CST-305: Runge-Kutta Solver

1 February 2024

Gerum Haile-Lincoln, Carlos Hernandez

Professor Ricardo Citro

Responsibilities

|  |  |  |
| --- | --- | --- |
| Task | Member(s) | Status |
| Brainstorm Algorithm | Carlos & Gerum | Completed |
| Implement ODEInt | Gerum | Completed |
| Implement Runge-Kutta | Carlos | Completed |
| Develop documentations | Carlos & Gerum | Completed |
| Documentation A-E | Carlos | Completed |
| Documentation F-I | Gerum | Completed |
| Python Implementation | Carlos & Gerum | Completed |

Manual Solving:

|  |  |  |  |
| --- | --- | --- | --- |
| Method: RUNGE-KUTTA METHOD | | | |
| Problem: | | | |
|  |  | | True Solution |
|  |  |
|  |  |  | 5 |
|  | 1.02 | 5.06 | 5.058 |
|  | 1.04 | 5.12 | 5.114 |
|  | 1.06 | 5.18 | 5.169 |
|  | 1.08 | 5.234 | 5.224 |
|  | 1.1 | 5.28 | 5.277 |

Scratch Work:

A hand holding a piece of paper

Description automatically generated

A paper with writing on it

Description automatically generated

A notebook with lines and numbers on it

Description automatically generated

Specific Problem Solved:

Runge-Kutta methods are used to solve ordinary differential equations, especially ones that are very complex to calculate by hand. Getting the averages of the interval by breaking the interval into subintervals helps in estimating the function’s value by the end of the interval. We calculate four intermediate k values at each step using the weighted averages of these values to estimate the slope. We use this slope to compute the function’s values at the next point.

Mathematical Approach to Solve it:

We need initial xo and yo values for the function. We also need an h value. For the k values at each function value we do: k1 = f(xn,yn); k2 = f(xn+h/2,yn+h/2\*k1); k3 = f(xn+h/2,yn+h/2\*k2); k4 = f(xn+h,yn+h\*k3). When we find all k values we find y1 and x1. In general we find the next y and x values this way: yn+1=yn+h/6\*(k1+2k2+2k3+k4); xn+1=xn+h. We keep going until our desired function values.

f. Approach for implementation in code

The main function implements odeint in order to solve the ordinary differential equation. We did this by defining the equation we wanted solved through a function called model, then defining our variables, and defining our x points using numpy’s arange() function. We then solve the ODE using a corresponding y variable which uses odeint which takes in the model, the starting y value and the x values. We then plotted this using matplotlib using the plot and label functions.

For the Runge-Kutta implementation, we used our own user-defined functions in order to solve the differential equation using the Runge-Kutta method. We used 3 helper functions, which were f, t4 and runge\_kutta. f acted as our equation, t4 acted as the t4 function shown on the padlet, and the runge\_kutta method implemented both of these helper functions in order to recursively build the solution and return it. The program then loops through all x values and builds a corresponding array of y values. We then plot the x and y values using matplotlib’s plot function.

g. First we define all of our starting values that are defined in the code.

A black background with white text

Description automatically generated

Then we have the 3 helper functions which represent different parts of the Runge-Kutta function shown on the padlet.

A screenshot of a computer program

Description automatically generated

Then the program loops through all of the x values and computes corresponding y values and plots it.

A screen shot of a computer program

Description automatically generated

Finally, the program outputs the graph. As shown.

A screen shot of a graph

Description automatically generated

h. Theory includes Runge-Kutta method for solving ordinary differential equations. The equations, function names and method of solving were based off Professor Ricardo Citro’s padlet. Code was written by Gerum and Carlos.

i. README will be attached to this project.