Sophia Ardell

Computational Physics

September 30, 2015

Homework 4

**Problem 1**

This code considers the trajectory of the Earth if it were to stop in its orbit (of radius R0) and started falling towards the sun. The solution for the Earth’s distance from the sun *r*, as a function of time *t*, can be expressed as:

where . This function, *r(t)*, is only known parametrically.

This code allows the user to decide how many data points they want. From there, a createdData object is made, which holds radius and time as a function of η.

These data points are then set with the set\_data\_points() function. In order for m data points to inclusively range from radius of 0 to the distance of earth from the sun, and given that r is a function of cos(η), η must range from 0 to PI. The code loops over 180 degrees with η increasing by a factor of , this way η includes both 0 and PI. The radius and time are calculated as a function of this. The createdData struct also holds functions allow the user to retrieve specific time or radius values, or the entire time or radius vector. From there, the main function creates vectors that hold radii and times. The maximum amount of days is also retrieved by taking the last value in the time vector, as it will be the greatest value and can serve as an upper bound. Because time is in seconds and we want our answer in days, the time value needs to increase by the amount of seconds in 10 days. For every 10 days, I used interpolation on the time and radius vectors, and calculated r as a function of a specific time input. All of this was wrote to a file. Results are as follows:

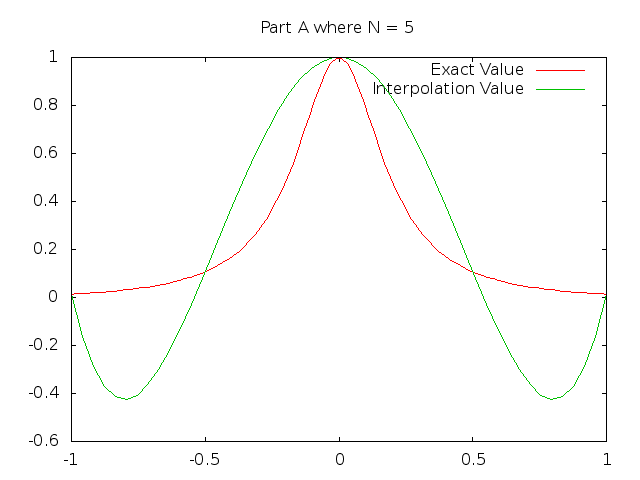
Days Distance (m)

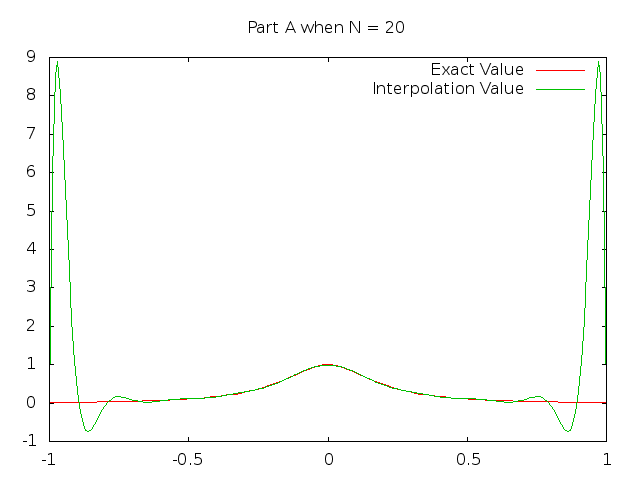
|  |  |
| --- | --- |
| 0 | 1.50E+11 |
| 10 | 1.47E+11 |
| 20 | 1.41E+11 |
| 30 | 1.29E+11 |
| 40 | 1.11E+11 |
| 50 | 8.45E+10 |
| 60 | 4.25E+10 |

**Problem 2**

This code runs for n equal to 5 and 20, as specified by the assignment. The code first takes input size and creates a larger value for interpolation size by multiplying n by 10. It then establishes vectors that hold the true values of both the original data set of n values [-1,1] and 10n values [-1,1] in accordance with function (3) given in the homework. The code then interpolates the data, and creates a vector that holds the interpolated values. The code writes the original x and y values, x and y values interpolated, relative error, and absolute error to a file for every index value. Relative error is calculated by the dy function included in the “interp\_1d.h” file. Absolute error is the magnitude of the difference between the interpolated value of y at a given x, and the true value of y at the same x. The code then presents the opportunity to find the coefficients of the original function. It uses both the method polcoe and polcof, which find coefficients using two different methods, and return them all in a vector. These values are written to a separate data file.

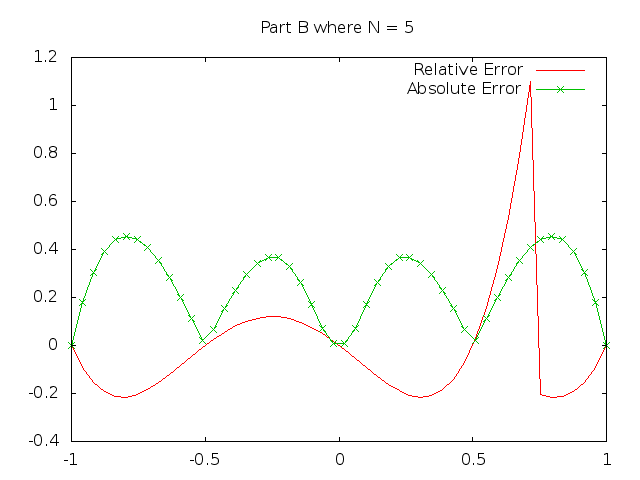
Part A:

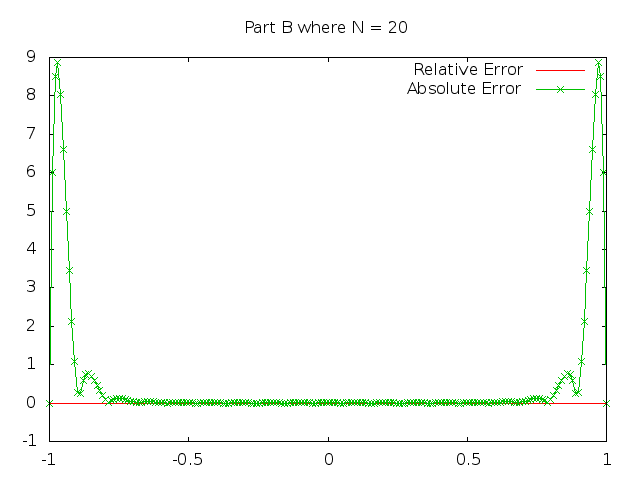




For part B, we were asked to see if the error estimates were a reflection of the true error of the interpolation. Even though the chart for N = 5 appears generally worse, when examining the scale of the graphs, it is actually much closer in value to the absolute error than the values in N = 20. Additionally, if you took the absolute value of the relative error, it would look even more similar. In N= 20, though it is good at estimating middle values, at the extremes, the estimate is 0, which is a far distance from an absolute error value of 9. Because of this extreme difference, it cannot be considered the better performing function.

This may be because of the imaginary poles in the function. Interpolation also assumes that the (n-1) derivatives are continuous, which is not the case here. Because of this, the derivative will be 0 near . You could verify this by taking the derivative of function (3).





Part C:

The function is an even function, meaning that

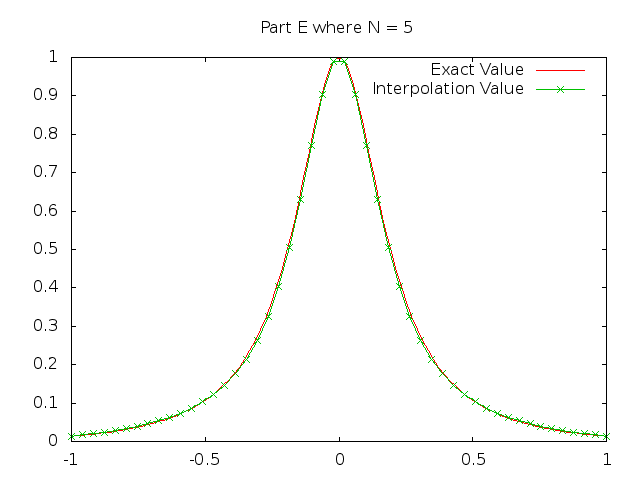
Therefore, all the odd order coefficients must be 0. The coefficient c0 is 1, given that at x = 0, the function is equal to 1 and all terms with an x in it vanish, so c0 is the only one left.

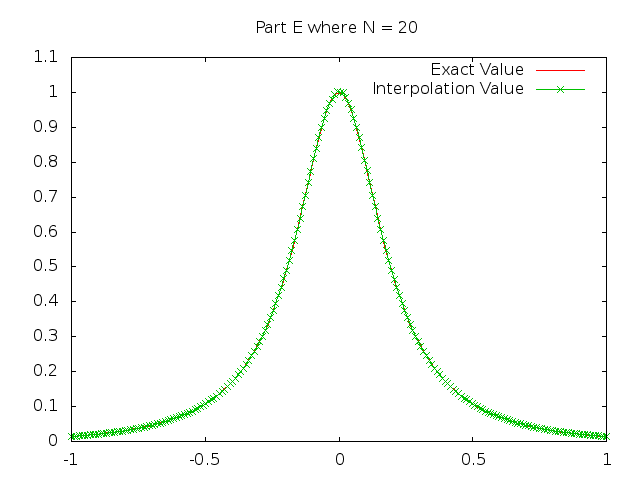
Part D:

I relatively find the properties I was expecting in C, as the odd indexes all have values that are approximately equal to zero. See attached data sheet for actual values.

Part E:

[implementing rational functions instead of polynomial]





It makes sense that the rational functions are a far better fit for this, given that they can model functions with poles, unlike polynomials.