IMPROVING THE RUNGE KUTTA METHOD

A Project by ~
Keshav Singh (2K19/EE/134)

Introduction

While sending a satellite to another planet, it is often necessary to make a course correction mid-way. The differential equations governing the motion are well known, the projected path can be calculated by solving the differential equations concerned. This is just one example where we use differential equations. Differential equations are of a great use in real life, governing everything from motion to The Laws of Thermodynamics. So it becomes extremely important to be able to solve these equations correctly and with precision. There are various tools/methods that come handy when it comes to solving differential equations, the **Runge Kutta Method** is one of them.

AIM of the PROJECT

In this project we focus on increasing the accuracy as well as performance of Runge-Kutta method in solving Differential Equations.

1) To increase the accuracy we take the order of Runge-Kutta method two steps ahead. We were introduced upto 4 orders of Runge-Kutta Method in our classes, but here we take it forward to two more orders introducing 5th and 6th order Runge-Kutta Methods.

2) To enhance the performance of Runge-Kutta Methods we will implement an adaptive step-size modulation in the Runge-Kutta algorithm using which the program will accordingly alter the step size intelligently to reduce calculation time without compromising with the accuracy.

Need for the topic

As discussed in Introductory slide, Runge-Kutta methods are used to solve differential equations governing the path of motion of satellites, where a mistake of even the order of 10^{-9} can lead to a massive disaster. So it becomes extremely necessary to constantly be updated with the algorithms that provide us with most accurate results. Currently, 4^{th} order Runge Kutta method is the most accurate algorithm for finding the solutions of a differential equation. So in this project, we're going to try to improve the accuracy as well as performance of the Runge-Kutta Methods.

Runge kutta method

- Runge-Kutta methods are a class of implicit and explicit iterative methods which judiciously uses the information on the 'slope' at more than one point and making use of temporal discretization for approximating the solutions of ordinary differential equations.

- These methods were developed around 1900 by the German mathematicians ***Carl Runge*** and ***Wilhelm Kutta.***

- This method is by far gives the most accurate output, so much so that it is used in most of the online calculators.

Implicit Runge-Kutta Method

Runge–Kutta methods are methods for the numerical solution of the ordinary differential equation

$$\frac{dy}{dt} = f(t, y).$$

Explicit Runge-Kutta methods take the form

$$\begin{aligned}
y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i \\
k_1 &= f(t_n, y_n), \\
k_2 &= f(t_n + c_2 h, y_n + h(a_{21} k_1)), \\
k_3 &= f(t_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2)), \\
&\vdots \\
k_i &= f\left(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right).
\end{aligned}$$

Stages for implicit methods of s stages take the more general form

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right)$$

Each method is defined by its Butcher tableau, which puts the coefficients of the method in a table as follows:

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Butcher's Tableau for RK4

Classical RK-4 method uses the following Butcher Tableau for solving a Differential Equation

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

for n = 0, 1, 2, 3, ..., using

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$

RK-5 and RK-6 working formula

This is a cropped image of the working formula for Runge-Kutta 5th and 6th order Methods from the book

"Numerical Analysis" 9th Edition by ~

Richard L. Burden
Youngstown State University
J. Douglas Faires
Youngstown State University

**Original Formula was developed by JH. Verner

The Runge-Kutta-Verner method (see [Ve]) is based on the formulas

5th Order $$w_{i+1} = w_i + \frac{13}{160}k_1 + \frac{2375}{5984}k_3 + \frac{5}{16}k_4 + \frac{12}{85}k_5 + \frac{3}{44}k_6 \quad \text{and}$$

6th Order $$\tilde{w}_{i+1} = w_i + \frac{3}{40}k_1 + \frac{875}{2244}k_3 + \frac{23}{72}k_4 + \frac{264}{1955}k_5 + \frac{125}{11592}k_7 + \frac{43}{616}k_8,$$

where

$$k_1 = hf(t_i, w_i),$$

$$k_2 = hf\left(t_i + \frac{h}{6}, w_i + \frac{1}{6}k_1\right),$$

$$k_3 = hf\left(t_i + \frac{4h}{15}, w_i + \frac{4}{75}k_1 + \frac{16}{75}k_2\right),$$

$$k_4 = hf\left(t_i + \frac{2h}{3}, w_i + \frac{5}{6}k_1 - \frac{8}{3}k_2 + \frac{5}{2}k_3\right),$$

$$k_5 = hf\left(t_i + \frac{5h}{6}, w_i - \frac{165}{64}k_1 + \frac{55}{6}k_2 - \frac{425}{64}k_3 + \frac{85}{96}k_4\right),$$

$$k_6 = hf\left(t_i + h, w_i + \frac{12}{5}k_1 - 8k_2 + \frac{4015}{612}k_3 - \frac{11}{36}k_4 + \frac{88}{255}k_5\right),$$

$$k_7 = hf\left(t_i + \frac{h}{15}, w_i - \frac{8263}{15000}k_1 + \frac{124}{75}k_2 - \frac{643}{680}k_3 - \frac{81}{250}k_4 + \frac{2484}{10625}k_5\right),$$

$$k_8 = hf\left(t_i + h, w_i + \frac{3501}{1720}k_1 - \frac{300}{43}k_2 + \frac{297275}{52632}k_3 - \frac{319}{2322}k_4 + \frac{24068}{84065}k_5 + \frac{3850}{26703}k_7\right).$$

Problem Definition

$$\frac{dy}{dx} = \frac{2x+1}{2y-1}$$

given at x=0, y=2.17928556.

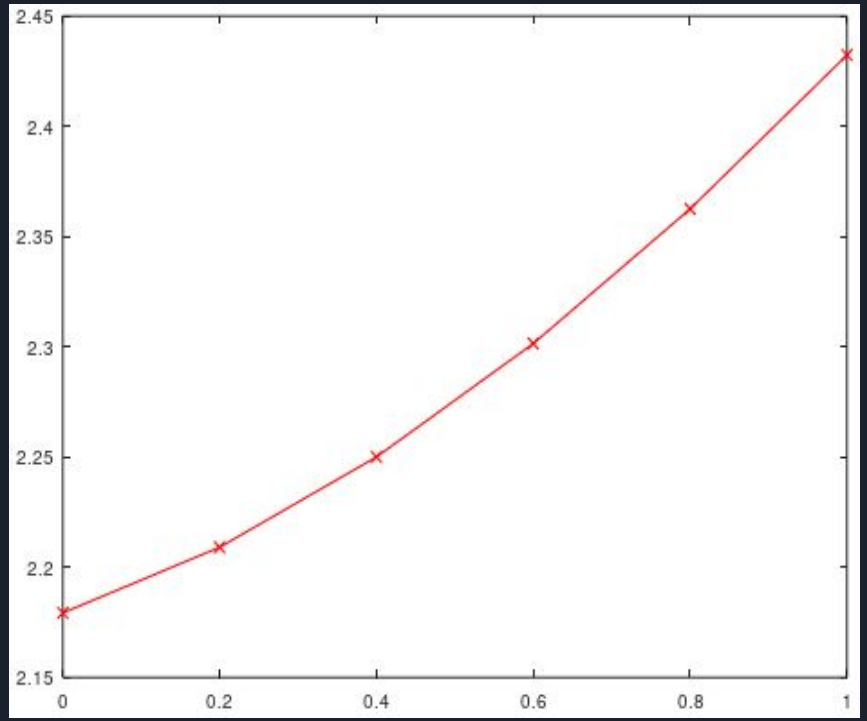
Find y at x=1.

On solving manually,

Solution : y(1) = 2.6932

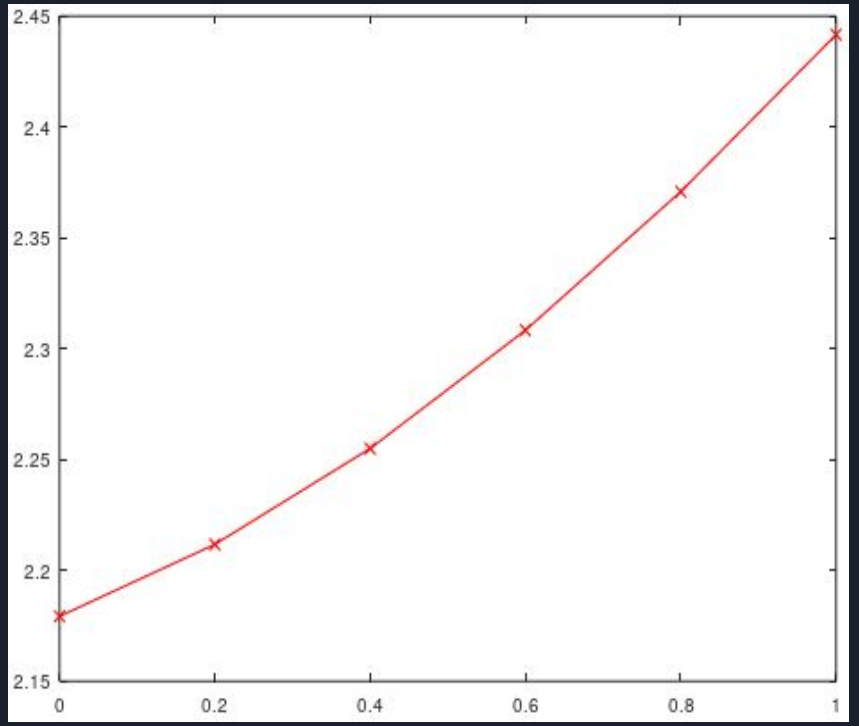
First order method

```
>> RK1
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.209060
 2 iterations gives x3 =0.400000 and y3 =2.250018
 3 iterations gives x4 =0.600000 and y4 =2.301446
 4 iterations gives x5 =0.800000 and y5 =2.362508
 5 iterations gives x6 =1.000000 and y6 =2.432307>>
```



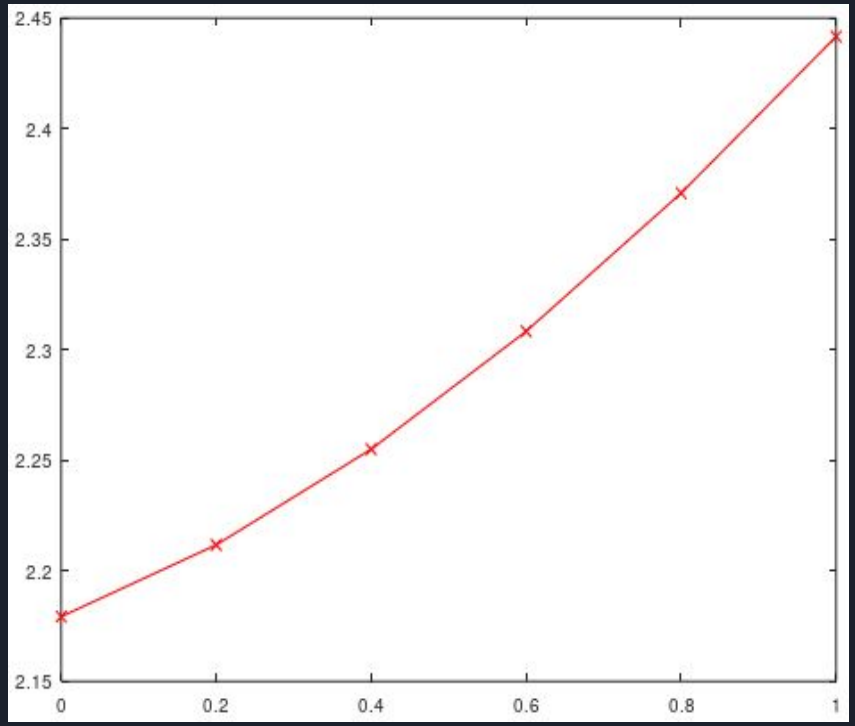
Second order method

```
>> RK2
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.211726
 2 iterations gives x3 =0.400000 and y3 =2.254997
 3 iterations gives x4 =0.600000 and y4 =2.308319
 4 iterations gives x5 =0.800000 and y5 =2.370834
 5 iterations gives x6 =1.000000 and y6 =2.441654>>
```



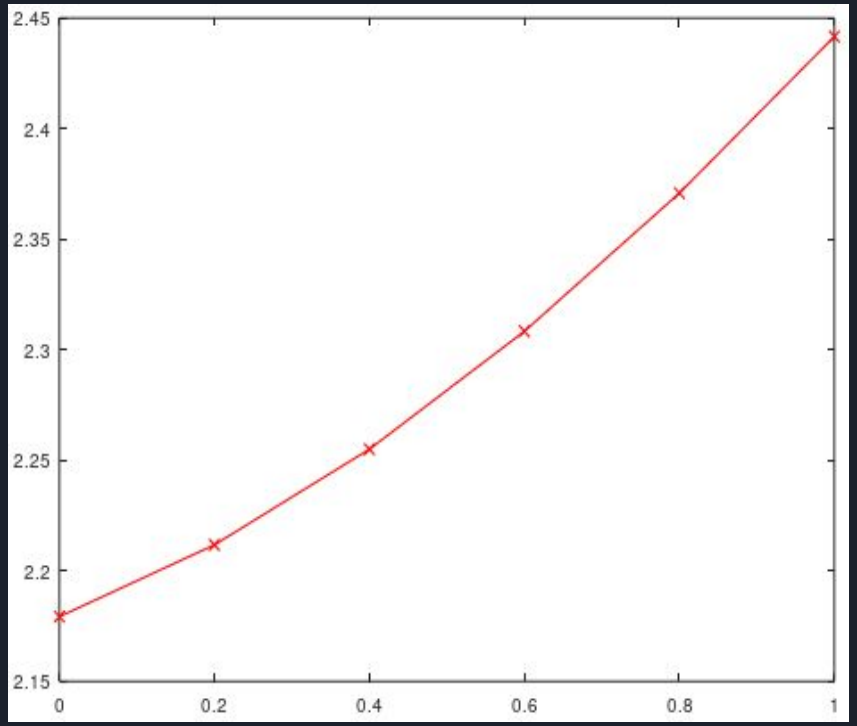
Third order method

```
>> RK3
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.211724
 2 iterations gives x3 =0.400000 and y3 =2.254992
 3 iterations gives x4 =0.600000 and y4 =2.308313
 4 iterations gives x5 =0.800000 and y5 =2.370828
 5 iterations gives x6 =1.000000 and y6 =2.441648>>
```



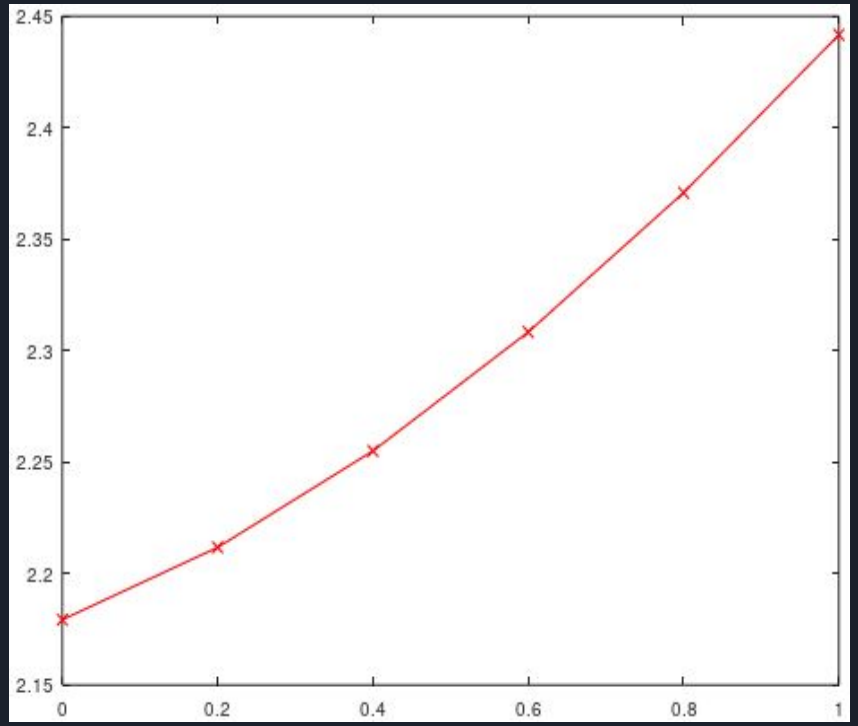
Fourth Order Method

```
>> RK4
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.211724
 2 iterations gives x3 =0.400000 and y3 =2.254993
 3 iterations gives x4 =0.600000 and y4 =2.308314
 4 iterations gives x5 =0.800000 and y5 =2.370829
 5 iterations gives x6 =1.000000 and y6 =2.441649>>
```



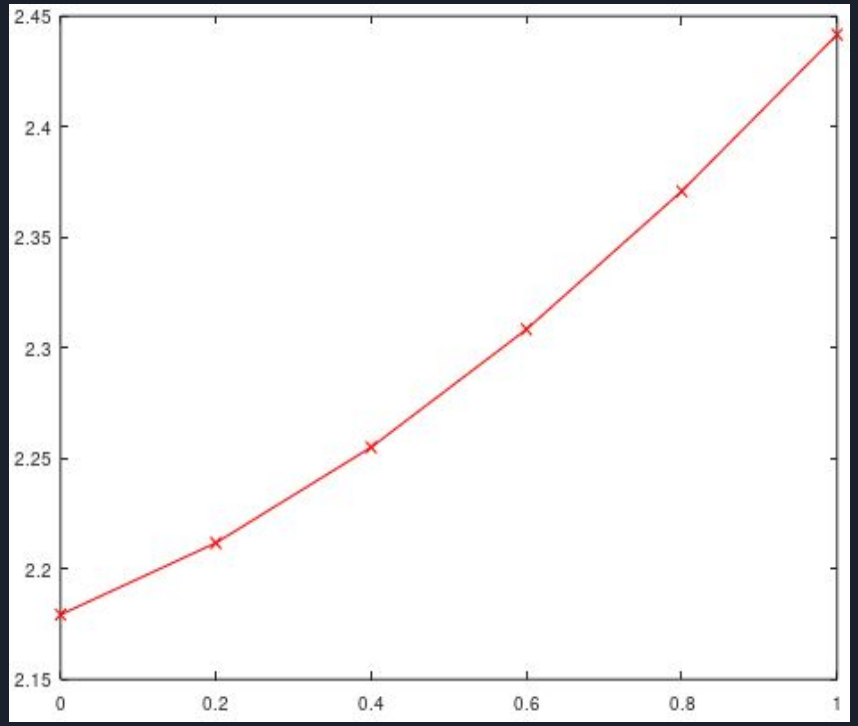
Fifth order method

```
>> RK5
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.211724
 2 iterations gives x3 =0.400000 and y3 =2.254993
 3 iterations gives x4 =0.600000 and y4 =2.308314
 4 iterations gives x5 =0.800000 and y5 =2.370829
 5 iterations gives x6 =1.000000 and y6 =2.441649>>
```



Sixth Order Method

```
>> RK6
initial value of x : 0
initial value of y : 2.17928556
number of iterations you will be performing : 5
value of h : 0.1
for what value of x you would like to estimate y : 1

 1 iterations gives x2 =0.200000 and y2 =2.211724
 2 iterations gives x3 =0.400000 and y3 =2.254993
 3 iterations gives x4 =0.600000 and y4 =2.308314
 4 iterations gives x5 =0.800000 and y5 =2.370829
 5 iterations gives x6 =1.000000 and y6 =2.441649>>
```



Observation Table

We implemented Runge-Kutta orders 1-6 all of them using a 5 step algorithm with step size =0.1 for purpose of comparison and these were the final values of y(1) estimated ~

METHOD	**y(1)**
RK-1	**2.432307**
RK-2	2.441654
RK-3	2.441648
RK-4	2.441649
RK-5	2.441649
RK-6	2.441649

Observations

- From the results, we observe that the most optimum algorithm for solving a differential equation is the RK-4 method.
- In some cases,like this RK-2 method might dominate, but mostly, RK-4 dominates.
- What's interesting to observe is that the results do not get better if we increase the order of algorithm, i.e. 5^{th} and 6^{th} runge kutta method generate the same result as RK-4.
- But by increasing the order of the algorithm, we increase the computational stages and thus increasing time of calculation.

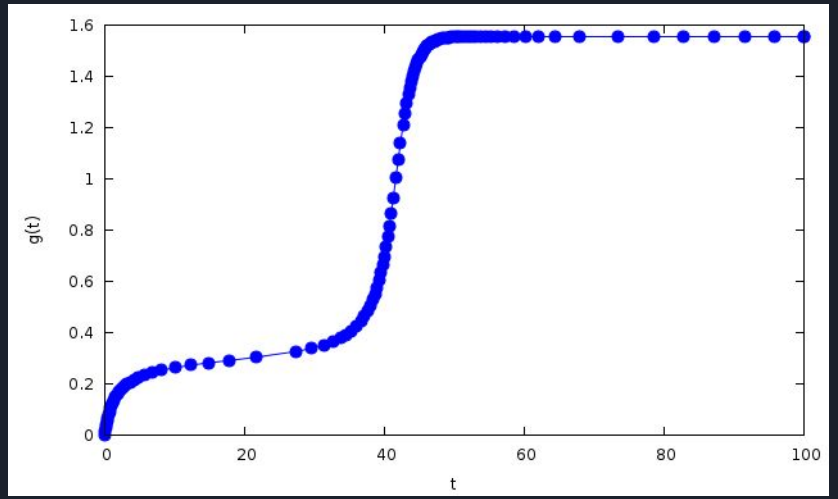
Hence, the most optimum algorithm for solving a differential equation is the RK-4 method.

What is Adaptive Step-Size Modulation ?

Considering the Function g(t) given below, we can see that for 10<t<30 the function is almost flat and increasing the step-size won't affect the results a lot.

But for 30<t<50, the function is rather steep and reducing the step size would yield a better result.

Therefore, implementing an adaptive step-size algorithm which can alter the step-size intelligently according to slope values in each iteration without human intervention could definitely yield a
Better performing Runge-Kutta Differential Equation Solver.



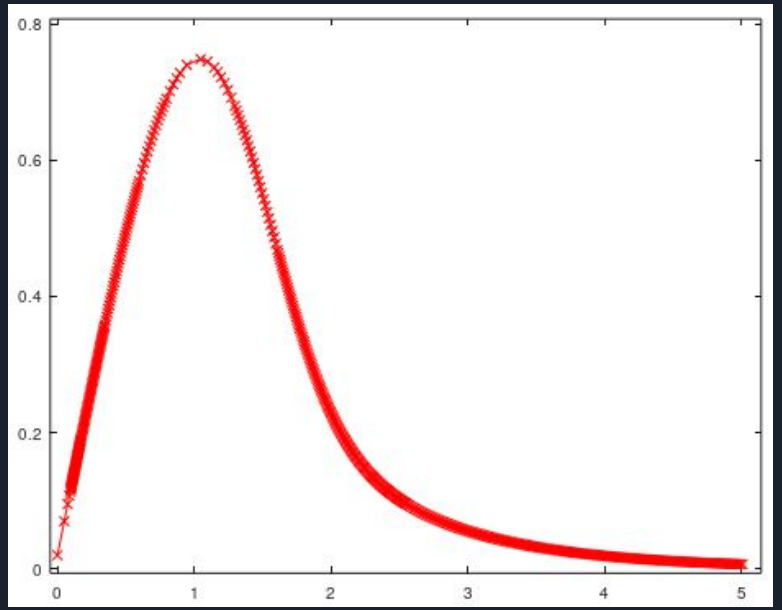
An example to understand step size modulation better.

Example Problem : $dy/dx = e^{y} - y.e^{x}$

Initial Conditions : Y(0)= 0.02

To evaluate : y(5)

Observation : Step size i.e. gap between successive steps is changing throughout the curve intelligently to evaluate the result without human intervention.



Our Initial Problem

$$\frac{dy}{dx} = \frac{2x+1}{2y-1}$$

given at x=0, y=2.17928556.

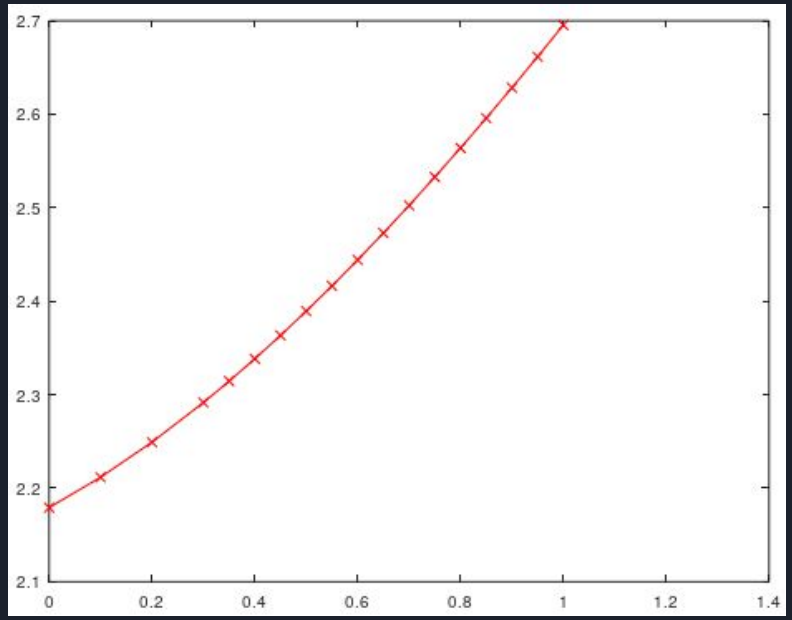
Find y at x=1.

On solving manually,

Solution : y(1) = 2.6932

Runge Kutta 4 with step size modulation

```
>> RKM4
initial value of x : 0
initial value of y : 2.17928556
for what value of x you would like to estimate y : 1
Initial Step Size : 0.1
The estimated value of function y =2.695450 at x =1.000000>>
```



The plot of y vs x depicts the change in step size which is decreasing with increase in slope.

Final Comparison of Results.

	RK-4 Method Answer	RK-4 with adaptive step size modulation
Calculated Answer	2.441649	2.695450
Actual answer on solving manually	2.6932	2.6932
No. of steps	5	17
Initial Step Size	0.1	0.1
Error %	9.3402%	0.0835

Final Conclusions

- The RK4 method is the best among other orders of Runge Kutta Method when considering accuracy as well as performance.
- When Intelligent step size modulation was added to the RK4 algorithm the accuracy increased a lot while compromising a little on calculation speed for the problem under consideration
- The new error was 0.0835 % compared to an error of 9.3402 % without step size modulation.

- Our aim to improve Runge Kutta Method was achieved by applying step size modulation to RK4 algorithm.

References

- “Numerical Analysis” 9th Edition by ~
  Richard L. Burden & *J.Douglas Faires*
  Youngstown State University Youngstown State University

- www.wikipedia.com
- www.stackexchange.com
- www.octave.sourceforge.io
- www.ieeexplore.ieee.org

ACKNOWLEDGEMENT

This project wouldn’t have had been possible without the help of
Mrs. Uma Nangia Ma’am [HOD,Electrical Department, DTU]

I would like to heartily thank Uma Nangia Ma’am for providing me with this idea of project and guiding me through the course of creation of this program.

Thank You