A Comparative Study On Second Order, Third Order, Fourth Order& Butcher's Fifth Order Runge-Kutta Methods With First Order Initial Value Problem(IVP) Through Manually And MATLAB.

Surendra Kumar Srivastava
Assistant Professor
Jayoti Vidyapeeth Women's University

Shivani Tiwari
Research Scholar
Jayoti Vidyapeeth Women's University

ABSTRACT

This present paper focused on the error analysis of first order ordinary differential equations through Runge-Kutta methods and Butcher's 5th order Runge-Kutta method for solving initial value problem (IVP) through manually and MATLAB. In order to achieve higher accuracy in the solution of ordinary differential equations concerning the several functions, the step size needs to be very small. Numerical solution of ODE contains two types of error. One of them is Round off errors and second of them is truncation errors. Both errors generally occur when ODE (Initial value problem as well as Boundary value problem) are solved numerically. Round off error is originated to represent the significant figures in computers while truncation errors arise when approximations are used to estimate some quantity.

KEYWORDS: Initial value problem, Runge-Kutta method, Butcher's 5th order Runge-Kutta method, MATLAB.

1. INTRODUCTION

1.1 SECOND ORDER RUNGE-KUTTA METHOD (RK2)

$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$

Where $k_1 = f(x_n, y_n)$

$k_2 = f(x_n + h, y_n + hk_1)$

For n = 0, 1, 2…….

1.2 THIRD ORDER RUNGE-KUTTA METHOD (RK3)

$y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3)$

Where $k_1 = \mathrm{f}(x_n, y_n)$

$k_2 = \mathrm{f}(x_n + \frac{h}{2}, y_n + h\frac{k_1}{2})$

$k_3 = \mathrm{f}(x_n + h, y_n - hk_1 + 2hk_2)$

For n = 0, 1, 2,.....

1.3 FOURTH ORDER RUNGE-KUTTA METHOD (RK4)

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

Where, $k_1 = \mathrm{f}(x_n, y_n)$

$k_2 = \mathrm{f}(x_n + \frac{h}{2}, y_n + h\frac{k_1}{2})$

$k_3 = \mathrm{f}(x_n + \frac{h}{2}, y_n + h\frac{k_2}{2})$

$k_4 = \mathrm{f}(x_n + h, y_n + hk_3)$

For n = 0, 1, 2.....

1.4 BUTCHER'S 5TH ORDER RUNGE-KUTTA METHOD (RK5)

$y_{n+1} = y_n + \frac{h}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6)$

Where, $x_{n+1} = x_n + \mathrm{h}$

$k_1 = \mathrm{f}(x_n, y_n)$

$k_2 = \mathrm{f}(x_n + \frac{h}{4}, y_n + k_1 h\frac{1}{4})$

$k_3 = \mathrm{f}(x_n + \frac{h}{4}, y_n + h\frac{k_1}{8} + h\frac{k_2}{8})$

$k_4 = \mathrm{f}(x_n + \frac{h}{2}, y_n - hk_2\frac{1}{2} + \mathrm{h}\,k_3)$

$k_5 = \mathrm{f}(x_n + h\frac{3}{4}, y_n + hk_1\frac{3}{16} + \frac{9}{16}hk_4)$

$k_6 = \mathrm{f}(x_n + h, y_n - \frac{3}{7}hk_1 + \frac{2}{7}hk_2 + \frac{12}{7}hk_3 - \frac{12}{7}hk_4 + \frac{8}{7}hk_5)$

For n = 0, 1, 2,.....

2 STATEMENT OF THE PROBLEM

Let us examine the initial value problem of first order ODE of the form

$\frac{dy}{dx}$ = f(x, y(x))

With initial condition y(x_0) = y_0 .In which we investigate the error by (1.1), (1.2), (1.3) and (1.4) through numerical approach which is shown also by graph.

3 NUMERICAL DISCUSSION

PROBLEM 3.1: Considering the initial value problem $\frac{dy}{dx}$ = x +y with initial condition y(0) = 1 in the interval 0 ≤ x ≤ 0.2. The exact solution of this problem is y = 2 e^x – 1 – x. We are taking step size h = 0.1

Table:1(ComparisonTable)Manually

Methods	Approximate value Y(x)=y(0.1)	Approximate value Y(x)=y(0.2)	Absolute Error(0.1)=|YTrue-Approximate value|	Absolute Error(0.2)=|YTrue-Approximate value|
RK2	1.11	1.24205	0.000342	0.000756
RK3	1.110333	1.242786	0.000009	0.0000195
RK4	1.110342	1.242805	0.00000016	0.0000005
Butcher 5	1.1113063	1.2448289	0.000964	0.002022

Here the value of x as like as Runge-Kutta 2 < Runge-Kutta 3<**Runge-Kutta 4** ≈ **Exact value**< Butcher's 5 RK5.For each value of x, absolute error is like as Runge-Kutta 4 < Runge-Kutta 3< Runge-Kutta 2<Butcher 5 Runge-Kutta**.**

Now, considering the same initial value problem y' = y + x, y(0) = 1 in the interval [0, 3]. Taking step size h = 0.2. The exact solution is given by $2e^x$ - 1 - x. Absolute Errors are obtained through MATLAB that shown in **Table: 2**

And command for these methods are mentioned

Runge-kutta second order method:

```
%rk2:runge kutta of second order

clc;
clear all;
close all;
```

```
  % y' = y+x ode condition
f = @(x,y) y+x;
fex = @(x) 2*exp(x)-x-1; % exact solution
a= 0;
b= 3;
n =15;
h=(b-a)/n;
y(1) =1;  %initial value
i = 0;
for x= a:h:b
    i = i+1;
    K1 = f(x,y(i));   %initializing solution
 K2  = f(x+h,y(i)+h*K1);
y(i+1) = y(i)+ h*(0.5*K1 + 0.5*K2);
g(i) = fex(x);
xx(i) = x;
Error(i) = abs(g(i) - y(i)); %error obtain
end
%plot result
plot(xx,y(1:n+1),'b',xx,g,'g','LineWidth',2)
legend('RK2','Exact solution')
xlabel('x')
ylabel('y')
title('RK2 vs exact solution')
figure
plot(xx,Error,'g','LineWidth',2)
legend('Exact curve')
xlabel('x')
ylabel('y')
```

Runge-kutta third order method:

```
%rk3:runge kutta of third order
clc;
clear all;
close all;
 % y' = y+x ode condition
 f = @(x,y) y+x;
fex = @(x) 2*exp(x)-x-1; % exact solution
a= 0;
b= 3;
n =15;
h=(b-a)/n;
y(1) =1;  %initial value
i = 0;
for x= a:h:b
    i = i+1;
K1 = f(x,y(i));    %initializing solution
K2 = f(x+h*0.5,y(i)+h*K1*0.5);
K3 = f(x+h,y(i)-h*K1+2*K2*h);
y(i+1)=y(i)+h*(1/6)*(K1+4*K2+K3);
```

```
g(i) = fex(x);
xx(i) = x;
Error(i) = abs(g(i) - y(i)); %error obtain
end
%plot result
plot(xx,y(1:n+1),'k',xx,g,'g','LineWidth',2)
legend('RK3','Exact solution')
xlabel('x')
ylabel('y')
title('RK3 vs exact solution')
```

Runge kutta fourth order method:

```
%rk4:runge kutta of fourth order
clc;
clear all;
close all;
% y' = y+x ode condition
f = @(x,y) y+x;
fex = @(x) 2*exp(x)-x-1; % exact solution
a=0;
b= 3;
n =15;
h=(b-a)/n;
y(1) =1;      %initial value
i = 0;
for x= a:h:b
    i = i+1;
K1 = f(x,y(i));  %initializing solution
K2 = f(x+h*0.5,y(i)+h*K1*0.5);
K3 = f(x+h*0.5, y(i)+h*K2*0.5);
K4 = f(x+h,y(i)+h*K3);
y(i+1) =y(i)+h*(1/6)*(K1 +2*K2+2*K3+ K4);
 g(i) = fex(x);
xx(i) = x;
 Error(i) = abs(g(i) - y(i));  %error obtain
end
%plot result
plot(xx,y(1:n+1),'c',xx,g,'g','LineWidth',2)
legend('RK4','Exact solution')
xlabel('x')
ylabel('y')
 title('RK4 vs exact solution')
```

Butcher's 5th order Runge-Kutta method:

```
% butcher 5th order runge kutta method
clc;
clear all;
```

```
close all;
% y' = y+x ode condition
f = @(x,y) y+x;
fex = @(x) 2*exp(x)-x-1; % exact solution
a=0;
b= 3;
n =15;
h=(b-a)/n;
y(1) =1;      %initial value
i = 0;
for x= a:h:b
    i = i+1;
K1 = f(x,y(i));  %initializing solution
K2 = f(x+h*(1/4),y(i)+h*(1/4)*K1);
K3 = f(x+h*0.5, y(i)+ h*K1*(1/8)+h*K2*(1/8));
K4 = f(x+h*0.5,y(i)- h*K2*0.5+h*K3);
K5 =f(x+3*h*(1/4),y(i)+(3/16)*h*K1+(9/16)*h*K4);
K6 = f(x+h,y(i)- 3*h*K1*(1/7)+2*h*K2*(1/7)+(12/7)*h*K3-(12/7)*h*K4 +
(8/7)*h*K5);
y(i+1) =y(i)+h*(1/90)*(7*K1 +32*K3+12*K4+ 32*K5 +7*K6);
g(i) = fex(x);
xx(i) = x;
 Error(i) = abs(g(i) - y(i));  %error obtain
end
%plot result
plot(xx,y(1:n+1),'r',xx,g,'g','LineWidth',2)
legend('butcher5','Exact solution')
xlabel('x')
ylabel('y')
title('butcher5 vs exact solution')
```

Command for multiple graph for Runge-Kutta's method, Butcher's 5 and Exact solution

```
clc;
clear all;
close all;
% y' = y+x ode condition
f1 = @(x,y1) y1+x;
fex1 = @(x) 2*exp(x)-x-1;    % exact solution
f2 = @(x,y2) y2+x;
fex2 = @(x) 2*exp(x)-x-1;
f3 = @(x,y3) y3+x;
fex3 = @(x) 2*exp(x)-x-1;
f4 = @(x,y4) y4+x;
fex4 = @(x) 2*exp(x)-x-1;
a=0;
b= 3;
n =15;
```

```
h=(b-a)/n;
y1(1) =1; %initial value
y2(1) =1;
y3(1) =1;
y4(1) =1;
i = 0;
for x= a:h:b
    i = i+1;
    K1 = f1(x,y1(i));
    K2 = f1(x+h*(1/4),y1(i)+h*(1/4)*K1);
     K3 = f1(x+h*0.5, y1(i)+ h*K1*(1/8)+h*K2*(1/8));
     K4 = f1(x+h*0.5,y1(i)- h*K2*0.5+h*K3);
     K5 = f1(x+3*h*(1/4),y1(i)+ (3/16)*h*K1+(9/16)*h*K4);
       K6 = f1(x+h,y1(i)- 3*h*K1*(1/7)+2*h*K2*(1/7) + (12/7)*h*K3
-(12/7)*h*K4 + (8/7)*h*K5);
y1(i+1) =y1(i)+h*(1/90)*(7*K1 +32*K3+12*K4+ 32*K5 +7*K6);
g1(i) = fex1(x);
xx(i) = x;
Error(i) = abs(g1(i) - y1(i));    %error obtain
 P1 = f2(x,y2(i));          %initializing solution
P2  = f2(x+h,y2(i)+h*P1);
y2(i+1) = y2(i)+ h*(0.5*P1 + 0.5*P2);
g2(i) = fex2(x);
xx(i) = x;
Error(i) = abs(g2(i) - y2(i));
L1 = f3(x,y3(i));         %initializing solution
     L2 = f3(x+h*0.5,y3(i)+h*L1*0.5);
    L3 = f3(x+h*0.5, y3(i)+h*L2*0.5);
     L4 = f3(x+h,y3(i)+h*L3);
y3(i+1) =y3(i)+h*(1/6)*(L1 +2*L2+2*L3+ L4);
g3(i) = fex3(x);
xx(i) = x;
Error(i) = abs(g3(i) - y3(i));
M1 = f4(x,y4(i));         %initializing solution
     M2 = f4(x+h*0.5,y4(i)+h*K1*0.5);
     M3 = f4(x+h, y4(i)-h*M1+2*h*M2);
y4(i+1) =y4(i)+h*(1/6)*(M1 +4*M2+M3);
g4(i) = fex4(x);
xx(i) = x;
Error(i) = abs(g4(i) - y4(i));     %error obtain
end
%plot result
plot(xx,y1(1:n+1),'r',xx,g1,'g',xx,y2(1:n+1),'b',xx,g2,'g',xx,y3(1:n+1),
'c',xx,g3,'g',xx,y4(1:n+1),'k',xx,g4,'g','LineWidth',2)
xlabel('x')
ylabel('y')
title('butcher5 vs exact solution vs rk2 vs rk4 VS RK3')
```

Errors are obtained through MATLAB software for RUNGE-KUTTA second order, third order, fourth order &Butcher's 5 order Runge-Kutta methods that shown below when step size h = 0.2

Table: 2 (comparison table h= 0.2) MATLAB

x- value	RK2	RK3	RK4	Butcher 5
0	0	0	0	0
0.2000	0.0028	0.0001	0.0000	0.0041
0.4000	0.0068	0.0003	0.0000	0.0092
0.6000	0.0125	0.0006	0.0000	0.0153
0.8000	0.0204	0.0010	0.0000	0.0228
1.0000	0.0311	0.0015	0.0001	0.0320
1.2000	0.0456	0.0023	0.0001	0.0432
1.4000	0.0650	0.0032	0.0001	0.0568
1.0000	0.0907	0.0045	0.0002	0.0736
1.8000	0.1245	0.0062	0.0002	0.0940
2.0000	0.1688	0.0084	0.0003	0.1189
2.2000	0.2267	0.0113	0.0004	0.1493
2.4000	0.3019	0.0150	0.0006	0.1865
2.6000	0.3953	0.0199	0.0008	0.2319
2.8000	0.5249	0.0262	0.0010	0.2874
3.0000	0.6865	0.0342	0.0014	1.3551

Table: 2 shows the absolute error of Runge-Kutta methods and Butcher's 5 order method with step size h = 0.2. These absolute error values are like Runge-Kutta 4 < Runge-Kutta 3 <Butcher 5 Runge-Kutta< Runge-Kutta 2.

ERROR REPRESENTATION BY GRAPH

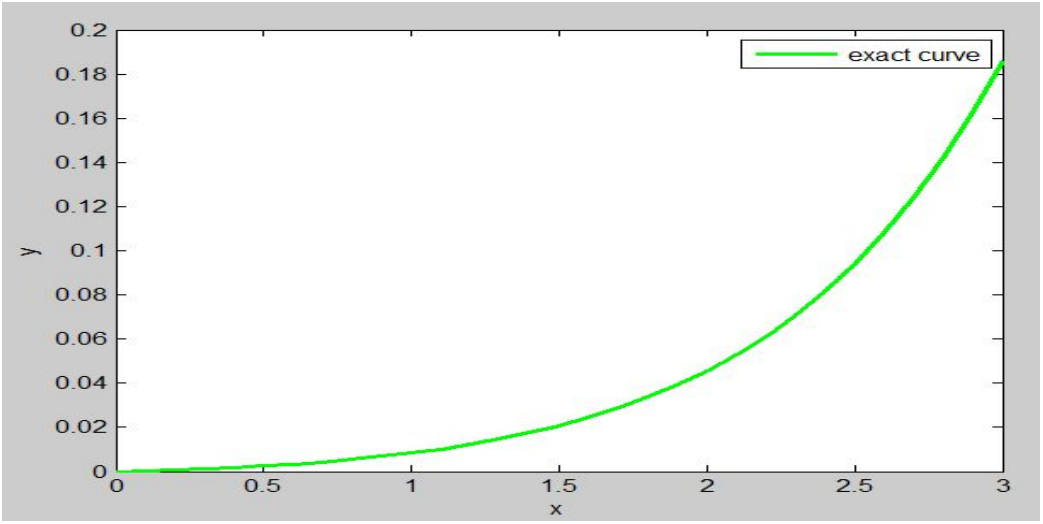

Figure 4.1 Exact curve

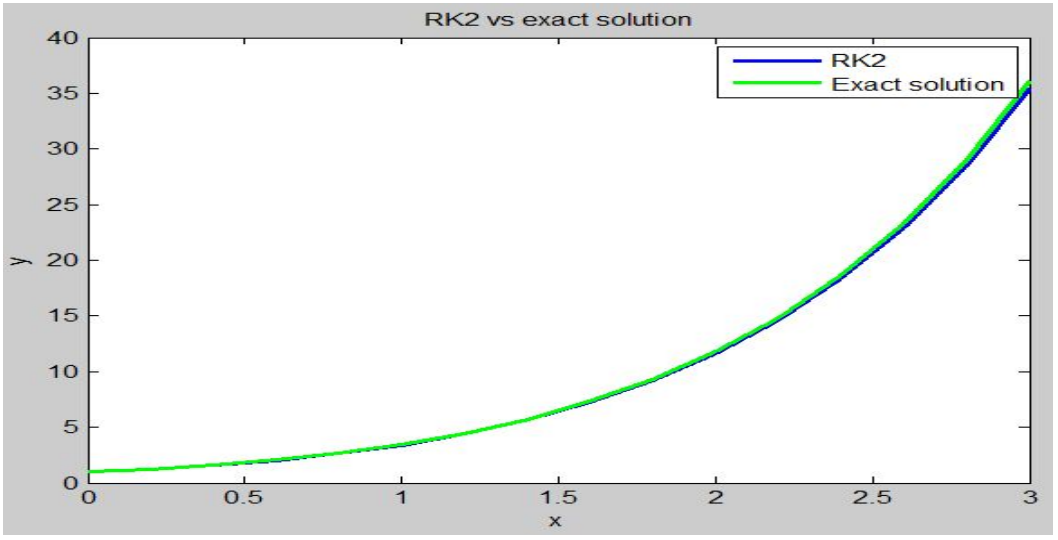

Figure 4.2 Relation of ODE between exact curve & Runge-Kutta second order method

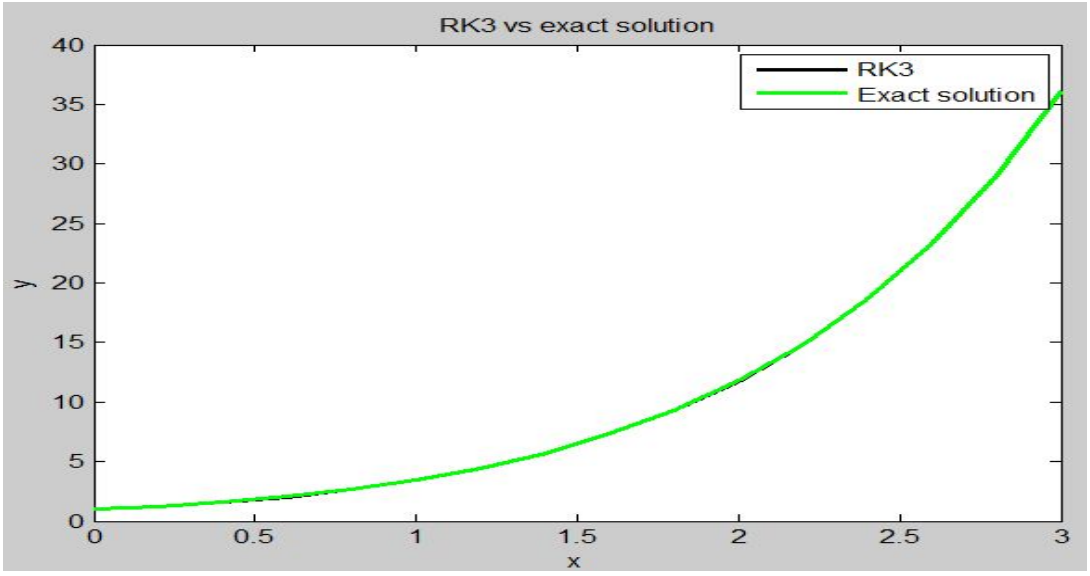

Figure 4.3 Relation of ODE between exact curve & Runge-Kuttathird order method

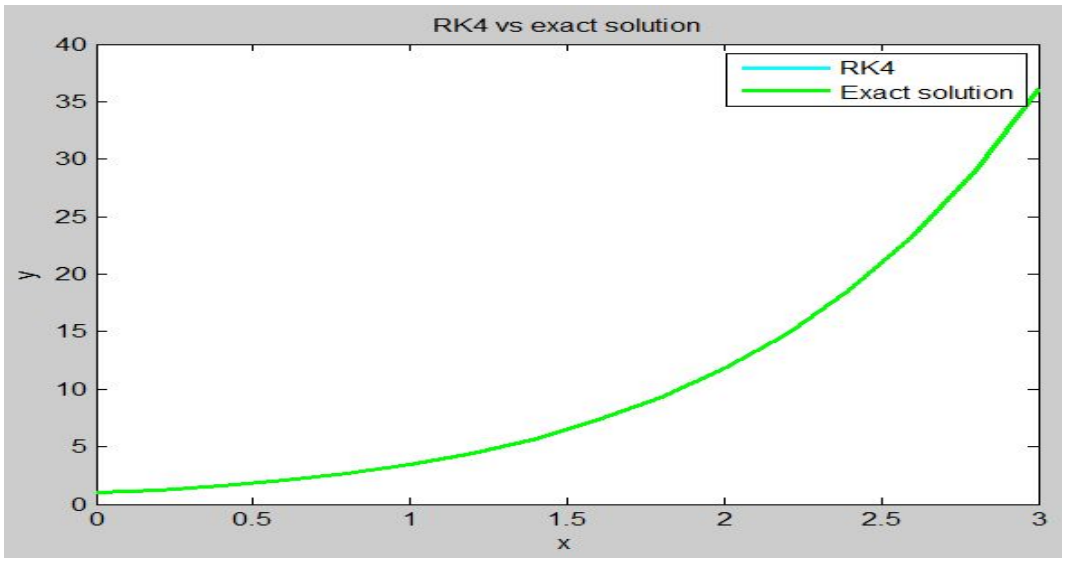

Figure 4.4 Relation of ODE between exact curve & Runge-Kutta fourth order method

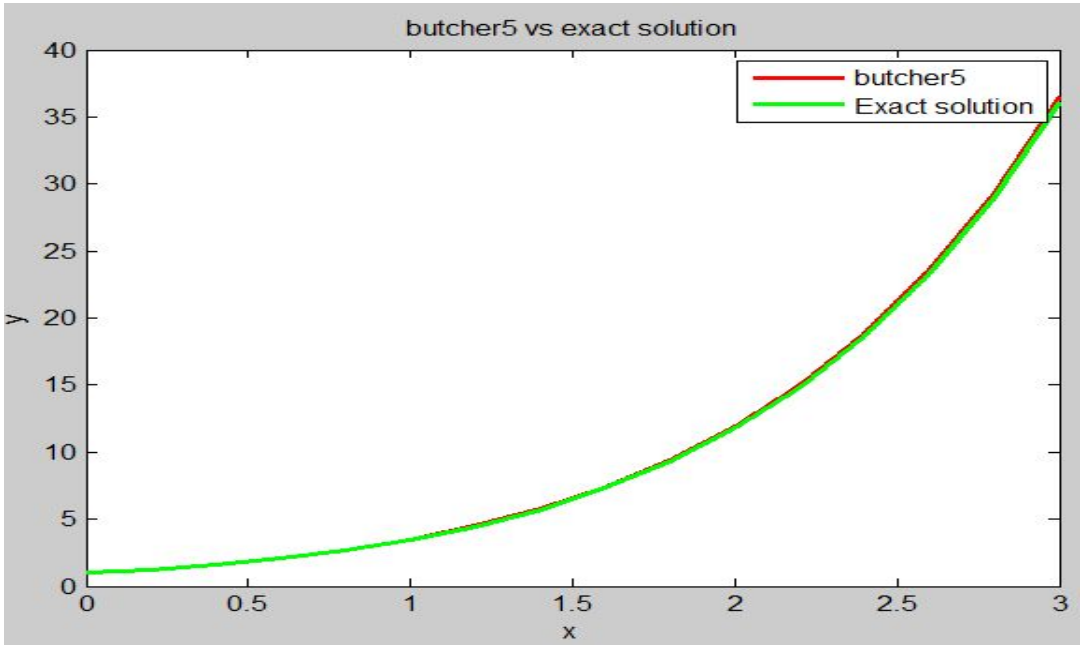

Figure 4.5 Relation of ODE between exact curve & Butchers fifth order method

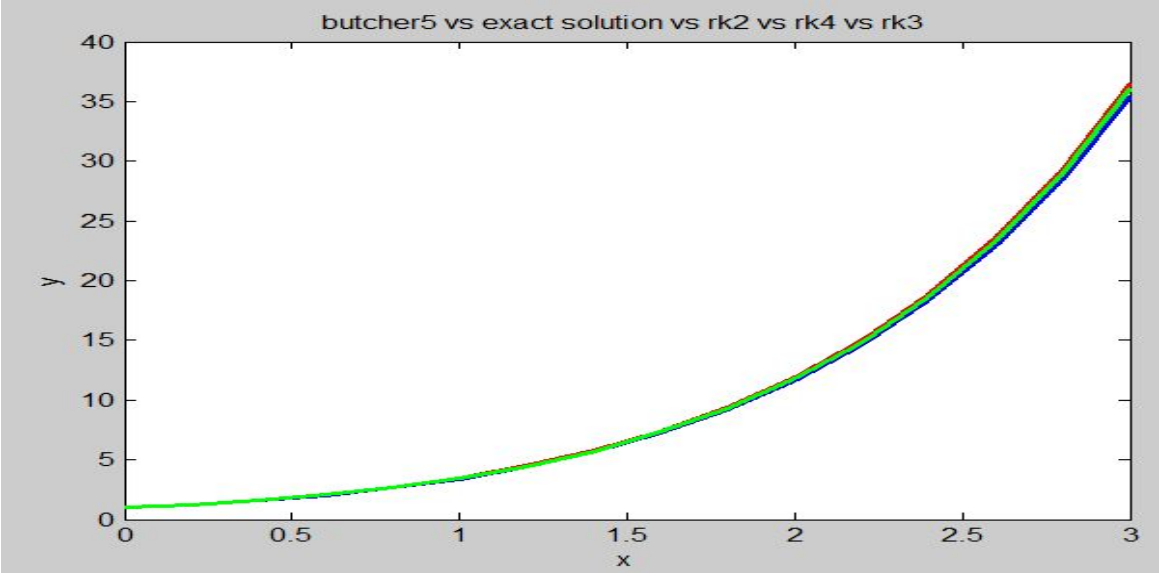

Figure 4.6 Relation of ODE among exact curve, Butchers fifth order RK and Runge-Kutta second, third, fourth order method

Figure 4.2, 4.3, 4.4 and 4.5 show the function $2e^x - 1 - x$ on the interval [0, 3] with 15 steps using MATLAB. In these figures blue, black, cyan and red colours curves show the approximate solution curve obtained by Runge-Kutta second order method, third order method, fourth order method and butcher's 5^{th} order method respectively and green colour curve indicates the solution of exact curve in **Figure 4.1**. Here, RK2 solution curve is below to exact curve. RK3 solution curve is almost near to exact curve as displayed**.** RK4 solution curve is just lie on the exact solution curve and Butcher 5^{th} solution curve is just above to the solution curve. After making loop for common graph we obtain **Figure 4.5** in which we can clearly see that exact solution curve is in between Butcher's 5^{th} order Runge-Kutta & RK2 solution curve and RK3 solution curve is just near to the exact curve and RK4 is just lie on the exact solution curve.

5. CONCLUSION

This present paper has focused comparative study on among various numerical methods like RK2, RK3, RK4 and Butcher's 5 Runge-Kutta method through manually and MATLAB. Table 1, 2 and 3 and Figure 4.5 obtained from several mentioned methods through manually and MATLAB consequently are concluded that Runge-Kutta fourth order method is more appropriate than others. Particularly, the reliability of solution of Runge-Kutta fourth order method is much closer to that of the exact solution and error is almost negligible compared to other methods.

Thus, we can apply Runge-Kutta fourth order method for solving first order ordinary differential equations of initial value problem through manually for small interval and also use MATLAB coding as mentioned for large interval as well as small interval.

REFERENCES

1. Butcher J.C. Fifth order Runge-Kutta methods, BIT Numerical Mathematics, 35(2), 1995
2. C. Senthilnathan, A numerical solution of Initial value problems for Ordinary differential equations with Euler and Higher order of Runge-Kutta method using MATLAB, IJESI, VOLUME 7, Issue 4 Ver. III, April 2018
3. Claudio Faria Lopes Junior, Silva ed. Int. Journal of Engineering, Vol. 8, Issue 5(Part-II) May 2018, pp24-29.
4. Dr. B.S. Grewal, Numerical Methods in engineering & science with programs in C & C++, Khanna Publication 9(2010)
5. Hossain et. al., A comparative study on fourth and Butcher's fifth order Runge-Kutta methods with third order initial value problem(IVP), Applied and computational mathematics, 6(6), 2017
6. J.C. Butcher, On Runge-Kutta methods of high order, J. Austral. Math Soc. 4(1964) 179-194
7. J.C. Butcher, The Numerical Analysis of Ordinary Differential Equation, John Wiley & sons, New York, 1987
8. J.C. Butcher, A history of Runge-kutta methods, Applied numerical mathematics 20(1996) 247-260.
9. Ince, E.L, Ordinary Differential Equations, 1st Edition, 1956, Dover publications, Inc, New York
10. Mathews and Fink, Numerical Methods Using MATLAB, Pearson Publication. 9(2004) 500-502