Numerical Methods of Ordinary Differential Equations - Runge-Kutta Method(RK4)

By Gopesh Gaba

Ordinary Differential Equations

An Ordinary Differential Equation(ODE) is a differential equation that contains one or more functions of one independent variable variable.

They can classified into

- Linear ODE
- Non-Linear ODE
- Homogeneous
- Non-Homogeneous
- Autonomous

Significance of Solving Initial Value Problems for Ordinary Differential Equations

ODE’s occur in many studies such as mathematics, social sciences and natural sciences.

So, studying ODE’s helps us understand the phenomena related to these subjects better by providing us with a way to solve various problems related to them.

The fields include analytical mechanics, geometry,physics, astronomy, biology, chemistry, economics and many more.

Motivation behind Developing The Runge Kutta Method

- The Runge Kutta Methods is a class of higher order methods developed for solving initial value problems of ordinary differential equations.
- One of the major drawbacks of higher order method of taylor series is that we have to differentiate and new functions in order to obtain a higher order error.
- But to overcome this runge kutta method was developed which uses weighted average of slopes to obtain a higher order of error
- In this project, we shall discuss the RK4 or the Runge Kutta Method of Order 4.

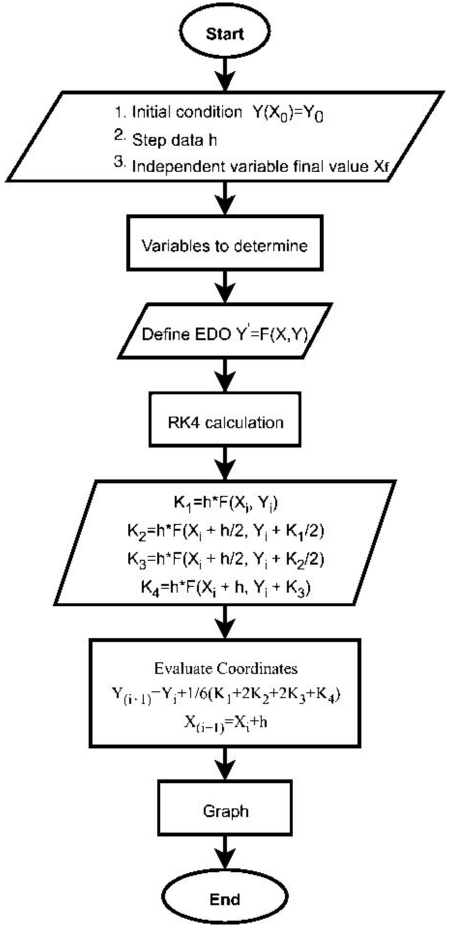
- As the name suggests this method uses the weighted average of four slopes in order to numerically estimate the values of the differential equations.
- Here dy/dx=f(x,y) and y(t(i)) is approximated to u(i) and u(i+1) = u(i) + (K1 + 2*K2 + 2*K3 +K4)/6, where K1 = h*f(t(i),u(i)), K2 = h*f(t(i)+h/2,u(i)+K1/2), K3 = h*f(t(i)+h/2,u(i)+K2/2), K4 = h*f(t(i)+h,u(i)+K3), for i=1,2,....,n. Here (K1 + 2*K2 + 2*K3 +K4)/6 term is the weighted average of the slope.
- We can calculate K1,K2,K3,K4 at each step from the f(x,y) without having to make any additional function.

Algorithm

Image Source for this heading - https://www.researchgate.net/figure/Flow-chart-of-the-Runge-Kutta-4th-order-method-in-MATLAB-R_fig2_351203205

Algorithm of RK4 for one independent and one dependent variable

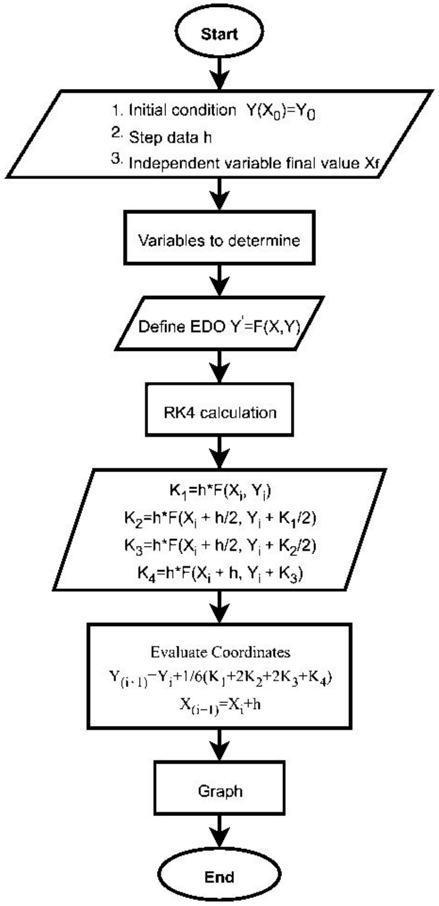
1. Have the user enter the initial x and y coordinates.
2. Have the user enter the function f(x,y) where dy/dx=f(x,y).
3. Have the user enter the target value and the number of steps OR Have the user enter the step size and the number of steps.



4. Create two row matrices of size n+1(n is the number of steps). ‘t’ stores the values of x coordinates. ‘u’ stores the y coordinates.

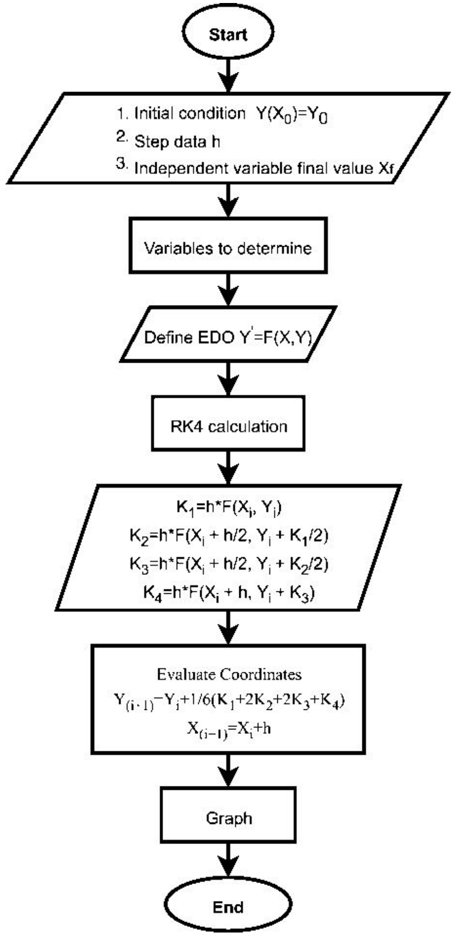
5. t(1)=x initial. With t(i+1)=t(i)+h, where i = 1,2,...n.

6. u(1)= y initial. Then we take 4 variable K1,K2,K3,K4. Here u(i)=y(t(i)). And u(i+1) = u(i) + (K1 + 2*K2 + 2*K3 + K4)/6, K1 = h*f(t(i),u(i)), K2 = h*f(t(i)+h/2,u(i)+K1/2), K3 = h*f(t(i)+h/2,u(i)+K2/2), K4 = h*f(t(i)+h,u(i)+K3). For i=1,2,....,n.


Start
1. Initial condition $Y(X_0)=Y_0$
2. Step data h
3. Independent variable final value Xf
Variables to determine
Define EDO $Y'=F(X,Y)$
RK4 calculation
$K_1=h*F(X_i, Y_i)$
$K_2=h*F(X_i + h/2, Y_i + K_1/2)$
$K_3=h*F(X_i + h/2, Y_i + K_2/2)$
$K_4=h*F(X_i + h, Y_i + K_3)$
Evaluate Coordinates
$Y_{(i-1)}=Y_i+1/6(K_1+2K_2+2K_3+K_4)$
$X_{(i-1)}=X_i+h$
Graph
End

7. Display t and u in tabular form by naming t as X Coordinate and corresponding u next to it as Y Coordinate.

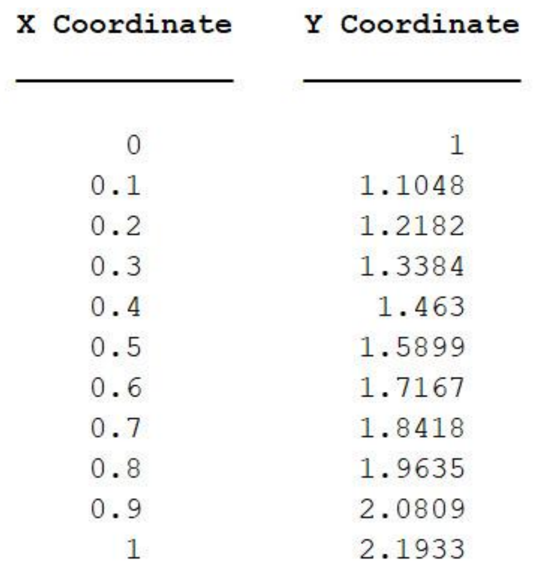
8. Plot the graph of x and y by using plot function keeping t on x-axis and u on y-axis


Start
1. Initial condition $Y(X_0)=Y_0$
2. Step data h
3. Independent variable final value Xf
Variables to determine
Define EDO $Y'=F(X,Y)$
RK4 calculation
$K_1=h*F(X_i, Y_i)$
$K_2=h*F(X_i + h/2, Y_i + K_1/2)$
$K_3=h*F(X_i + h/2, Y_i + K_2/2)$
$K_4=h*F(X_i + h, Y_i + K_3)$
Evaluate Coordinates
$Y_{(i-1)}=Y_i+1/6(K_1+2K_2+2K_3+K_4)$
$X_{(i-1)}=X_i+h$
Graph
End

Example

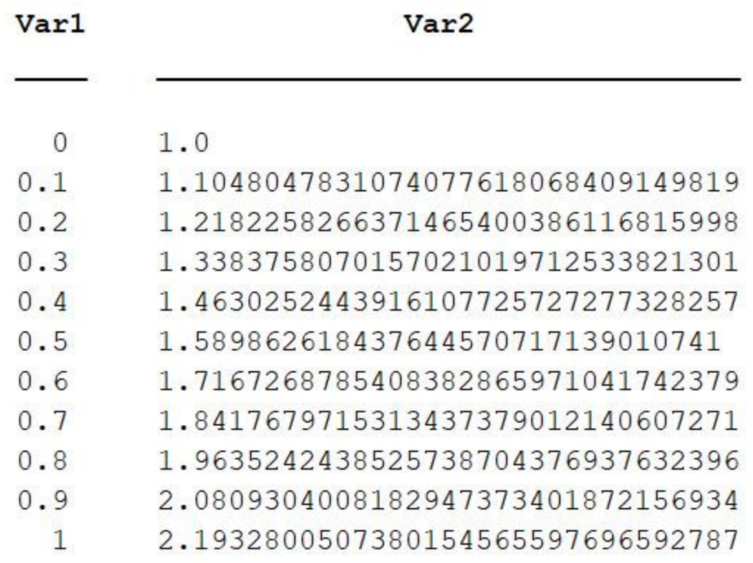
We take $dy/dx=f(x,y) = y/(1+x^2)$ and initial point (0,1). The solution obtained by RK4

Exact Solution is $\ln y = \tan^{-}x$



```
X Coordinate    Y Coordinate
____________    ____________

      0                1
    0.1           1.1048
    0.2           1.2182
    0.3           1.3384
    0.4            1.463
    0.5           1.5899
    0.6           1.7167
    0.7           1.8418
    0.8           1.9635
    0.9           2.0809
      1           2.1933
```



```
Var1                 Var2
____    _________________________________

  0     1.0
0.1     1.104804783107407761806840914981 9
0.2     1.2182258266371465400386116815998
0.3     1.3383758070157021019712533821301
0.4     1.4630252443916107725727277328257
0.5     1.589862618437644570717139010741
0.6     1.7167268785408382865971041742379
0.7     1.8417679715313437379012140607271
0.8     1.9635242438525738704376937632396
0.9     2.0809304008182947373401872156934
  1     2.1932800507380154565597696592787
```

Similarly We can design the algorithm for runge kutta order 4 for one independent and multiple dependent variables

By tweaking the algorithm given previously, we can convert in into an algorithm for ODE's with multiple dependent variables. If we have x as independent variable and y_1 , y_2 ,.......,y_n as dependent variables. And we are given $y_i' = f_i(x, y_1 , y_2 ,\dots\dots, y_n)$. And initial values of y_1 , y_2 ,.......,y_n at some particular initial x_0 . Then we can use runge kutta method to find approximate the values of system of these ODE's.

In this project, I have made a program for 1 independent and 2 dependent variables.

The algorithm for one independent and two dependent is as follows

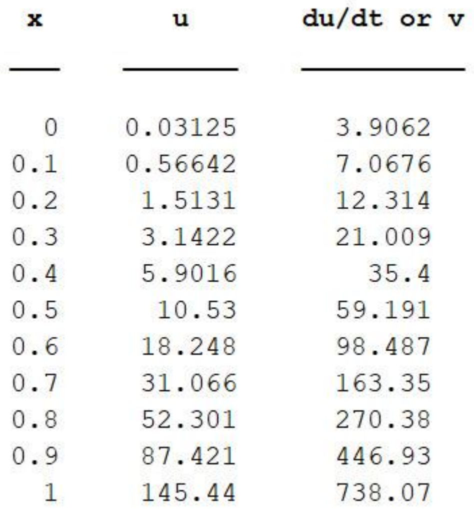
- We take inputs similar to the previous algorithm with addition of a new function z’ = g(x,y,z), value of z at initial x, and this time y’ = f(x,y,z). Then we create F(x,y,z) = [f(x,y,z);g(x,y,z)] to make coding easier.
- Then we proceed in similar way by initializing row matrix t, but this time we make a matrix A with two rows. It’s first row stores the approximated values of y and second contains the approximated values of z.
- Then we start the loop for i=1:n, in this loop K1=h.*F(t(i),A(1,i),A(2,i)), K2=h.*F(t(i)+h/2,A(1,i)+K1(1,1)/2,A(2,i)+K1(2,1)/2), K3=h.*F(t(i)+h/2,A(1,i)+K2(1,1)/2,A(2,i)+K2(2,1)/2), K4=h.*F(t(i)+h,A(1,i)+K3(1,1),A(2,i)+K3(2,1)), and A(:,i+1) = A(:,i) + (K1+2.*K2+2.*K3+K4)/6.
- Now we print a table whose first column is t, second column is A(1,:) and third column is A(2,:). Here first column is x, second column is approx of u and third column is approx of v.(Example at end in real world examples)

Similarly we can also design the algorithm for runge kutta order 4 for second order differential equations or higher

In the case of second order differential equation we do it in a similar manner to the one with 2 variables. Here we have the independent variable x and the dependent variable y. The user input y” = f(x,y,y’). So we assume y’ to be another variable, let’s say z. And y’=z and z’ = f(x,y,z). Hence we obtain one independent variable and two dependent variables and using this we solve our higher order ODE using the previous method.

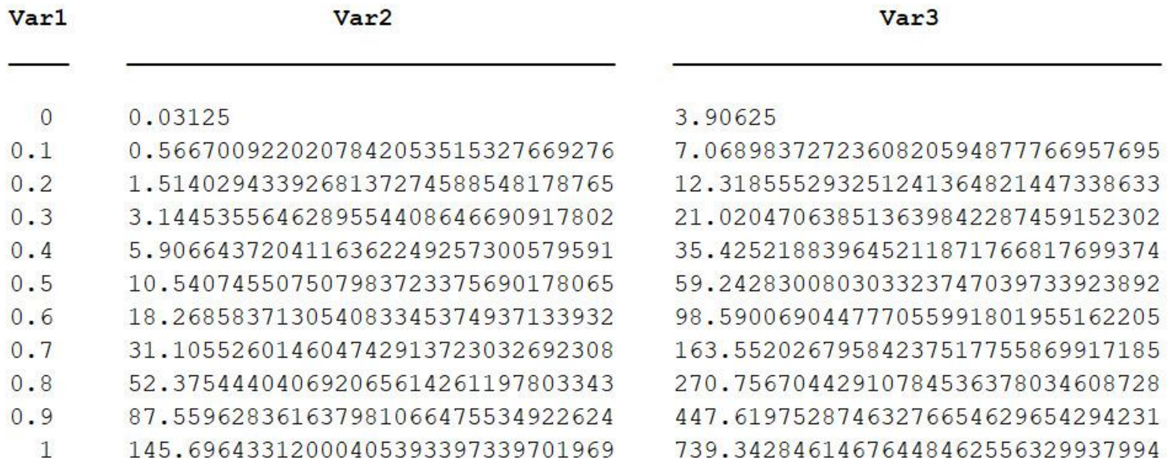
Example

If $y'' = 6y' - 5y + e^{-3x}$, with $y(0)=1/32$ and $y'(0)=125/32$. Then through RK4 we obtain



x	u	du/dt or v
0	0.03125	3.9062
0.1	0.56642	7.0676
0.2	1.5131	12.314
0.3	3.1422	21.009
0.4	5.9016	35.4
0.5	10.53	59.191
0.6	18.248	98.487
0.7	31.066	163.35
0.8	52.301	270.38
0.9	87.421	446.93
1	145.44	738.07

The exact solution of the IVP is $y = -e^{x} + e^{5x} + 1/32*(e^{-3x})$. And the exact values obtained would be:



Var1	Var2	Var3
0	0.03125	3.90625
0.1	0.5667009220207842053515327669276	7.0689837272360820594877766957695
0.2	1.5140294339268137274588548178765	12.318555293251241364821447338633
0.3	3.1445355646289554408646690917802	21.020470638513639842287459152302
0.4	5.9066437204116362249257300579591	35.425218839645211871766817699374
0.5	10.540745507507983723375690178065	59.242830080303323747039733923892
0.6	18.268583713054083345374937133932	98.590069044777055991801955162205
0.7	31.105526014604742913723032692308	163.55202679584237517755869917185
0.8	52.375444040692065614261197803343	270.75670442910784536378034608728
0.9	87.559628361637981066475534922624	447.61975287463276654629654294231
1	145.69643312000405393397339701969	739.34284614676448462556329937994

Here in first image u is y and v is y’ and in next image var 1 is x, var 2 is y and var 3 is y’

https://www.cuemath.com/calculus/second-order-differential-equation/

Convergence Criteria

http://www.iact.ugr-csic.es/personal/julyan_cartwright/papers/rkpaper/node13.html#:~:text=which%20shows%20that%20consistency%20is,initial%20value%20problem%2C%20passes%20through%20.

Numerical Analysis by Richard L. Burden and J. Douglas Faires.

Convergence of a One Step difference equation method

A one step difference equation method for solving an IVP of an ODE is said to be convergent with respect to the ODE it approximates if

$$\lim_{h\to 0} \max_{1\le i\le N} |w_i - y(t_i)| = 0$$

Convergence of Runge Kutta Order 4

An RK4 method is said to be convergent with respect to an ODE if and only if it is consistent.

A method is said to be consistent if

$$\lim_{h \to 0} \max_{1 \le i \le N} |\tau_i(h)| = 0.$$

Here the term $\tau_i(h)$ is the local truncation error, at ith step.

Error Order

https://archive.org/details/discretevariable0000henr/page/96/mode/2up

The Error order of a pth order method

The general rule of thumb is that the error order of a pth order method is p+1, i.e the local truncation error is proportional to h^{p+1} (or error is $O(h^{p+1})$).

The error order of runge kutta order 4

By calculation, the local truncation error of RK4, or it’s error order comes out to be $O(h^5)$.

Computational Costs

Looking at the algorithm we can see that the program calls the function f(x,y) 4 times in each loop, where the loop runs n times(n is the number of steps). So, in totality the function f(x,y) is called 4*n times. So, the computational cost depends the number of steps and time complexity required to compute the value from the function. To be more precise the computational cost in time complexity is n*(time required to compute f(x,y)) and space complexity is O(n).

Advantages and Disadvantages of Runge Kutta Method

Advantages

- Over Taylor Series Method - Runge Kutta does not require the computation of higher order derivatives of f(x,y).
- Over Multistep Method - Compared to multistep methods, Runge kutta methods are considered to be much more stable and accurate. Additionally, they do not require multiple starting points so they are self starting.

Disadvantages

- One of the biggest drawback of runge kutta methods over multistep methods(like Adam Bashforth and Adam Moulton) is that they are more computationally expensive as compared to multistep methods.

Real World Examples

Some examples are

- Applying Kirchoff's Law in Circuits - When we apply kirchhoff's law in a circuit containing different components like resistors, capacitors, inductors, etc, we can get complicated differential equations which are really hard to solve using normal methods. For solving these numerical methods for IVP of ODE’s like the RK4 come in handy.

Example

We apply Kirchhoff's Law on the following circuit:

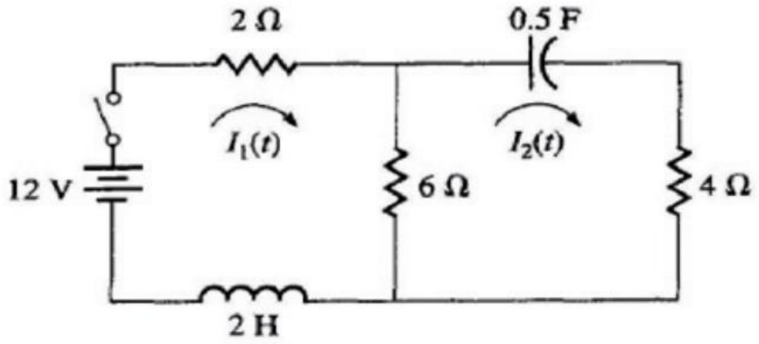
The exact equation of the the circuit for IVP $I_1(0) = 0$

And $I_2(0)=0$ comes out to be $I_1(t) = -3.375e^{-2t} + 1.875e^{-0.4t} = 1.5$ and $I_2(t) = -2.25e^{-2t} + 2.25e^{-0.4t}$.

And the differential equations are

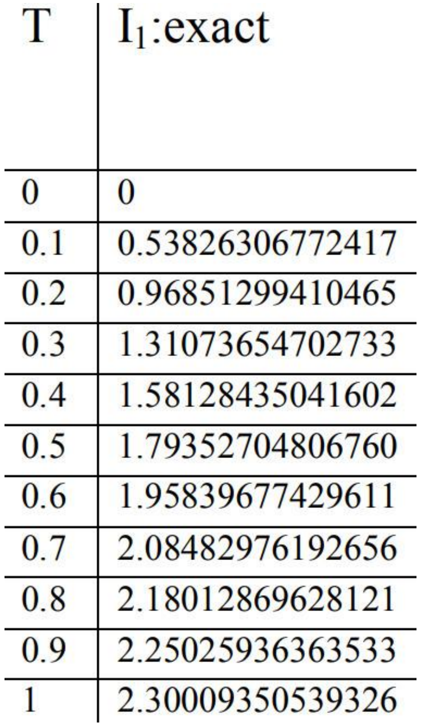
$I_1' = -4I_1 + 3I_2 + 6$

$I_2' = -2.4I_1 + 1.6I_2 + 3.6$

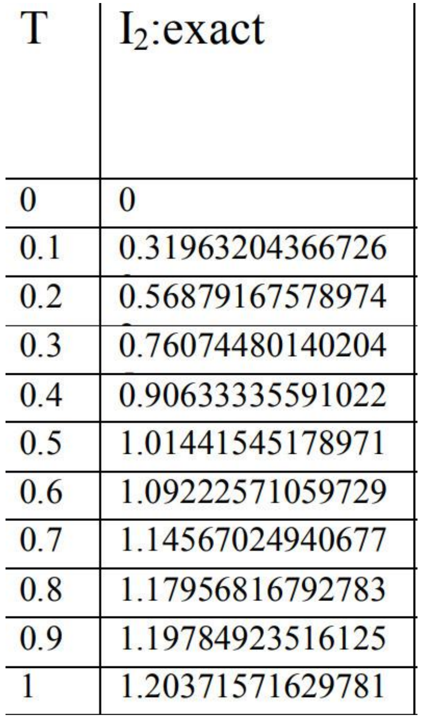


- https://www.researchgate.net/publication/337317358_On_Initial_Value_Problems_and_Its_Applications

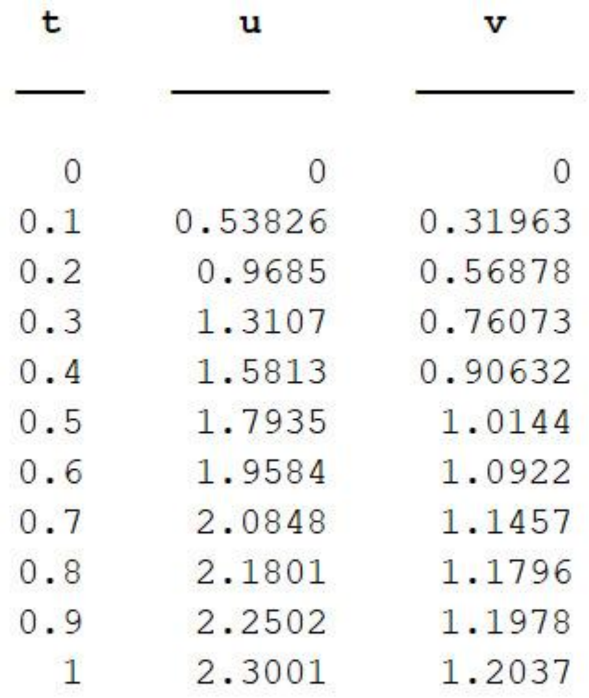
The exact answer and the answer from RK4 code come out to be (here u = I_1 and v = I_2)



T	I_1:exact
0	0
0.1	0.53826306772417
0.2	0.96851299410465
0.3	1.31073654702733
0.4	1.58128435041602
0.5	1.79352704806760
0.6	1.95839677429611
0.7	2.08482976192656
0.8	2.18012869628121
0.9	2.25025936363533
1	2.30009350539326



T	I_2:exact
0	0
0.1	0.31963204366726
0.2	0.56879167578974
0.3	0.76074480140204
0.4	0.90633335591022
0.5	1.01441545178971
0.6	1.09222571059729
0.7	1.14567024940677
0.8	1.17956816792783
0.9	1.19784923516125
1	1.20371571629781



t	u	v
0	0	0
0.1	0.53826	0.31963
0.2	0.9685	0.56878
0.3	1.3107	0.76073
0.4	1.5813	0.90632
0.5	1.7935	1.0144
0.6	1.9584	1.0922
0.7	2.0848	1.1457
0.8	2.1801	1.1796
0.9	2.2502	1.1978
1	2.3001	1.2037

- Modelling flow of medication inside human body - When a person ingests medication pills, the pills dissolve in GI tract. From there it dissolves into the bloodstream and other parts of the body. Pharmaceutical companies are interested in modelling the flow of these medications inside the body and numerical methods involving IVP’s of ODE’s can be used for that.

https://www.researchgate.net/publication/337317358_On_Initial_Value_Problems_and_Its_Applications

- In Oscillatory Systems - Oscillatory Systems can be written in form of differential equations and their solutions can be called IVP. These can be solved using numerical methods for ODE’s. These may be used to understand motion of simple machines to huge celestial objects.
- Other examples may include weather modelling, stock trends, modelling the spread of infectious diseases etc.

Thank You