Homework 5: Program Report

John Wu

29 April, 2020

1 Executive Summary

In class, we've been looking at derivatives and integrals as of late. Now, we've moved to ordinary differential equations. In particular, we take a look at the Euler's method - a subset of the larger branch of Taylor's Series method – and the Runge Kutta method. The implementation of these methods were very simple and yielded expected results. In my testing I have yet to find a combination of values and functions that would cause inconsistencies between the outputs of my code or otherwise creating errors in my code.

2 The Math

The math behind the code is really quite simple, at least for the methods and step sizes that we are using.

Euler's Method: For Euler's method, we are simply and mainly interested in only the first derivative of any function. Euler's method is nice due to being the same in principle as the Taylor series method, we only need to concern ourselves with the first iteration each time because it is of order 1. There is no need to compute any higher order derivative than the first and there is no need to compute any factorials in the computation of our estimation. The equation used is as shown below:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Where y is the value of the original function and f derivative of the original function. H, as always, is our step size.

4th Order Runge Kutta Method: This one is also quite simple, though a bit more computationally tedious. To put it simply, in this method we simply take the averages centered about the step sizes we use to compute the value of the original function at the intended point. The equation is as shown below:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

These k values are essentially the points that we are taking our average about, they are calculated by the following formulas:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + .5h, y_i + .5k_1h)$$

$$k_3 = f(x_i + .5h, y_i + .5k_2h)$$

$$k_4 = f(x_i + h, y_i + k_3 h)$$

3 Implementation

My code uses the variables given to us in the assignment. Y is my output, with inputs fname, a, b, s, and n. These inputs are used to pass into my program, the file that holds the function we will be using, the starting point of our interval, the ending point of our interval, the initial value given to us, and the number of steps we will be using. We combine, b, a, and n in order to calculate h, our step size. In my code, I created multiple temporary variables to store the results of my previous iteration, which I could've used the values I stored in my array to the same effect, but I chose to keep the temporary variables for the sake of readability. The for loops I've created in the programs follow very similar structures, I take the values from the current function and derive the value for my next function from that.

4 Results

In the end, the results are basically how I imagined them to be. I utilized multiple of the homework problems that we had already solved to test my code ensuring that the results matched up. Afterwards, I could be sure that if the values were correct for this set of intervals and step sizes, all variations proceeding it should follow suit. Therefore, I turned to attempting to use values that would make my code either return errors or give impossible values to no avail. I used oscillatory functions, and steep functions such as sines and $1/x^t$. Sine functions and steep functions seem to work as expected, giving some sort of number as a return. The accuracy of these numbers are unconfirmed however. I hoped to use these numbers to cause some deviation or error in my code, but quickly realized during testing that it would only be likely in the event that I used some number less than 0. Testing this steep function with 0s expectantly gave me numbers of both great magnitude and undefined magnitudes.

5 Conclusion

Given the results as discussed in the previous section, I think it's safe to say that the program works as intended. I think it would be interesting to test more functions that are known to return such oddities to observe their effects on my code even further. I struggled most on keeping track of the values I needed to compute my next iteration for both functions. Keeping track of what iteration I was on, along with getting the correct x value and y value were a struggle for me. Eventually I found that I would be able to get the correct x value by multiplying the current iteration I was on by my step size and keeping a temporary variable handy at all times.