CST-305: Project 2 – Runge-Kutta-Fehlberg (RKF) for ODE

Asher Shores

College of Science, Engineering, and Technology, Grand Canyon University

CST-305: Principles of Modeling and Simulation Lecture & Lab

Dr. Ricardo Citro

October 3, 2021

**Project 2 – Runge-Kutta-Fehlberg (RFK) for ODE**

The objectives of this assignment are split into two parts: theoretical and practical. First by solving a mathematical problem using the RK4 method. Then by using the same computational problem as part one, however a program is written to solve it using the RK4 method and measure the performance of the computer.

**Responsibilities and Completed Tasks**

During the first part, the differential equation was solved manually on paper (demonstrated below) to demonstrate the RK4 method and its’ precision. A complementary python program was written to solve the same ODE using the same method recursively. The script also runs the ODE and solves via ODEint from SciPy. Both are plotted and graphed to compare visually as well as compared analytically using percent error.

**System Performance Context Description**

The Runge-Kutta method clocked in at 3.3729ms and ODEint took slightly longer at 4.748ms (typically). In general, Runge-Kutta should be much longer however, the machine this was run on hosts a Intel-CoreI9 processor and other strong measures. ODEint is optimized for such purposes while numerically solving via Runge-Kutta should generally be slower.

**Specific Problem Solved**

The specific problem that was solved was . The derivative of y in terms of x is equal to y divided by the quantity of minus y + the natural log of x. The initial conditions were x0 = 2, y0 = 1 and h = 0.3.

**Mathematical Approach**

Diagram, schematic

Description automatically generated

A picture containing diagram

Description automatically generated

Diagram, schematic

Description automatically generated with medium confidenceDiagram, schematic

Description automatically generatedDiagram, schematic

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Method: RUNGE-KUTTA METHOD | | | |
| Problem: | | | |
|  |  | | True Solution |
|  |  |
|  |  |  | 1 |
|  | 2.3 | 0.9399 | 0.939920 |
|  | 2.6 | 0.9292 | 0.929199 |
|  | 2.9 | 0.9511 | 0.951140 |
|  | 3.2 | 0.9942 | 0.994182 |
|  | 3.5 | 1.0503 | 1.050343 |

**Approach for Implementation in Code**

A picture containing diagram

Description automatically generated

**Screenshots Depicting Key Phases in the Program Execution**

***Code***

Text

Description automatically generated

Defining ODE model for use in solving

Text

Description automatically generatedRunge-Kutta numerical algorithm

Text

Description automatically generated

Solve using ODEint

***Code Output***

***Text

Description automatically generated***

***Graphs***

Chart

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated

References

“Learn Numerical Methods: Algorithms, Pseudocodes & Programs.” *Codesansar*, https://www.codesansar.com/numerical-methods/.

“Numpy Reference¶.” *NumPy Reference - NumPy v1.22.dev0 Manual*, https://numpy.org/devdocs/reference/.

“Runge-Kutta 4th Order Method to Solve Differential Equation.” *GeeksforGeeks*, 21 June 2021, https://www.geeksforgeeks.org/runge-kutta-4th-order-method-solve-differential-equation/.