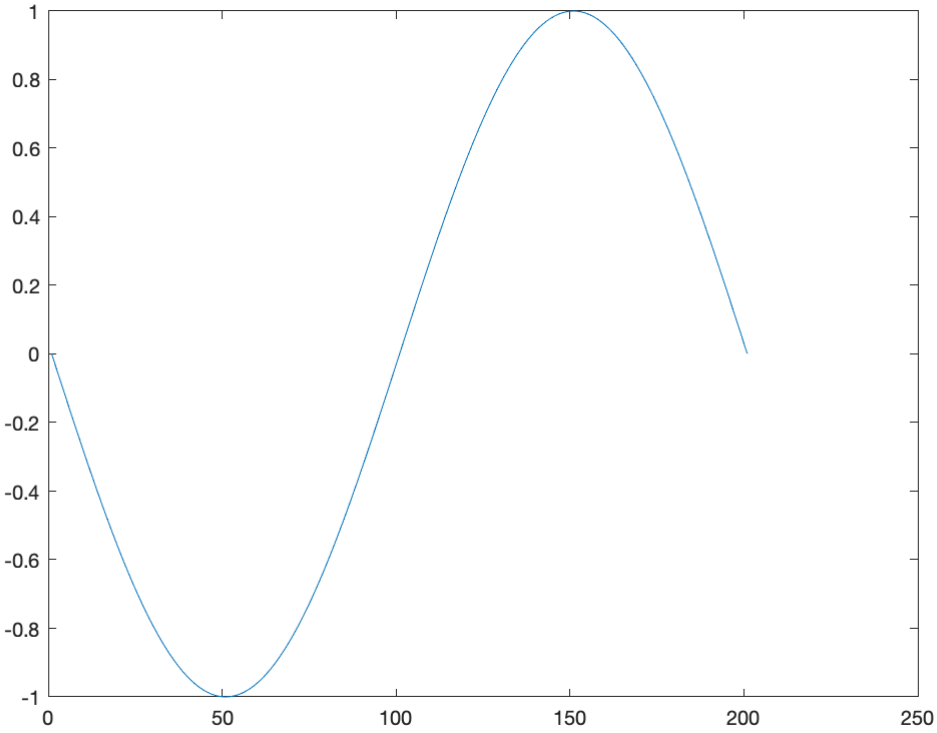
ENME202 Matlab

PLOTS

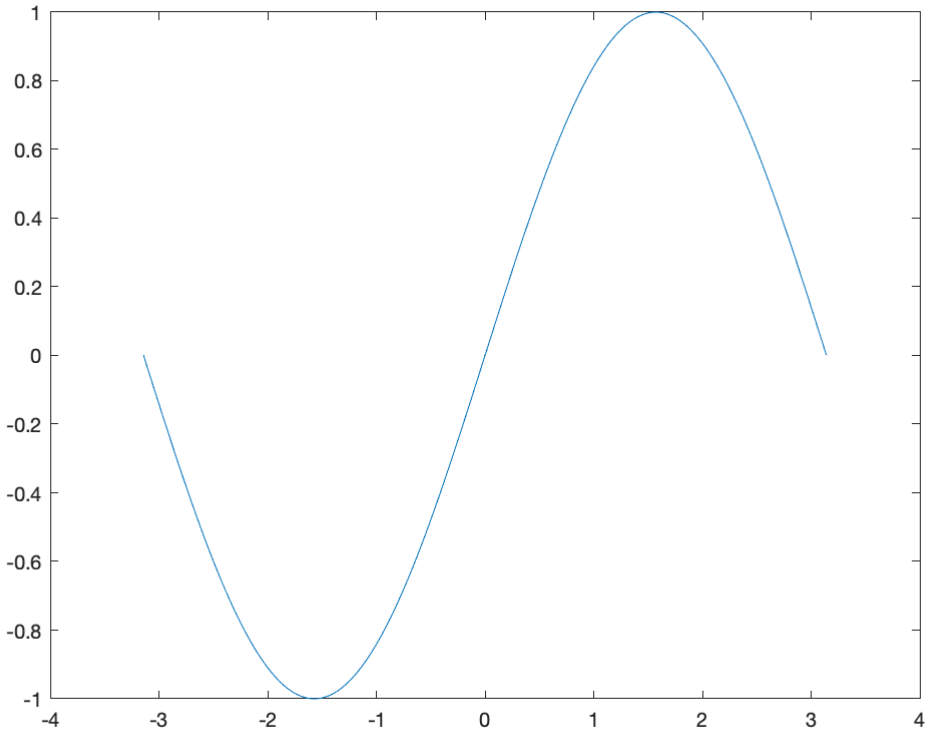
Simple plots are easily created by passing arrays to the **plot()** function. When passing a single array y as plot(y), Matlab will generate a graph with the array index values on the x-axis, and the corresponding array values on the y-axis:

```
theta = (-1:.01:1)*pi;
y = sin(theta);
plot(y)
```



When passing two arrays x and y as plot(x,y) the resulting graph will contain points with coordinates x(i) and y(i) for each index i in the arrays:

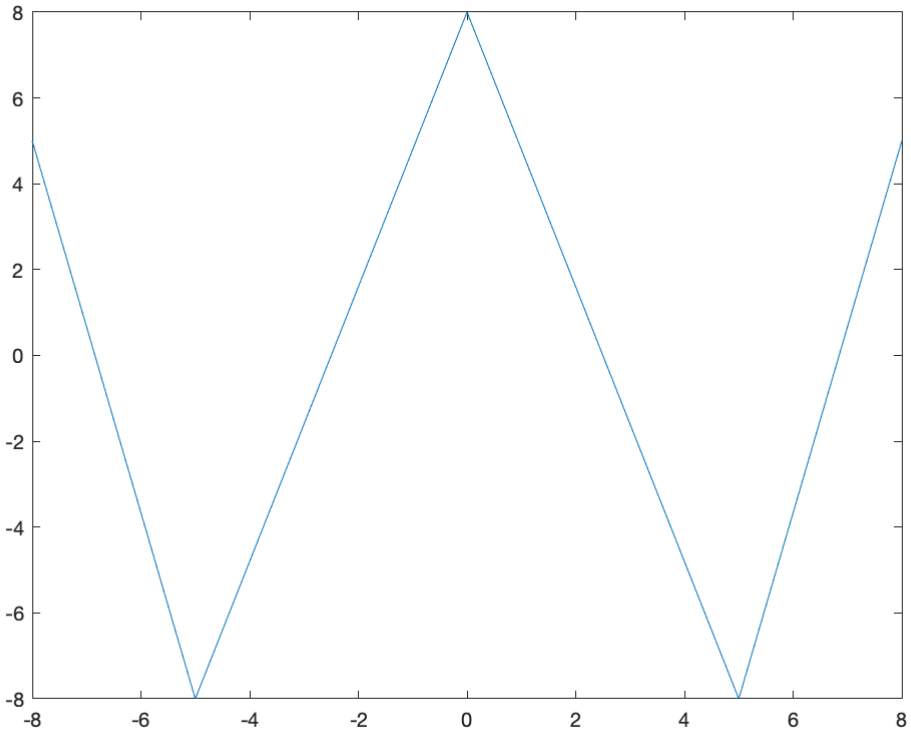
```
clf                % clears the plot window
plot(theta,y)
```



Note that arrays theta and y must have the same length

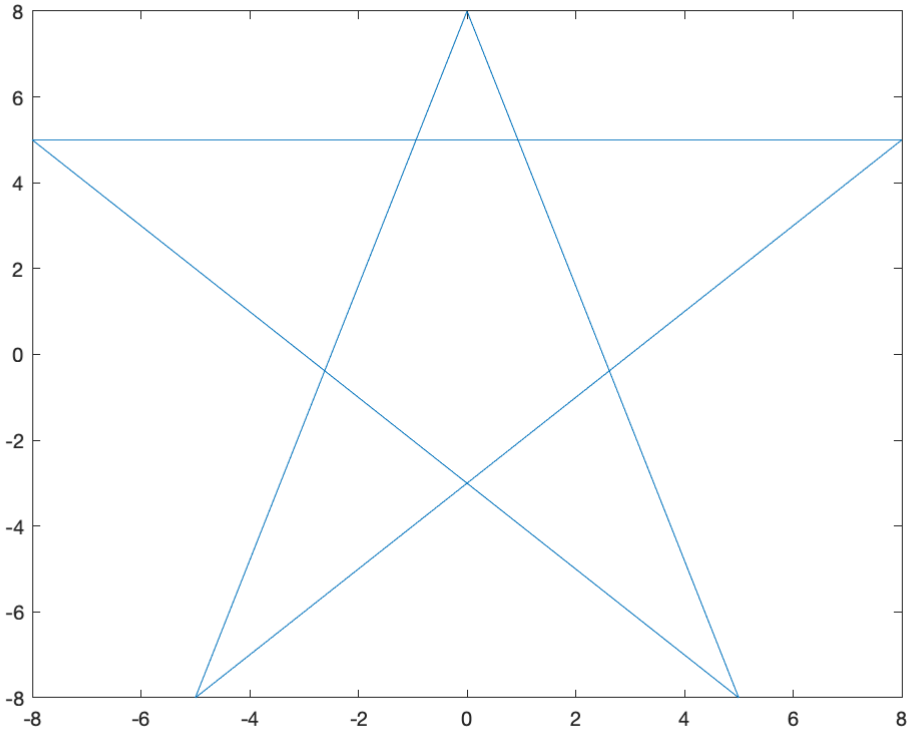
By default, each pair of (theta,y) values is shown as a single dot on the graph, and Matlab draws a line connecting the dots **in sequence**, which can lead to unexpected results if the point pairs are not properly ordered:

```
x = [-8 -5 0 0  5 8];    % increasing x values
y = [ 5 -8 8 8 -8 5];
plot(x,y)
```



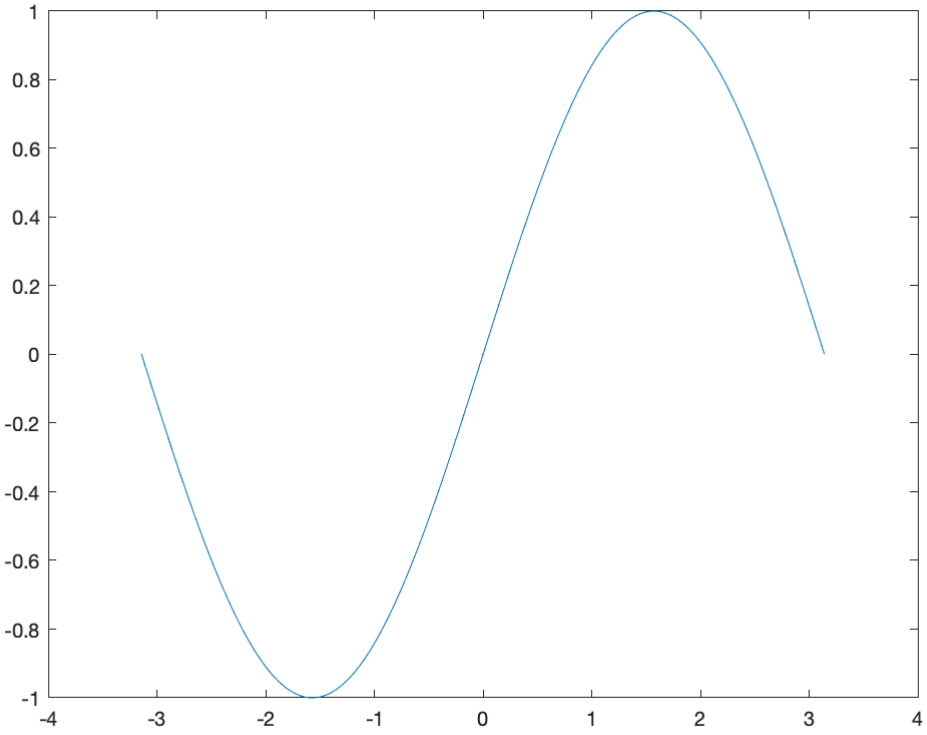
vs:

```
x = [0 -5 8 -8  5 0];    % same x,y pairs, but different order
y = [8 -8 5  5 -8 8];
plot(x,y)
```



Try another example:

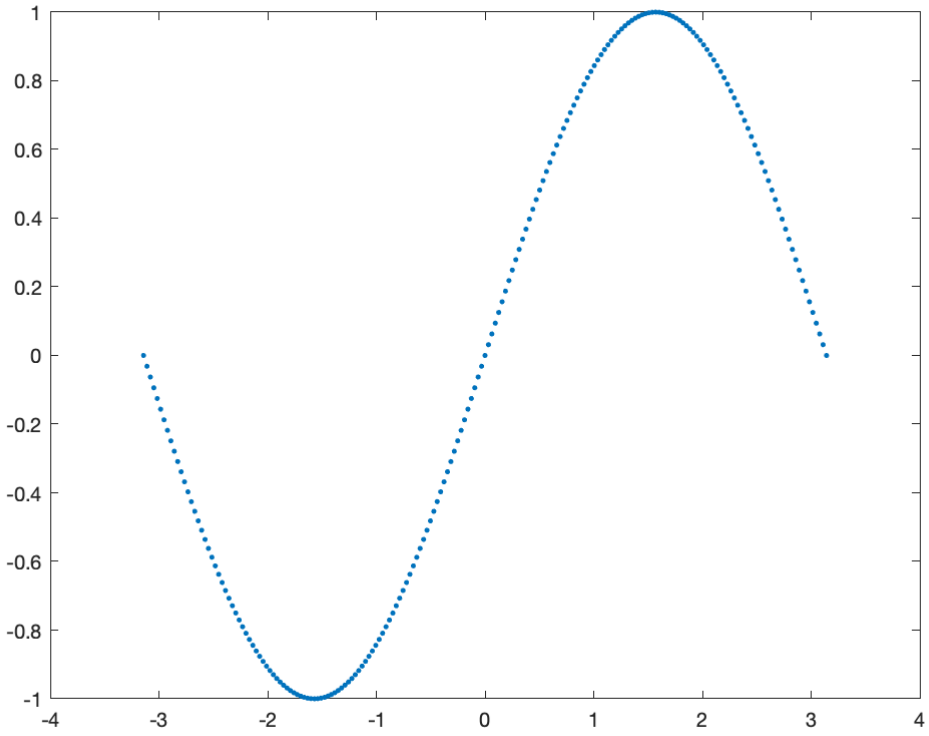
```
x = (-1:.01:1)*pi;
y = sin(x);
plot(x,y)
```



By default, the plot linearly interpolates between x,y data points.

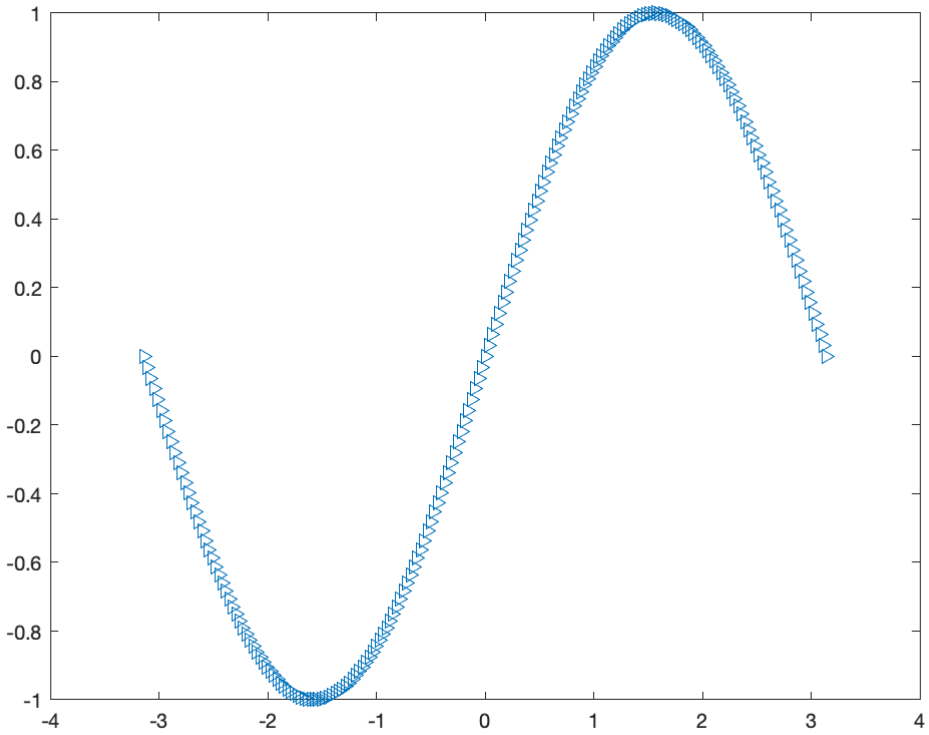
Can plot just the data points using a third argument in plot():

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



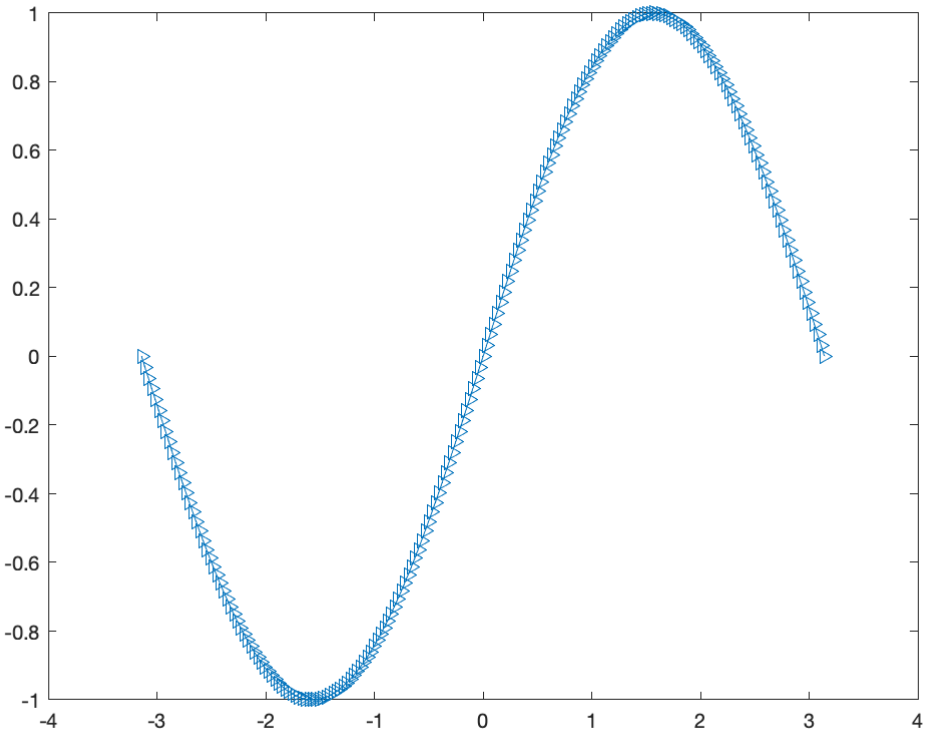
A variety of point markers are available, like triangles:

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



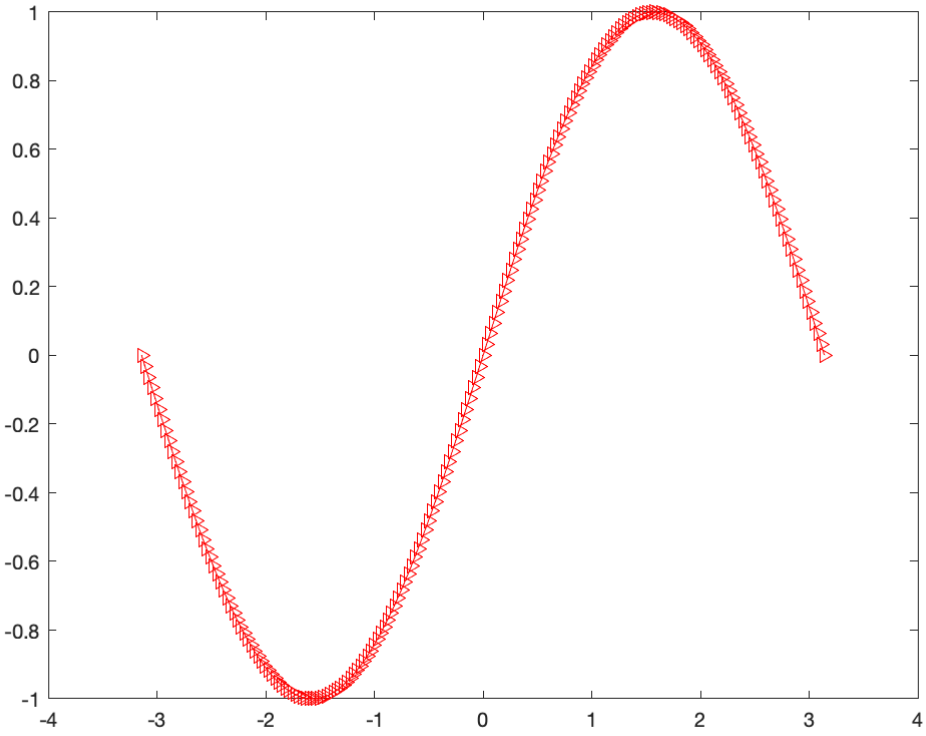
Draw the data with markers, *and* the interpolated line between points

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



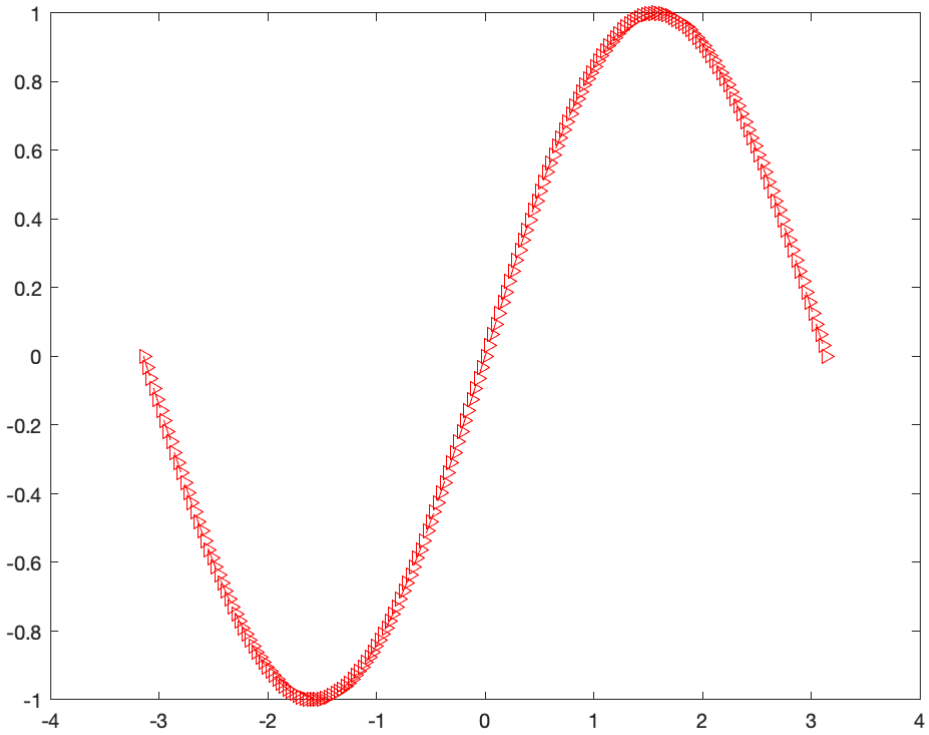
Chage the color of markers and lines ("r","g","b","k", etc colors)

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



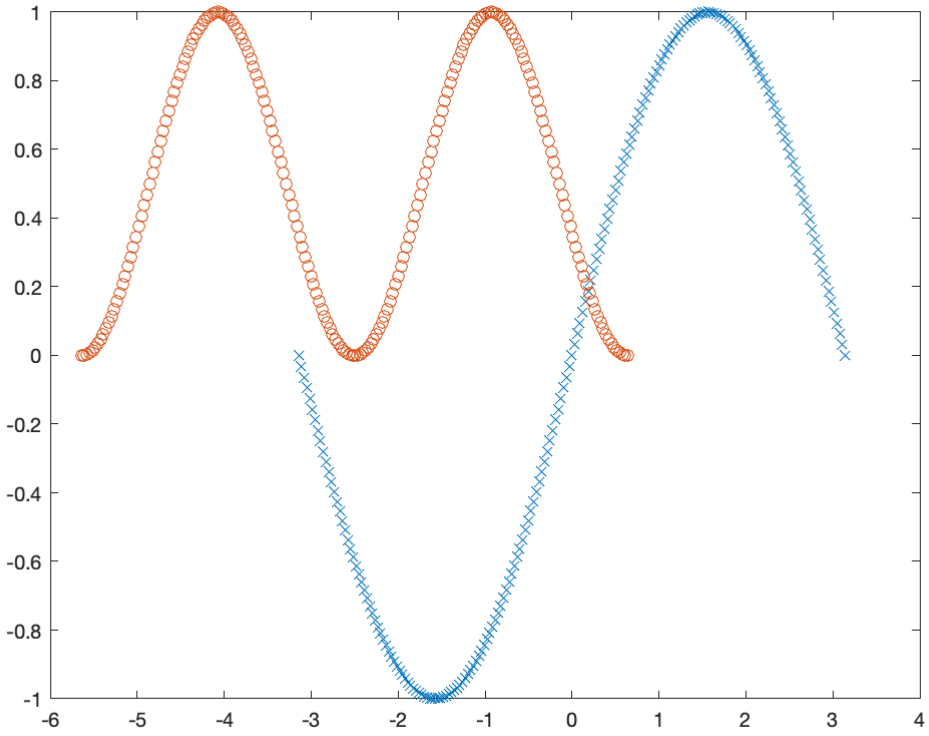
Control the line style (dashed "--", dotted ":", etc)

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



Combine multiple plots in a single function call:

```
plot(x,y,'x', x-2.5,y.^2,'o')
```



Many combinations of colors, markers, and linestyles, check the help file for details:

```
help plot
```

```
  plot   Linear plot.
     plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
     then the vector is plotted versus the rows or columns of the matrix,
     whichever line up.  If X is a scalar and Y is a vector, disconnected
     line objects are created and plotted as discrete points vertically at
     X.

     plot(Y) plots the columns of Y versus their index.
     If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)).
     In all other uses of plot, the imaginary part is ignored.

     Various line types, plot symbols and colors may be obtained with
     plot(X,Y,S) where S is a character string made from one element
     from any or all the following 3 columns:

            .      .-                      .                        -.-
```

Formatting plots

Adding grids, axis labels, and titles (note the "see also" options under "help plot"):

```
plot(x,y)
grid
xlabel('x')
ylabel('sin(x)')
title('Plot of sin(x)');
```

The range for each axis can be specified using the axis command, with syntax: axis([XMIN XMAX YMIN YMAX])

```
axis([-2 2 -1.5 3])
```

Each new plot command will completely overwrite previous plot and formatting. To keep the previous plot, put new plot in a different "figure" window.

Create a new figure window and put a different plot in it:

```
figure(2)
plot(x, sin(2*x))
```

Figures can be saved to different graphics formats (png, pdf, etc.).

We won't do a lot with multiple figure windows in this class but the functionality is there to open up as many different figure windows as you like, with different plots on each.

To "redirect" subsequent plots to a specific, already open figure window, type **figure(n)** before the next plot command, where "n" is the number of the window you want to use. If a window with that number doesn't exist, a new window will be opened and given the number indicated by "n".

Get rid of an unneeded figure window with the close command

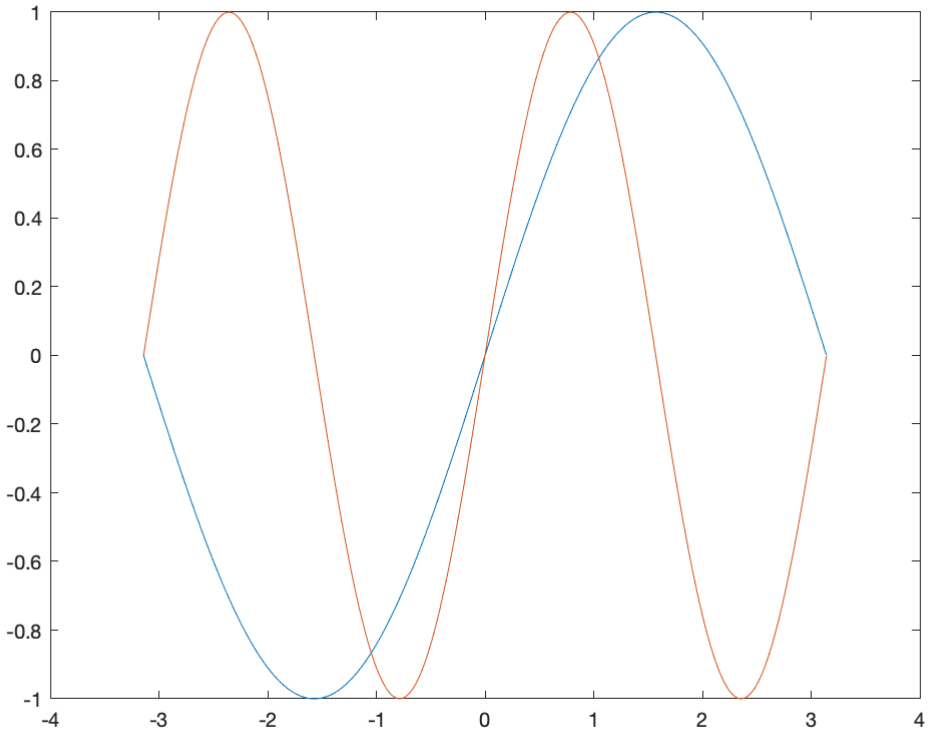
```
close(2)
```

We can also close all figure windows:

```
close all
```

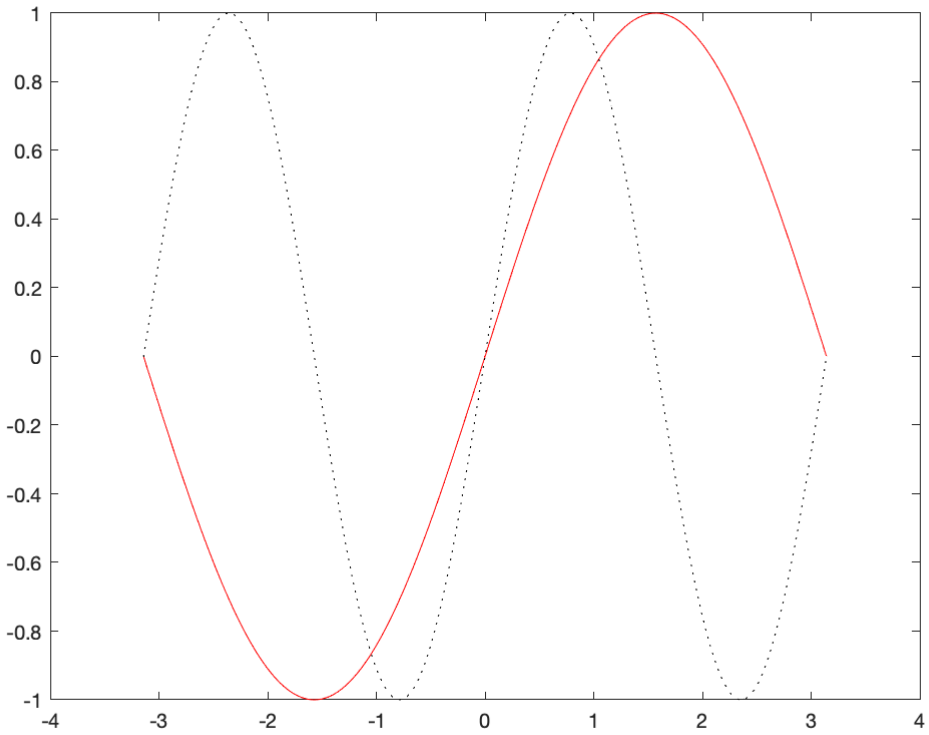
As we saw previously, multiple plots can be placed in a single figure by specifying 2 (or more) x-y datasets in a single plot command:

```
plot(x, y, x, sin(2*x))
```



Can control the color and linestyle of each plot separately:

```
plot(x, y, 'r-', x, sin(2*x), 'k:')
```

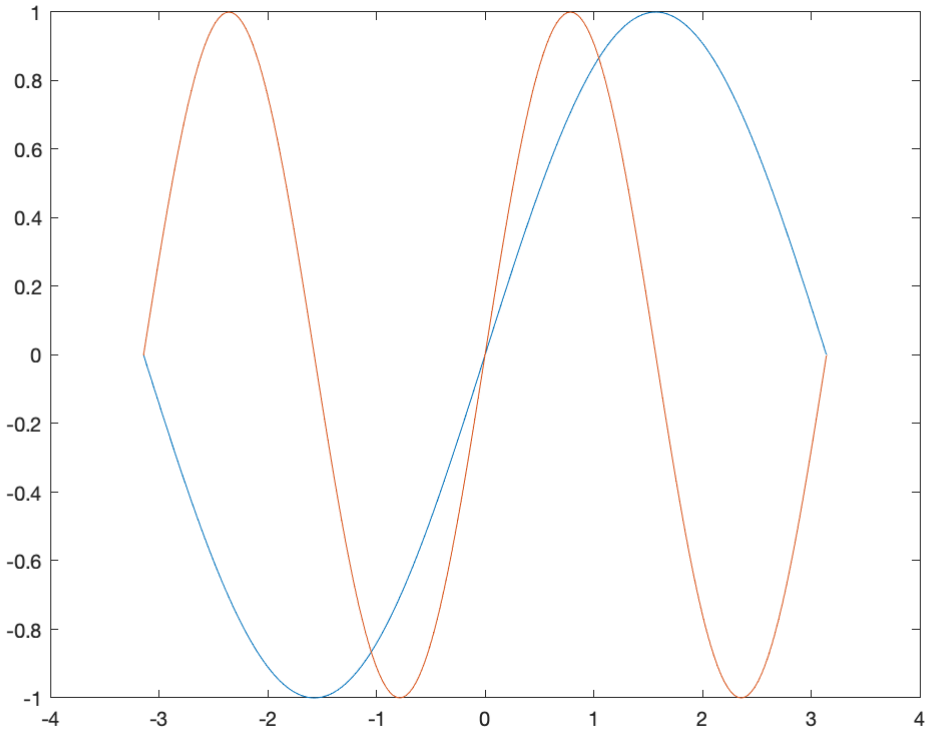


Can alternately use "hold on" to tell Matlab to keep the previous plot in a window in order to overlay an additional plot onto it at a later time

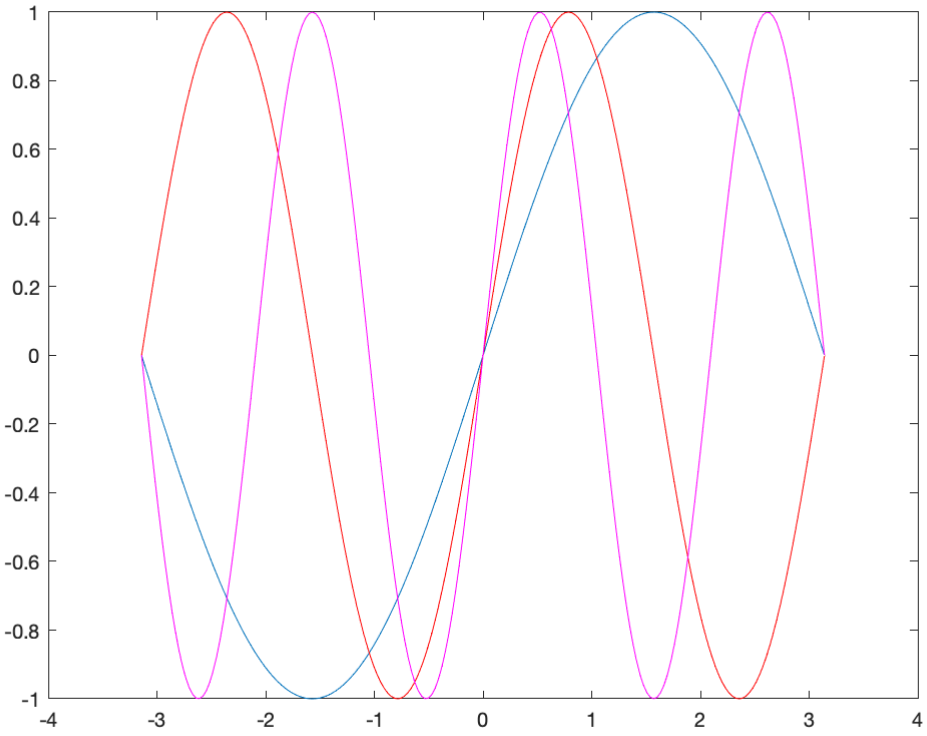
```
plot(x,y)
hold on
plot(x, sin(2*x))
```

Use "hold off" to revert to original plot behavior (replacing the old plot completely when a new plot command is given)

```
hold off
```

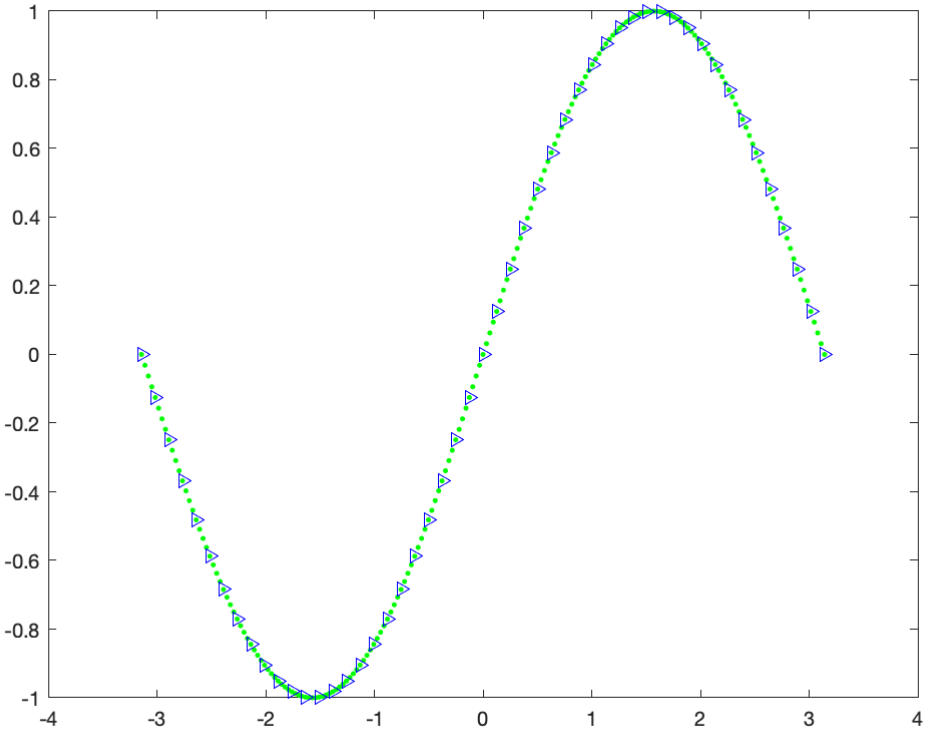



```
plot(x,y)  % plot from above is replaced entirely here
hold on
plot(x, sin(2*x), 'r') % different colored plot overlaid
plot(x, sin(3*x), 'm') % yet another plot overlaid; 3 total now
hold off
```



To make the graph have one color, and the data points another, plot twice with hold on:

```
plot(x,y,'g.')  % draw the original graph
hold on
plot(x(1:4:end),y(1:4:end),'b>') % Mark every 4th datum a different color
```



Clear a figure without removing the window with "clf":

```
clf
```

Multiple *independent* plots can also be created in a single figure window using the "subplot" command. You are not responsible for this in ENME202, but it is a useful command to be aware of:

```
help subplot
```

```
  subplot Create axes in tiled positions.
     H = subplot(m,n,p), or subplot(mnp), breaks the Figure window
     into an m-by-n matrix of small axes, selects the p-th axes for
     the current plot, and returns the axes handle.  The axes are
     counted along the top row of the Figure window, then the second
     row, etc.  For example,

         subplot(2,1,1), PLOT(income)
         subplot(2,1,2), PLOT(outgo)

     plots income on the top half of the window and outgo on the
     bottom half. If the CurrentAxes is nested in a uipanel the
     panel is used as the parent for the subplot instead of the
     current figure.
```

A final note:

Matlab plots are extremely powerful, and we are just touching the surface here. There is a wide variety of 2D and 3D plot options, and your code can also manipulate plots at a very low level using more advanced commands than covered here.