IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 17, Issue 4 Ser. III (Jul. – Aug. 2021), PP 01-09
www.iosrjournals.org

Demonstration Study on Runge-Kutta Fourth Order Method by Using MATLAB Programming

Pushap Lata Sharma & Ashok Kumar
Department of Mathematics, Himachal Pradesh University, Summer Hill, Shimla, India

Abstract
In this paper we have solved initial value problem (IVP) for ordinary differential equations (ODE) by using Runge-Kutta fourth order method only. Despite various analytical methods for finding the solution of initial value problems, there are large number of ordinary differential equations which have no analytical solutions. In that case we have to solve ordinary differential equations by using numerical methods. There are numerous numerical methods such as Euler's, Heun's, Runge-Kutta third order and Runge-Kutta fourth order methods. Runge-Kutta fourth order method is the powerful numerical technique to solve initial value problems. In this paper different four examples have been solved by using MATLAB programming and numerical results have been shown in tables and graphs.

Keyword: *Initial value problem, ordinary differential equation, Runge-Kutta fourth order method and MATLAB programming.*

Date of Submission: 10-08-2021 Date of Acceptance: 25-08-2021

I. Introduction:

In this paper Runge-Kutta fourth order method (RK4) is used to solve Initial value problems for ordinary differential equations by using MATLAB Programming. Maximum problems in physical sciences, Life sciences and Engineering are solved by differential equations. Differential equations solutions are very important part to develop the various models in Physics and Engineering. But a minimum number of differential equations have been solved analytically to obtain the solution of differential equation under given conditions. Therefore, we have used numerical method to obtain the solution of the initial value problems by using MATLAB programming. Differential equations are the most important mathematical tools used in providing models in the mathematics, physics, banking, engineering, elasticity, astronomy, dynamics, biology, chemistry, medicine environmental science and many other areas. Many researchers have studied the various kind of differential equations and many complicated systems that can be expressed quite precisely with mathematical expressions. Many differential equations emerging in applications are so complicated that in sometimes inappropriate to have solution formulas. In this case numerical methods come up with a powerful different tool for solving the differential equations under the prescribed initial condition or conditions. Runge-Kutta method has the advantage of being the most widely used numerical medium, since it gives calculable values, starting values and particularly capable when the computation of higher order derivatives are complicated. It gives a higher accuracy than on Euler's method and also possesses the advantage of requiring only the function values at some selected points on the sub-intervals. Moreover, it is easy to change step length for special procedures necessary for starting, which minimize the computing time. The Runge-Kutta method is very useful. The inherent error in the Runge-Kutta method hard to be assessed. From the literature review analysis, we came across plenty of work have been carried out to find the numerical solutions of initial value problems by using Runge - Kutta first, second, third, and fourth order methods. Gowri et al. [2017] solved differential equation problems by using Runge - Kutta fourth order method. A comparative exploration on different numerical methods for solving ordinary differential equations by Arefin et al. [2020]. Hossen et al. [2019] performed comparative investigation on numerical solution of initial value problems by using modified Euler's method and Runge-Kutta method. They concluded that Runge-Kutta fourth order method gave more accurate results. Sharma and Kumar [2021] have written a review paper after reviewing many research papers on Runge- Kutta methods to study numerical solutions of initial value problems in ordinary differential equations, they came to know that the Runge- Kutta fourth order method gives more accurate results. Sharma [2021] also has analysed that the Runge-Kutta fourth order method is the best among all the remaining three methods therefore, in this paper Runge-Kutta fourth order method (RK4) is used to solve initial value problems for ordinary differential equations by using MATLAB programming.

Problem Formulation

In this paper, we have taken Runge-Kutta fourth Order method to find numerical solutions of the initial value problem (IVP) of the first-order ordinary differential equations of the form

$$y' = f(x, y(x)), \qquad x \in (x_0, x_n), \qquad y(x_0) = y_0 \tag{1}$$

Where $y' = \frac{dy}{dx}$ and $f(x, y(x))$ is the given function and $y(x)$ is the solution of the equation (1). In this paper, we find a solution of the equation on a finite interval(x_0, x_n), starting with initial point x_0. A continuous approximation to the solution $y(x)$ will not be obtained; instead, approximations to y will be generated at various values.

Runge-Kutta's Fourth Order Method

The fourth order Runge-Kutta's method is used to find the numerical solution of linear or non-linear ordinary differential equations. The universal formula for Runge-Kutta's fourth order method is

$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \qquad n = 0,1,2,3, \ldots$

$k_1 = hf(x_n, y_n)$

$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$

$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$

$k_4 = hf(x_n + h, y_n + k_3)$

$\Delta y = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

But $\quad y_{n+1} = y_n + \Delta y$

Therefore $\quad y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Numerical Experiments

Numerical Experiment 1: Consider the initial the problem $y' = 2e^x - 0.4y$, with the initial condition $y(0) = 5$, on the interval $1 \le x \le 3$.

MATLAB Programming for above Numerical Experiment:

```
% Runge-Kutta 4th order with MATLAB
% It calculates an ODE using Runge-Kutta 4th order method
% Equation to solve: Y'=2*exp(x)-0.4*Y; Y(0)=5; x=[1,3];
clc, clear all, close all
% instruction to write results on external file
fid=fopen('Expotential.m','w');
h=0.1; a=1; b=3;       % h is the step size, x=[a,b] x-range
x=a:h:b;               % computes x-array up to x=3
y=zeros(1,numel(x));   % Memory preallocation
y(1)=5;                % initial condition; in MATLAB indices start at 1
Fyt=@(x,y)2*exp(x)-0.4*y;      % change the function as you desire
            % the function is the expression after (x,y)
% table title
fprintf(fid,'%7s %7s %7s %7s %7s %7s %7s\n','i','x(i)','k1','k2','k3','k4','y(i)');
for ii=1:1:numel(x)
   k1=Fyt(x(ii),y(ii));
   k2=Fyt(x(ii)+0.5*h,y(ii)+0.5*h*k1);
   k3=Fyt((x(ii)+0.5*h),(y(ii)+0.5*h*k2));
   k4= Fyt((x(ii)+h),(y(ii)+h*k3));
   y(ii+1)=y(ii)+(h/6)*(k1+2*k2+2*k3+k4);   % min equation
   % table data
   fprintf(fid,'%7d %7.2f %7.3f %7.3f',ii, x(ii), k1, k2);
   fprintf(fid,' %7.3f %7.3f %7.3f\n', k3, k4, y(ii));
end
y(numel(x))=[ ];   % erase the last computation of y(n+1)
% Solution PLOT:
plot(x,y,'ok')
xlim([1,3])
```

```
ylim([5,25])
title('RK-4--Numerical Solution---');
ylabel('Y-Approximate Value');    xlabel('X-Subdivision');    legend('RK4');
grid on
fclose(fid);
```

Table 1: Approximate solutions of the Numerical Experiment 1

i	x(i)	k1	k2	k3	k4	y(i)
1	1.00	3.437	3.647	3.642	3.863	5.000
2	1.10	3.862	4.093	4.089	4.331	5.365
3	1.20	4.331	4.585	4.579	4.846	5.774
4	1.30	4.846	5.125	5.119	5.413	6.232
5	1.40	5.412	5.720	5.714	6.037	6.745
6	1.50	6.037	6.376	6.369	6.725	7.317
7	1.60	6.724	7.098	7.090	7.483	7.954
8	1.70	7.482	7.894	7.886	8.318	8.664
9	1.80	8.318	8.772	8.763	9.240	9.453
10	1.90	9.240	9.740	9.730	10.257	10.330
11	2.00	10.256	10.809	10.798	11.379	11.304
12	2.10	11.378	11.988	11.976	12.617	12.385
13	2.20	12.616	13.290	13.276	13.984	13.584
14	2.30	13.983	14.726	14.712	15.493	14.913
15	2.40	15.492	16.313	16.296	17.159	16.385
16	2.50	17.158	18.064	18.046	18.999	18.016
17	2.60	18.998	19.999	19.979	21.031	19.823
18	2.70	21.030	22.136	22.114	23.276	21.823
19	2.80	23.275	24.496	24.471	25.755	24.036
20	2.90	25.754	27.103	27.076	28.494	26.485

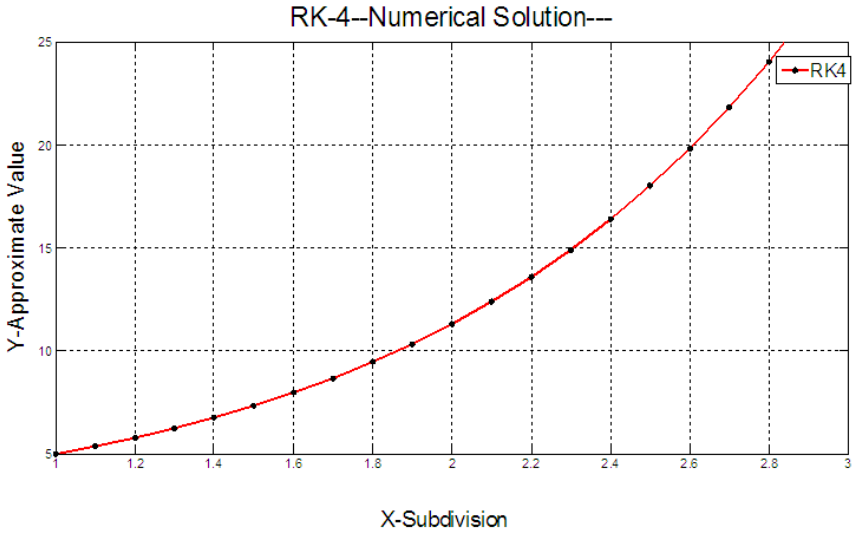

Figure 1: Graphically representation for the solutions of table 1.

Numerical Experiment 2:

Consider the initial value problem $y' = -4y + 6x^2 + 3x,$ with the initial condition $y(0) = \frac{1}{12}$; on the interval $0 \leq x \leq 2$.

MATLAB Programming for above Numerical Experiment:

```
% Runge-Kutta 4th order with MATLAB
% It calculates an ODE using Runge-Kutta 4th order method
% Equation to solve: Y'=-4*y+6*x^2+3*x; Y(0)=1/2; x=[0,2];
clc, clear all, close all
% instruction to write results on external file
fid=fopen('final11.m','w');
h=0.1; a=0; b=2;    % h is the step size, x=[a,b] t-range
x=a:h:b;          % computes t-array up to t=2
```

```
y=zeros(1,numel(x));  % Memory preallocation
y(1)=1/2; % initial condition; in MATLAB indices start at 1
Fyt=@(x,y)-4*y+6*x^2+3*x; % change the function as you desire
            % the function is the expression after (t,y)
% table title
fprintf(fid,'%7s %7s %7s %7s %7s %7s %10s\n','i','x(i)','k1','k2','k3','k4','y(i)');
for ii=1:1:numel(x)
   k1=Fyt(x(ii),y(ii));
   k2=Fyt(x(ii)+0.5*h,y(ii)+0.5*h*k1);
   k3=Fyt((x(ii)+0.5*h),(y(ii)+0.5*h*k2));
   k4= Fyt((x(ii)+h),(y(ii)+h*k3));
   y(ii+1)=y(ii)+(h/6)*(k1+2*k2+2*k3+k4);   % min equation
   % table data
   fprintf(fid,'%7d %7.2f %7.3f %7.3f',ii, x(ii), k1, k2);
   fprintf(fid,' %7.3f %7.3f %10.6f\n', k3, k4, y(ii));
end
y(numel(x))=[ ];   % erase the last computation of y(n+1)
% Solution PLOT:
plot(x,y,'ok')
xlim([0,2])
ylim([0,6])
title('RK-4--Numerical Solution---');
ylabel('Y-Approximated values');    xlabel('X-Subdivision');    legend('RK4');
grid on
fclose(fid);
```

Table 2: Approximate solutions of the Numerical Experiment 2

i	x(i)	k1	k2	k3	k4	y(i)
1	0.00	-2.000	-1.435	-1.548	-1.021	0.500000
2	0.10	-1.041	-0.608	-0.694	-0.283	0.350220
3	0.20	-0.299	0.046	-0.023	0.310	0.284751
4	0.30	0.297	0.583	0.526	0.807	0.285693
5	0.40	0.796	1.042	0.992	1.239	0.341045
6	0.50	1.229	1.448	1.404	1.627	0.442760
7	0.60	1.618	1.820	1.779	1.986	0.585447
8	0.70	1.978	2.167	2.130	2.326	0.765487
9	0.80	2.318	2.500	2.463	2.653	0.980459
10	0.90	2.645	2.821	2.786	2.971	1.228736
11	1.00	2.963	3.135	3.101	3.283	1.509228
12	1.10	3.275	3.445	3.411	3.591	1.821207
13	1.20	3.583	3.752	3.718	3.896	2.164181
14	1.30	3.889	4.056	4.023	4.200	2.537823
15	1.40	4.192	4.359	4.326	4.502	2.941912
16	1.50	4.495	4.661	4.628	4.804	3.376302
17	1.60	4.796	4.962	4.929	5.105	3.840893
18	1.70	5.098	5.263	5.230	5.406	4.335619
19	1.80	5.398	5.564	5.531	5.706	4.860435
20	1.90	5.699	5.864	5.831	6.006	5.415311
21	2.00	5.999	6.164	6.131	6.307	6.000229

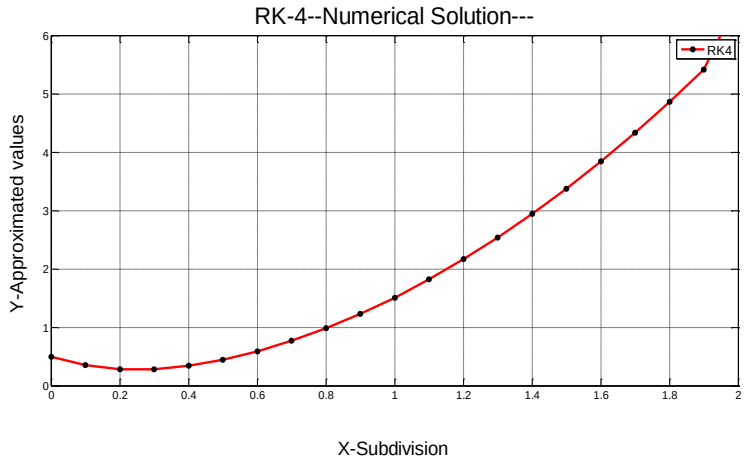

Figure 2: Graphically representation for the solutions of table 2.

Numerical Experiment 3:
Consider the initial value problem
$y' = 3log(\sin(x)) - 1.5y$ with the initial condition $y(0) = 6$ on the interval $1 \leq x \leq 3$.

MATLAB Programming for above Numerical Experiment:

```
% Runge-Kutta 4th order with MATLAB
% It calculates an ODE using Runge-Kutta 4th order method
% Equation to solve: Y'=3*log(sin(x))-1.5*Y; Y(0)=6; x=[1,3];
clc, clear all, close all
% instruction to write results on external file
fid=fopen('Logsin.m','w');
h=0.1; a=1; b=3;    % h is the step size, x=[a,b] x-range
x=a:h:b;          % computes x-array up to x=3
y=zeros(1,numel(x));  % Memory preallocation
y(1)=6; % initial condition; in MATLAB indices start at 1
Fyt=@(x,y)3*log(sin(x))-1.5*y; % change the function as you desire
              % the function is the expression after (x,y)


% table title
fprintf(fid,'%7s %7s %7s %7s %7s %7s %7s\n','i','x(i)','k1','k2','k3','k4','y(i)');
for ii=1:1:numel(x)
   k1=Fyt(x(ii),y(ii));
   k2=Fyt(x(ii)+0.5*h,y(ii)+0.5*h*k1);
   k3=Fyt((x(ii)+0.5*h),(y(ii)+0.5*h*k2));
   k4= Fyt((x(ii)+h),(y(ii)+h*k3));
   y(ii+1)=y(ii)+(h/6)*(k1+2*k2+2*k3+k4);   % min equation
   % table data
   fprintf(fid,'%7d %7.2f %7.3f %7.3f',ii, x(ii), k1, k2);
   fprintf(fid,' %7.3f %7.3f %7.3f\n', k3, k4, y(ii));
end
y(numel(x))=[ ];   % erase the last computation of y(n+1)
% Solution PLOT:
plot(x,y,'ok')
xlim([1,3])
ylim([-2,8])
title('RK-4--Numerical Solution---');
ylabel('Y-Approximated Values');    xlabel('X-Subdivision');    legend('RK4');
grid on
fclose(fid);
```

Table 3: Approximate solutions of the Numerical Experiment 3						
i	**x(i)**	**k1**	**k2**	**k3**	**k4**	**y(i)**
1	1.00	-9.518	-8.713	-8.773	-8.030	6.000
2	1.10	-8.033	-7.358	-7.409	-6.787	5.125
3	1.20	-6.789	-6.226	-6.268	-5.749	4.385
4	1.30	-5.751	-5.282	-5.318	-4.886	3.760
5	1.40	-4.888	-4.499	-4.529	-4.172	3.229
6	1.50	-4.174	-3.854	-3.878	-3.586	2.777
7	1.60	-3.587	-3.326	-3.346	-3.109	2.390
8	1.70	-3.110	-2.900	-2.916	-2.727	2.056
9	1.80	-2.727	-2.562	-2.574	-2.427	1.765
10	1.90	-2.428	-2.301	-2.311	-2.201	1.508
11	2.00	-2.201	-2.109	-2.116	-2.040	1.277
12	2.10	-2.040	-1.980	-1.984	-1.939	1.066
13	2.20	-1.939	-1.908	-1.911	-1.895	0.867
14	2.30	-1.894	-1.893	-1.893	-1.907	0.676
15	2.40	-1.907	-1.936	-1.934	-1.980	0.487
16	2.50	-1.979	-2.043	-2.038	-2.121	0.293
17	2.60	-2.120	-2.226	-2.218	-2.350	0.088
18	2.70	-2.349	-2.512	-2.500	-2.705	-0.134
19	2.80	-2.703	-2.959	-2.940	-3.272	-0.386
20	2.90	-3.268	-3.708	-3.675	-4.301	-0.682
21	3.00	-4.294	-5.272	-5.199	-7.180	-1.054

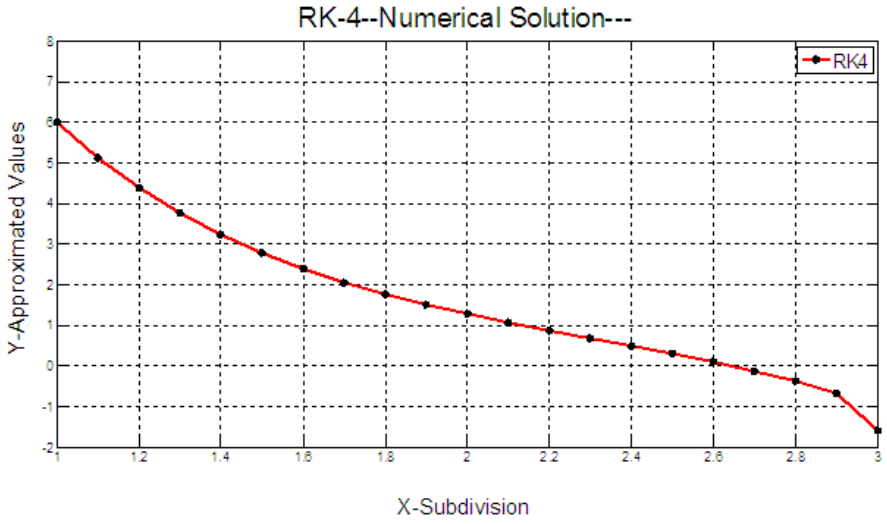

Figure 3: Graphically representation for the solutions of table 3.

Numerical Experiment 4:
Consider the initial value problem$y' = 4\log(x) - 2y$ with the initial condition $y(0) = 4$ on the interval $1 \leq x \leq 3$.

MATLAB Programming for above Numerical Experiment:

```
% Runge-Kutta 4th order with MATLAB
% It calculates an ODE using Runge-Kutta 4th order method
% Equation to solve: Y'=4*log(x)-2*Y; Y(0)=4; x=[1,3];
% Author Ido Schwartz, modified by Marco Arocha
clc, clear all, close all
% instruction to write results on external file
fid=fopen('Log.m','w');
h=0.1; a=1; b=3;    % h is the step size, x=[a,b] x-range
x=a:h:b;          % computes x-array up to x=3
y=zeros(1,numel(x));  % Memory preallocation
y(1)=4; % initial condition; in MATLAB indices start at 1
Fyt=@(x,y)4*log(x)-2*y; % change the function as you desire
```

```
            % the function is the expression after (x,y)
% table title
fprintf(fid,'%7s %7s %7s %7s %7s %7s %7s\n','i','x(i)','k1','k2','k3','k4','y(i)');
for ii=1:1:numel(x)
   k1=Fyt(x(ii),y(ii));
   k2=Fyt(x(ii)+0.5*h,y(ii)+0.5*h*k1);
   k3=Fyt((x(ii)+0.5*h),(y(ii)+0.5*h*k2));
   k4= Fyt((x(ii)+h),(y(ii)+h*k3));
   y(ii+1)=y(ii)+(h/6)*(k1+2*k2+2*k3+k4);   % min equation
   % table data
   fprintf(fid,'%7d %7.2f %7.3f %7.3f',ii, x(ii), k1, k2);
   fprintf(fid,' %7.3f %7.3f %7.3f\n', k3, k4, y(ii));
end
y(numel(x))=[ ];   % erase the last computation of y(n+1)
% Solution PLOT:
plot(x,y,'ok')
xlim([1,3])
ylim([0,6])
title('RK-4--Numerical Solution---');
ylabel('Y-Approximated values');    xlabel('X-Subdividions');    legend('RK4');
grid on
fclose(fid);
```

Table 4: Approximate solutions of the Numerical Experiment 4

i	x(i)	k1	k2	k3	k4	y(i)
1	1.00	-8.000	-7.005	-7.104	-6.198	4.000
2	1.10	-6.205	-5.407	-5.486	-4.760	3.293
3	1.20	-4.765	-4.125	-4.189	-3.607	2.747
4	1.30	-3.612	-3.099	-3.151	-2.685	2.331
5	1.40	-2.689	-2.279	-2.320	-1.949	2.017
6	1.50	-1.951	-1.625	-1.658	-1.362	1.787
7	1.60	-1.364	-1.104	-1.130	-0.895	1.622
8	1.70	-0.897	-0.691	-0.712	-0.526	1.510
9	1.80	-0.528	-0.365	-0.381	-0.235	1.439
10	1.90	-0.236	-0.109	-0.121	-0.007	1.402
11	2.00	-0.007	0.092	0.082	0.171	1.390
12	2.10	0.171	0.248	0.240	0.309	1.399
13	2.20	0.308	0.367	0.361	0.414	1.423
14	2.30	0.413	0.458	0.454	0.493	1.459
15	2.40	0.493	0.526	0.523	0.551	1.505
16	2.50	0.551	0.575	0.573	0.594	1.557
17	2.60	0.593	0.610	0.609	0.623	1.614
18	2.70	0.623	0.634	0.633	0.642	1.675
19	2.80	0.641	0.648	0.647	0.652	1.738
20	2.90	0.652	0.655	0.655	0.657	1.803
21	3.00	0.657	0.657	0.657	0.657	1.869

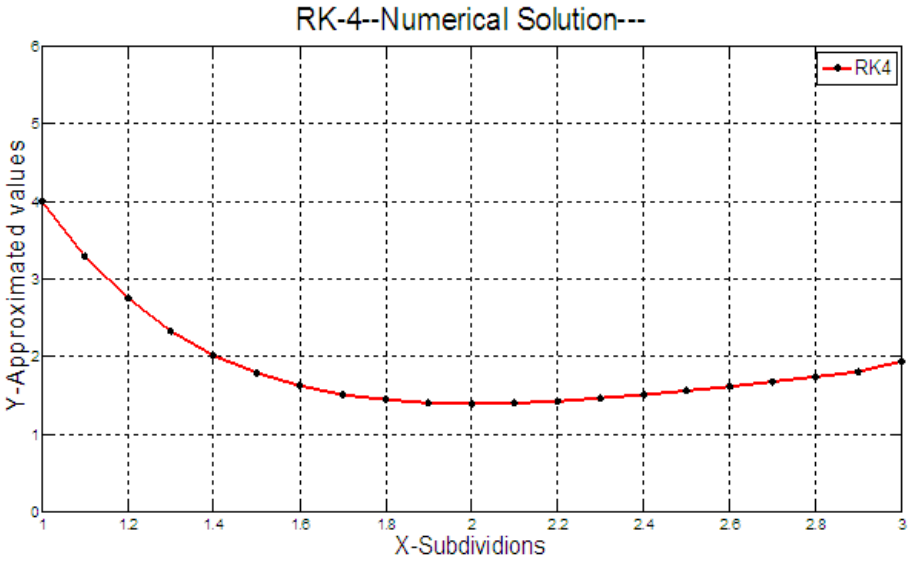

Figure 4: Graphically representation for the solutions of table 4.

II. Result and Discussion

In the present investigation we have analysed for different numerical problems by using Runge-Kutta fourth order method with the help of MATLAB programming. The results are compiled in tables 1 to 4 simultaneously results are also graphically represented in figures 1 to 4. Islam [2015], Hossen et.al [2019], Jamali [2019] Sharma [2021] and Sharma and Kumar [2021] have proved by taking different numerical examples the Runge-Kutta 4^{th} order method is the best. We have applied this method to solve different initial value problems which cannot be solved analytically

III. Conclusion:

The problem which cannot be solved analytically, can be easily handled by Runge-Kutta fourth order method which has also been recommended by earlier workers.

Acknowledgment

Thanks are due to the Himachal Pradesh University, Shimla for providing funds to carry out this work.

REFERENCES

[1]. Arefin, M.A. Gain, B. Karim R. and Hossain. S. (2020). A Comparative Exploration on Different Numerical Methods for Solving Ordinary Differential Equations, Journal of Mechanics of Continua and Mathematical Sciences, Vol. 15, pp. 1-11.

[2]. Gowri, P. Priyadharsini S. and Maheswari. T. (2017). A case study on Runge-Kutta fourth order differential equations and Its application, Impeial Journal of Interdisciplinary Research, Vol. 3(2), pp. 134-139.

[3]. Hossen, M. Ahemed, Z. Kabir R. and Hossain. Z. (2019). A Comparative Investigation on Numerical Solution Of Initial Value Problem by Using Modified Euler's Method and Runge-Kutta Method, IOSR Journal of Mathematics, Vol. 15, pp. 40-45.

[4]. Islam. M.A. (2015). Accuracy Analysis of Numerical solutions of initial value problems (IVP) for ordinary differential equations (ODE), IOSR Journal of Mathematics, Vol. 11(3), pp. 18-23.

[5]. Islam. M.A. (2015). Accurate solutions of initial value problems for ordinary differential equations with the fourth order Runge-Kutta method, Journal of Mathematics Research, Vol. 7(3), pp. 41-45.

[6]. Islam. M.A. (2015). A Comparative Study on Numerical Solutions of Initial Value Problems (IVP) for Ordinary Differential Equations (ODE) with Euler and Runge-Kutta Methods, American Journal of Computational Mathematics, Vol. 5(03), pp. 393-404.

[7]. Jamali. N. (2019). Analysis and Comparative Study of Numerical Methods to Solve Ordinary Differential Equation with Initial Value Problem, International Journal of Advanced Research (IJAR), Vol. 7(5), pp. 117-128.

[8]. Sharma P.L. and Kumar. A. (2021). Review Paper on The Runge-Kutta Methods to Study Numerical Solutions of Initial Value Problems in Ordinary Differential Equations, IASET International journal of applied Mathematics and Statistical Sciences (IJAMSS), Vol. 107, pp. 45-54.
[9]. Sharma. P.L. (2021). Analysis of different Numerical Methods for Solving Initial Value Problems in Ordinary Differential Equations, Journal of Physical Sciences, Vol. 1(1), pp. 135-142.

Pushap Lata Sharma, et. al. "Demonstration Study on Runge-Kutta Fourth Order Method by Using MATLAB Programming." *IOSR Journal of Mathematics (IOSR-JM)*, 17(4), (2021): pp. 01-09.