Opening Discussion

■ What did we talk about last class?
■ Do you have any questions about the reading?
We finished last time looking at how to use fzero. We should go ahead and finish that discussion to see if we can use it to find the roots of a function.
Another common problem is trying to find the smallest or largest values that a function takes in a certain range.

Mathematically, these “local extrema” are points where the first derivative is zero so you just do root finding on that.

“Hill climbing” methods come in many forms. Their simplest forms wind up working in a manner very similar to root finding.

Matlab provides you with methods for doing minimization of functions: fminbnd and fminsearch.
Interpolation is the process of estimating values between data points.
Matlab has the functions interp1 and interp2 for doing interpolation in 1-D and 2-D.
The last argument is a string that tells the type of interpolation: linear, cubic, spline, and nearest.
Let's play with some 1-D interpolation and see how the different methods work.
Matlab can also do 2-D interpolation with interp2. This is easy to do if you have a 2-D array of data and want to find one point.

To construct a finer mesh we need the meshgrid function that will give us an easy way to represent all points on a grid. It takes two 1-D arrays and returns two 2-D arrays.

Let's play with this as well. We can use the mesh function to plot out surfaces.
Matlab provides a simple mechanism for us to deal with polynomials. Row arrays are can be viewed as the coefficients on polynomials.

Given this form, roots finds the roots of that polynomial.

Given the roots, poly will return the polynomial with those roots. (This one can be fairly easily done by hand.)

The conv function will multiply two polynomials.

Addition can be done easily if the polynomials have the same number of terms. Otherwise one will need to be padded with zeros.

For division use deconv.
You can take derivatives of polynomials with polyder.

You can integrate a polynomial with polyint. Remember that this must be passed in a constant along with the polynomial.

The polyval function will evaluate a polynomial at one value or for an array of values.
Given data, you can use the polyfit to fit a polynomial to it. The arguments are the x and y values followed by the order of the polynomial you want back.

In general you should use lower order polynomials, they are typically better behaved.

It turns out this is just a wrapper function for solving a system of linear equations, typically a system that is overspecified.
Because we are moving through topics slower than I have done in previous years (though I feel good about the depth of coverage) I've pushed back assignments 4 and 5.