Pranav Rajan
February 22, 2019
Professor Varun Shankar

Polynomial Interpolation Analysis

Synopsis

The purpose of this assignment was to learn about finite differences and how scientists compute approximations of integrals and derivatives given very little data for a function.

1.

(a) Generate an O(h 2) (second-order) approximation to the first derivative of some function f(x) using 3 points, x0, x1 and x2 using the polynomial interpolation approach. Give the finite difference formulas for each of the points x0, x1, and x2. Assume they are equally spaced, with spacing h.

The first thing that I did was create a number line with x0 and x1 and x2 evenly spaced with a distance h between each of these points.

$$p(x) = \sum_{k=0}^{N} y_k \ell_k(x), \tag{9}$$

$$\ell_k(x) = \frac{\prod\limits_{\substack{j=0,\\ j\neq k}}^{N} (x - x_j)}{\prod\limits_{\substack{j=0,\\ j\neq k}}^{N} (x_k - x_j)}. \tag{10}$$

Using the above equations for lagrange polynomial interpolation, I calculated the lagrange bases functions for l0, l1,l2.

$l_0(x) = (x-x_1)*(x-x_2)/(x_0-x_1)*(x_0-x_2)$

$l_1(x) = (x-x_0)*(x-x_2)/(x_1-x_0)*(x_1-x_2)$

$l_2(x) = (x-x_0)*(x-x_1)/(x_2-x_0)*(x_2-x_1)$

Using the lagrange bases, I simplified these terms into expressions containing h which represents the space between points

$l_0(x) = (x-x_1)*(x-x_2)*1/2h^2$

$l_1(x) = (x-x_0)*(x-x_2)*-1/h^2$

$l_2(x) = (x-x_0)*(x-x_1)*1/2h^2$

Using the simplified lagrange bases I then computed the derivate for each of the simplified lagrange bases above

$l_0'(x) = (2x-x_1-x_2)*1/2h^2$

$l_1'(x) = (2x-x_0-x_2)*-1/h^2$

$l_2'(x) = (2x-x_0-x_1)*1/2h^2$

Using the derivatives of the lagrange basis, I computed the derivative of the lagrange polynomial interpolation

$p'(x) = y0*(2x-x_0-x_1)*1/2h^2 – y1*(2x-x_0-x_2)/h^2 + y2*(2x-x_0-x_1)/ 2h^2$

After computing the derivative for the lagrange interpolation polynomial, I evaluated x0, x1, and x2 in this equation. Computing the derivative at x0 yields the forward difference, computing the derivative at x1 yields the center difference and computing the derivative at x2 yields the backward difference. I also simplified these equations in terms of h such that all the derivatives at the above respective points only contain h.

p'(x0) = -3y0/2h + 2y1/h – y2/2h (forward difference)

p'(x1) = -y0/2h + y2/2h (center difference)

p'(x2) = y0/2h-2y1/h+3y2/2h (backward difference)

b) Using the error term for polynomial interpolation, show that the error above is O(h 2).

To compute the error, I used the following equation from the lecture notes

$$f^{(n)}(x_j) = p^{(n)}(x_j) + \frac{f^{(N+1)}(\xi_{x_j})}{(N+1)!} e^{(n)}(x_j). \tag{15}$$

The following equations are the results I got from computing the error of the first derivative at x0,x1, and x2. The p'(x0), p'(x1), p'(x2) refer to the finite difference equations derived in part a.

$f'(x0) = p'(x0)+c*(2h^2)$

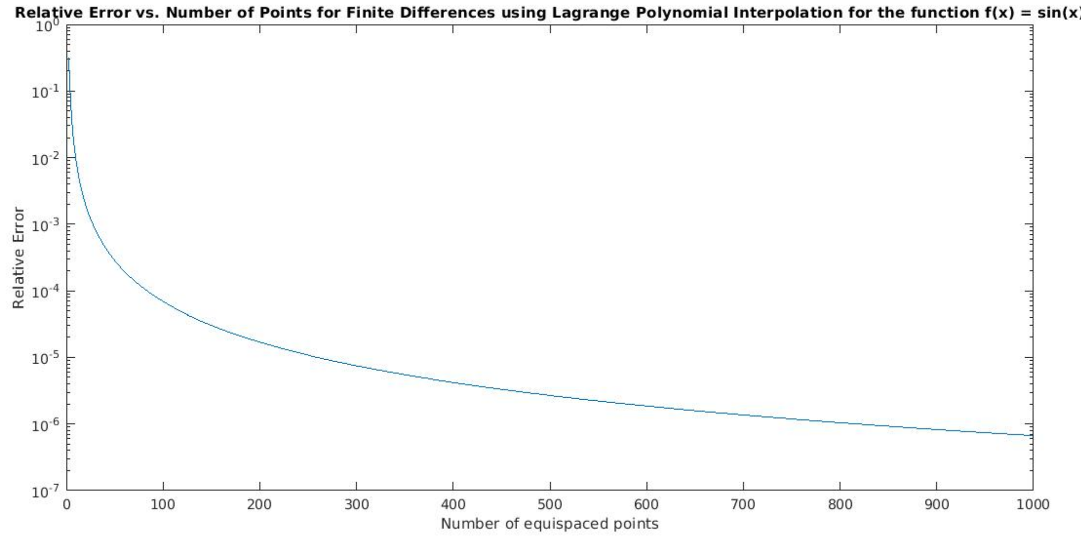
$f'(x1) = p'(x1)+c*(h^2)$

$f'(x2) = p'(x2)+c*(2h^2)$

If we observe the rate of growth for each of these equations, we see that the term that dominates each equation is the $c*(2h^2)$ or the $c*(h^2)$. Thus since h^2 dominates, the order for the error is $O(h^2)$

c) Check out the matlab code submitted for this assignment

d)



No. The graph does not match the error that I derived for part b. Looking at the graph, as the number of points increases the error is decreasing so as the number of points increases towards infinity, I would say that the error converges to some finite number which seems like it would be satisfy the $O(h^2)$ order for the error in part b.

2) (a) Generate an approximation to R b a f(x)dx using a quadratic polynomial interpolant to f. Assume you have 3 equispaced points, including the end points a and b; that is, let x0 = a, x1 = (a + b)/2 and x2 = b be the 3 points.

For this part, I used the following equation

$$\int_{x_0}^{x_1} f(x)dx \approx y_0 \int_{x_0}^{x_1} \ell_0(x)dx + y_1 \int_{x_0}^{x_1} \ell_1(x)dx, \qquad (32)$$

Using matlab, I computed the integral for the quadrature for three points for some function f(x). As mentioned in the assignment 2 discussion, I have left my result in terms of x0, x1, x2 and y0, y1, y2. The following result is what I computed. Instead of going from x0 to x1 as the bounds for the integrals, I set the bounds to be from x0 to x2 as specified by Professor Shankar in the assignment since we are finding the integral over three data points.

-y0((x0-x2)*(2x0-3x1+x2))/6*(x0-x1) – y1$(x0-x2)^3$/6*(x0-x1)*(x1-x2)+ y2((x0-x2)*(x0-3x1+2x2))/6*(x1-x2)

(b) Derive the corresponding error term for this quadrature rule

Professor Shankar announced on Friday that we can skip this question.

(c) What is the problem with using equally-spaced points to generate quadrature rules as the number of points increases? How do you alleviate this problem?

The problem with using equally-spaced points to generate quadrature rules is that the Runge Phenomenon will occur. As shown in assignment 1, the error increases as we increase the number of points when we have equally spaced points. Another way to analyze this is to observe the error for polynomial interpolation and the error for quadrature in the following order

$$f(x) - p_N(x) = \frac{f^{(N+1)}(\xi_x)}{(N+1)!} \prod_{k=0}^{N} (x - x_k). \tag{15}$$

Polynomial Interpolation Error

$$\left| \int_{x_0}^{x_1} f(x)dx - \int_{x_0}^{x_1} p(x)dx \right| \leq \frac{M_2}{2!} \int_{x_0}^{x_1} e(x)dx, \tag{36}$$

$$\leq \frac{M_2}{2} \int_{x_0}^{x_1} (x - x_0)(x - x_1)dx, \quad \leq \frac{M_2}{12}(x_1 - x_0)^3. \tag{37}$$

Quadrature Error

As we increase the number of points in both of these the M2/12 in the quadrature error f(N+1) term in the polynomial interpolation error grow bigger much faster than the product of (x-xk) terms where k = 0..N. This issue can be fixed by using Gaussian quadrature (based on Legendre zeros) and Clenshaw Curtis Quadrature (which uses Chebyshev spaced points).