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

Kyungchan Im

Grand Canyon University

Course number: CST - 305

Professor Ricardo Citro

Due Feb 5, 2023

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

Introduction Paragraph

**PART 1: Runge Kutta Calculation**

The Given ODE is .

Calculate manually for .

[ Runge Kutta Equation ]

[ RKE Calculation ]

1. Point

1. Point

1. Point

1. Point

1. Point

**Table Chart for RKF**

**Manual Calculation vs Python Program**

|  |  |  |  |
| --- | --- | --- | --- |
| Method: RUNGE-KUTTA METHOD | | | |
| Problem: | | | |
|  |  | | True Solution  (ODE) |
|  |  |
|  |  |  | 1 |
|  | 2.3 | 0.9399 | 0.939906 |
|  | 2.6 | 0.9292 | 0.929179 |
|  | 2.9 | 0.9511 | 0.951118 |
|  | 3.2 | 0.9941 | 0.994162 |
|  | 3.5 | 1.050 | 1.050326 |

**True Solution Execution from a Python Code (ODE)**

**Text

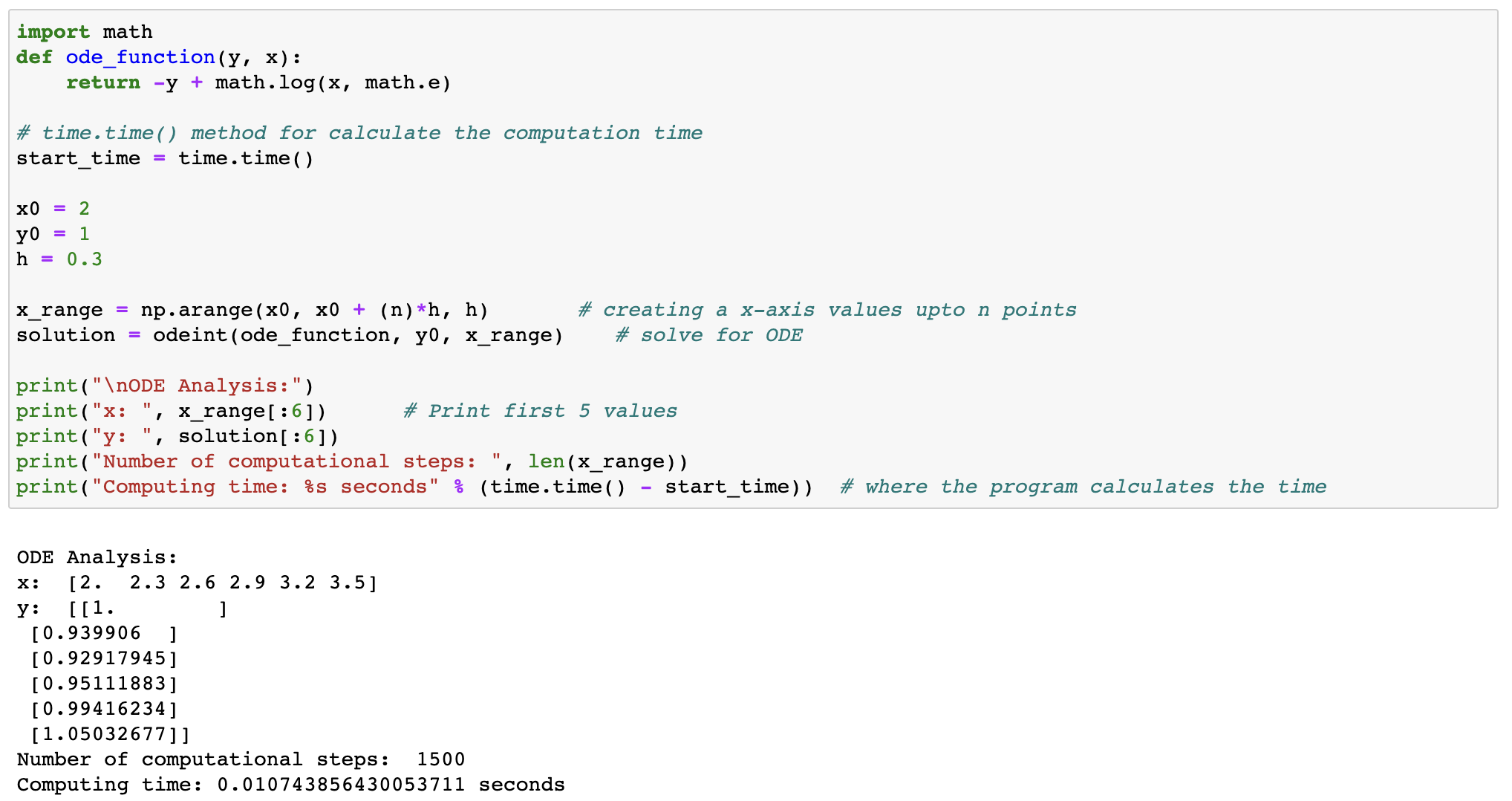
Description automatically generated**

**Analysis**

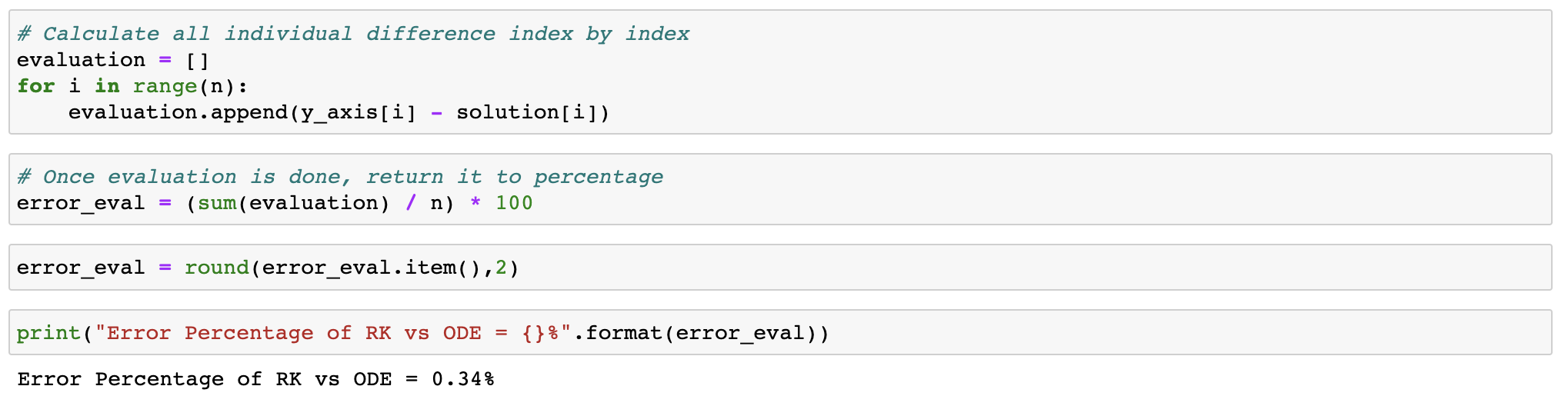
I performed both manual calculations and used a Python program to solve an ODE using the Runge-Kutta-Fehlberg method. The comparison of the results from both methods can provide an estimation of the precision of the calculations. I found that the manual calculations had a low error, and although there were some differences between the manual calculations and the program's calculations due to rounding the decimals, the results were very close. This highlights the importance of carefully checking the results and considering multiple methods when estimating the precision of numerical calculations. To ensure a high level of precision in the calculations, it is crucial to use formal mathematical rigor when writing and discussing the solution and intermediary steps. Additionally, it is useful to analyze the error between the calculated solution and the true solution to determine the accuracy of the calculations. One common way to do this is to calculate the relative error, which is the difference between the calculated and true solutions divided by the true solution. A smaller relative error indicates higher precision. By following these best practices, one can ensure a high level of precision in their numerical calculations.

**PART 2: Solve the ODE**

**[ Execution of ODE Python Function ]**

****

**[ Error Evaluation ]**



I ran a computational program to solve numerical problems and measured its average execution time. After conducting 10 trials, I found that the average time of execution was approximately 0.012277 seconds. The program is designed to select the appropriate number of computations based on the specific problem being solved. This time, I ran the program 1500 times to compare the error between the Runge-Kutta method and ODE (ordinary differential equations) in solving the problem. The error difference between the two methods was found to be very close. I calculated the error by finding the sum of the differences between each data point and dividing it by the total number of data points. The result showed that the percentage error was less than 1%. This indicates that both methods produced similarly accurate results for this problem.

**[ Visualization ]**

Chart

Description automatically generated

The results from the Runge-Kutta method and the ODE solutions are very similar, with the error percentage being less than 1%. This is reflected in the two graphs, where the red line represents the Runge-Kutta solutions, and the green dots represent the ODE solutions. The number of points in the graph is 1500, and the graph displays the range of both the minimum and maximum values of the x and y axis. The two graphs are a testament to the accuracy of the Runge-Kutta method. This method is widely used in numerical analysis due to its ability to accurately approximate solutions to ODEs, even with a relatively small number of points. By comparing the results of both the manual calculations and the program's calculations, one can be confident in the precision of the results. The close agreement between the two sets of results supports the conclusion that the Runge-Kutta method is a reliable tool for solving ODEs. The purpose for this project is to see the evaluation of RungeKutta, and by modeling and programming the both method, we are able to tell the two methods are efficient for solving ODE.

**References**

SciPy v1.10.0 Manual. (n.d.). scipy.integrate.odeint - Retrieved February 3, 2023, https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

Kong, Q., Siauw, T., & Bayen, A. M. (2021). *Python programming and numerical methods: A guide for engineers and scientists*. Academic Press, an imprint of Elsevier.