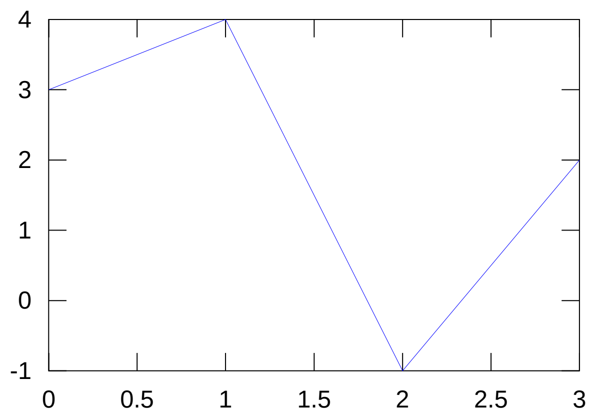
1 Basic Plots

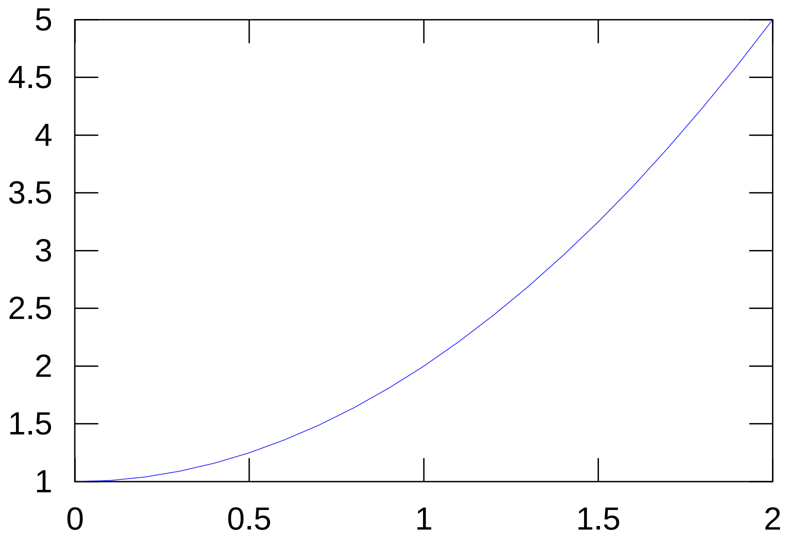
If x is a vector of 10 x-coordinates and y is a vector of 10 y-coordinates, then plot(x,y) plots 10 points where the x-coordinates come from the x vector and the y-coordinates come from the y vector. The points are connected with straight lines.

```
octave-3.2.3:11> x=[0,1,2,3];
octave-3.2.3:12> y=[3,4,-1,2];
octave-3.2.3:13> plot(x,y)
```



It is important that the x and y vectors must have the same number of entries. This is usually done by first constructing a vector of x entries, and then using x to make a vector of the same length with the y-coordinates. For example:

```
octave-3.2.3:11> x=[0:0.1:2];
octave-3.2.3:12> y=1+x.^2;
octave-3.2.3:13> plot(x,y)
```

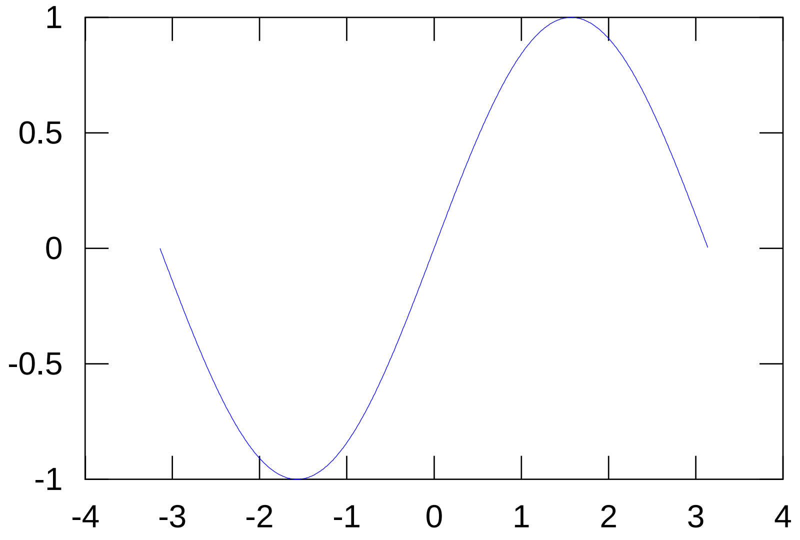


In this next example, we make a plot of the function $\sin(x)$ over the range $-\pi \leq x \leq \pi$. To create the vector of x-coordinates, it would be helpful if Octave knew about the value of π; it does:

```
octave-3.2.3:22> x=[-pi:0.01:pi];
```

Octave's sin function behaves nicely when you give it a vector of input; it returns a vector of the same size where the sin function has been applied to each entry. This is exactly what you want for making a plot.

```
octave-3.2.3:23> y=sin(x);
octave-3.2.3:24> plot(x,y)
```



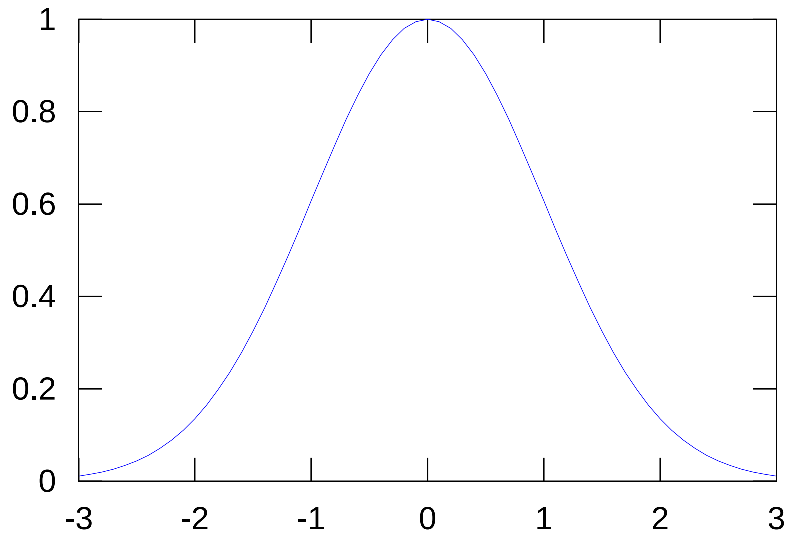
Most of the mathematical functions you know are available in Octave, and behave the same way when you feed them vector inputs.

Octave function	Mathematical function
`sin`	sin
`cos`	cos
`tan`	tan
`sqrt`	$\sqrt{x}$ (square root)
`exp`	exp (i.e. $\exp(x) = e^x$)
`log`	ln (logarithm base e)
`log10`	$\log_{10}$ (logarithm base 10)
`atan`	arctan (the inverse function of tan)
`asin`	arcsin
`factorial`	$x!$ (factorial)
`abs`	$\lvert x \rvert$ (absolute value)

2 Inline functions

Sometimes it is handy to make your own mathematical functions. For example, you might be working with the Gaussian function $g(x) = e^{-x^2/2}$. You can plot this function as follows

```
octave-3.2.3:88> x=[-3:0.1:3];
octave-3.2.3:89> plot(x,exp(-x.^2/2))
```



This function is the well-known bell curve. The following commands make a new function gauss that behaves much the same way as `sin`, `cos`, and so forth.

```
octave-3.2.3:90> gauss = @(t) exp(-t.^2/2)
gauss =

@(t) exp (-t .^ 2 / 2)
```

These commands say that gauss is a function that takes one input (called `t` for the purposes of defining the function) and returns $e^{-t^2/2}$. You can now use your new function:

```
octave-3.2.3:91> gauss(0)
ans =  1
octave-3.2.3:92> gauss(1)
ans =  0.60653
octave-3.2.3:93> 1/sqrt(e)
ans =  0.60653
octave-3.2.3:94> gauss(2)
ans =  0.13534
octave-3.2.3:95> 1/(e*e)
ans =  0.13534
octave-3.2.3:96> gauss([0,1,2])
```

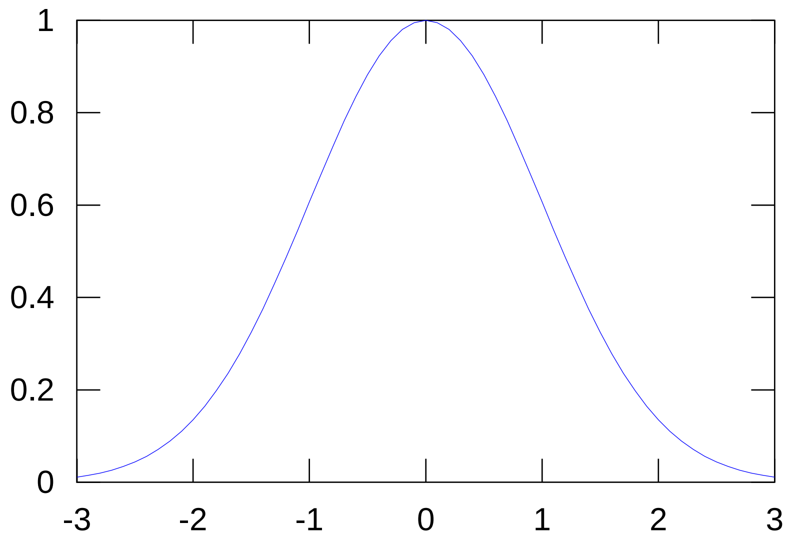
```
ans =

   1.00000   0.60653   0.13534
```

Take a moment and figure out what was going on in the computations above. Why is gauss(2) the same as $1/e^2$? (Notice that Octave knows about the number e, it's just e!).

We can easily plot the gauss function:

```
octave-3.2.3:97> x=[-3:0.1:3];
octave-3.2.3:98> plot(x,gauss(x))
```

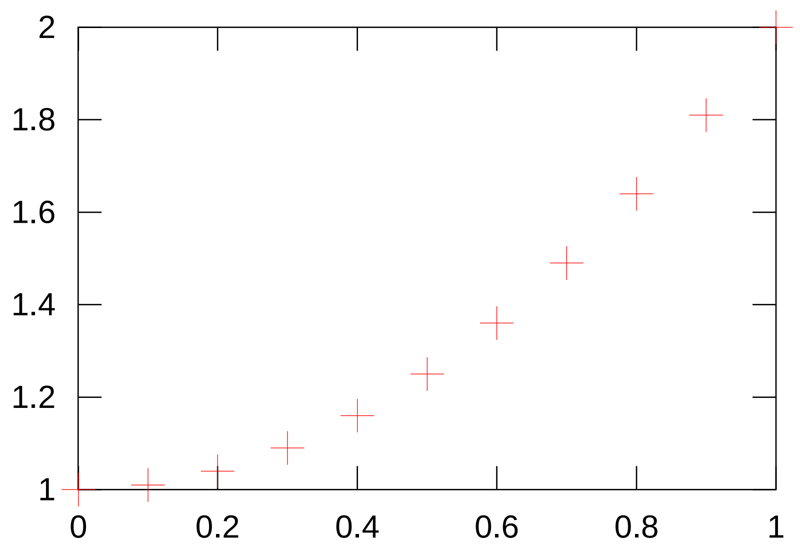


3 Properties of plots

The plot command allows you to specify certain properties of the graph. For example, you might want a different color line, or just points plotted rather than a line.

```
octave-3.2.3:42> x=[0:0.1:1];
octave-3.2.3:43> y=1+x.^2;
octave-3.2.3:44> plot(x,y,'r+')
```

The extra argument 'r+' specifies that that red crosses should be used for the plot:



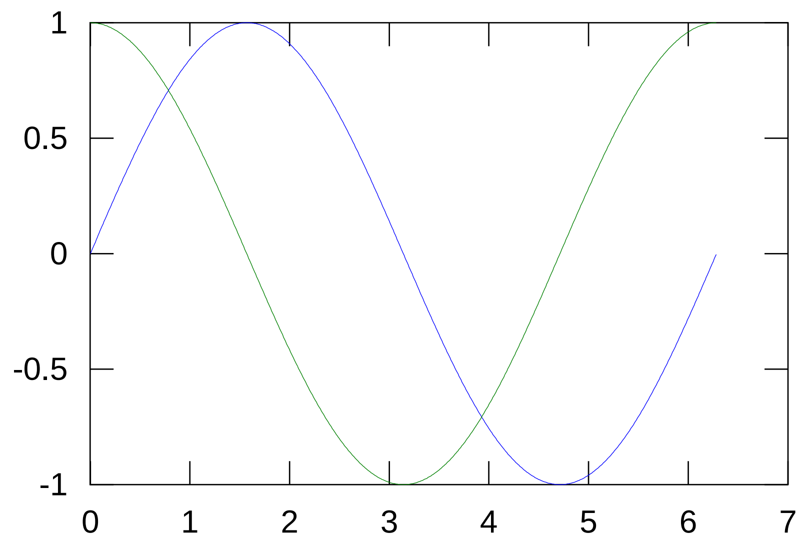
Some color styles: `r` (red), `g` (green), `b` (blue), `c` (cyan), `m` (magenta), `k` (black).

Some symbol styles: `.` (dots), `s` (squares or diamonds), `+` (crosses), (’*’) (stars), `o` (circles or triangles).

4 Plotting more than one graph

You can plot more than one graph at once with the plot command. Just specify each x and y coordinate vector.

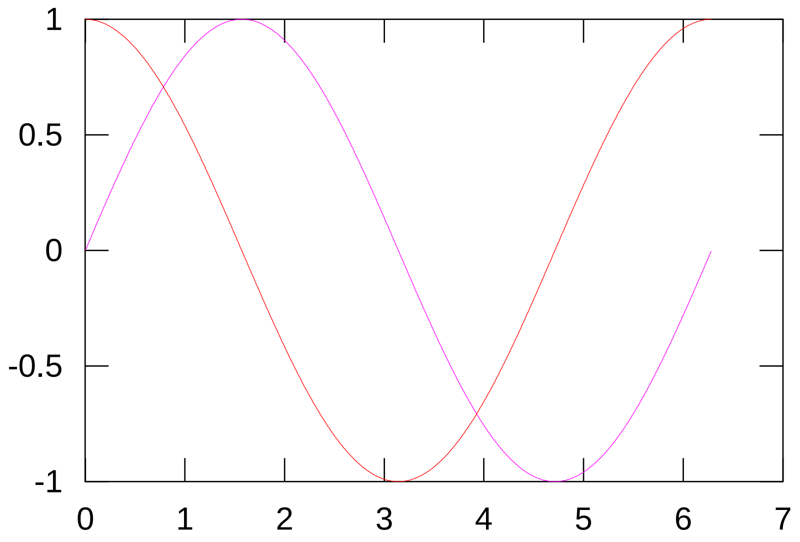
```
octave-3.2.3:53> x=[0:0.01:2*pi];
octave-3.2.3:54> plot(x,sin(x),x,cos(x))
```



Octave picks colors for each curve. You can change them if you want.

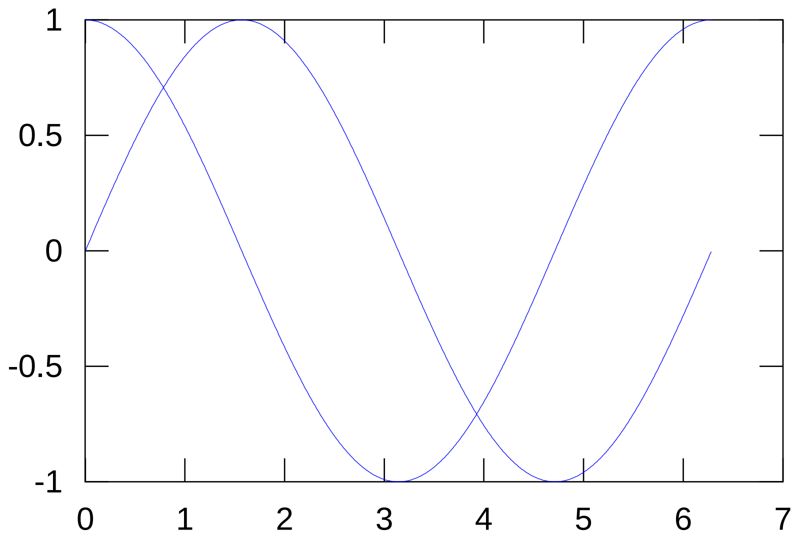
```
octave-3.2.3:53> x=[0:0.01:2*pi];
```

```
octave-3.2.3:54> plot(x,sin(x),'m',x,cos(x),'r')
```



If you have already made a plot, you can add more curves to that plot using the `hold` command.

```
octave-3.2.3:60> x=[0:0.01:2*pi];
octave-3.2.3:61> plot(x,sin(x))
octave-3.2.3:62> hold on
octave-3.2.3:63> plot(x,cos(x))
octave-3.2.3:64> hold off
```

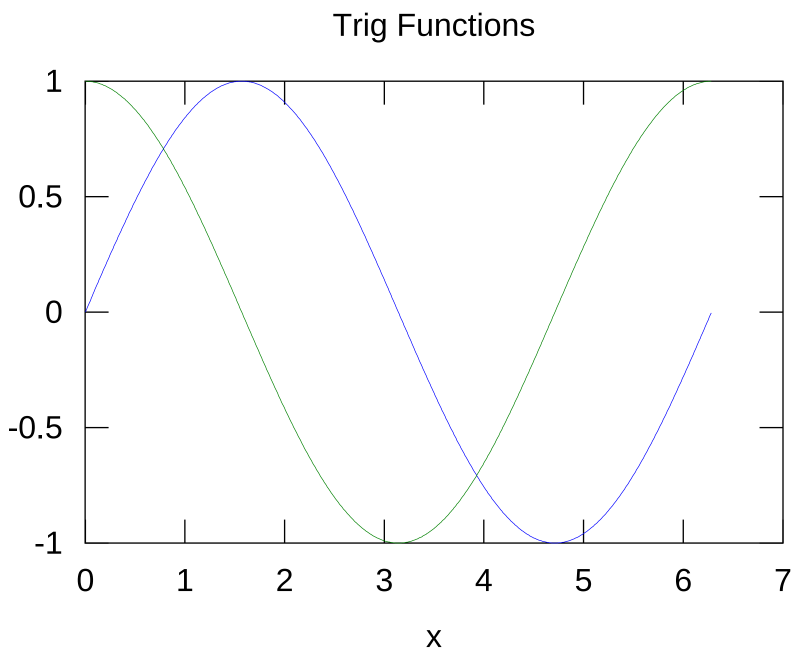


Everything you plot between `hold on` and `hold off` will appear in the same figure.

5 Labels

It can be helpful to add some text labels to a plot. Here are some examples:

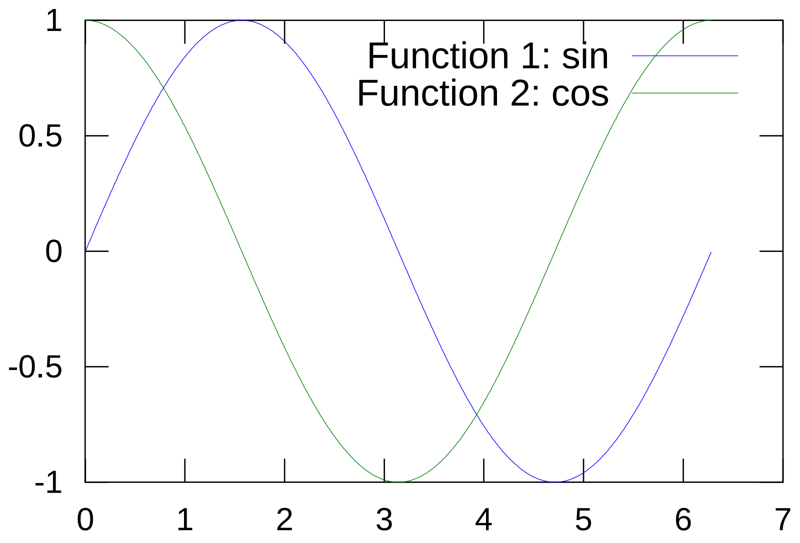
```
octave-3.2.3:66> plot(x,sin(x),x,cos(x))
octave-3.2.3:67> title('Trig Functions')
octave-3.2.3:68> ylabel('y')
octave-3.2.3:69> xlabel('x')
```



Unfortunately, on the Mac the labels are sometimes cut off (my plot above is missing the *y* label). Free software sometimes has this kind of glitch.

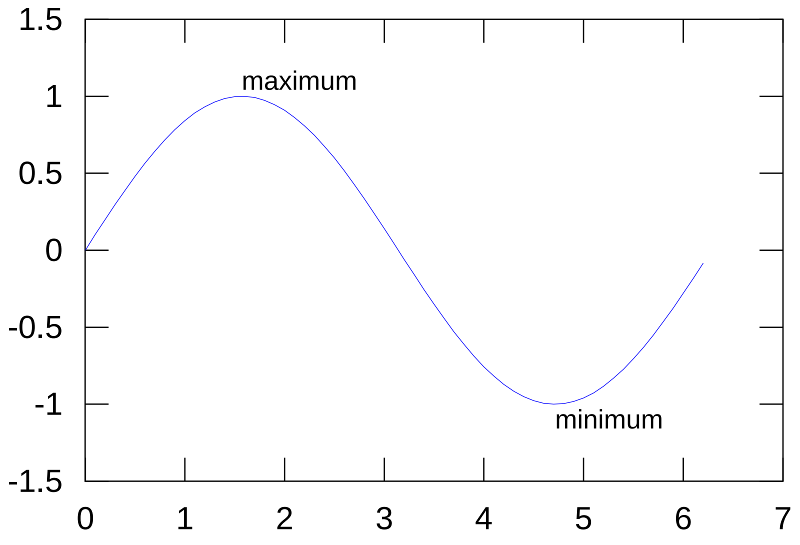
It can be helpful to add a legend to a plot as well.

```
octave-3.2.3:66> plot(x,sin(x),x,cos(x))
octave-3.2.3:68> legend('Function 1: sin','Function 2: cos')
```



You might want at some point to add a text label somewhere in a plot

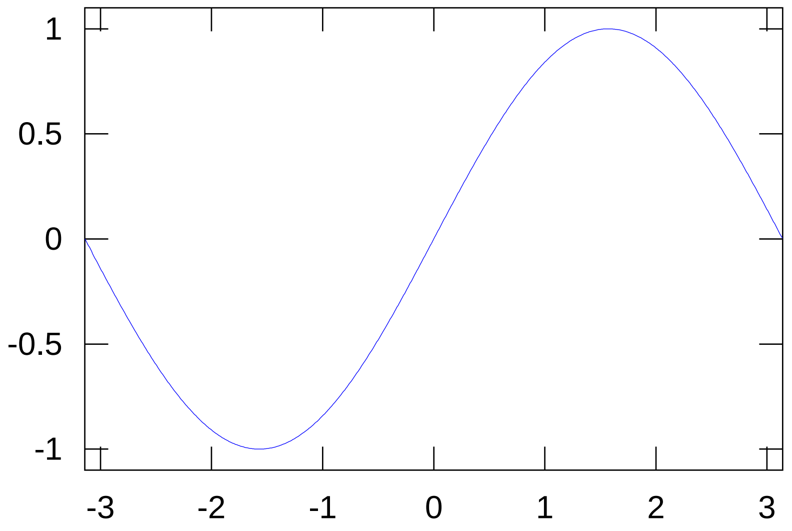
```
octave-3.2.3:113> x=[0:0.1:2*pi];
octave-3.2.3:114> plot(x,sin(x))
octave-3.2.3:115> text(pi/2,1.1,'maximum')
octave-3.2.3:116> text(3*pi/2,-1.1,'minimum')
```



6 Miscellaneous

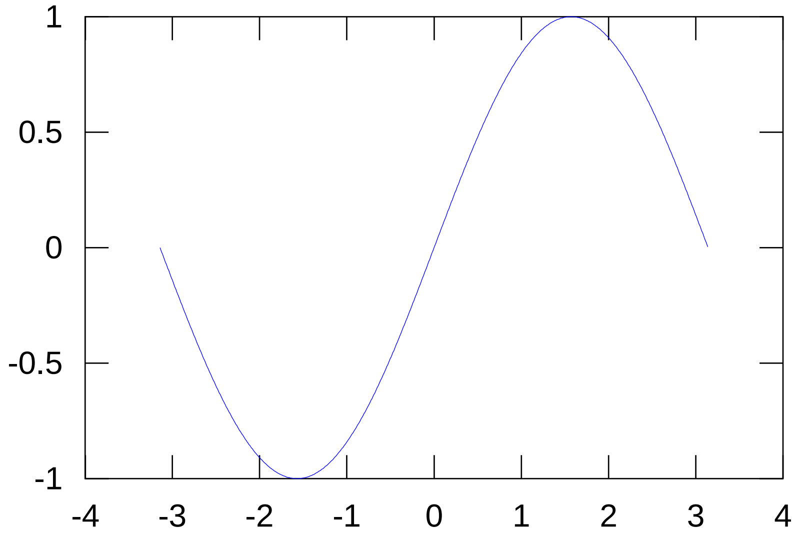
You might want to adjust the viewing region of a plot. Use `axis([xmin,xmax,ymin,ymax])`. Example:

```
octave-3.2.3:151> x=[-pi:0.01:pi];
octave-3.2.3:152> plot(x,sin(x))
octave-3.2.3:153> axis([-pi,pi,-1.1,1.1])
```



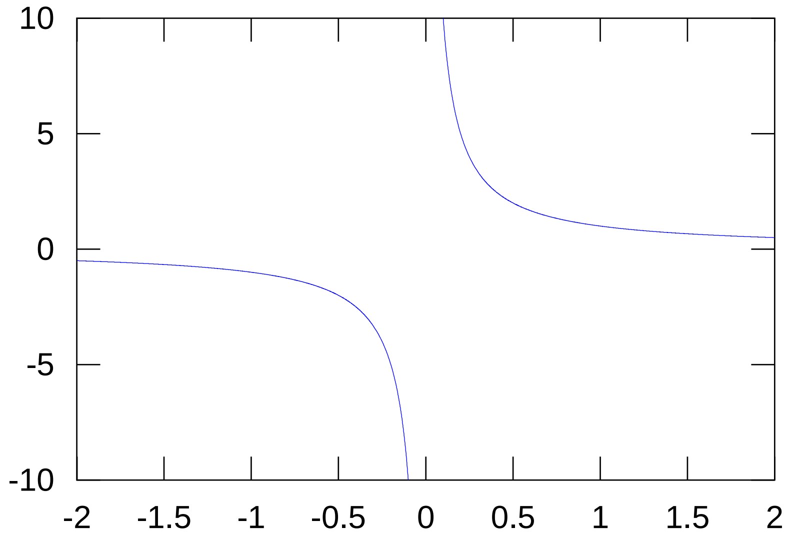
Without the `axis` command the plot would have looked like:

```
octave-3.2.3:151> x=[-pi:0.01:pi];
octave-3.2.3:152> plot(x,sin(x))
```



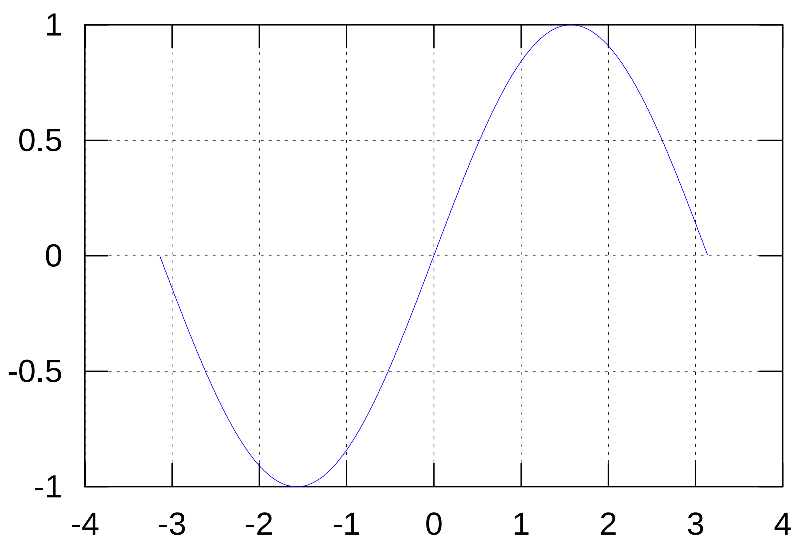
This is especially helpful for plotting functions that have singularities, like $1/x$

```
octave-3.2.3:151> x=[-2:0.001:2];
octave-3.2.3:152> plot(x,1./x)
octave-3.2.3:153> axis([-2,2,-10,10])
```



You might want to add a grid to the background of a plot. Use `grid`. Example:

```
octave-3.2.3:151> x=[-pi:0.01:pi];
octave-3.2.3:152> plot(x,sin(x))
octave-3.2.3:153> grid
```



7 Getting help

Octave has a help facility. If you enter `help plot`, you will see a help page for the plot command. The help pages can be a bit cryptic to read sometimes, but they can point you in the right direction. If a help page has more than one screen, you use the space-bar or 'f' to advance to the next page ('b' is used to go back, and 'q' is used to stop reading the help page.)

The Octave plotting commands are related to Matlab plotting commands. So you can use Google to search for `matlab plot` and find a Matlab help page as well. Keep in mind, however, that Matlab's plotting facility is much more advanced than Octave's, so not everything you find will work.

8 Exercises

Before attempting these exercises, you should enter all the commands in this tutorial into Octave. Everything you need to complete these exercises can be found in the tutorial.

Exercise 1:

Plot the curves $y = Ce^x$ for $C = 1$, $C = 1/2$, $C = 0$, $C = -1/2$, and $C = -1$ over the range $-1 \leq x \leq 1$ all in the same figure. Add a helpful legend to your plot. Hand in a printout of your plot.

Exercise 2:

Use Octave to generate a nice plot for your work in Section 1.2, problem 15. Hand in a printout of your plot.

Exercise 3:

Let $\text{logistic}(x) = \dfrac{1}{1+e^{-x}}$.

a. Following the example from Section 2, use Octave to define a function `logisitic` for this function.

b. Verify that your function works correctly by computing `logistic(0)`, `logisitic(1)`, and `logistic([0,1])`. Do you obtain the right answers? (Hint: if you have a error when you test with vector input, think about the dot operators `.*`, `./` and so forth.)

c. Plot the `logistic` function over the range $-2 \leq x \leq 2$. Add a red square or diamond that marks the point $(1, \text{logistic}(1))$.

Hand in a transcript of the Octave commands you used in parts a) through c) as well as a printout of your plot.