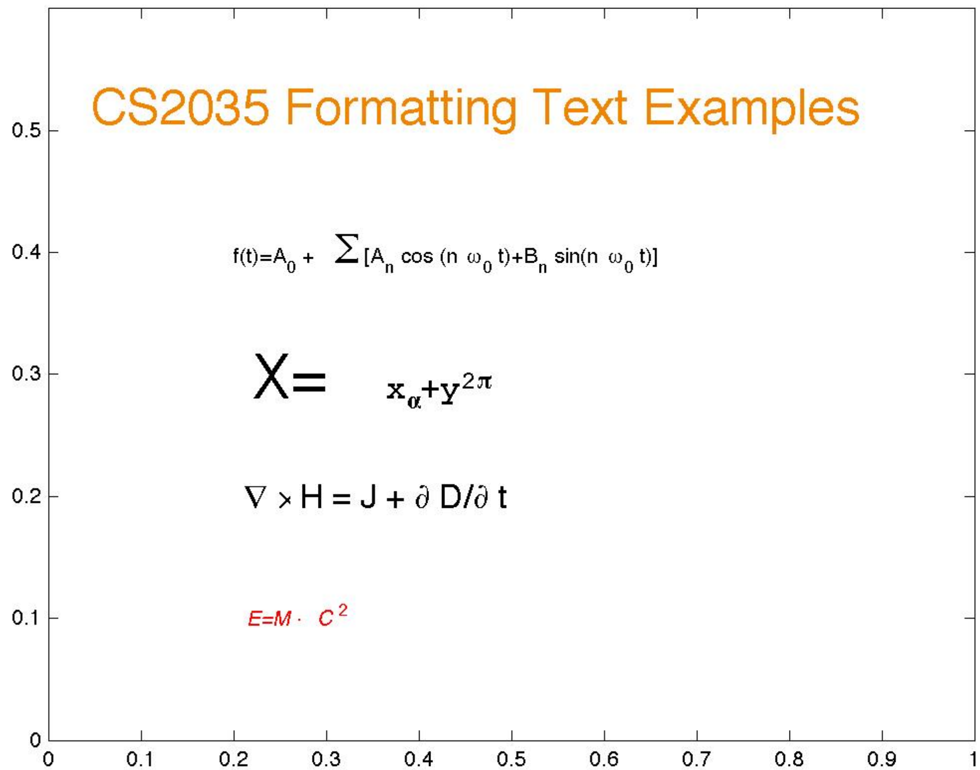
\exists	$\exists$	\sim	$\sim$	\propto	$\propto$
\ni	$\ni$	\leq	$\leq$	\partial	∂
\cong	$\cong$	\infty	∞	\bullet	$\bullet$
\approx	$\approx$	\clubsuit	$\clubsuit$	\div	$\div$
\Re	$\Re$	\diamondsuit	$\diamondsuit$	\neq	$\neq$
\oplus	$\oplus$	\heartsuit	$\heartsuit$	\aleph	$\aleph$
\cup	$\cup$	\spadesuit	$\spadesuit$	\wp	$\wp$
\subseteq	$\subseteq$	\leftrightarrow	$\leftrightarrow$	\oslash	$\oslash$
\in	$\in$	\leftarrow	$\leftarrow$	\supseteq	$\supseteq$
\lceil	$\lceil$	\Leftarrow	$\Leftarrow$	\subset	$\subset$
\cdot	$\cdot$	\uparrow	$\uparrow$	\o	ø
\neg	$\neg$	\rightarrow	$\rightarrow$	\nabla	∇
\times	$\times$	\Rightarrow	$\Rightarrow$	\ldots	$\ldots$
\surd	$\surd$	\downarrow	$\downarrow$	\prime	$\prime$
\varpi	ϖ	\circ	$\circ$	\O	Ø
\rangle	$\rangle$	\pm	$\pm$	\mid	$\mid$
\langle	$\langle$	\geq	$\geq$	\copyright	©

Some more Tex (Latex) characters.

- The following MatLab code in file **L07plot).m** shows how to do this:

```
close all
axis([0 1 0 0.6]);
box on
text(0.0,0.52,['\fontsize{24} ' ...
     '\color[rgb]{0.9467 0.5203 0.0} ' ...
     ' CS2035 Formatting Text Examples']);
text(0.2,0.4,['f(t)=A_0 + ' ...
              '\fontsize{20} \Sigma' ...
              '\fontsize{10} [A_n cos ' ...
              '(n \omega_0 t)+B_n ' ...
              'sin(n \omega_0 t)]']);
text(0.2,0.3,['\fontsize{30} X=' ...
              '\fontname{courier} \fontsize{16}' ...
              '\bf x_{\alpha}+y^{2\pi}']);
text(0.2,0.2,['\fontsize{16} \nabla \times H' ...
              ' = J + \partial D/\partial t']);
text(0.2,0.1,'\color{red} \it E=M \cdot C^{\rm 2}');
print plotO.jpg -djpeg
```

produces



Formatted text example.