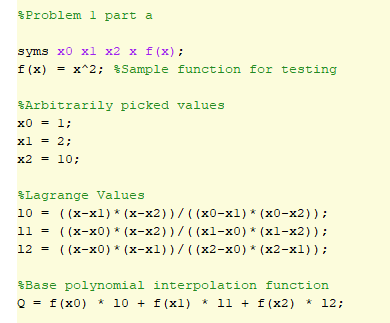
Jonathan Pilling

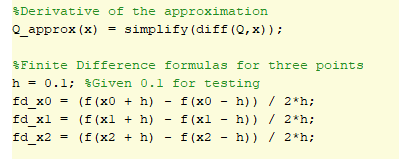
HW2 Writeup

CS 3200

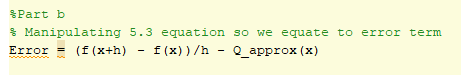
1. Problem 1 Finite Differences
   1. Starting with a sample function x^2, I set up some arbitrarily picked values for x0, x1, and x2. Using these values in the LaGrange formulas, and then plugging this into our polynomial interpolant formula I got a function that was similar to the starting function.



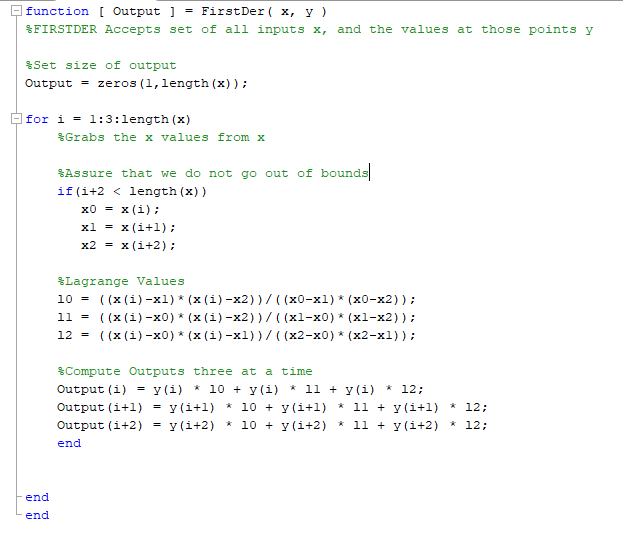
Taking the derivative of this Q function gave me the derivative of the original function. Using the finite distance formulas for points I was able to compute the values for x0, x1, and x2.

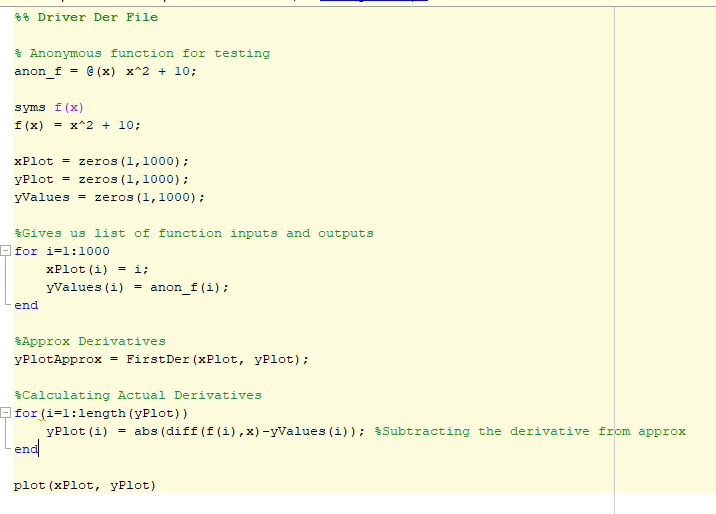


* 1. Manipulating Equation 5.3 from the notes gave me a formula where you subtract from the terms with h’s in it, the derived, derivative approximation. Given below, with test points this gave me error estimate equal to h.

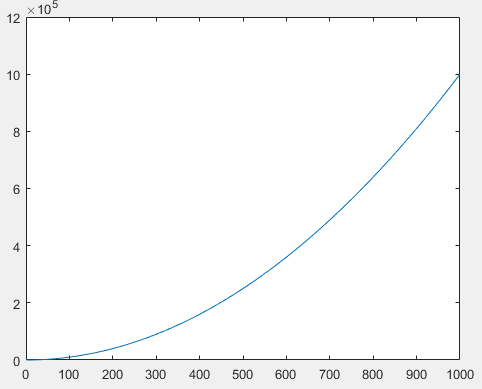


* 1. Writing the FirstDer() function in Matlab, I wrote it as specified in the assignment. Taking two inputs x and y, I iterated through these valid values up to three points. With these grabbed points I calculated LaGrange values and calculated three outputs at a time. In DriverDer.m I did a similar thing. Getting y values for the anonymous function in a for loop, I used these values to approx. derivatives by calling FirstDer function. Then in another loop I calculated all the actual derivatives, and found the error between approximations and actual derivative values. Below is some snippets of my code from FirstDer() and DriverDer.



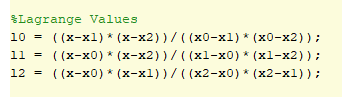


* 1. Plotting the relative error, this did increase as the number of points increased. Matching my derivation from part b.



The x axis here is the x values for FirstDer(). The y axis is the error of approximation.

1. Problem 2 Quadrature
   1. I generated an approx. for the function -x^2 + 1 after declaring the function f(x) I set x0, x1, and x2 as values within the integral. These were sort of given in assignment specification, I chose the range -10 to 10 for my integral. LaGrange formulas were setup with LaGrange values x0, x1, and x2.

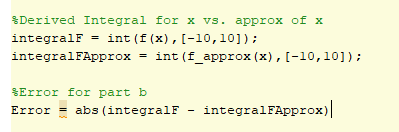


After this I used the Polynomial Interpolant equation to derive an approximation.



F(x) is in this equation was the original function I was trying to approximate.

* 1. The error value for the approximation is the Integral of the function minus the Integral of our approximation. If we got a negative value for this, we would need the absolute value. Since my approximation was simple, I always got an error of 0. Adding more points could change this error value.



* 1. Similar to interpolation, ends up being good in the middle of the integral but bad at the function extrema. To alleviate, one solution is to divide [a,b] into n parts, use trapezoidal rule on each part individually.