LESSON 05: PLOTTING

We've toyed a lot with how to perform calculations, but it's not awfully useful if
we can't visualize them. Thankfully, MATLAB offers us a wide array of plotting
functionality to help us visualize our data.

Graphic Objects

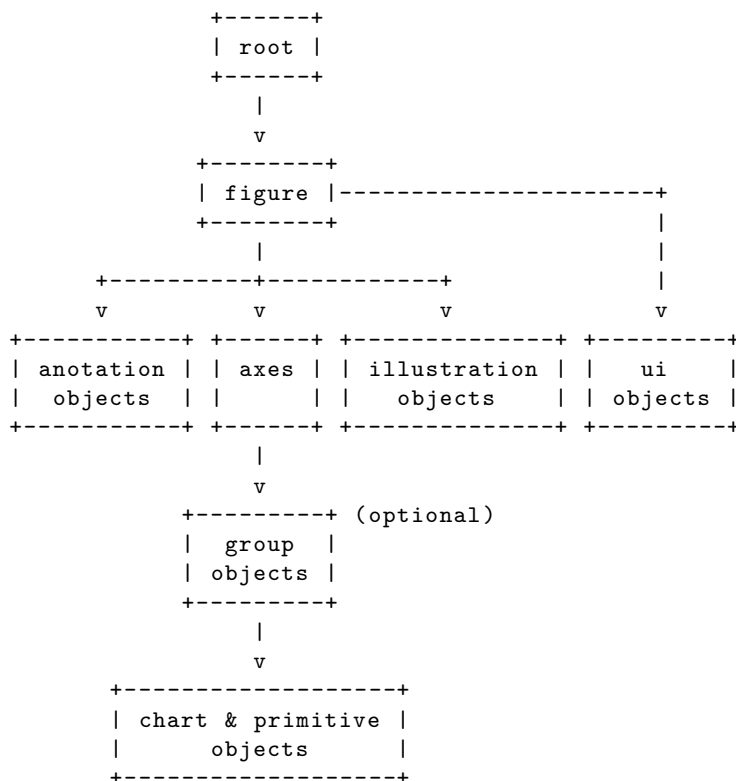MATLAB organizes graphical objects into a hierarchical structure[1]:

```
                    +------+
                    | root |
                    +------+
                       |
                       v
                  +--------+
                  | figure |---------------------+
                  +--------+                      |
                      |                           |
          +----------+-----------+                |
          v          v           v                v
    +----------+ +------+ +--------------+ +---------+
    | anotation | | axes | | illustration | |   ui    |
    |  objects  | |      | |    objects   | | objects |
    +----------+ +------+ +--------------+ +---------+
                     |
                     v
                +---------+ (optional)
                |  group  |
                | objects |
                +---------+
                     |
                     v
          +------------------+
          | chart & primitive |
          |      objects      |
          +------------------+
```

Figure 1: Organization of Graphics Objects

---

[1] https://www.mathworks.com/help/matlab/graphics-objects.html

2-D Plots

There's not much point in explaining. Let's jump right into an example.

```matlab
1  x = linspace(0, 2 * pi, 1e3);
2  y = sin(x);
3
4  figure;
5  plot(x, y);
```
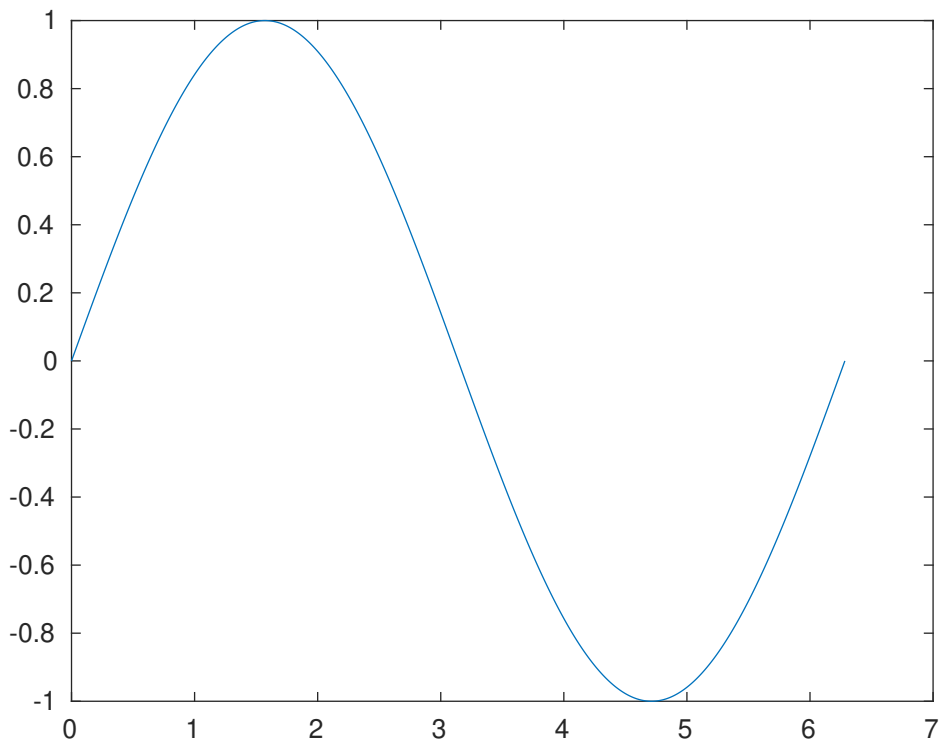

Here `figure` creates a new figure that `plot` can interact with. Although we successfully plotted our sine wave, it's quite ugly. Let's take the liberty to make it a little more pretty.

```matlab
1   x = linspace(0, 2 * pi, 1e3);
2   y = sin(x);
3
4   figure;
5   plot(x, y);
6   title('sin(x)');
7   xlabel('x [rad]');
8   xlim([x(1), x(end)]);
9   xticklabels({'0', '\pi/2', '\pi', '3\pi/2', '2\pi'});
10  xticks([0, pi / 2, pi, 3 * pi / 2, 2 * pi]);
11  ylabel('y');
```
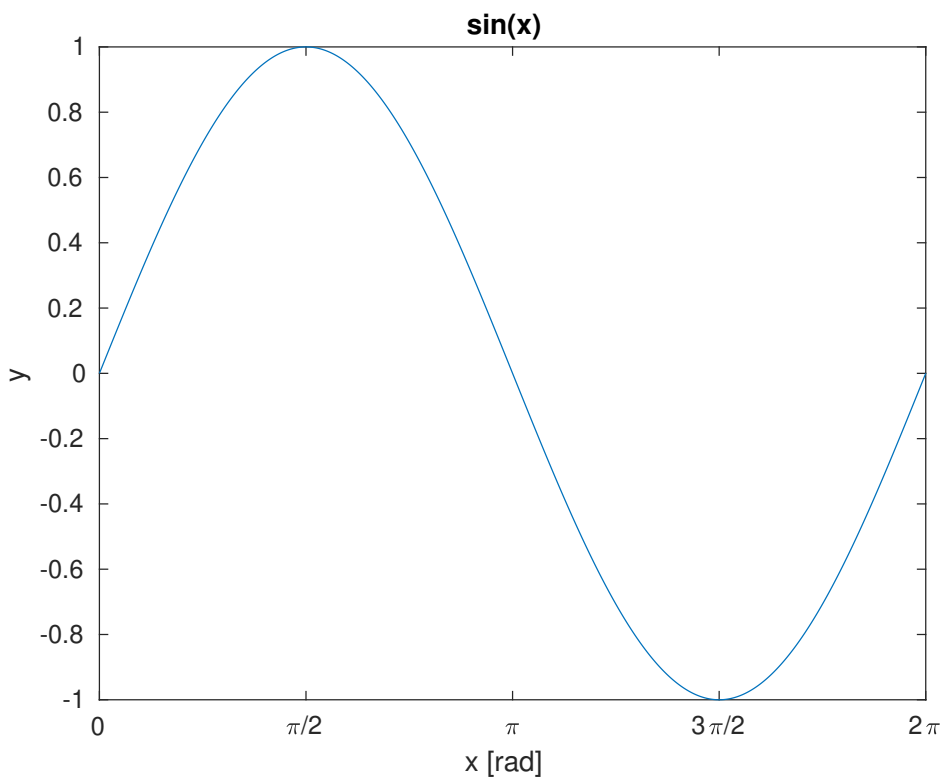

Retaining Plots

Say we wanted to plot multiple functions on the same plot:

```matlab
1   x = linspace(0, 2 * pi, 1e3);
2   f = {@sin, @cos, @tan};
3
4   figure;
5   hold on;
6   for i = 1:numel(f)
7       plot(x, f{i}(x));
8   end
9   xlabel('x [rad]');
10  legend('sin(x)', 'cos(x)', 'tan(x)');
11  xlim([x(1), x(end)]);
12  xticklabels({'0', '\pi/2', '\pi', '3\pi/2', '2\pi'});
13  xticks([0, pi / 2, pi, 3 * pi / 2, 2 * pi]);
14  ylim([-2, 2]);
```

Aside—
`hold` on can be a back stabber! Sometimes you may experience buggy behavior if you call it before the first plot.

github.com/jacobkoziej/jk-ece210
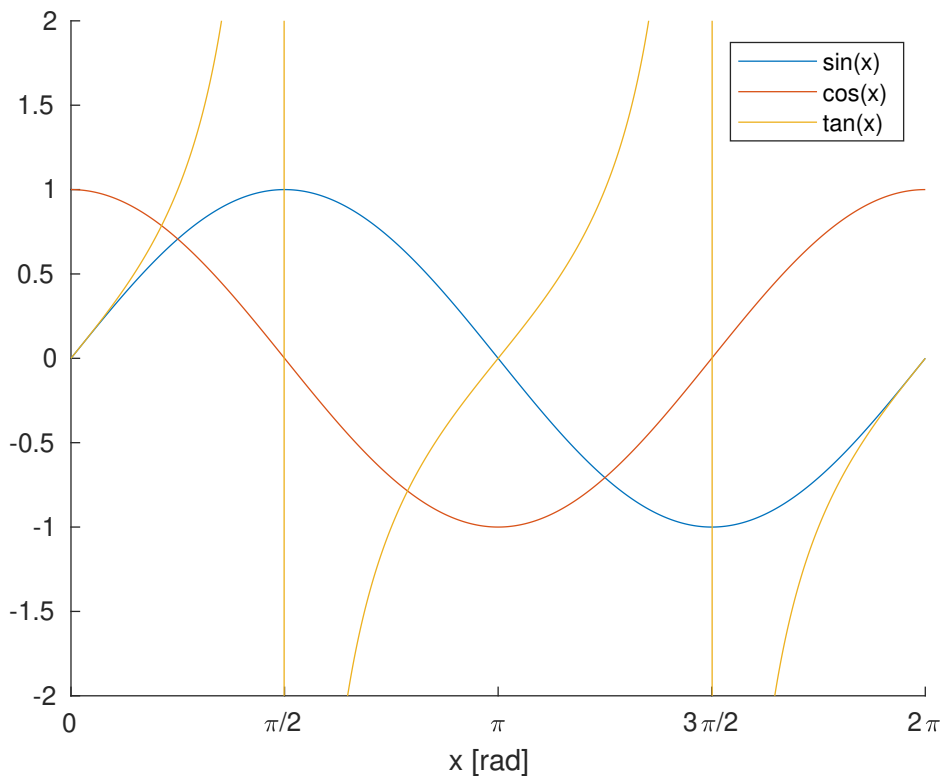
Subplots

We can have multiple plots in a figure by doing the following:

```matlab
t = linspace(-10, 10, 1000);
n = -10:10;

figure;

subplot(2, 1, 1);
plot(t, sinc(t));
title('sinc(t)');

subplot(2, 1, 2);
stem(n, sinc(n), 'filled');
title('sinc[n]');
```
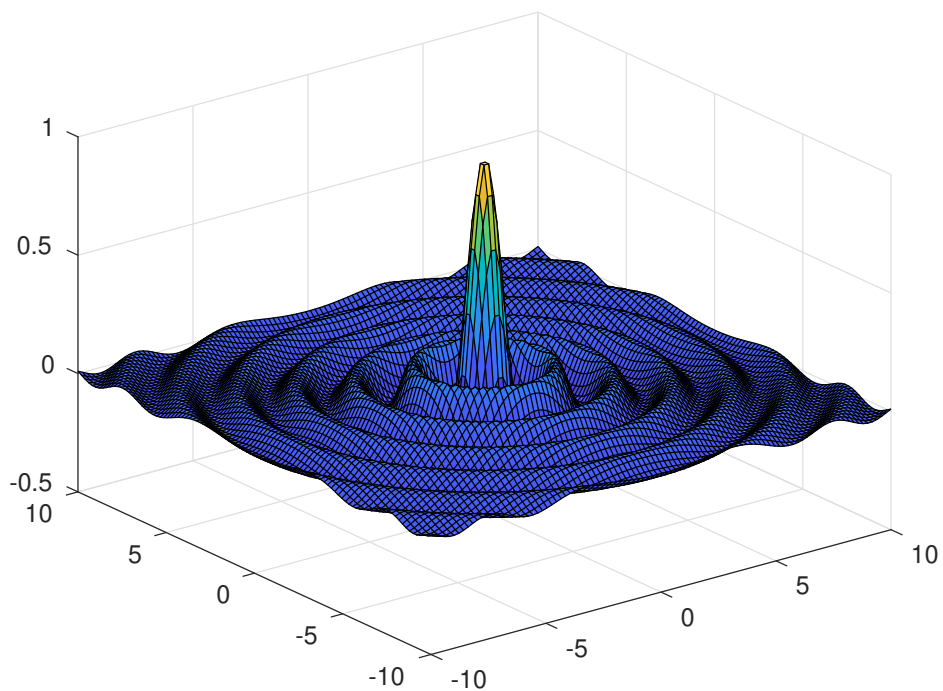
Aside—
Notice the zero-crossing property of the sinc function!

sinc(t)

sinc[n]

　　　　　github.com/jacobkoziej/jk-ece210
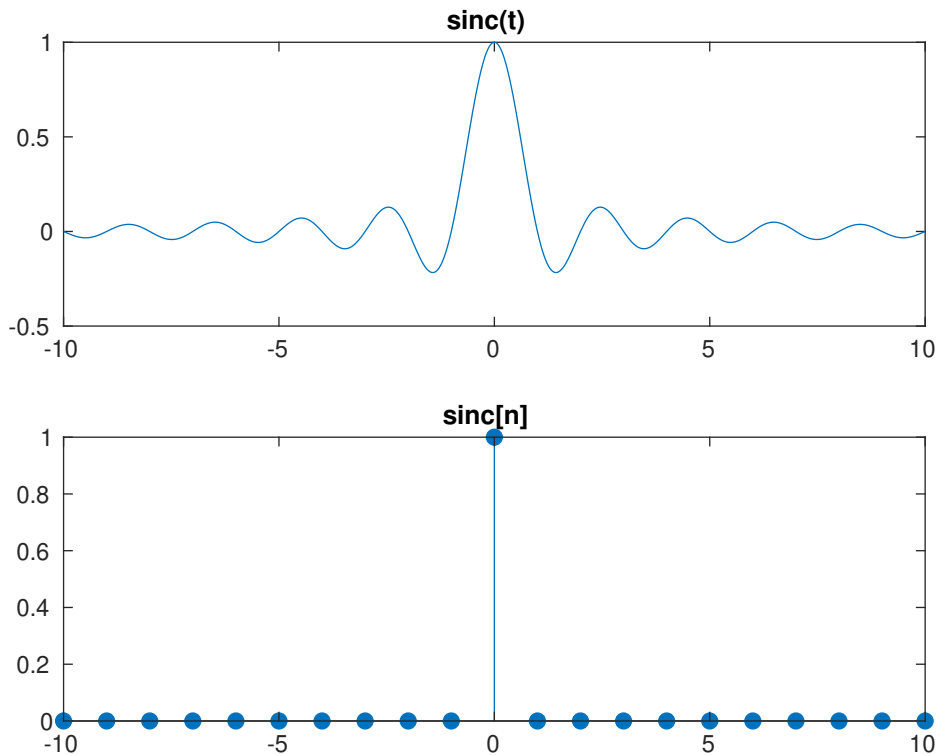
3-D Plots

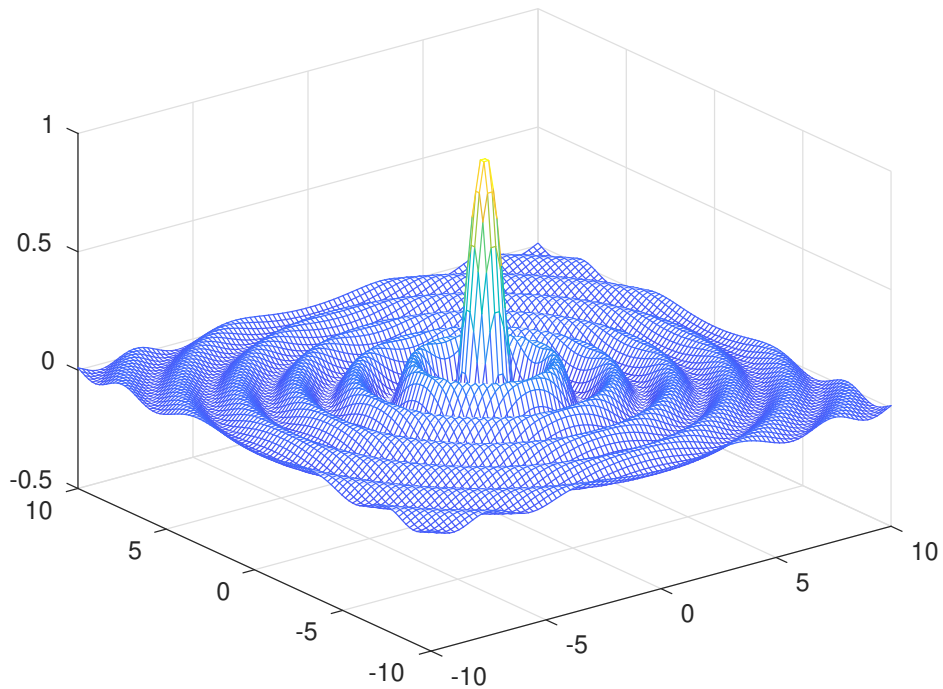MATLAB is also capable of making 3-D plots:

```matlab
1  x = linspace(-10, 10, 100);
2  y = x;
3
4  [X, Y] = meshgrid(x, y);
5  R = sqrt(X.^2 + Y.^2);
6
7  figure;
8  surf(X, Y, sinc(R));
```


You can think of `meshgrid` as the Cartesian product of the two input vectors split over two output matrices. Having these matrices allows us to evaluate functions over a 2-D plane.

We can also use a mesh instead:

```
1  figure;
2  mesh(X, Y, sinc(R));
```


More Plotting Functionality

This barely scratches the surface of MATLAB's extensive plotting functionality. Once again, the online documentation is a great starting point for the interested reader: `https://www.mathworks.com/help/matlab/2-and-3d-plots.html`

`github.com/jacobkoziej/jk-ece210`